R Scripts

Data Visualization

Data_visualization.R

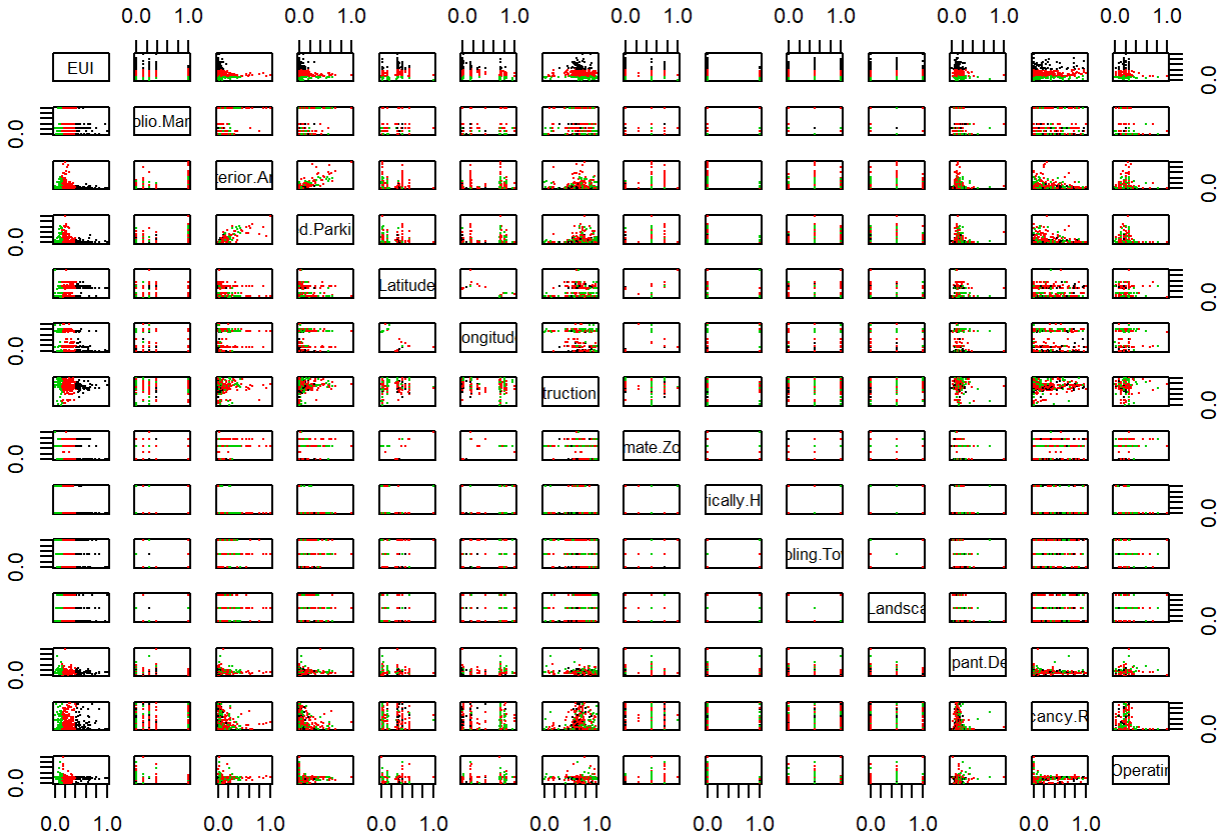
carle

Sat Oct 13 08:22:22 2018

```
####Load and PreProcess Data ####
set.seed(1234)
setwd("C:/Users/carle/Documents/Data")
getwd()
```

```
## [1] "C:/Users/carle/Documents/Data"
```

```
# Load Data (use an individual year dataset)
data <- read.csv(file.choose(), row.names = 1, check.names = FALSE, na.strings = "NA", sep = "
,", stringsAsFactors = TRUE, header = TRUE)
#Rename Columns to have 'syntactically valid' names
cnames <- cbind("EUI","Qualitative.EUI","Actual.E_C.Energy","Actual.Thermal.Energy", "Actual.T
otal.Energy",
                "Portfolio.Manager","Net.Rentable.Area","Exterior.Area","Gross.Floor.Area", "E
nclosed.Parking.Area",
                "Latitude","Longitude","Construction.Year","No.of.Structures",
                "Building.Class","Closest.Major.City","Climate.Zone","Electrically.Heated","Co
oling.Tower","Soft.Landscaping",
                "Occupant.Density","Vacancy.Rate","Weekly.Operating.Hours")
colnames(data) <- cnames
rnames <- row.names(data)

# Order the factor variable Qualitative.EUI
data[,"Qualitative.EUI"]=ordered(data[,"Qualitative.EUI"], levels = c("Poor","Fair","Good"))

#normalize
numerics <- unlist(lapply(data, is.numeric))
Maximums <- as.data.frame(lapply(data[,numerics], max))
Minimums <- as.data.frame(lapply(data[,numerics], min))
max.min <- rbind.data.frame(Maximums,Minimums)
normalize <- function(x){(x-min(x))/(max(x)-min(x))}
denormalize <- function(x,y){(x)*(max(max.min[,y])-min(max.min[,y]))+min(max.min[,y])}

# Reduce Variables for Visability
data.eui=data[,-c(2,3,4,5,7,9,14,15,16)]
norm.data <-as.data.frame(lapply(data.eui, normalize))
row.names(norm.data) <- rnames

#Plot
plot(norm.data, pch = 16, cex = .2, col=data[,2])
```


EUI
plio.Mar
erior.A
d.Parki
Latitude
ongitud
truction
mate.Zo
ically.H
bling.To
Landsca
pant.De
ancy.R
Operatin
0.0
1.0

```
#### Load Packages ####
library(hexbin)
library(RColorBrewer)
library(ggplot2)
library(dplyr)
library(maps)
#gpclibPermit()
library(maptools)
library(mapdata)
#library(gdal)

library(sp)
set.seed(1234)
#### Load Dataset ####
setwd("C:/Users/carle/Documents/Data")
data <- read.csv(choose.files(), row.names = 1, check.names = FALSE, na.strings = "NA", stringsAsFactors = TRUE, header = TRUE)

#Filter by region/water type and set colors
Good_EUI <- filter(data, Qualitative.EUI=="Good")
Good <- brewer.pal(6, "Paired")[4]
Fair_EUI <- filter(data, Qualitative.EUI=="Fair")
Fair <- brewer.pal(6, "Paired")[2]
Poor_EUI <- filter(data, Qualitative.EUI=="Poor")
Poor <- brewer.pal(6, "Paired")[6]

#display.brewer.all(n=NULL, type="all", select=NULL, exact.n=TRUE, colorblindFriendly=TRUE)

canada <- map_data("worldHires", "Canada")
NAmap_base <- ggplot() +
  geom_polygon(data = canada, aes(x=long, y = lat, group = group),
               fill = "white", color="black")+
  coord_fixed(xlim = c(-125, -65),  ylim = c(42, 60), ratio = 1.2)
NAmap_base
#Add Rivers
#fileName <- "http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/50m/physical/ne_50m_rivers_lake_centerlines.zip"
temp <- tempfile()
#download.file(fileName, temp, mode="wb")
unzip(temp)
library(rgdal)
#readOGR
shapeData <- readOGR("ne_50m_rivers_lake_centerlines.shp")
#unlink(c(temp, "ne_*"))
#shapeData@data$id <- rownames(shapeData@data)
#watershedPoints <- fortify(shapeData, region = "id")
#watershedDF <- merge(watershedPoints, shapeData@data, by = "id")
#NAmap2 <- NAmap1 + geom_path(data=watershedDF,
#                             aes(x = long, y = lat, group = group),
#                             color = 'grey', size=0.5)
#NAmap2
#Add Province boundaries
if (!file.exists("./src/ref/ne_50m_admin_1_states_provinces_lakes/ne_50m_admin_1_states_provinces_lakes.dbf")){
download.file(file.path('http://www.naturalearthdata.com/http/',
                        'www.naturalearthdata.com/download/50m/cultural',
                        'ne_50m_admin_1_states_provinces_lakes.zip'),
              f <- tempfile())
unzip(f, exdir = "./src/ref/ne_50m_admin_1_states_provinces_lakes")
rm(f)
}

region <- readOGR("./src/ref/ne_50m_admin_1_states_provinces_lakes", 'ne_50m_admin_1_states_provinces_lakes', encoding='UTF-8')
#View(regions)
regions <- subset(region, name %in% c("British Columbia", "Alberta", "Saskatchewan", "Manitoba", "Ontario", "Quebec", "New Brunswick", "Prince
Edward Island", "Nova Scotia", "Newfoundland and Labrador", "Yukon", "Northwest Territories", "Nunavut"))

NAmap_region=NAmap_base+geom_polygon(data = regions, aes(x=long, y = lat, group = group),
                            fill = NA, color="black")
NAmap_region

NAmap3 <- NAmap_region + geom_point(data=Poor_EUI, aes(x=Longitude, y=Latitude),
                                    fill=Poor,alpha = 0.1,colour="black",
                                    shape=25,size=4.5)
NAmap4 <- NAmap3 + geom_point(data=Fair_EUI, aes(x=Longitude, y=Latitude),
                              fill=Fair,alpha = 0.1,colour="black",
                              shape=21,size=4)
NAmapALL <- NAmap4 + geom_point(data=Good_EUI, aes(x=Longitude, y=Latitude),
                                fill=Good,alpha = 0.1,colour="black",
                                shape=22,size=2)
NAmapALL

NAmapPoor = NAmap_region + geom_point(data=Poor_EUI, aes(x=Longitude, y=Latitude),
                                      fill=Poor,alpha = 0.1, colour="black",
                                      shape=25,size=3.0)
NAmapPoor

NAmapFair = NAmap_region + geom_point(data=Fair_EUI, aes(x=Longitude, y=Latitude),
                                      fill=Fair,alpha = 0.1, colour="black",
                                      shape=21,size=3.0)
NAmapFair

NAmapGood = NAmap_region + geom_point(data=Good_EUI, aes(x=Longitude, y=Latitude),
                                      fill=Good,alpha = 0.1, colour="black",
                                      shape=22,size=3.0)
NAmapGood

plot(data)
```

PCA

Final_-_PCA__Shiny_Approach_.R

carle

Wed Oct 10 23:46:11 2018

```
# Reset Workplace Directory to pull files from aspecific folder
setwd("C:/Users/carle/Documents/Data")
getwd()
```

```
## [1] "C:/Users/carle/Documents/Data"
```

```
# Load Required Packages
library(FactoMineR)
```

```
## Warning: package 'FactoMineR' was built under R version 3.4.4
```

```
library(Factoshiny)
```

```
## Warning: package 'Factoshiny' was built under R version 3.4.4
```

```
## Loading required package: colourpicker
```

```
## Warning: package 'colourpicker' was built under R version 3.4.4
```

```
## Loading required package: shiny
```

```
## Warning: package 'shiny' was built under R version 3.4.4
```

```
##
## Attaching package: 'shiny'
```

```
## The following object is masked from 'package:colourpicker':
##
##     runExample
```

```
library(shiny)

# Load Data
data <- read.csv(file.choose(), row.names = 1, check.names = FALSE, na.strings = "NA", sep = "
,", stringsAsFactors = TRUE, header = TRUE)

# Order Qualitative
data[,"Qualitative.EUI"]=ordered(data[,"Qualitative.EUI"], levels = c("Poor","Fair","Good"))
```

```
# Sanity Check
is.ordered(data[,"Qualitative.EUI"])
```

```
## [1] TRUE
```

```
#str(data) <- Use if Errors are encountered.
#sapply(data, class) <- Use if Errors are encountered.

# Open Shiny Window
pca = PCAshiny(data)
```

```
##
## Listening on http://127.0.0.1:7702
```

```
## Warning in PCA(data.selec, quali.sup = choixquali, quanti.sup =
## choixquanti, : Missing values are imputed by the mean of the variable: you
## should use the imputePCA function of the missMDA package
```

Final_-_PCA__Prcomp_appraoch_.R

carle

Thu Oct 11 00:12:36 2018

```
##############################
####        PCA Script     ####
##############################

# Preprocessing
set.seed(1234)
setwd("C:/Users/carle/Documents/Data/2010-2015 Data")
getwd()
```

```
## [1] "C:/Users/carle/Documents/Data/2010-2015 Data"
```

```
# Load packages
library(MASS)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.4.3
```

```
library(scales)
```

```
## Warning: package 'scales' was built under R version 3.4.3
```

```
library(ggplot2)
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 3.4.4
```

```
# na.strings = "NA" populates every blank with "NA", which R will ignore
# stringsAsFactors = TRUE will let you read in 0's as real zeroes
# check.names = FALSE permits multiple words in Column names to be seperated by a space.
# ^ [R] prefers names to read as one word (syntactically valid): "a.b.c" rather than "a b c"
# Load Data - Script is prepared for the 2010|2015 Dataset
data <- read.csv(file.choose(), row.names = 1, check.names = FALSE, na.strings = "NA", sep = "
```

```
,", stringsAsFactors = TRUE, header = TRUE)
#Rename Columns to have 'syntactically valid' names
cnames <- cbind("x.2010.EUI","x.2015.EUI","D.EUI.percent","x.2010.Qualitative.EUI","x.2015.Qua
litative.EUI",
                "D.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2015.Actual.E_C","D.Actual.E
_C","x.2010.Actual.Thermal","x.2015.Actual.Thermal","D.Actual.Thermal",
                "x.2010.Actual.Energy.Use","D.Actual.Energy.Use","x.2015.Actual.Energy.Use", "
Portfolio.Manager","Portfolio.Manager.no.","Net.Rentable.Area",
                "Exterior.Area", "Gross.Floor.Area","Enclosed.Parking","Latitude",  "Longitude
","Construction.Year","No.of.Structures","Building.Class","Closest.Major.City","Climate.Zone",
                "Electrically.Heated","Cooling.Tower","Soft.Landscaping",   "Occupant.Density"
,"Vacancy.Rate","Weekly.Operating.Hours")
colnames(data) <-cnames
rnames <- row.names(data)

# Order Qualitative outcome variables
#sapply(data, class)
data[,c("x.2010.Qualitative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI")] <- lapply(data
[,c("x.2010.Qualitative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI")] , factor)
data[,"x.2010.Qualitative.EUI"]=ordered(data[,"x.2010.Qualitative.EUI"],levels = c("Poor","Fai
r","Good"))
data[,"x.2015.Qualitative.EUI"]=ordered(data[,"x.2015.Qualitative.EUI"],levels = c("Poor","Fai
r","Good"))
data[,"D.Qualitative.EUI"]=ordered(data[,"D.Qualitative.EUI"],levels = c("Poor","Fair","Good")
)
sapply(data[,c("x.2010.Qualitative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI")],is.orde
red)
```

```
## x.2010.Qualitative.EUI x.2015.Qualitative.EUI      D.Qualitative.EUI
##                   TRUE                   TRUE                   TRUE
```

```
#### Run 1 ####
i1.data<-data[,c("x.2010.Qualitative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI","x.2010
.Actual.E_C.Energy","x.2010.Actual.Thermal",
              "Portfolio.Manager.no.",  "Gross.Floor.Area", "Enclosed.Parking",
              "Latitude",   "Longitude",    "Construction.Year",    "No.of.Structures", "Build
ing.Class",
              "Electrically.Heated",    "Cooling.Tower",    "Soft.Landscaping", "Occupant.Dens
ity",
              "Vacancy.Rate",   "Weekly.Operating.Hours")]
#View(data)
##create Train and Test set
train = sample(1:nrow(i1.data), nrow(i1.data)*.80)
i1.data.train = i1.data[train,]
i1.data.test = i1.data[-train,]

#sapply(i1.data.train, class)
# Check % Variance for each feature because PCA doesn't work well if variance is too small
#nzv <- nearZeroVar(pca.i1.data, saveMetrics = TRUE)
#print(paste('Range:',range(nzv$percentUnique)))
#head(nzv)
#nzv
```

```
# Run Model
i1.pca <- prcomp(i1.data.train[,-c(1,2,3)], retx=TRUE, scale = TRUE, center = TRUE)
expl.var <- round(i1.pca$sdev^2/sum(i1.pca$sdev^2)*100)
i1.pca$sdev
```

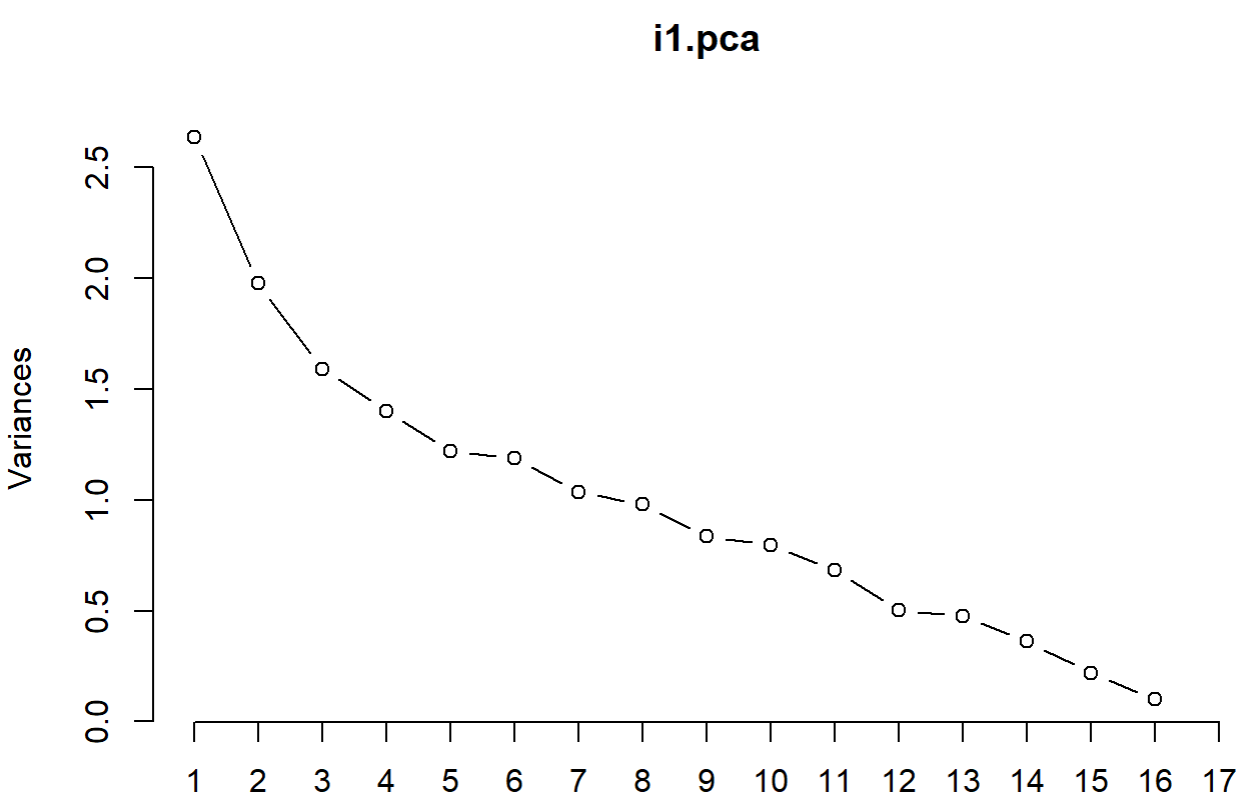
```
##  [1] 1.6232219 1.4061503 1.2614700 1.1834698 1.1038498 1.0905126 1.0183178
##  [8] 0.9895101 0.9135699 0.8920982 0.8265116 0.7082039 0.6910712 0.6007971
## [15] 0.4674370 0.3162656
```

```
page(i1.pca, method="print")
print(i1.pca)
```

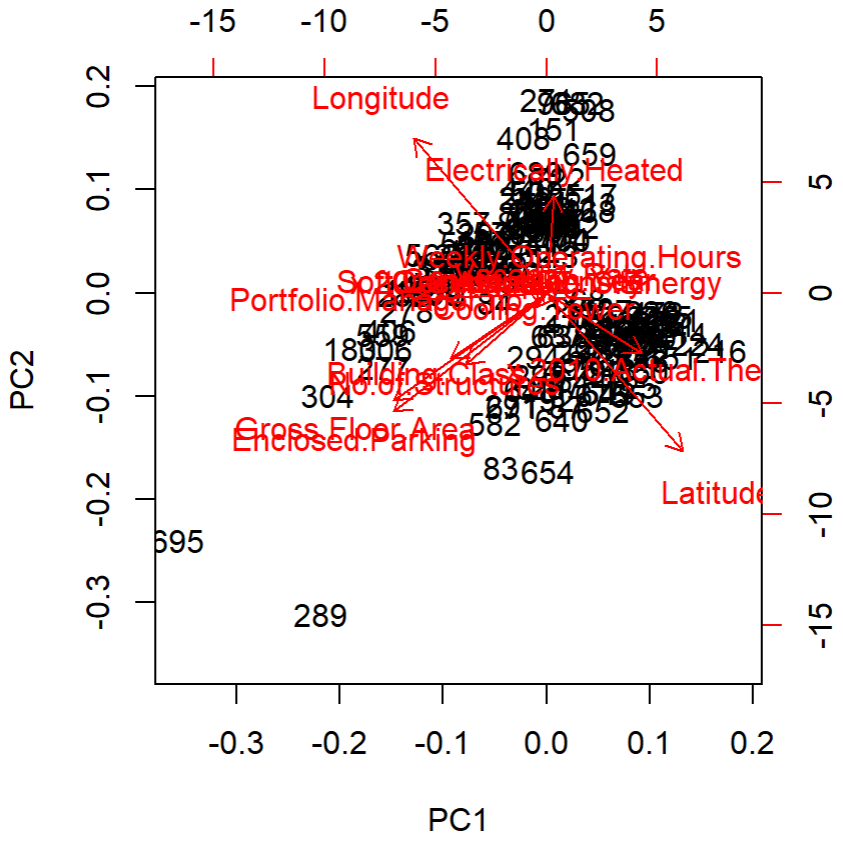
```
## Standard deviations (1, .., p=16):
##  [1] 1.6232219 1.4061503 1.2614700 1.1834698 1.1038498 1.0905126 1.0183178
##  [8] 0.9895101 0.9135699 0.8920982 0.8265116 0.7082039 0.6910712 0.6007971
## [15] 0.4674370 0.3162656
##
## Rotation (n x k) = (16 x 16):
##                                    PC1         PC2         PC3         PC4
## x.2010.Actual.E_C.Energy -0.02125529  0.02093894 -0.12314296  0.43211387
## x.2010.Actual.Thermal     0.26434346 -0.19219925  0.29569105  0.08111832
## Portfolio.Manager.no.    -0.36457940 -0.02037817  0.18244011  0.01723941
## Gross.Floor.Area         -0.42047044 -0.34105510  0.20756022  0.12336051
## Enclosed.Parking         -0.42197606 -0.37744407  0.11619340 -0.03262719
## Latitude                  0.37596310 -0.50465573 -0.01714489 -0.04864968
## Longitude                -0.36605032  0.49190556  0.11893797  0.03058100
## Construction.Year        -0.03932044  0.03599273 -0.09564599 -0.45980190
## No.of.Structures         -0.22312124 -0.22878698  0.24044281  0.23213373
## Building.Class           -0.26449474 -0.20977807 -0.47872962 -0.12175621
## Electrically.Heated       0.01901615  0.31030907  0.19444195  0.06787200
## Cooling.Tower            -0.02871213 -0.04502602  0.11452201 -0.43428252
## Soft.Landscaping         -0.19325969  0.02499688 -0.22937490 -0.37366108
## Occupant.Density         -0.06372036  0.02340046 -0.53730964  0.39047801
## Vacancy.Rate              0.01087921  0.04103008  0.13625150 -0.12941497
## Weekly.Operating.Hours    0.05051464  0.08918463  0.29351493  0.06808548
##                                   PC5         PC6          PC7         PC8
## x.2010.Actual.E_C.Energy -0.300054733  0.03635596 -0.337712965  0.57434072
## x.2010.Actual.Thermal     0.242291642  0.26438678  0.211567128 -0.21769912
## Portfolio.Manager.no.     0.397233914  0.17783005  0.093594604  0.37146368
## Gross.Floor.Area         -0.071727225  0.06516211 -0.052074064 -0.12933799
## Enclosed.Parking          0.039431884  0.04021516 -0.164140300  0.04553970
## Latitude                  0.074908935 -0.01900773 -0.045578528  0.12140654
## Longitude                -0.059277101  0.02339412  0.054962306 -0.10598246
## Construction.Year         0.307220940 -0.23083759 -0.602001919  0.09397158
## No.of.Structures         -0.081490756 -0.36704285  0.147942340 -0.08864992
## Building.Class            0.063824142 -0.12744294 -0.009921979 -0.30819164
## Electrically.Heated       0.396669120 -0.10473944  0.057847069  0.04250217
## Cooling.Tower            -0.464320080  0.24188360  0.210624009  0.30274618
## Soft.Landscaping         -0.099586958  0.31740523  0.207334789 -0.05722974
## Occupant.Density         -0.001094703  0.10748685  0.084375378 -0.13443289
## Vacancy.Rate             -0.320222152 -0.64293481  0.147217243 -0.05314830
## Weekly.Operating.Hours   -0.295199693  0.30164809 -0.542030502 -0.45899360
```

```
##                                   PC9         PC10         PC11          PC12
## x.2010.Actual.E_C.Energy -0.295717248 -0.15416101  0.02650680 -0.034221875
## x.2010.Actual.Thermal    -0.269869526 -0.36516306  0.11545919  0.451868610
## Portfolio.Manager.no.    -0.033869527 -0.15417766 -0.06712014  0.067591017
## Gross.Floor.Area         -0.028252642  0.09894103  0.25867761 -0.110473137
## Enclosed.Parking         -0.178387073  0.05611445  0.14865500  0.094350873
## Latitude                 -0.093311332  0.19338176 -0.03709433 -0.070630533
## Longitude                 0.116859597 -0.14968085  0.10407294  0.165512143
## Construction.Year         0.003603193 -0.05804938 -0.17684620  0.380983429
## No.of.Structures          0.175909778  0.14327769 -0.67191113  0.124449782
## Building.Class            0.016149681  0.03310802  0.30143521 -0.099183203
## Electrically.Heated      -0.411658343  0.66863938  0.10739297 -0.101565479
## Cooling.Tower             0.147085142  0.38847329  0.12412586  0.389517221
## Soft.Landscaping         -0.531797346 -0.11407306 -0.47196340 -0.253411236
## Occupant.Density         -0.145164823  0.24204823 -0.09841456  0.563675574
## Vacancy.Rate             -0.484309518 -0.17987232  0.18235952  0.148739577
## Weekly.Operating.Hours   -0.151127916  0.13943460 -0.11836039 -0.001491452
##                                  PC13         PC14          PC15         PC16
## x.2010.Actual.E_C.Energy -0.33625186  0.18772663 -0.013867329  0.01908723
## x.2010.Actual.Thermal    -0.35899039  0.13060966  0.009484909  0.09325669
## Portfolio.Manager.no.     0.51402384  0.43522549 -0.081051318 -0.05214267
## Gross.Floor.Area         -0.08785317 -0.22053519 -0.680662067  0.10195730
## Enclosed.Parking          0.02568842 -0.37417245  0.651484192 -0.04063199
## Latitude                  0.05637877  0.03829581 -0.119723706 -0.71209335
## Longitude                -0.24488919 -0.05975820 -0.017833225 -0.67267726
## Construction.Year        -0.14578585 -0.07806697 -0.217572924  0.05650845
## No.of.Structures         -0.23596671  0.19355497  0.055583725  0.02022979
## Building.Class           -0.23107202  0.59921736  0.108879582 -0.01629054
## Electrically.Heated      -0.20334075  0.06638679  0.024178551  0.05228438
## Cooling.Tower            -0.08657910  0.18146080 -0.007590742  0.07179923
## Soft.Landscaping         -0.10768631 -0.09389149 -0.089446380 -0.04078487
## Occupant.Density          0.26834877 -0.15836838 -0.111416943 -0.02983779
## Vacancy.Rate              0.30420703  0.05773718 -0.060753723 -0.02501373
## Weekly.Operating.Hours    0.25882797  0.29566163  0.061108168 -0.04081813
```

```
#Scree Plot

i1.pca_screeplot <-plot(i1.pca, npcs = 17, type = "lines")
```




```
i1.pca_biplot <-biplot(i1.pca)
```




```
#predict
pred_i1.pca = predict(object = i1.pca, newdata = i1.data.test[,-c(1,2,3)])

page(pred_i1.pca, method="print")
pred_i1.pca
```

```
##              PC1        PC2         PC3         PC4         PC5
## 12   1.272034220 -1.0348485  0.21670685  0.21547105  0.29109650
## 80  -0.900042055  0.2497169  0.23689466 -0.41834383 -0.41460846
## 85   0.709180078 -1.3426623  0.86309116 -0.57278137 -1.45771045
## 103  0.003727095  1.4995164 -0.22537858 -0.64177970 -1.27733569
## 132  1.282261522 -0.4439345 -0.86013276 -0.28020510 -0.33084236
## 144 -0.359629046  1.3374230 -1.95100794 -0.39472167  0.53037001
## 148  0.314256694  1.2575959 -1.20668549  1.38732303 -0.92276511
## 188 -3.125400183 -4.2291142  1.25401731  0.63448903  1.15503173
## 189 -3.701630036 -4.8848116  1.84834047  0.92985309  0.99744195
## 196 -0.894813945  0.9695702 -0.53062666 -0.18214465  0.77582091
## 199 -1.974312246  1.0118065  0.08526502 -1.38809974 -1.50900590
## 201 -3.518954771 -1.2146343  0.90359743  0.51680808  0.85826420
## 213  2.331288137 -0.5044983  1.66993364 -0.47926813 -0.74675998
## 253 -0.378569783  0.9606932 -0.66250346 -0.44885263 -0.45295555
## 254  0.875904217  1.1699988  1.67255605  0.52974732 -0.78762255
## 269 -0.361114467 -1.0224511 -1.11341755 -1.53188759  0.58045123
## 279 -1.282344313  0.1643722 -0.15953768  2.79541887  0.33532801
## 280 -3.200348344 -0.8649979  0.93019246 -0.21309003  0.26456272
```

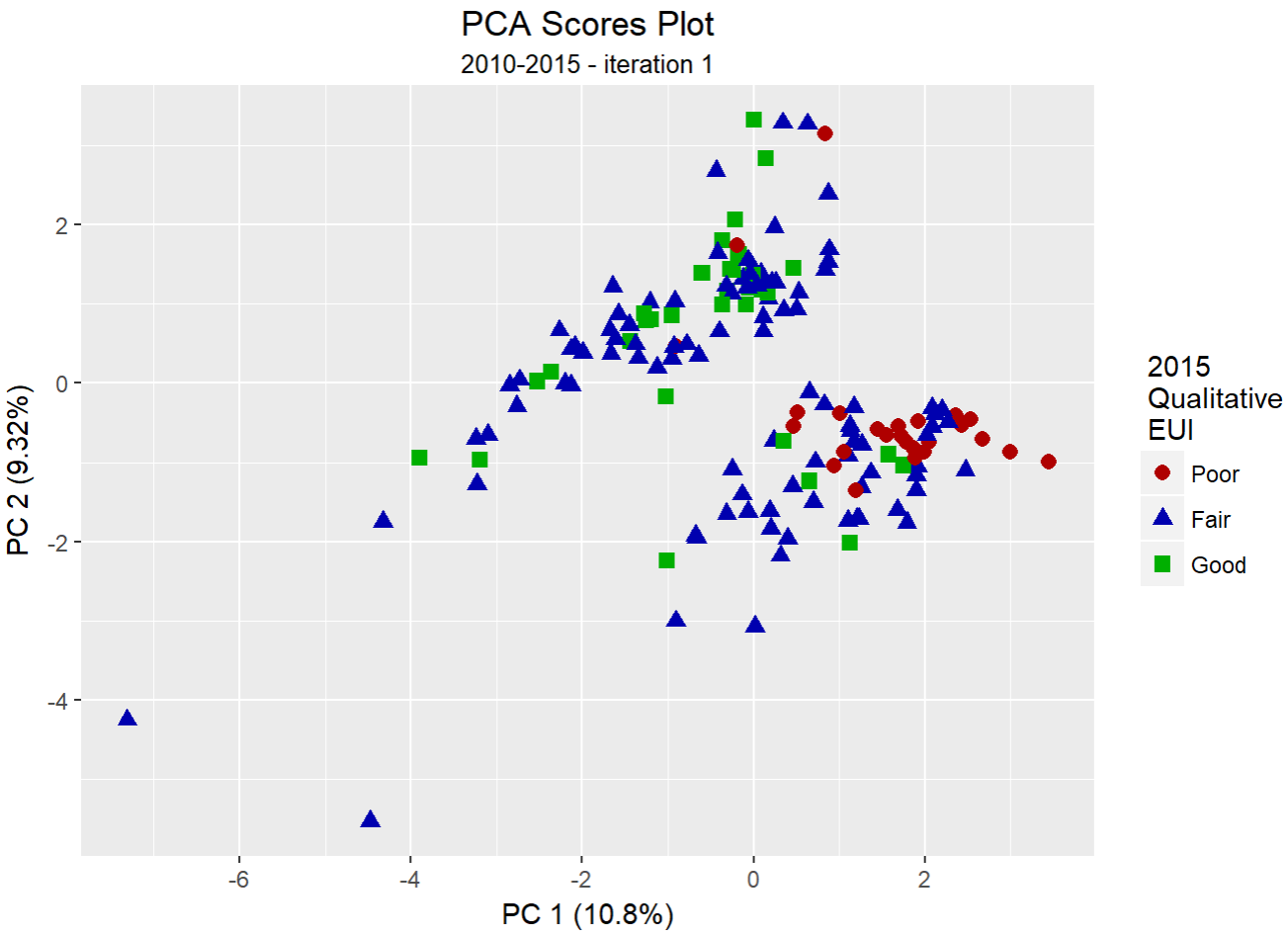
```
## 295 -0.024115731 -1.3214628 -0.19339924 -0.25367874  0.93849938
## 300 -1.440566776  0.6532659 -0.72557951  0.37684140  0.86365358
## 302 -4.098907020 -1.4013716  0.62553987  1.04723847  0.11155837
## 353 -1.014362946  0.7954364 -0.55798842 -0.17513900  0.64578959
## 470  0.635791379  1.4318149  1.20663967  2.24529075  0.18912660
## 477  0.949889391 -0.2342051 -0.60156135 -0.27222684  0.17674365
## 485 -0.071520776 -1.9259435 -0.54017335 -0.49484261  1.58670125
## 486 -0.526927901  1.3807072 -1.40876297  2.40853951 -0.27362220
## 487 -0.504375654  0.9437977 -0.73550745  0.63945455  1.51115102
## 491 -0.161086113  1.2687821 -0.49868342 -0.37265724  1.97409821
## 503  0.299735282  1.7594078  1.56614843 -0.65638702 -1.68556217
## 506  0.206972827  1.8525755  1.48001896 -0.52072509 -2.06199638
## 514  0.677542584  1.6667474  1.98529881 -0.49641370 -1.94734899
## 516  0.189901207  1.1547257  0.45499422 -0.71229654 -0.30653014
## 547  1.117584547 -0.6863237 -1.25977836 -0.23098755  0.06303308
## 565 -1.857067067  1.0478524 -1.33423161 -1.11196587  0.08379281
## 635  0.990077017 -1.0123080 -1.17236972 -1.63844184  0.08982798
## 645  1.117122981 -1.0062121 -0.23053013 -2.43388151  0.21264603
## 647  1.304055565 -0.5379928 -0.86792003 -1.10834878  1.31999806
## 655  1.060164315 -0.7944168 -1.12692871 -0.66616300  0.63461668
## 664  0.277326296  0.9089709  1.09047206  0.60091805 -0.93970484
## 667  0.686232576 -0.1434191 -2.41577901  0.01561975  0.47901588
##             PC6          PC7         PC8           PC9         PC10
## 12   0.13341945  1.391129904 -1.02279401 -0.6751114812 -0.967358401
## 80  -0.67781200  0.131531864 -0.99166708 -0.2840862998 -0.701377445
## 85  -1.50664377 -1.175739001 -0.79682463 -1.9084216495 -0.407156372
## 103 -1.97380368  0.586726683 -0.47104442 -0.3757306397 -0.847221689
## 132 -0.20851728 -0.829398614  0.99581682  0.5257694405  0.075879868
## 144 -0.03638641 -0.157227541 -1.25763182  1.1129864892  0.457271170
## 148  1.29337888 -1.716666539 -1.64480928 -0.6886381476  0.138230609
## 188 -0.17508354  0.174632437  0.27975090 -0.0856110652  0.642182985
## 189  0.22675055 -0.499455755  0.52411457 -0.4708882206  0.463468754
## 196  0.36102315  0.533739494  0.78025147  1.8430817181  0.282207217
## 199 -0.48486186  0.008822934  0.70236918 -1.5523891841 -0.470520673
## 201  0.30689226 -0.921001641 -0.15043547  0.4872691175  0.153219755
## 213 -0.60211394  0.320667447  0.70183022 -0.0000911263  0.064243605
## 253 -0.69543159  0.367160217 -0.03678143  1.6318533367  0.578663538
## 254  1.03416844  1.032032658 -0.17712388  1.4224551094  0.021381263
## 269  1.18697484  0.151080437  1.16334761 -0.2236400541  0.611314559
## 279  2.13511866 -0.996835470  1.00951558 -2.7172968421 -2.706994236
## 280  0.87715750 -1.398600369  0.33637391  0.8930438209  0.718710451
## 295 -0.08339213  0.964238980  0.63981644 -0.1555426332 -0.242854824
## 300  0.08918407  0.574687120  1.14745554  0.1139922241 -0.983646169
## 302 -0.50132488 -0.763844526 -0.43361696 -0.5559914463  0.344424222
## 353 -0.48412353  0.878600753  1.64157427  1.2876017084 -0.276763936
## 470  0.10604265  1.832975358 -0.80327732  1.1403420901 -0.862483839
## 477 -3.32465053 -1.175324258 -0.33619189 -0.4524516575 -0.322836626
## 485 -1.89243264  0.062268585 -1.26710707 -1.9060915038 -0.901689698
## 486 -1.15865811 -0.779407139 -0.22842747 -0.0183743094 -0.544420196
## 487 -0.65781396 -1.249543375 -0.46365662  1.2376485889 -0.735436312
## 491 -1.34098725 -0.721543126 -0.04734553  1.9941949808 -1.085753235
## 503 -1.26852154 -0.531499491  0.60170659  0.5416444927  0.029785964
## 506 -1.61447241 -0.652022429  0.90111929  0.2818236444  0.002779860
## 514 -1.71314135 -0.220223530  0.56368494 -0.4688636923 -0.690174847
```

```
## 516  0.35521023 -0.418370701 -0.48653503  1.7013300565  0.202541013
## 547 -0.03263774 -0.164123830 -0.18849859  0.4862911214  0.577960451
## 565  1.12820067  0.458975307  0.75789768  0.1651166247  0.298538027
## 635  0.65447623  1.209919459  0.47815634 -0.0636777519  0.290056221
## 645  1.15327718 -0.552559949 -0.60365335 -0.2252686081  0.382519778
## 647  0.29544173 -1.245475955 -1.17552464 -0.5107742526 -0.605579908
## 655 -0.45922776  0.691886500  1.04091362  1.6853766132  0.727949913
## 664  0.61141010  0.165959968  0.25623849  1.0832195202  0.428821274
## 667 -0.06654986 -1.585193393  0.12620669 -0.6058610137  0.002099647
##            PC11        PC12        PC13        PC14         PC15
## 12   0.89482154  0.37362648 -1.18381017  0.10483409 -0.711037632
## 80   0.95806399 -0.05642834 -0.63671393 -0.66335180  0.058303974
## 85   0.85821111 -0.56021615  0.33146651  0.46949941 -0.711562002
## 103  0.62800592 -0.71872665  0.16548191  0.18452260  0.253897159
## 132 -0.22261278 -1.99682164 -0.73041126  0.45311843  0.225099543
## 144 -0.40398999  0.85019314  0.05469041 -0.81621218 -0.373789273
## 148 -0.44170204  1.61773029 -0.17742362  0.68361040 -0.121306326
## 188 -0.44894963 -0.06649674 -0.09605039 -0.76605487  0.065594082
## 189 -0.08449848 -0.18223955 -0.71588760 -0.90628075  0.011153094
## 196  0.64582509  1.29401535  0.60231424  0.85440857 -0.333939004
## 199  0.01875727  0.13704275  1.25022144  1.13156518 -0.208569929
## 201  1.73235236  0.49196884  1.04104758 -1.09023412  1.406891241
## 213  0.45913903  0.89608756  0.86488587 -0.17459979 -0.077417274
## 253  1.04630435  0.91500229 -0.16199674 -0.13538482 -0.491763344
## 254  0.66482909  0.71138000 -0.15185783 -0.67133900 -0.029269165
## 269 -0.92207991 -0.02331799  0.89234327  0.73251763 -0.156299388
## 279 -0.13749166  0.14811696 -1.72373585  1.41943096 -0.353031808
## 280  1.49795819  1.16703369  0.68810271 -0.36984035  0.847718581
## 295  0.27847722 -0.79369232  1.31244908  0.60797951  0.248013902
## 300  0.33885008  0.26605392  0.02079383  0.40702707 -0.058712337
## 302 -0.92108661  1.47008416  0.60441495 -0.37565434  1.181224731
## 353  1.03752920  1.38947225  0.17733368  0.71726308 -0.395167348
## 470  0.53554397 -1.23126871  1.20285984 -1.10351412  0.487737552
## 477  0.44121978 -0.67422304  2.22469895  0.29008381 -0.022956057
## 485 -2.43707194  0.73566498  0.76895809  2.01802659 -0.381165071
## 486  0.64553297  0.19016482  0.76153427  0.20394814 -0.348587317
## 487  0.34401287  0.07472359 -0.15036642 -0.32954040  0.296452285
## 491  0.32427840 -0.75944465 -0.50322701 -0.26154842  0.003877933
## 503  0.49603960  0.65964697  1.09860225 -0.49186020 -0.013786188
## 506  0.56205883  0.54031839  1.21048395 -0.42086403 -0.037455488
## 514  0.83195318  1.32220758  0.90574257 -0.12933097 -0.048744668
## 516  0.71783628  0.61566316 -0.37137905  0.86703189  0.457869838
## 547 -0.23793850 -0.95629820  0.12729890 -0.11948454  0.144243623
## 565 -0.74075089  0.51897148  0.74089568  0.44166264  0.048744836
## 635 -0.46750900 -0.29723254 -0.78560214  0.01903402 -0.440866676
## 645 -0.83636904  0.19422721 -0.70055776  0.45919695 -0.450018511
## 647 -1.37462953 -1.23307842 -0.27222382  0.08432783 -0.288583277
## 655  0.78856925  0.77195055 -0.68329887 -0.21788615 -0.131924752
## 664  0.61452247  1.20138861  0.04144750 -1.13685789 -0.328419608
## 667 -1.57723033 -1.51838838  0.15547530 -0.16804819 -0.352087244
##            PC16
## 12  -0.29580376
## 80   0.17442049
## 85  -0.18930918
```

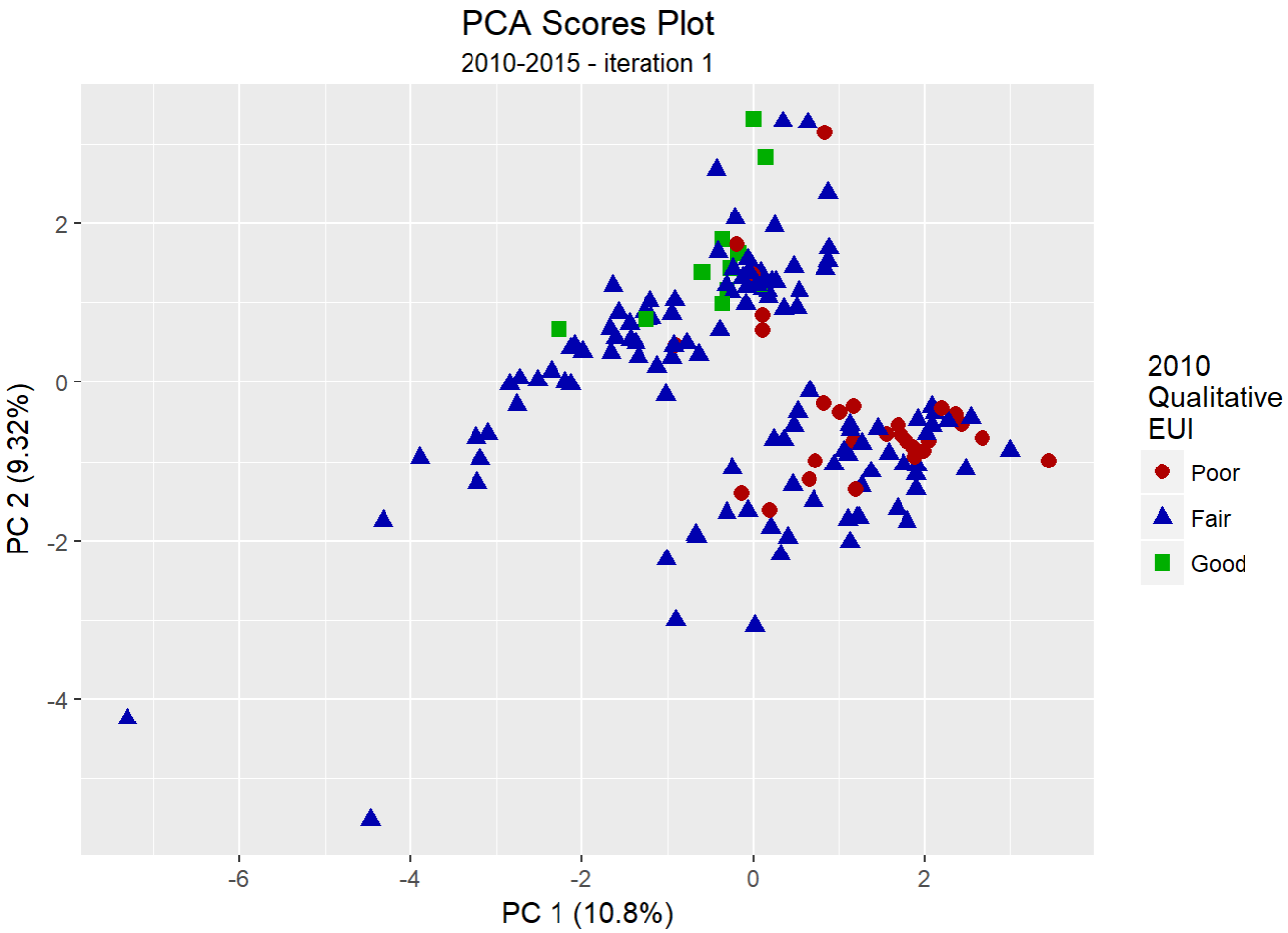
```
## 103 -0.05262305
## 132  0.38317829
## 144  0.10370895
## 148 -0.48081156
## 188 -0.08540747
## 189  0.04102166
## 196  0.11928799
## 199 -0.13712290
## 201 -0.01665398
## 213 -0.08705848
## 253  0.21949042
## 254  0.32128357
## 269  0.29721546
## 279  0.14632535
## 280  0.14741061
## 295  0.23538076
## 300  0.02700372
## 302 -0.05970263
## 353  0.20603600
## 470 -0.13406871
## 477  0.07674947
## 485 -0.97254873
## 486 -0.13810800
## 487 -0.02677059
## 491  0.08268301
## 503  0.09634207
## 506  0.06236467
## 514  0.17828466
## 516  0.19977008
## 547  0.31604422
## 565 -0.10342970
## 635  0.55800393
## 645 -0.06391106
## 647 -0.36072004
## 655 -0.21650757
## 664 -0.27170543
## 667  0.14747579
```

```
#Graph Pca 1 vs 2
i1.pca.PC1=i1.pca$x[,1]
i1.pca.PC2=i1.pca$x[,2]

Dep_Var=i1.data.train$x.2015.Qualitative.EUI

plot.i1.pca.1.2.2015.EUI.v1 <- ggplot(i1.data.train) + geom_point(aes(i1.pca.PC1, i1.pca.PC2,
colour = Dep_Var, shape = Dep_Var), size = 2.5) +
  labs(x = paste("PC 1 (", percent(i1.pca$sdev[1]/sum(i1.pca$sdev)), ")", sep=""),
       y = paste("PC 2 (", percent(i1.pca$sdev[2]/sum(i1.pca$sdev)), ")", sep=""),
       title="PCA Scores Plot", subtitle = "2010-2015 - iteration 1",
       colour = " 2015 \n Qualitative \n EUI", shape = " 2015 \n Qualitative \n EUI")+
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  theme( plot.title = element_text(hjust = 0.5),
```

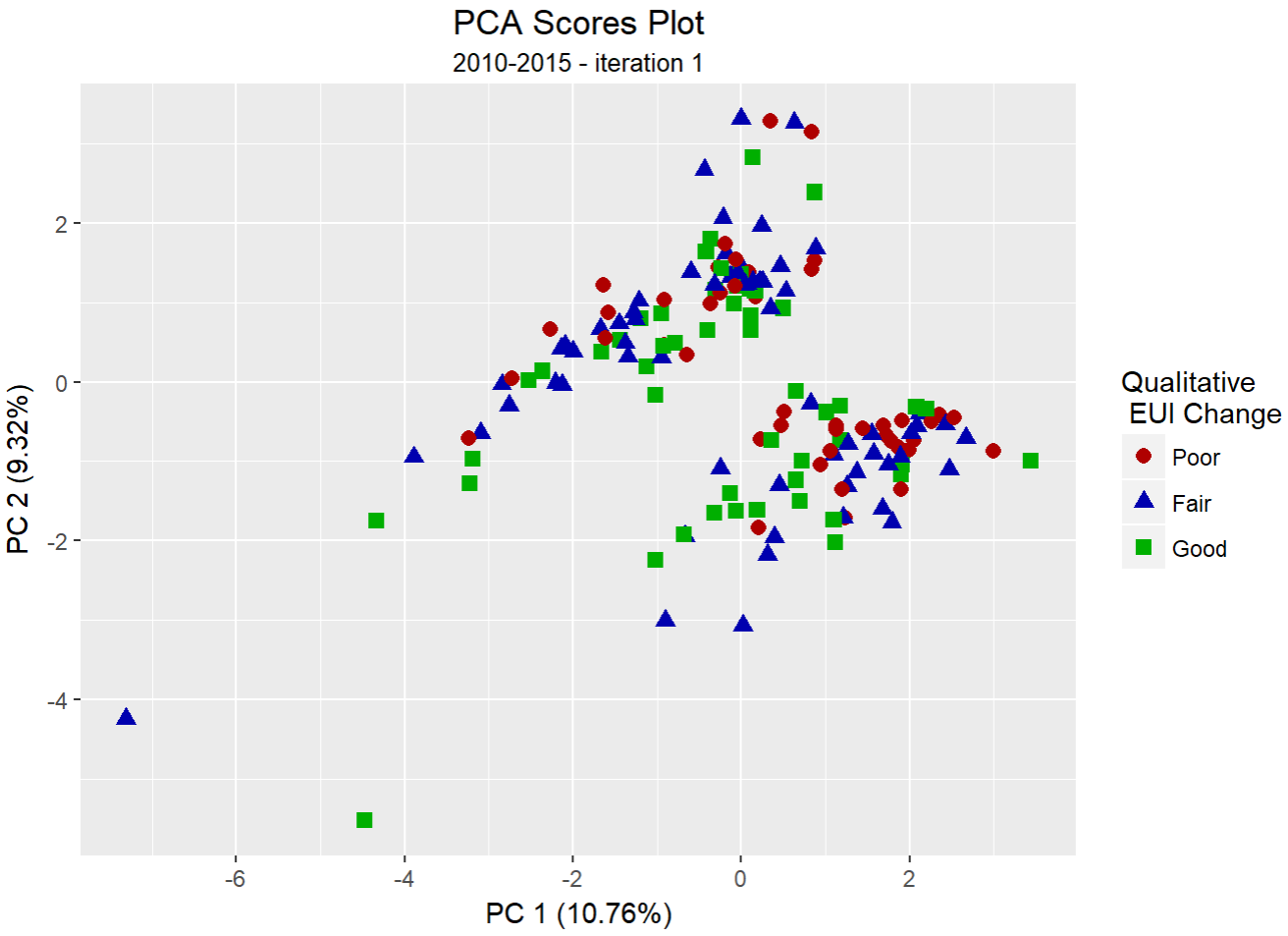
```
        plot.subtitle = element_text(hjust = 0.5))
plot.i1.pca.1.2.2015.EUI.v1
```



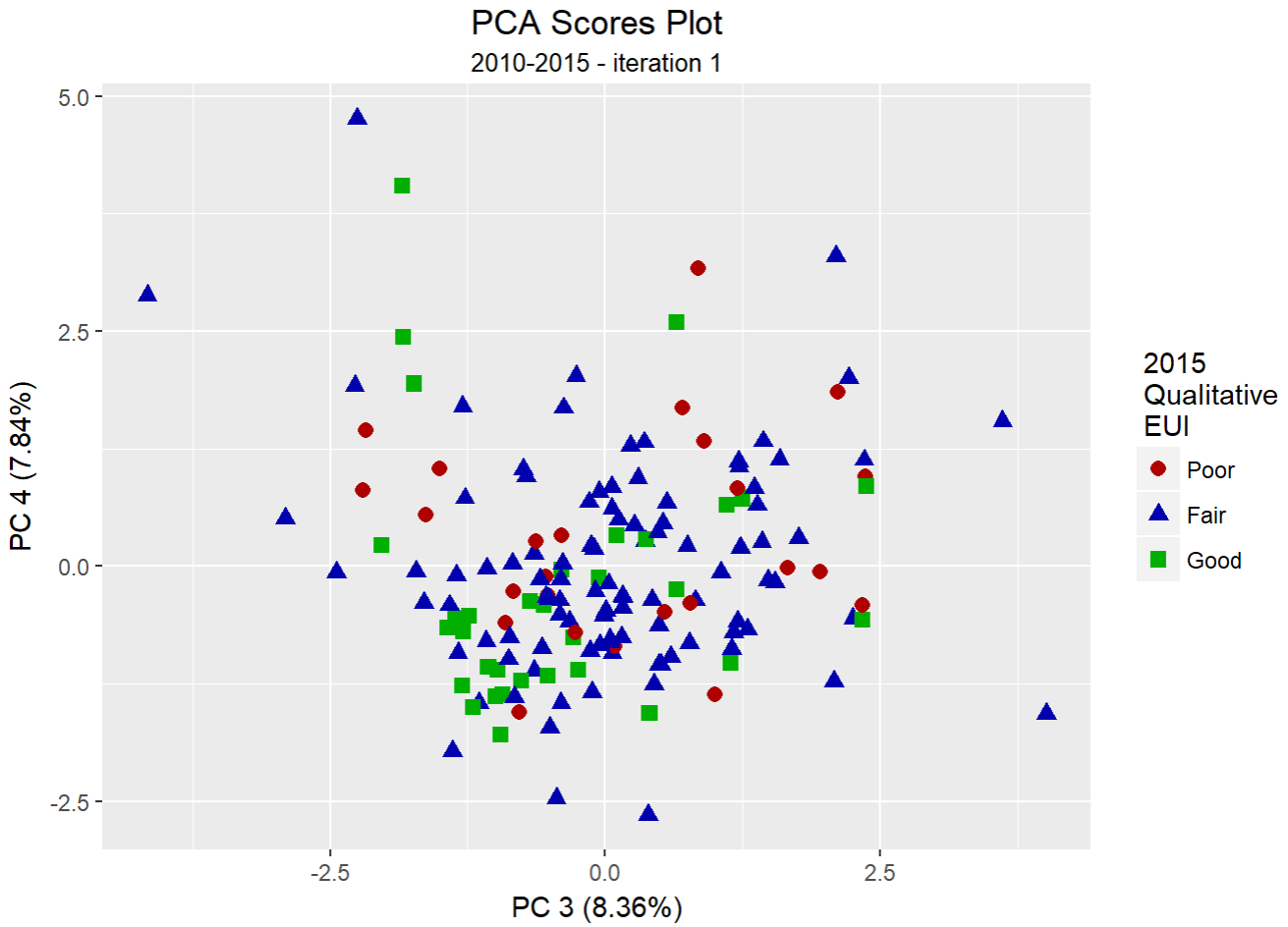

```
Dep_Var=i1.data.train$x.2010.Qualitative.EUI

plot.i1.pca.1.2.2010.EUI.v1 <- ggplot(i1.data.train) + geom_point(aes(i1.pca.PC1, i1.pca.PC2,
colour = Dep_Var, shape = Dep_Var), size = 2.5) +
  labs(x = paste("PC 1 (", percent(i1.pca$sdev[1]/sum(i1.pca$sdev)), ")", sep=""),
       y = paste("PC 2 (", percent(i1.pca$sdev[2]/sum(i1.pca$sdev)), ")", sep=""),
       title="PCA Scores Plot", subtitle = "2010-2015 - iteration 1",
       colour = " 2010 \n Qualitative \n EUI", shape = " 2010 \n Qualitative \n EUI")+
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  theme( plot.title = element_text(hjust = 0.5),
         plot.subtitle = element_text(hjust = 0.5))
plot.i1.pca.1.2.2010.EUI.v1
```



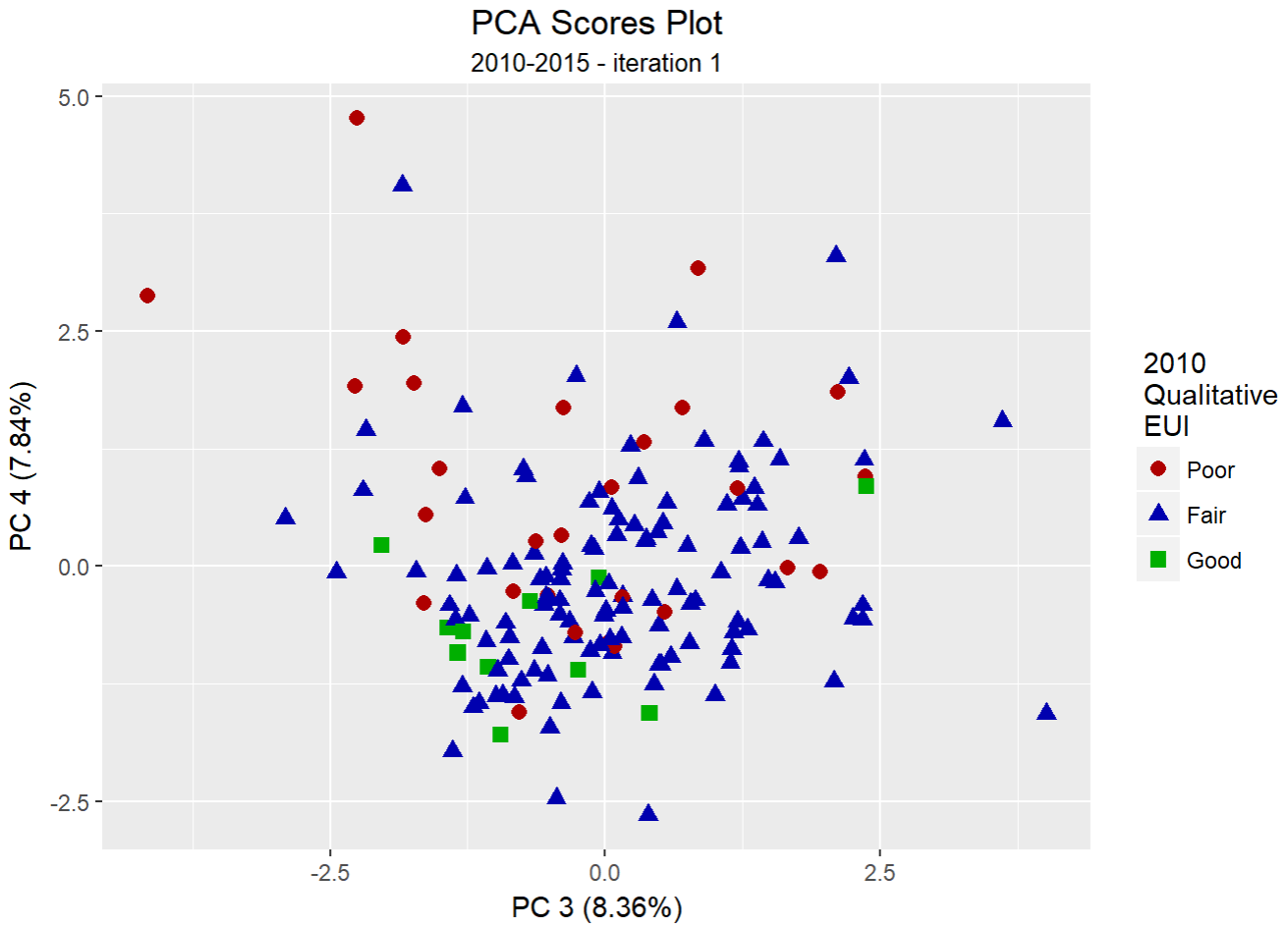

```
Dep_Var=i1.data.train$D.Qualitative.EUI

plot.i1.pca.1.2.EUI.Change.v1 <- ggplot(i1.data.train) + geom_point(aes(i1.pca.PC1, i1.pca.PC2
, colour = Dep_Var, shape = Dep_Var), size = 2.5) +
  labs(x = paste("PC 1 (", percent(i1.pca$sdev[]/sum(i1.pca$sdev)), ")", sep=""),
       y = paste("PC 2 (", percent(i1.pca$sdev[2]/sum(i1.pca$sdev)), ")", sep=""),
       title="PCA Scores Plot", subtitle = "2010-2015 - iteration 1",
       colour = "Qualitative \n EUI Change", shape = "Qualitative \n EUI Change")+
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  theme( plot.title = element_text(hjust = 0.5),
         plot.subtitle = element_text(hjust = 0.5))
plot.i1.pca.1.2.EUI.Change.v1
```



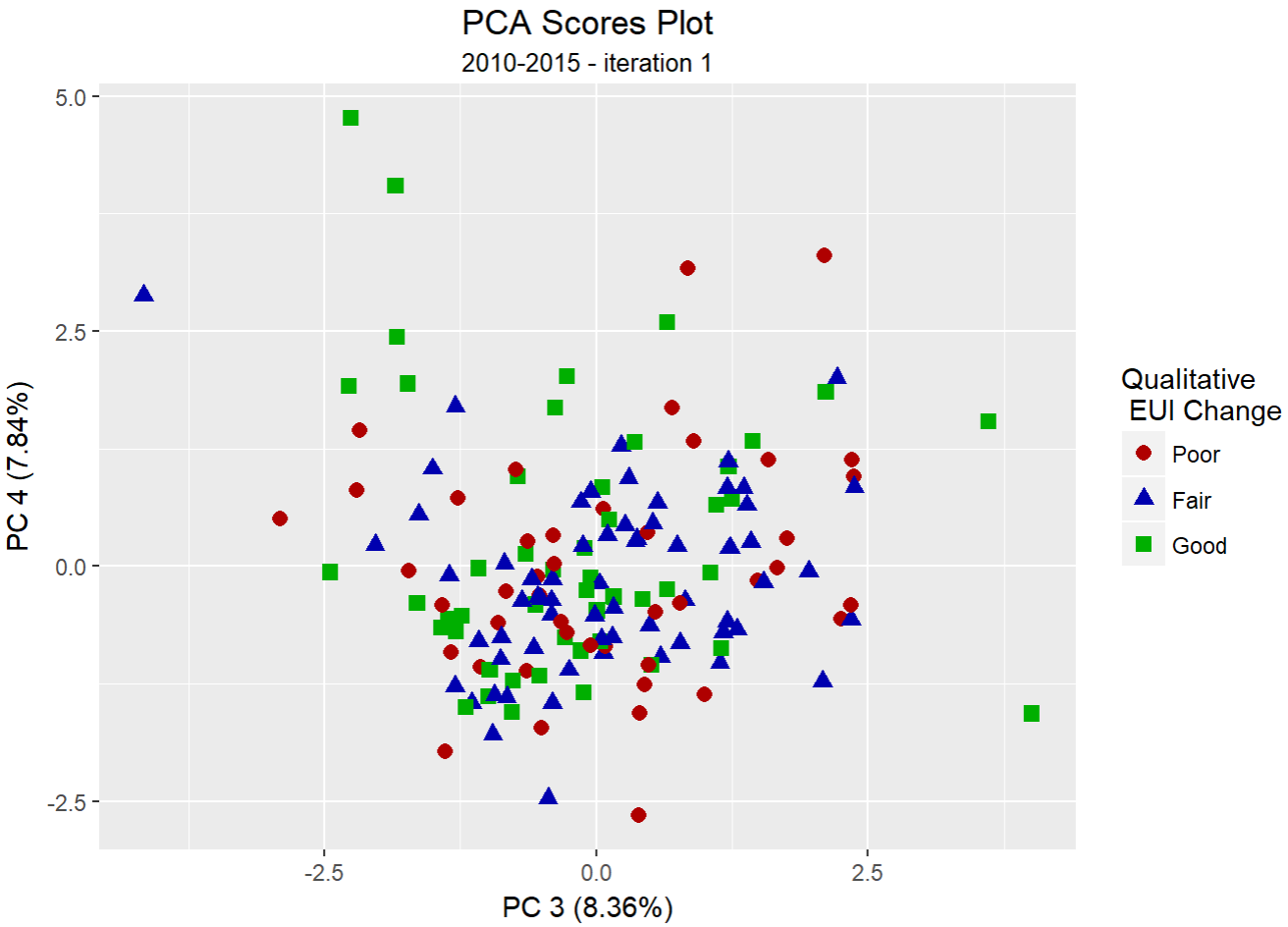

```
#Graph Pca 3 vs 4
i1.pca.PC3=i1.pca$x[,3]
i1.pca.PC4=i1.pca$x[,4]

Dep_Var=i1.data.train$x.2015.Qualitative.EUI

plot.i1.pca.3.4.2015.EUI.v1 <- ggplot(i1.data.train) + geom_point(aes(i1.pca.PC3, i1.pca.PC4,
colour = Dep_Var, shape = Dep_Var), size = 2.5) +
  labs(x = paste("PC 3 (", percent(i1.pca$sdev[3]/sum(i1.pca$sdev)), ")", sep=""),
       y = paste("PC 4 (", percent(i1.pca$sdev[4]/sum(i1.pca$sdev)), ")", sep=""),
       title="PCA Scores Plot", subtitle = "2010-2015 - iteration 1",
       colour = " 2015 \n Qualitative \n EUI", shape = " 2015 \n Qualitative \n EUI")+
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  theme( plot.title = element_text(hjust = 0.5),
         plot.subtitle = element_text(hjust = 0.5))
plot.i1.pca.3.4.2015.EUI.v1
```




```
Dep_Var=i1.data.train$x.2010.Qualitative.EUI

plot.i1.pca.3.4.2010.EUI.v1 <- ggplot(i1.data.train) + geom_point(aes(i1.pca.PC3, i1.pca.PC4,
colour = Dep_Var, shape = Dep_Var), size = 2.5) +
  labs(x = paste("PC 3 (", percent(i1.pca$sdev[3]/sum(i1.pca$sdev)), ")", sep=""),
       y = paste("PC 4 (", percent(i1.pca$sdev[4]/sum(i1.pca$sdev)), ")", sep=""),
       title="PCA Scores Plot", subtitle = "2010-2015 - iteration 1",
       colour = " 2010 \n Qualitative \n EUI", shape = " 2010 \n Qualitative \n EUI")+
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  theme( plot.title = element_text(hjust = 0.5),
         plot.subtitle = element_text(hjust = 0.5))
plot.i1.pca.3.4.2010.EUI.v1
```




```
Dep_Var=i1.data.train$D.Qualitative.EUI
plot.i1.pca.3.4.EUI.Change.v1 <- ggplot(i1.data.train) + geom_point(aes(i1.pca.PC3, i1.pca.PC4
, colour = Dep_Var, shape = Dep_Var), size = 2.5) +
  labs(x = paste("PC 3 (", percent(i1.pca$sdev[3]/sum(i1.pca$sdev)), ")", sep=""),
       y = paste("PC 4 (", percent(i1.pca$sdev[4]/sum(i1.pca$sdev)), ")", sep=""),
       title="PCA Scores Plot", subtitle = "2010-2015 - iteration 1",
       colour = "Qualitative \n EUI Change", shape = "Qualitative \n EUI Change")+
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  theme( plot.title = element_text(hjust = 0.5),
         plot.subtitle = element_text(hjust = 0.5))
plot.i1.pca.3.4.EUI.Change.v1
```



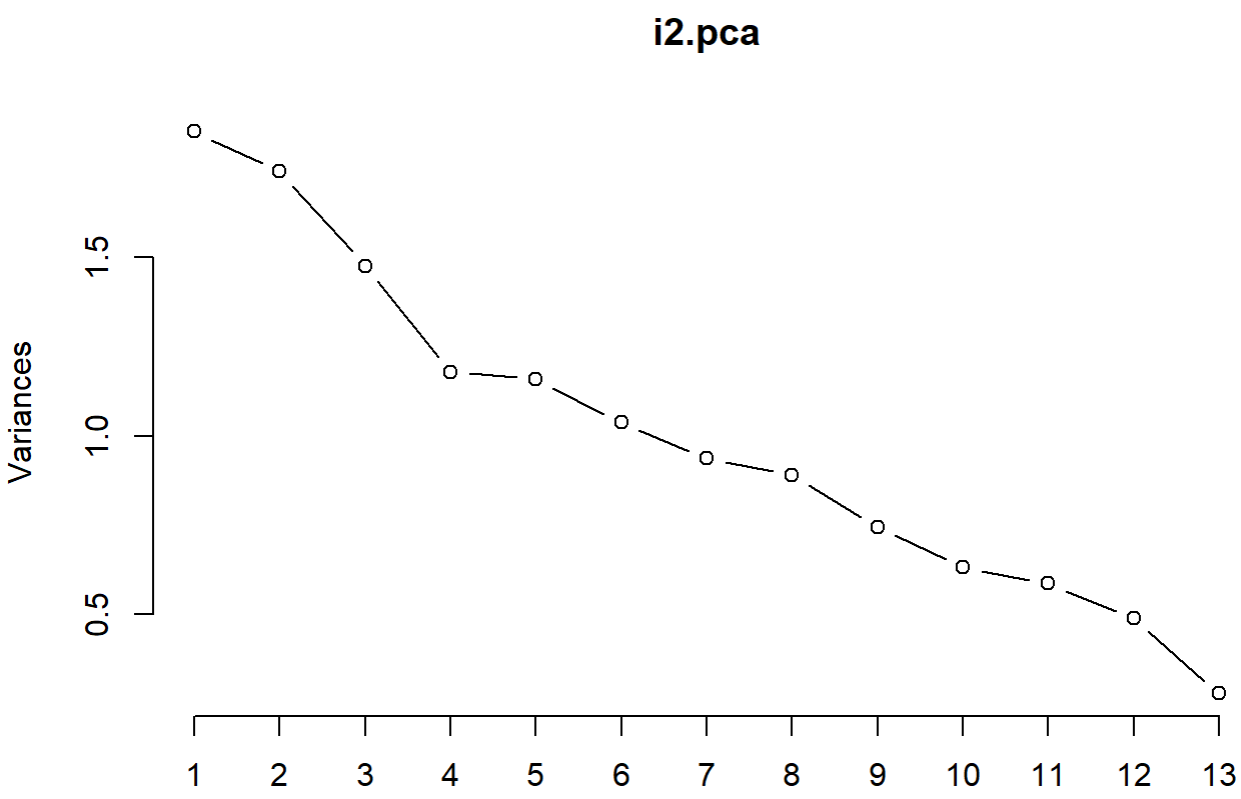

```
#### Run 2 ####

# Excludes Climate Zone, Total Energy. Exterior Area,Net Area,.2010EUI and delta or 2015 Value
s)
i2.data<-data[,c("x.2010.Qualitative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI",
              "Portfolio.Manager.no.","Exterior.Area", "x.2010.Actual.Energy.Use",
                "Construction.Year",    "No.of.Structures", "Building.Class", "Climate.Zone",
              "Electrically.Heated",    "Cooling.Tower",    "Soft.Landscaping", "Occupant.Dens
ity",
              "Vacancy.Rate",   "Weekly.Operating.Hours")]
#create Train and Test set
train = sample(1:nrow(i2.data), nrow(i2.data)*.80)
i2.data.train = i2.data[train,]
i2.data.test = i2.data[-train,]

#Pca
#sapply(i2.data.train, class)

i2.pca <- prcomp(i2.data.train[,-c(1,2,3)], retx=TRUE, scale = TRUE, center = TRUE)
expl.var <- round(i2.pca$sdev^2/sum(i2.pca$sdev^2)*100)
page(i2.pca, method="print")
i2.pca
```

```
## Standard deviations (1, .., p=13):
##  [1] 1.3612462 1.3193417 1.2151682 1.0861342 1.0762062 1.0186390 0.9679566
```

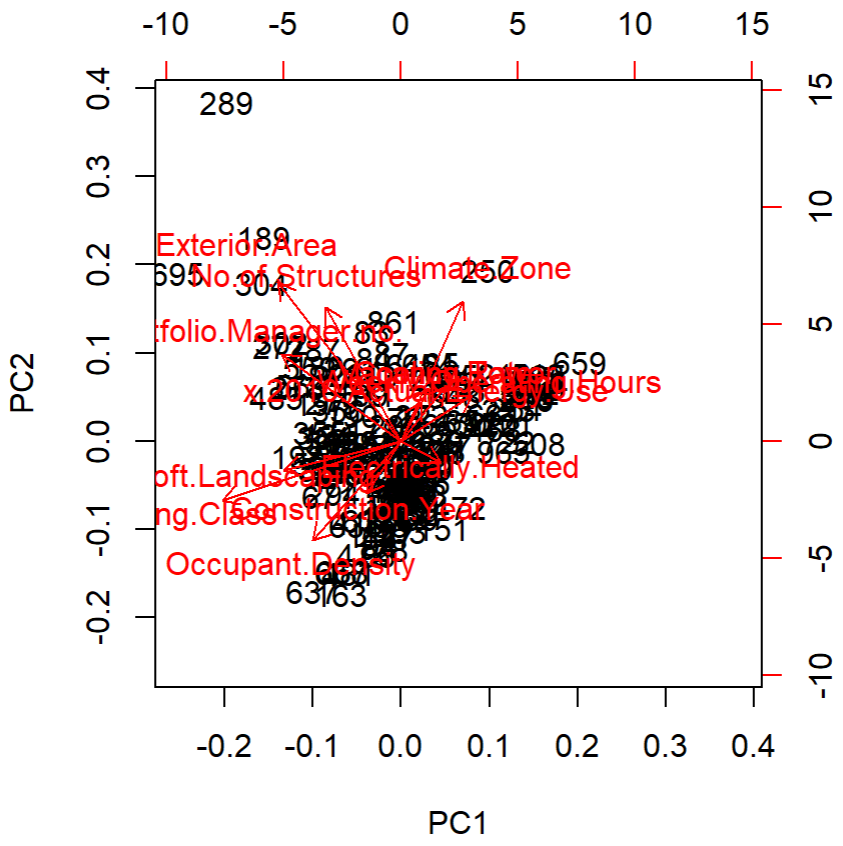
```
##  [8] 0.9430338 0.8631288 0.7947735 0.7650788 0.6987144 0.5269946
##
## Rotation (n x k) = (13 x 13):
##                                  PC1         PC2          PC3         PC4
## Portfolio.Manager.no.    -0.37016671  0.28085423 -0.070106561  0.19866253
## Exterior.Area            -0.38345623  0.50958090 -0.064468244  0.04153295
## x.2010.Actual.Energy.Use  0.06294606  0.12514278 -0.538580589  0.11101885
## Construction.Year        -0.10682122 -0.16884319  0.412692859  0.13506977
## No.of.Structures         -0.23578777  0.42879154 -0.054139884 -0.39779957
## Building.Class           -0.55631142 -0.18947691  0.123777858 -0.01513749
## Climate.Zone              0.19117442  0.44921314  0.004142822  0.32274441
## Electrically.Heated       0.12415082 -0.06875181 -0.173498332  0.16984839
## Cooling.Tower             0.13486107  0.17280405  0.423439615  0.30238699
## Soft.Landscaping         -0.36590481 -0.09221765  0.190269428  0.39767171
## Occupant.Density         -0.27360094 -0.31832934 -0.420562929 -0.02618501
## Vacancy.Rate              0.08600676  0.17091861  0.281758992 -0.56405736
## Weekly.Operating.Hours    0.22180423  0.14860837 -0.102022938  0.26139257
##                                  PC5         PC6         PC7          PC8
## Portfolio.Manager.no.     0.31382493 -0.24472095  0.40425629 -0.123659425
## Exterior.Area            -0.04774402  0.04745507 -0.17821032  0.099957023
## x.2010.Actual.Energy.Use -0.20642208  0.07598085  0.32296127 -0.451277469
## Construction.Year         0.41501910  0.42800596 -0.04760155 -0.375032404
## No.of.Structures          0.10675044  0.13886877 -0.07188516  0.227287131
## Building.Class           -0.06595523  0.12077953 -0.28700250 -0.007462445
## Climate.Zone              0.10144724  0.03012645 -0.40227663 -0.234123237
## Electrically.Heated       0.63257221 -0.05957944  0.17512079  0.519440054
## Cooling.Tower            -0.34648164 -0.37283895 -0.02208778  0.273347328
## Soft.Landscaping         -0.26350795  0.13538703  0.45479445  0.113695501
## Occupant.Density         -0.16214927  0.01835049 -0.20951913  0.225553995
## Vacancy.Rate             -0.10525940  0.16371451  0.40525464  0.020990777
## Weekly.Operating.Hours   -0.17703310  0.72532078  0.05254434  0.339276950
##                                  PC9        PC10        PC11        PC12
## Portfolio.Manager.no.    -0.13021031  0.42004658 -0.26954102  0.18892015
## Exterior.Area             0.04797750  0.29153858  0.32035982 -0.22580770
## x.2010.Actual.Energy.Use  0.33567093 -0.18101943  0.05274135 -0.40145957
## Construction.Year         0.12094192  0.02970641 -0.32122169 -0.34244708
## No.of.Structures         -0.20770196 -0.51811959 -0.33839462 -0.21117234
## Building.Class            0.33931043 -0.01319977  0.36307034  0.00371036
## Climate.Zone              0.31755392 -0.23791567 -0.11024156  0.50967349
## Electrically.Heated       0.37265638 -0.16117288  0.20162572 -0.10038123
## Cooling.Tower             0.27728184  0.05213167 -0.27169491 -0.41097826
## Soft.Landscaping         -0.06445427 -0.46720119  0.03260020  0.25824247
## Occupant.Density          0.30477273  0.14386691 -0.58608914  0.13044765
## Vacancy.Rate              0.51623963  0.15204332 -0.02013853  0.26016678
## Weekly.Operating.Hours   -0.12258555  0.29816674 -0.02877726  0.01155988
##                                 PC13
## Portfolio.Manager.no.    -0.31763110
## Exterior.Area             0.54791383
## x.2010.Actual.Energy.Use -0.11067924
## Construction.Year         0.19290931
## No.of.Structures         -0.20289535
## Building.Class           -0.53903845
## Climate.Zone             -0.01657793
## Electrically.Heated       0.03388575
```

```
## Cooling.Tower            -0.15691726
## Soft.Landscaping          0.25602798
## Occupant.Density          0.22527084
## Vacancy.Rate              0.07465972
## Weekly.Operating.Hours   -0.26417909
```

```
#Scree Plot
i2.pca_screeplot <-plot(i2.pca, npcs = 13, type = "lines")
```




```
i2.pca_biplot <-biplot(i2.pca)
```




```
#predict
pred_i2.pca = predict(object = i2.pca, newdata = i2.data.test[,-c(1,2,3)])
page(pred_i2.pca, method="print")
pred_i2.pca
```

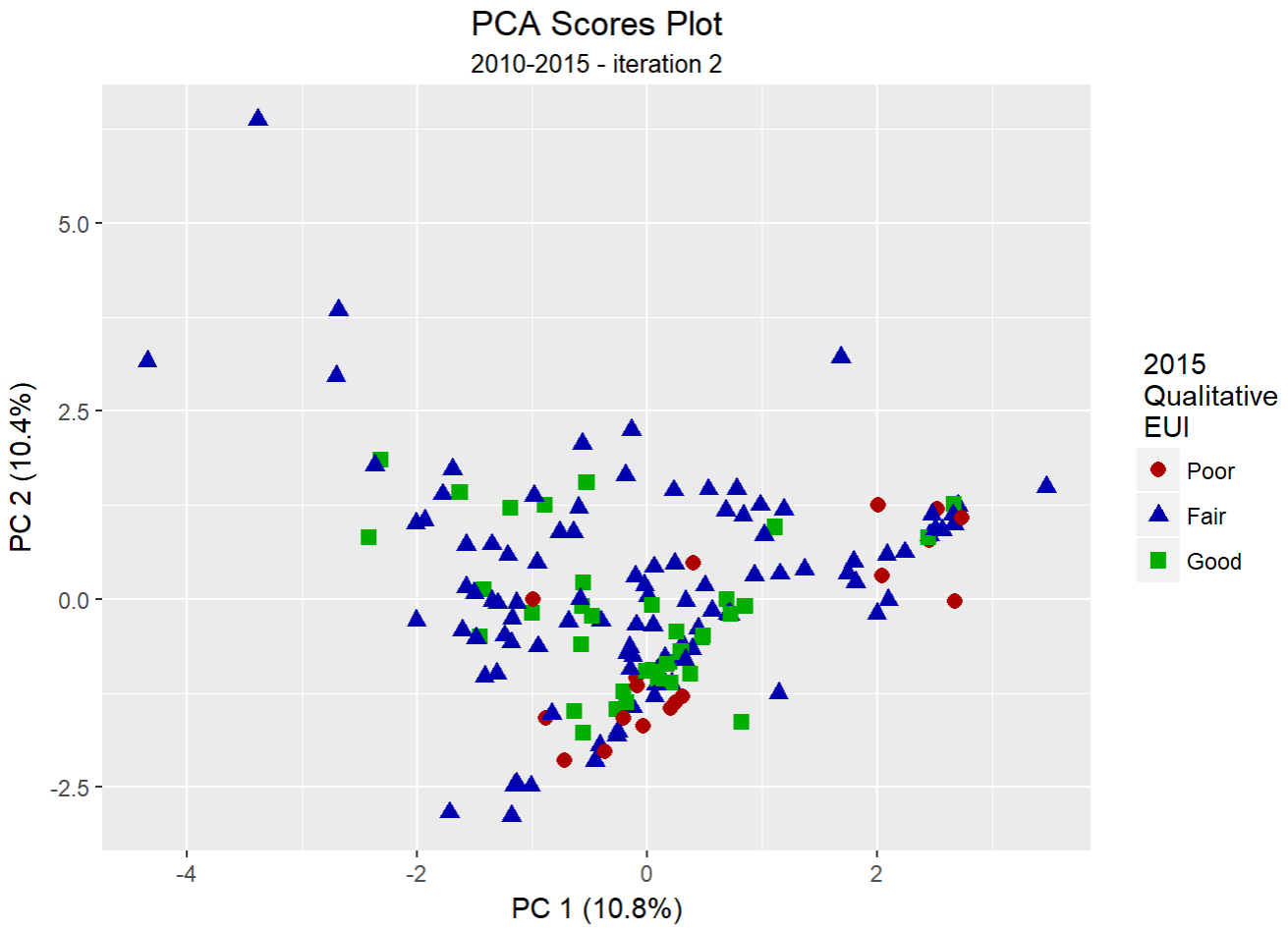
```
##                PC1         PC2         PC3         PC4         PC5
## 81  -0.3381328417 -1.50859706  0.12720667 -0.17955953 -0.27704782
## 86   0.8827593580  0.03470834 -0.22950802 -0.59373632 -1.31360978
## 88   0.3479166452 -0.03104976  0.20020843 -0.13982493 -0.35795656
## 92  -1.2724172308 -0.48994608 -0.59687738 -2.01093206 -0.58313869
## 98   0.7329611169 -0.07240532  1.30562043 -1.80981813 -0.57831771
## 102  0.6634925331 -0.32500164  0.87305811 -1.00855128 -0.43710647
## 124  0.1630979190 -1.48415716 -0.72503920 -0.22942031 -0.77620666
## 126  0.2813683808 -1.13021087  0.54763209 -2.27373172 -1.01185258
## 129 -0.4373528218 -2.11012138 -1.70666519 -0.26598335 -1.15604822
## 130 -0.6116637103 -2.43644924 -1.10932149 -0.36868302 -0.85593654
## 181 -1.1202692759  0.45564023 -0.71030646  0.81075861  0.13034606
## 188 -2.5646109961  3.15595443 -0.42141310 -0.05878795  0.80261472
## 195 -1.6327631668 -0.12152041  0.71329672  1.63380670 -0.15074714
## 196 -0.5145380986 -0.22770875  0.31673997  0.50216514  0.63962737
## 211  2.1293659007  0.94686462 -1.52732352  0.95153563 -0.53506496
## 213  2.6421981830  1.22141867  0.57793618 -0.42566712 -0.23191824
## 214  2.7189634557  1.19915698 -0.32231296  0.47335737 -0.39143695
## 248  0.9984915696 -0.12509325  0.72769816  1.47088338 -0.63608183
## 251 -0.9235199303 -0.52811480  0.21847449  1.38412218 -1.30379676
```

```
## 269 -1.7258693892 -1.15028148  0.97377745  1.13673320 -0.22369519
## 270 -1.0311178071  1.43912618 -0.25303996 -0.25169900  0.59106900
## 271  1.8847817135 -0.08054610 -1.16942544  0.54605429  5.93123326
## 306 -1.4631851826  2.20350545  0.07075295 -0.52804472  0.64072091
## 406 -2.1115492669  0.67801966  0.54821075  0.23762614  1.05384033
## 440  0.1353463862 -0.91875822  1.59355815 -2.21944512 -0.39846038
## 507  2.1896391072  0.95571545 -0.18995522  0.47520049 -0.30272027
## 509  2.4964873064  1.25098799 -1.21649258  0.31655325 -0.78071076
## 516  1.0290752679 -0.16863974  0.82328473  0.48267968 -0.01573541
## 517  2.0165327259  0.02961478 -0.59446024 -1.00812544  0.87238866
## 543  0.0455671630 -0.52566857 -1.77935242 -0.68820354  0.51003415
## 562 -0.1621158940  0.52134163 -0.38119732  1.68909271 -0.63761740
## 570  0.0009749734  0.37831000  1.35289469  0.44878594  0.93645569
## 582 -0.7170776145 -0.88537047  0.32137750 -0.16778109 -0.28761220
## 636  1.6350915023 -0.66017654  0.61558962 -1.00378741 -0.29515367
## 644  0.8144006021  0.65999991  0.39575567  0.67557023 -0.31673580
## 645  0.0678455523 -0.04598545  1.65697109  2.09659948 -0.31156050
## 649 -0.0180015774  0.12686088  1.05075316  1.41414346 -0.72444137
## 662  2.1662981212  0.45770239  0.11651967 -0.18563210 -0.42976144
## 673  0.0081856618 -0.85736763  0.51305943 -1.07957103 -0.68005542
## 682  2.3938217786  0.12963295  0.56274872  0.26271087  2.79066931
##             PC6          PC7         PC8         PC9         PC10
## 81  -0.07583858 -1.420343086  0.80475013  0.08389644 -0.111173378
## 86   0.77210210  0.221237781  0.78408321  0.44094921  0.002900863
## 88   0.17668571 -0.692041683 -0.40520598  0.88463596 -0.628187193
## 92  -0.25611558 -0.210357310  1.22620290 -0.77067820 -1.306445213
## 98   1.17711498  0.975808991 -0.19406335  1.81177115  0.048738592
## 102  0.10380372  0.229855958 -0.14906984  0.58398993 -0.483184260
## 124 -0.19263522  0.640807030 -0.43467602 -0.58509455 -0.631035799
## 126  0.34103434  1.677260454  0.09102831  1.05011370  0.112530066
## 129 -0.13897911  0.229625494 -0.01365032  0.11824408 -0.331152943
## 130 -0.10619773 -0.218818478  0.47738723 -0.13656574 -0.080284921
## 181 -0.62410638 -0.008169052 -0.14170418 -0.41070357  0.739258453
## 188 -0.42308506 -0.985821882  0.03326536 -0.43207314 -0.349613868
## 195 -1.17324404  0.336349824  0.08060007 -0.01002422 -0.019002433
## 196 -1.55962130 -0.939902294 -0.13900100  0.11412353  1.183183326
## 211  0.47456253  1.067831536 -1.51526615  0.15148038 -0.873360698
## 213 -0.48294877  0.219590994 -0.34110383  0.70781935  0.391987901
## 214 -0.25227739  0.082737429 -0.63929965  0.36129219  0.174851823
## 248 -0.21350950  0.635000593  0.07576322 -0.58565298 -0.926921771
## 251 -0.03525910 -0.560728208  0.91278697  0.46421567 -0.429198799
## 269 -0.77286428  1.241792088  0.54193239 -0.79279661  0.475747560
## 270 -0.36628715 -0.849084424  0.45825823 -0.21915736  0.144513818
## 271 -1.07727063  1.242955156  2.75728469  0.16483040  0.100037525
## 306 -0.74137274 -0.981443591  0.54044614 -0.24019383  0.388128923
## 406  0.45539551 -1.180321494 -0.85134507  0.64431889  1.715769273
## 440  0.66504740  0.151329147  0.66854698  1.84712787  0.356514248
## 507 -0.33794500 -0.024123889 -0.18683971 -0.40715101  0.607203169
## 509  0.16369894  1.236594121 -1.55821369  0.98788682  0.187728266
## 516 -0.08913384 -1.095814801  0.14449109 -0.14257977  0.538277406
## 517  0.97780741  0.283917042 -0.79066664 -0.78076852  0.583909700
## 543  0.51088657 -0.312481110 -1.67403906  0.61974648  0.213086786
## 562 -1.51680564  1.144634354  0.55181051 -0.77815547  0.100266494
## 570 -1.13927840 -0.587727477 -0.19520030 -0.38413120  1.249172196
```

```
## 582  1.22539901 -0.081035315  0.30944414 -0.45170546  0.679998192
## 636 -1.11313009  0.262781147  0.84837450 -0.96643965  1.005946321
## 644 -0.03846999 -1.836987388  0.26642333  0.48603792  0.562624336
## 645  0.85913991 -0.503645078 -0.19753462  0.17246649 -1.008442244
## 649 -1.27147909 -0.276587019 -0.50897308  0.23198039 -1.796066271
## 662 -0.80290627 -0.350462073  0.42105928 -0.25888579  0.575250621
## 673 -0.82682446 -0.490197606  0.95051524  0.11030472  1.187791137
## 682  0.14284532  3.651154267  3.39527830  3.94420519 -1.448672165
##            PC11        PC12        PC13
## 81  -0.66578713  0.96366986  0.41105257
## 86   0.91647026  0.98892982 -0.73078319
## 88   0.35202176  0.83178491  0.09609900
## 92  -0.37622331 -1.10392718 -0.07850194
## 98   0.36067751  0.94222595 -0.05221601
## 102  0.84213349  0.76664961 -0.31153567
## 124  1.06562829 -1.45551073 -0.61332739
## 126  0.69779580 -0.18891849 -0.14878937
## 129 -0.15521921 -1.25213871 -0.11190225
## 130 -0.52004036 -0.82465200  0.11602252
## 181  0.14642758  0.39135357 -0.39283713
## 188 -0.04375456 -0.27573992  0.60078029
## 195 -0.85049118  0.21440137 -0.03432819
## 196 -1.23916979 -0.26359377 -0.92743796
## 211 -0.23794187 -0.26191416  0.15631656
## 213 -0.82737206  0.14155205  0.16136383
## 214 -0.77500037 -0.54157952 -0.17604687
## 248 -0.97715283 -0.44188497  1.30824641
## 251 -0.52394261 -0.08845599  0.65861971
## 269 -0.74598508 -0.80958131 -0.33419338
## 270 -1.90871724 -1.21289635 -1.51504275
## 271  1.04632892  0.66381493  0.42811454
## 306 -1.16245328 -1.29489420 -0.79361276
## 406  1.17997664 -0.59004839 -1.41395205
## 440 -0.66961101  1.82796055  0.55683067
## 507 -0.43541519 -1.21594118  0.23651929
## 509 -0.51199706 -2.14328222 -0.06614966
## 516 -0.03543564 -0.72774382 -1.27386656
## 517 -0.26823755  0.53097953  0.57486954
## 543  0.17673948 -0.19074277 -0.38545048
## 562 -1.54075209  0.45797293  0.92853670
## 570 -0.35475865 -0.27646566 -1.28157583
## 582  0.74778798 -1.60380707  0.79186679
## 636 -0.66613519 -1.33344608  0.74360659
## 644  0.06557640 -0.33928209 -0.54024443
## 645 -0.19188285 -0.14150654 -0.10587161
## 649  0.67460383  0.20261445 -0.09878752
## 662 -0.75008412 -0.44498640  0.48877020
## 673  0.18767217 -1.41815435 -0.13434847
## 682  0.27423830 -0.01025741  1.49346742
```

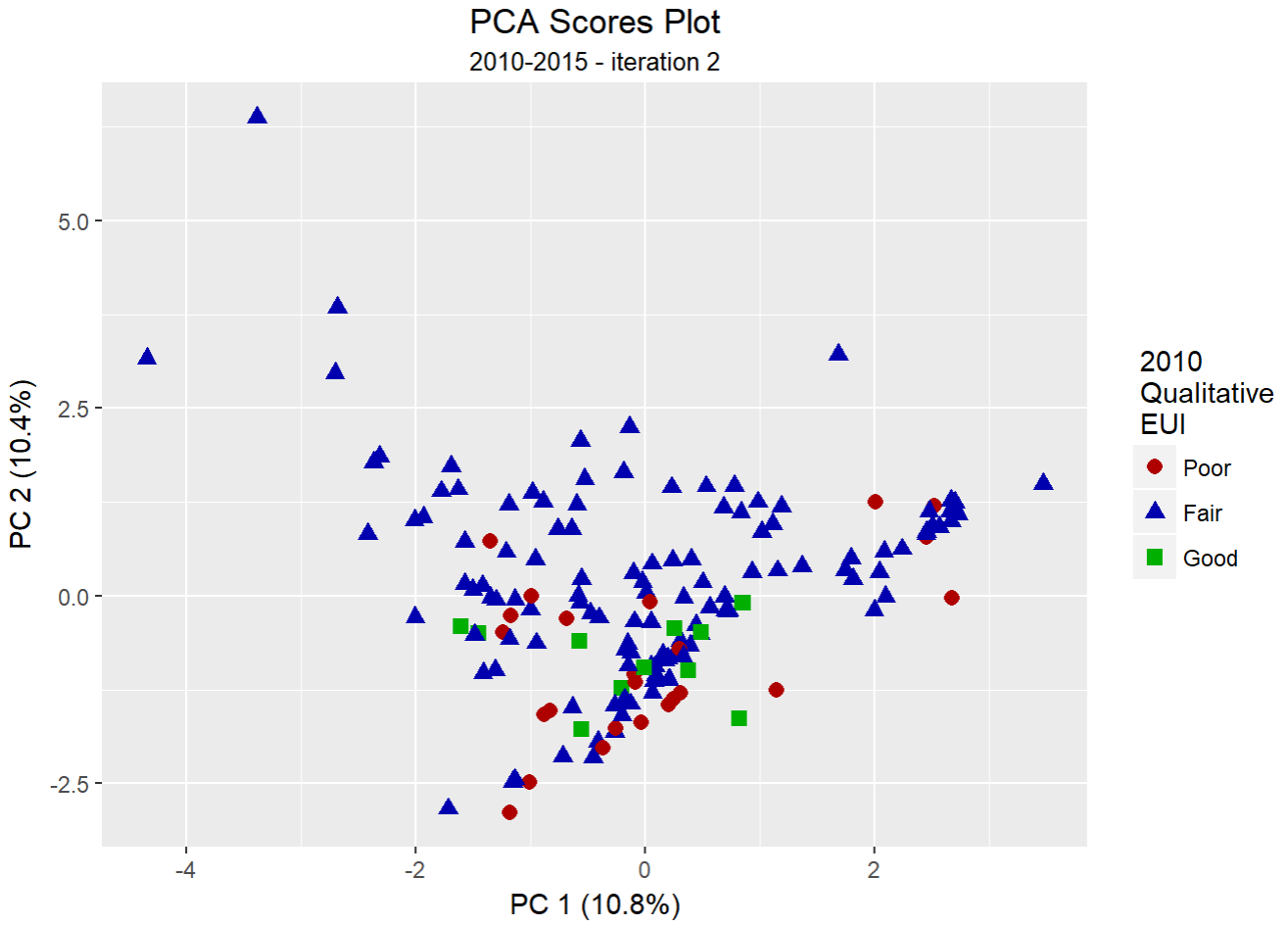
```
#Graph Pca 1 vs 2
i2.pca.PC1=i2.pca$x[,1]
i2.pca.PC2=i2.pca$x[,2]
```

```
Dep_Var=i2.data.train$x.2015.Qualitative.EUI

plot.i2.pca.1.2.2015.EUI <- ggplot(i2.data.train) + geom_point(aes(i2.pca.PC1, i2.pca.PC2, col
our = Dep_Var, shape = Dep_Var), size = 2.5) +
  labs(x = paste("PC 1 (", percent(i2.pca$sdev[1]/sum(i2.pca$sdev)), ")", sep=""),
       y = paste("PC 2 (", percent(i2.pca$sdev[2]/sum(i2.pca$sdev)), ")", sep=""),
       title="PCA Scores Plot", subtitle = "2010-2015 - iteration 2",
       colour = " 2015 \n Qualitative \n EUI", shape = " 2015 \n Qualitative \n EUI")+
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  theme( plot.title = element_text(hjust = 0.5),
         plot.subtitle = element_text(hjust = 0.5))
plot.i2.pca.1.2.2015.EUI
```

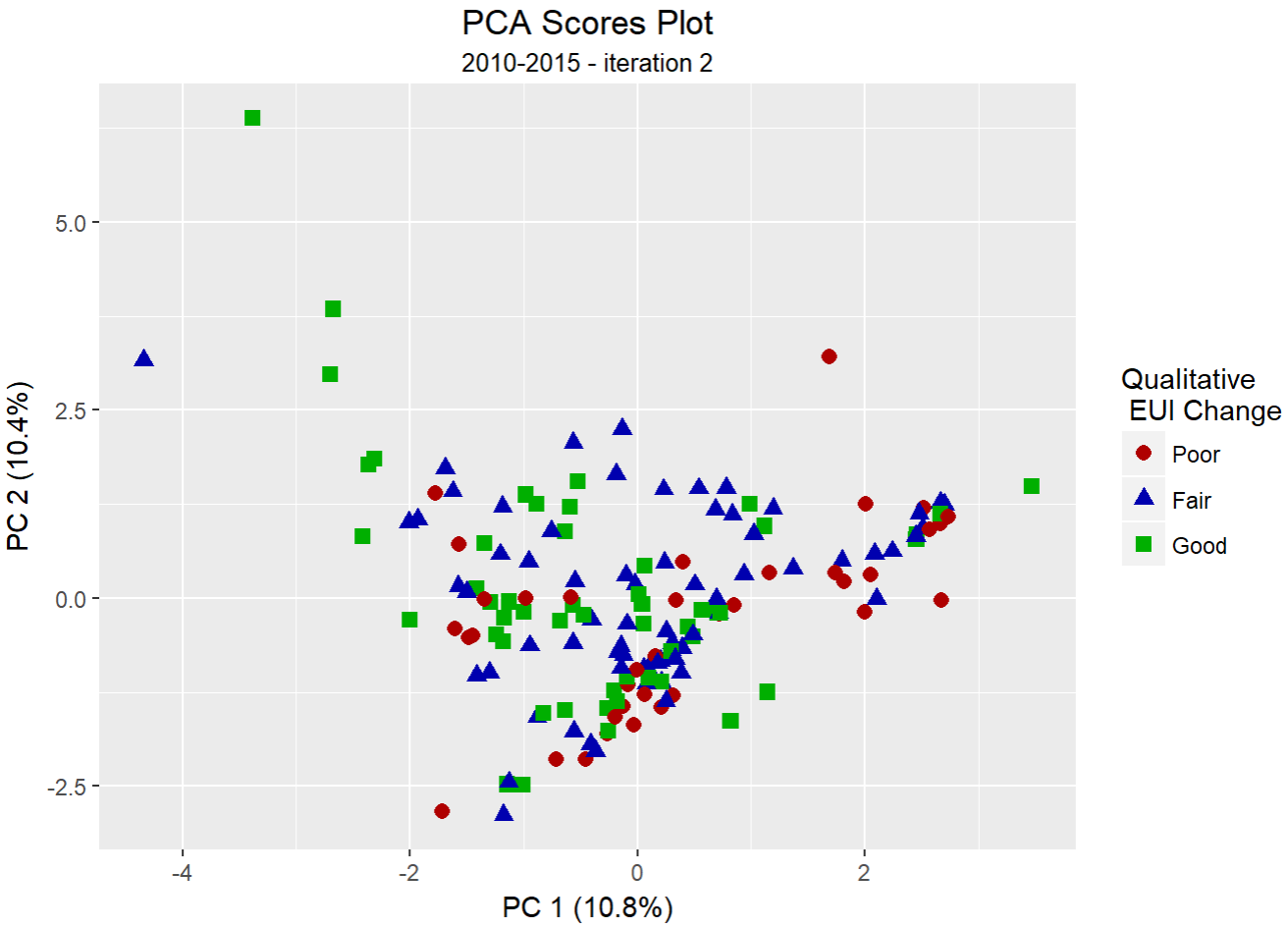



```
Dep_Var=i2.data.train$x.2010.Qualitative.EUI
plot.i2.pca.1.2.2010.EUI <- ggplot(i2.data.train) + geom_point(aes(i2.pca.PC1, i2.pca.PC2, col
our = Dep_Var, shape = Dep_Var), size = 2.5) +
  labs(x = paste("PC 1 (", percent(i2.pca$sdev[1]/sum(i2.pca$sdev)), ")", sep=""),
       y = paste("PC 2 (", percent(i2.pca$sdev[2]/sum(i2.pca$sdev)), ")", sep=""),
       title="PCA Scores Plot", subtitle = "2010-2015 - iteration 2",
       colour = " 2010 \n Qualitative \n EUI", shape = " 2010 \n Qualitative \n EUI")+
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  theme( plot.title = element_text(hjust = 0.5),
         plot.subtitle = element_text(hjust = 0.5))
```

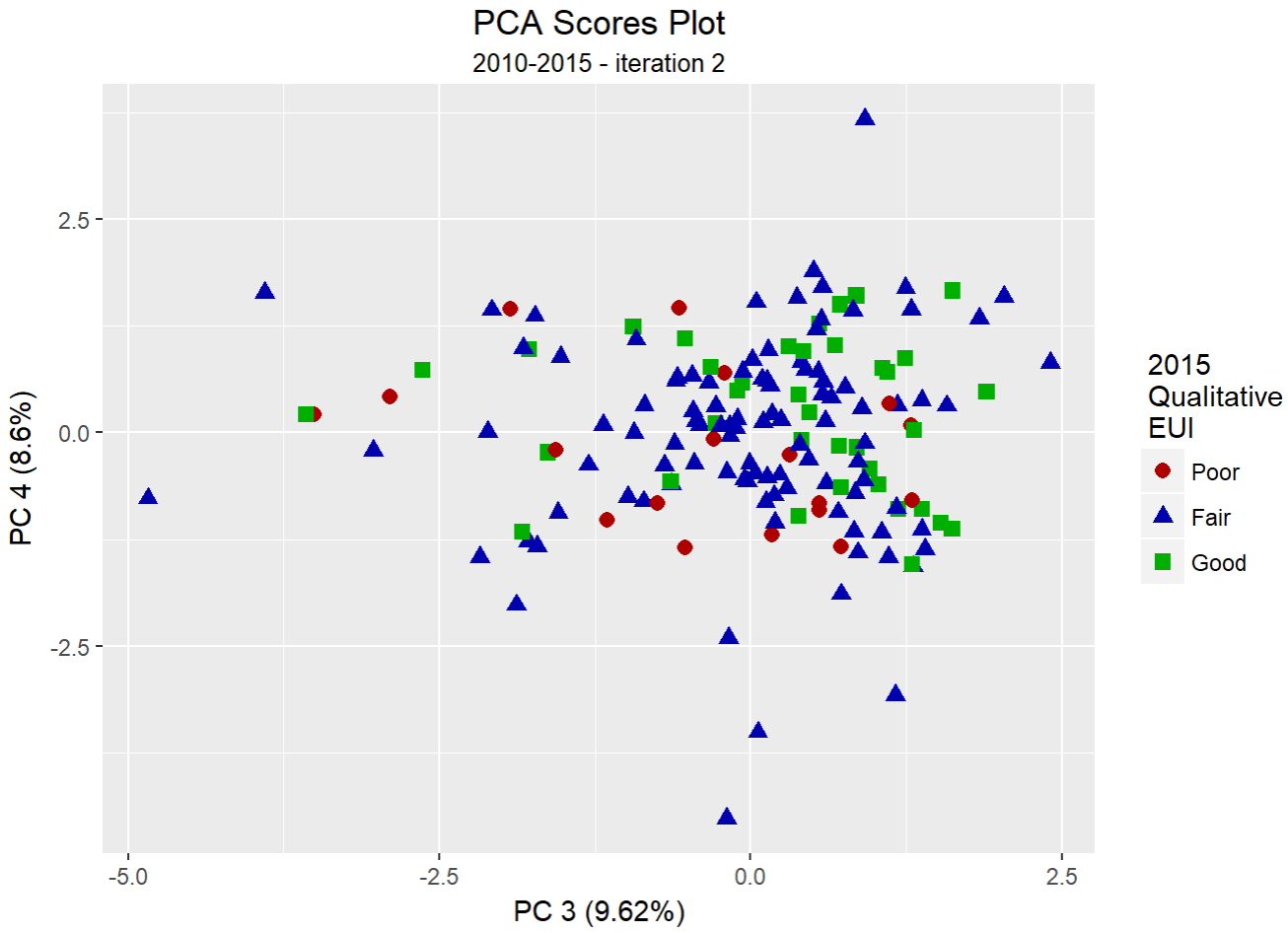
```
plot.i2.pca.1.2.2010.EUI
```



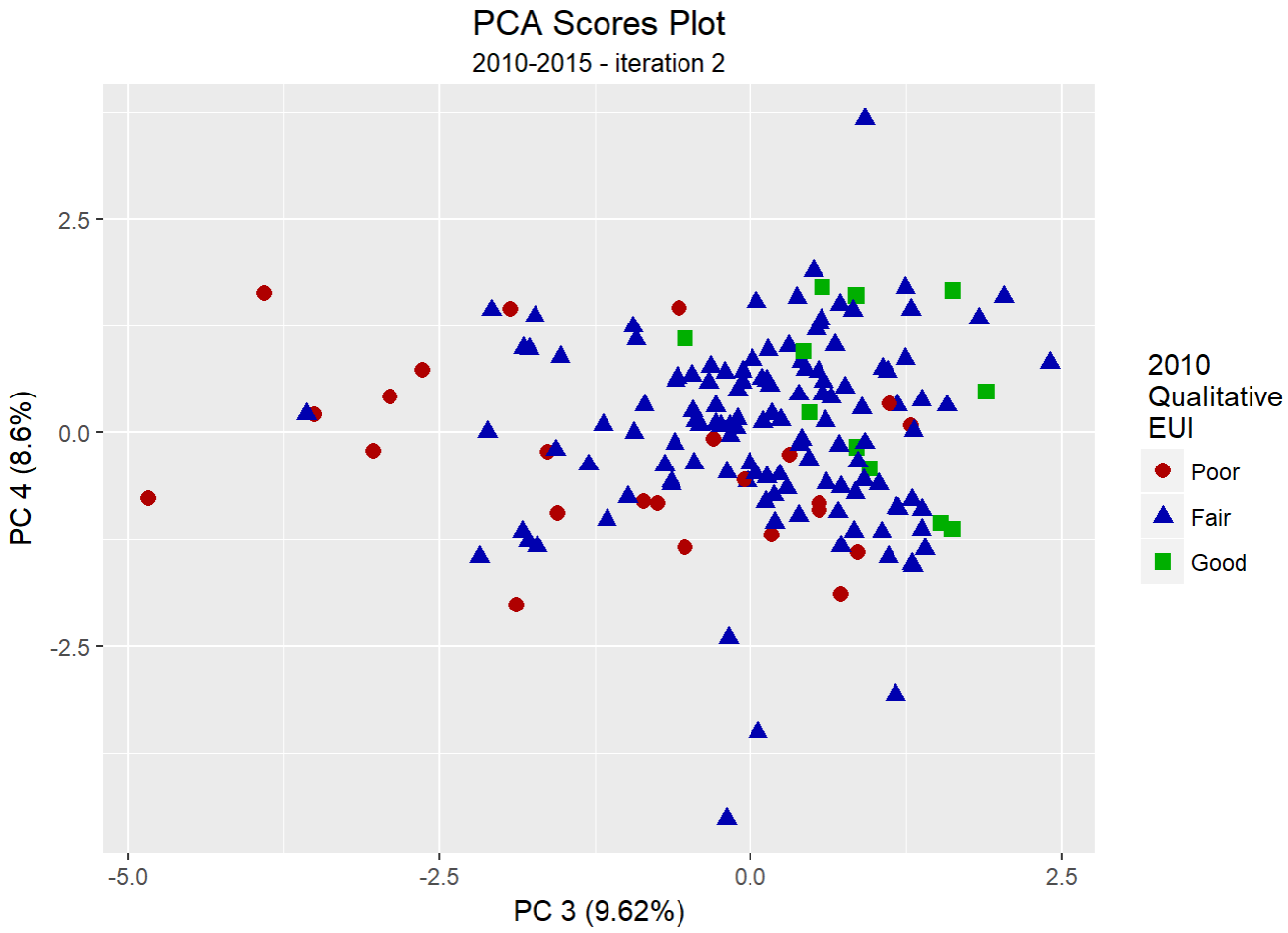

```
Dep_Var=i2.data.train$D.Qualitative.EUI
plot.i2.pca.1.2.2010.EUI <- ggplot(i2.data.train) + geom_point(aes(i2.pca.PC1, i2.pca.PC2, col
our = Dep_Var, shape = Dep_Var), size = 2.5) +
  labs(x = paste("PC 1 (", percent(i2.pca$sdev[1]/sum(i2.pca$sdev)), ")", sep=""),
       y = paste("PC 2 (", percent(i2.pca$sdev[2]/sum(i2.pca$sdev)), ")", sep=""),
       title="PCA Scores Plot", subtitle = "2010-2015 - iteration 2",
       colour = "Qualitative \n EUI Change", shape = "Qualitative \n EUI Change")+
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  theme( plot.title = element_text(hjust = 0.5),
         plot.subtitle = element_text(hjust = 0.5))
plot.i2.pca.1.2.2010.EUI
```



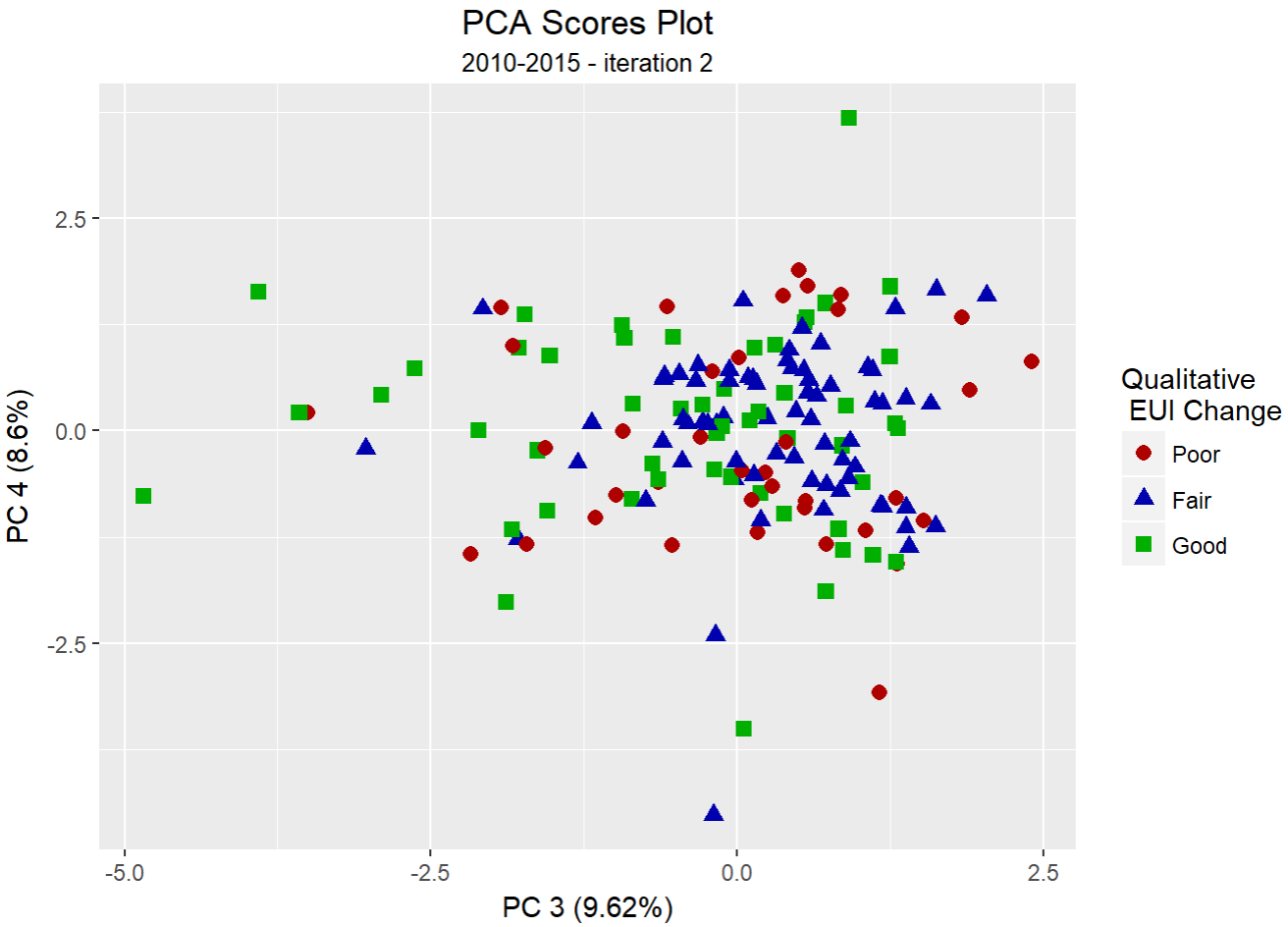

```
#Export

#Graph Pca 3 vs 4
i2.pca.PC3=i2.pca$x[,3]
i2.pca.PC4=i2.pca$x[,4]

Dep_Var=i2.data.train$x.2015.Qualitative.EUI

plot.i2.pca.3.4 <- ggplot(i2.data.train) + geom_point(aes(i2.pca.PC3, i2.pca.PC4, colour = Dep
_Var, shape = Dep_Var), size = 2.5) +
  labs(x = paste("PC 3 (", percent(i2.pca$sdev[3]/sum(i2.pca$sdev)), ")", sep=""),
       y = paste("PC 4 (", percent(i2.pca$sdev[4]/sum(i2.pca$sdev)), ")", sep=""),
       title="PCA Scores Plot", subtitle = "2010-2015 - iteration 2",
       colour = " 2015 \n Qualitative \n EUI", shape = " 2015 \n Qualitative \n EUI")+
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  theme( plot.title = element_text(hjust = 0.5),
         plot.subtitle = element_text(hjust = 0.5))
plot.i2.pca.3.4
```




```
Dep_Var=i2.data.train$x.2010.Qualitative.EUI

plot.i2.pca.3.4 <- ggplot(i2.data.train) + geom_point(aes(i2.pca.PC3, i2.pca.PC4, colour = Dep
_Var, shape = Dep_Var), size = 2.5) +
  labs(x = paste("PC 3 (", percent(i2.pca$sdev[3]/sum(i2.pca$sdev)), ")", sep=""),
       y = paste("PC 4 (", percent(i2.pca$sdev[4]/sum(i2.pca$sdev)), ")", sep=""),
       title="PCA Scores Plot", subtitle = "2010-2015 - iteration 2",
       colour = " 2010 \n Qualitative \n EUI", shape = " 2010 \n Qualitative \n EUI")+
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  theme( plot.title = element_text(hjust = 0.5),
         plot.subtitle = element_text(hjust = 0.5))
plot.i2.pca.3.4
```




```
Dep_Var=i2.data.train$D.Qualitative.EUI

plot.i2.pca.3.4 <- ggplot(i2.data.train) + geom_point(aes(i2.pca.PC3, i2.pca.PC4, colour = Dep
_Var, shape = Dep_Var), size = 2.5) +
  labs(x = paste("PC 3 (", percent(i2.pca$sdev[3]/sum(i2.pca$sdev)), ")", sep=""),
       y = paste("PC 4 (", percent(i2.pca$sdev[4]/sum(i2.pca$sdev)), ")", sep=""),
       title="PCA Scores Plot", subtitle = "2010-2015 - iteration 2",
       colour = "Qualitative \n EUI Change", shape = "Qualitative \n EUI Change")+
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  theme( plot.title = element_text(hjust = 0.5),
         plot.subtitle = element_text(hjust = 0.5))
plot.i2.pca.3.4
```


PCA Scores Plot
2010-2015 - iteration 2
2.5
0.0
-2.5
PC 4 (8.6%)
-5.0
-2.5
0.0
2.5
PC 3 (9.62%)
Qualitative
EUI Change
Poor
Fair
Good

LDA

Final_LDA_Script.R

carle

Wed Oct 10 21:41:11 2018

```
##############################
####        LDA Script     ####
##############################

# Preprocessing
set.seed(1234)
setwd("C:/Users/carle/Documents/Data/2010-2015 Data")
getwd()
```

```
## [1] "C:/Users/carle/Documents/Data/2010-2015 Data"
```

```
library(MASS)
library(Momocs)
```

```
## Warning: package 'Momocs' was built under R version 3.4.4
```

```
## This is Momocs 1.2.9
```

```
##
## Attaching package: 'Momocs'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.3
```

```
library(scales)
```

```
## Warning: package 'scales' was built under R version 3.4.3
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:Momocs':
##
##     rescale
```

```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 3.4.4
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:Momocs':
##
##     combine
```

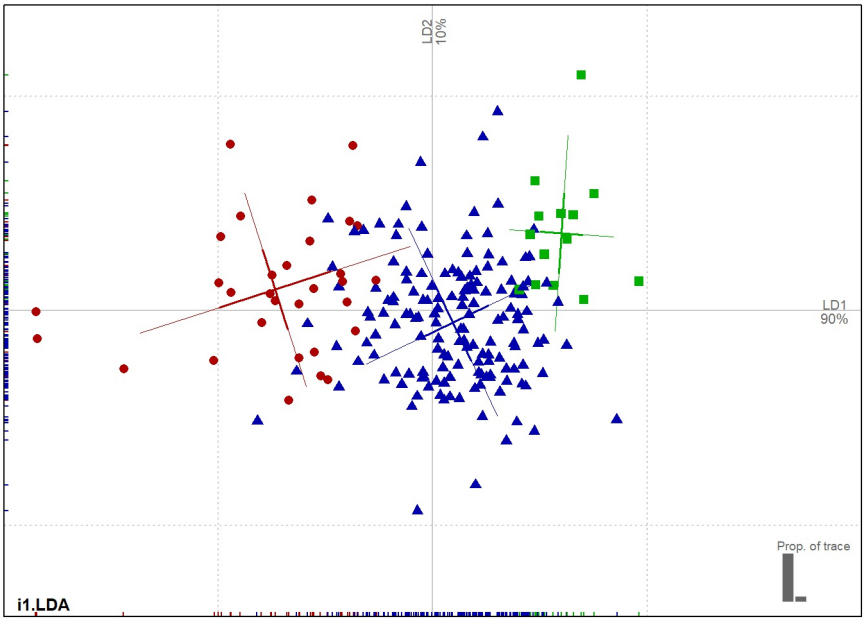
```
# na.strings = "NA" populates every blank with "NA", which R will ignore
# stringsAsFactors = TRUE will let you read in 0's as real zeroes
# check.names = FALSE permits multiple words in Column names to be seperated by a space.
# ^ [R] prefers names to read as one word (syntactically valid): "a.b.c" rather than "a b c"

# Load Data - Script is prepared for the 2010|2015 Dataset
data <- read.csv(file.choose(), row.names = 1, check.names = FALSE, na.strings = "NA", sep = ",", stringsAsFact
ors = TRUE, header = TRUE)
#Rename columns to ensure they have 'syntactically valid' names
cnames <- cbind("x.2010.EUI","x.2015.EUI","D.EUI.percent","x.2010.Qualitative.EUI","x.2015.Qualitative.EUI",
                "D.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2015.Actual.E_C","D.Actual.E_C","x.2010.Actua
l.Thermal","x.2015.Actual.Thermal","D.Actual.Thermal",
                "x.2010.Actual.Energy.Use","D.Actual.Energy.Use","x.2015.Actual.Energy.Use", "Portfolio.Manager
","Portfolio.Manager.no.","Net.Rentable.Area",
                "Exterior.Area", "Gross.Floor.Area","Enclosed.Parking","Latitude",  "Longitude","Construction.Y
ear","No.of.Structures","Building.Class","Closest.Major.City","Climate.Zone",
                "Electrically.Heated","Cooling.Tower","Soft.Landscaping",   "Occupant.Density","Vacancy.Rate","
Weekly.Operating.Hours")
colnames(data) <-cnames
rnames <- row.names(data)

# Order Qualitative outcome variables
sapply(data, class)
```

```
##               x.2010.EUI               x.2015.EUI            D.EUI.percent
##                "numeric"                "numeric"                "numeric"
##   x.2010.Qualitative.EUI   x.2015.Qualitative.EUI        D.Qualitative.EUI
##                 "factor"                 "factor"                 "factor"
## x.2010.Actual.E_C.Energy        x.2015.Actual.E_C             D.Actual.E_C
##                "numeric"                "numeric"                "numeric"
##    x.2010.Actual.Thermal    x.2015.Actual.Thermal         D.Actual.Thermal
##                "numeric"                "numeric"                "numeric"
## x.2010.Actual.Energy.Use      D.Actual.Energy.Use x.2015.Actual.Energy.Use
##                "numeric"                "numeric"                "numeric"
##        Portfolio.Manager    Portfolio.Manager.no.        Net.Rentable.Area
##                 "factor"                "integer"                "integer"
##            Exterior.Area         Gross.Floor.Area         Enclosed.Parking
##                "integer"                "integer"                "integer"
##                 Latitude                Longitude        Construction.Year
##                "numeric"                "numeric"                "integer"
##         No.of.Structures           Building.Class       Closest.Major.City
##                "integer"                "integer"                 "factor"
##             Climate.Zone      Electrically.Heated             Cooling.Tower
##                "integer"                "integer"                "integer"
##         Soft.Landscaping         Occupant.Density             Vacancy.Rate
##                "integer"                "numeric"                "numeric"
##   Weekly.Operating.Hours
##                "numeric"
```

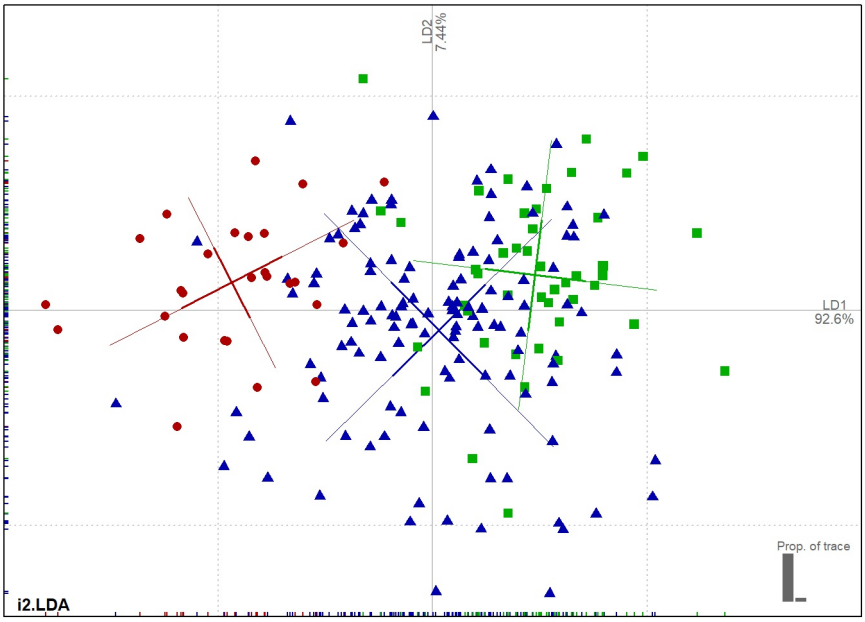
```
data[,c("x.2010.Qualitative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI")] <- lapply(data[,c("x.2010.Quali
tative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI")] , factor)
data["x.2010.Qualitative.EUI"]=ordered(data["x.2010.Qualitative.EUI"],levels = c("Poor","Fair","Good"))
data["x.2015.Qualitative.EUI"]=ordered(data["x.2015.Qualitative.EUI"],levels = c("Poor","Fair","Good"))
data["D.Qualitative.EUI"]=ordered(data["D.Qualitative.EUI"],levels = c("Poor","Fair","Good"))
sapply(data[,c("x.2010.Qualitative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI")],is.ordered)
```

```
## x.2010.Qualitative.EUI x.2015.Qualitative.EUI      D.Qualitative.EUI
##                   TRUE                   TRUE                   TRUE
```

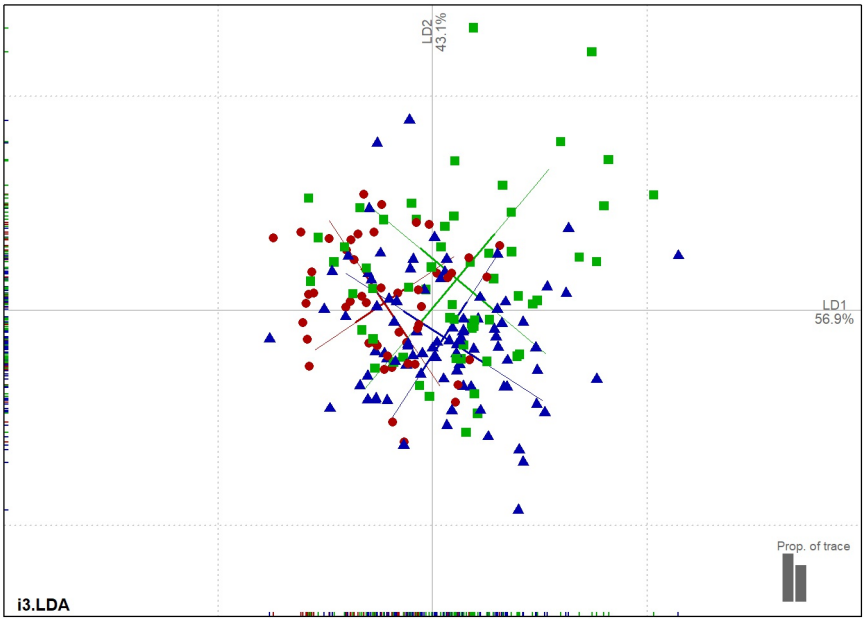
```
#Remove outliers (as determined from PCA)
outliers<-c("256","186","289")
data<-data[!rownames(data) %in% outliers,]


################################
####    LDA iteration 1   ####
################################

# Select outcome variable (i.e. x.2010.Qualitative.EUI) and independent variables for iteration 1 (i1)
i1.data<-data[,c("x.2010.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
                 "Portfolio.Manager.no.",   "Gross.Floor.Area", "Enclosed.Parking",
                 "Latitude",    "Longitude",    "Construction.Year",    "No.of.Structures", "Building.Class",
                 "Electrically.Heated", "Cooling.Tower",    "Soft.Landscaping", "Occupant.Density",
                 "Vacancy.Rate",    "Weekly.Operating.Hours")]

# perform LDA
i1.LDA <- LDA(i1.data[-1], fac=i1.data$x.2010.Qualitative.EUI, retain=1)
plot(i1.LDA,points = TRUE, labelsgroups = FALSE,
 col = c("Poor" ="#AF0000", "Fair"="#0000AF", "Good"= "#00AF00"), cex=1, pch = c(16,17,15), box=TRUE)
```



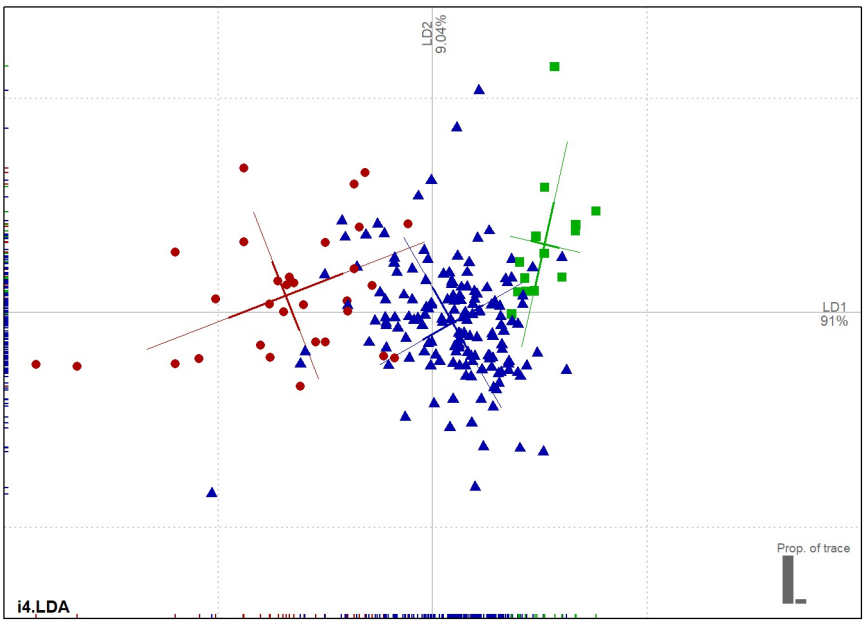

```
##############################
####    LDA iteration 2   ####
##############################

# Select outcome variable (i.e. x.2015.Qualitative.EUI) and independent variables for iteration 2 (i2)
i2.data<-data[,c("x.2015.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
                 "Portfolio.Manager.no.", "Gross.Floor.Area", "Enclosed.Parking",
                 "Latitude", "Longitude", "Construction.Year",    "No.of.Structures", "Building.Class",
                 "Electrically.Heated",   "Cooling.Tower",    "Soft.Landscaping", "Occupant.Density",
                 "Vacancy.Rate",  "Weekly.Operating.Hours")]

# perform LDA
i2.LDA <- LDA(i2.data[-1], fac=i2.data$x.2015.Qualitative.EUI, retain=1)

plot(i2.LDA,points = TRUE, labelsgroups = FALSE,
     col = c("Poor" ="#AF0000", "Fair"="#0000AF", "Good"= "#00AF00"), cex=1, pch = c(16,17,15), box=TRUE)
```



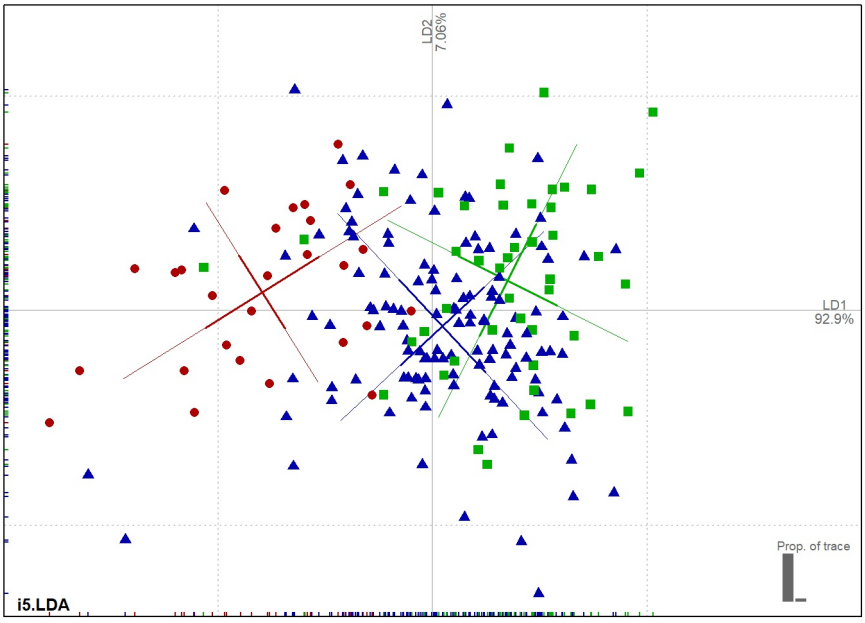

```
##############################
####    LDA iteration 3   ####
##############################

# Select outcome variable (i.e. x.D.Qualitative.EUI) and independent variables for iteration 3 (i3)
i3.data<-data[,c("D.Qualitative.EUI", "x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
                 "Portfolio.Manager.no.", "Gross.Floor.Area", "Enclosed.Parking",
                 "Latitude",   "Longitude",    "Construction.Year",    "No.of.Structures", "Building.Class",
                 "Electrically.Heated",   "Cooling.Tower",    "Soft.Landscaping", "Occupant.Density",
                 "Vacancy.Rate",  "Weekly.Operating.Hours")]

# perform LDA
i3.LDA <- LDA(i3.data[-1], fac=i3.data$D.Qualitative.EUI, retain=1)
plot(i3.LDA,points = TRUE, labelsgroups = FALSE,
     col = c("Poor" ="#AF0000", "Fair"="#0000AF", "Good"= "#00AF00"), cex=1, pch = c(16,17,15), box=TRUE)
```



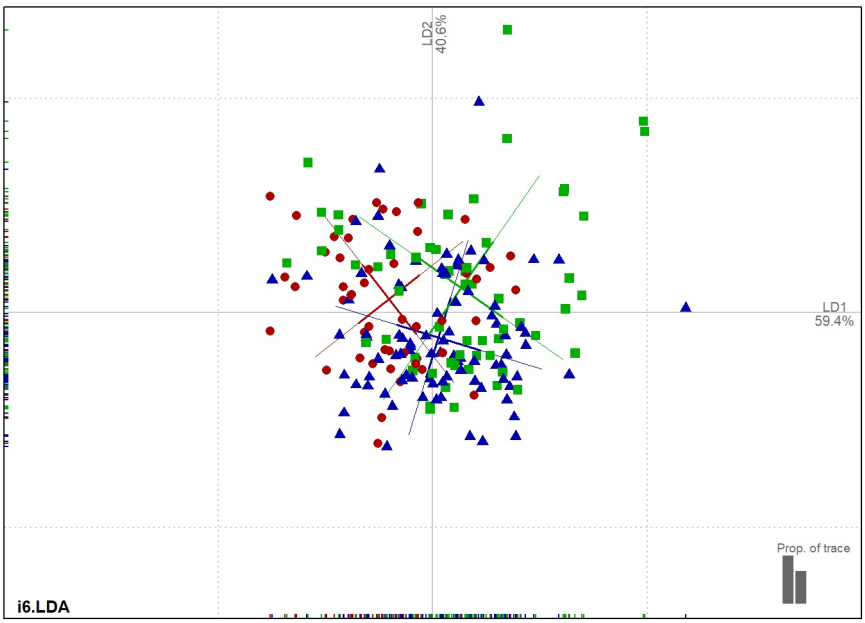

```
##############################
####    LDA iteration 4   ####
##############################

# Select outcome variable (i.e. x.2010.Qualitative.EUI) and independent variables for iteration 4 (i4)
i4.data<-data[,c("x.2010.Qualitative.EUI", "Portfolio.Manager.no.","Exterior.Area", "x.2010.Actual.Energy.Use",
                 "Construction.Year",    "No.of.Structures", "Building.Class", "Climate.Zone",
                 "Electrically.Heated", "Cooling.Tower",    "Soft.Landscaping", "Occupant.Density",
                 "Vacancy.Rate",    "Weekly.Operating.Hours")]

# perform LDA
i4.LDA <- LDA(i4.data[-1], fac=i4.data$x.2010.Qualitative.EUI, retain=1)
plot(i4.LDA,points = TRUE, labelsgroups = FALSE,
     col = c("Poor" ="#AF0000", "Fair"="#0000AF", "Good"= "#00AF00"), cex=1, pch = c(16,17,15), box=TRUE)
```



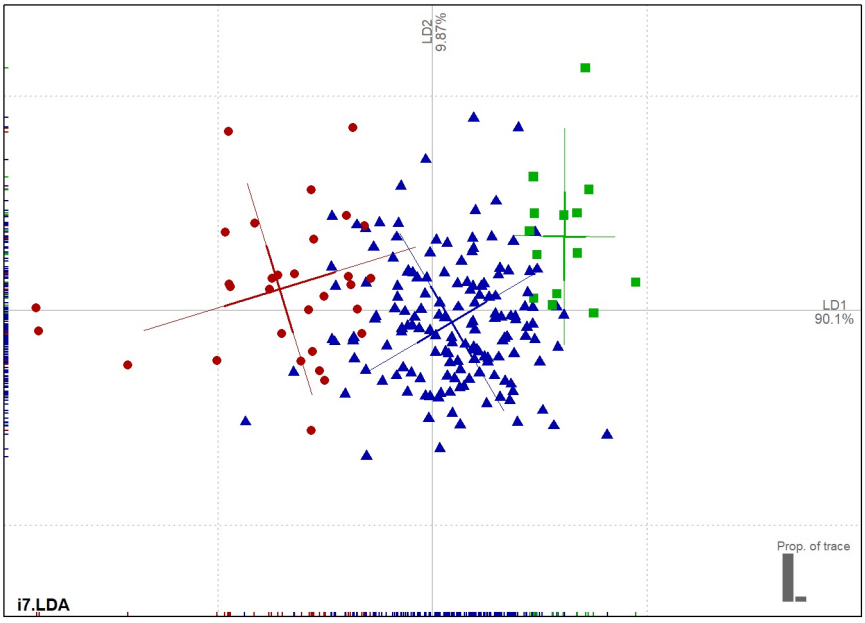

```
##############################
####    LDA iteration 5   ####
##############################

# Select outcome variable (i.e. x.2015.Qualitative.EUI) and independent variables for iteration 5 (i5)
i5.data<-data[,c("x.2015.Qualitative.EUI", "Portfolio.Manager.no.","Exterior.Area", "x.2010.Actual.Energy.Use",
                 "Construction.Year",    "No.of.Structures", "Building.Class", "Climate.Zone",
                 "Electrically.Heated", "Cooling.Tower",    "Soft.Landscaping", "Occupant.Density",
                 "Vacancy.Rate",    "Weekly.Operating.Hours")]

# perform LDA
i5.LDA <- LDA(i5.data[-1], fac=i5.data$x.2015.Qualitative.EUI, retain=1)
p5=plot(i5.LDA,points = TRUE, labelsgroups = FALSE,
     col = c("Poor" ="#AF0000", "Fair"="#0000AF", "Good"= "#00AF00"), cex=1, pch = c(16,17,15), box=TRUE)
```



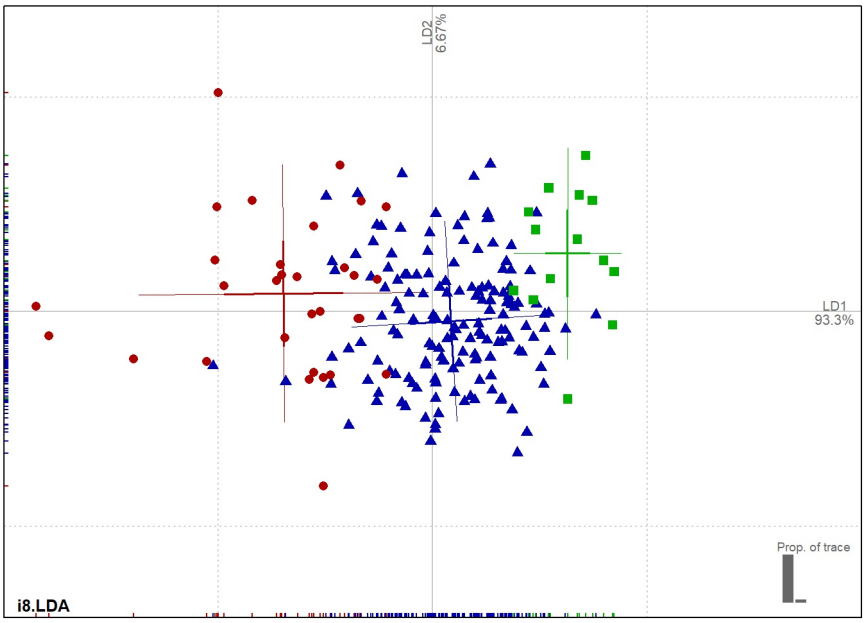

```
##############################
####    LDA iteration 6   ####
##############################

# Select outcome variable (i.e. D.Qualitative.EUI) and independent variables for iteration 6 (i6)
i6.data<-data[,c("D.Qualitative.EUI", "Portfolio.Manager.no.","Exterior.Area", "x.2010.Actual.Energy.Use",
                 "Construction.Year",    "No.of.Structures", "Building.Class", "Climate.Zone",
                 "Electrically.Heated", "Cooling.Tower",    "Soft.Landscaping", "Occupant.Density",
                 "Vacancy.Rate",    "Weekly.Operating.Hours")]

# perform LDA
i6.LDA <- LDA(i6.data[-1], fac=i6.data$D.Qualitative.EUI, retain=1)
plot(i6.LDA,points = TRUE, labelsgroups = FALSE,
     col = c("Poor" ="#AF0000", "Fair"="#0000AF", "Good"= "#00AF00"), cex=1, pch = c(16,17,15), box=TRUE)
```




```
##############################
####    LDA iteration 7   ####
##############################

# Select outcome variable (i.e. x.2010.Qualitative.EUI) and independent variables for iteration 7 (i7)
i7.data<-data[,c("x.2010.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
                 "Portfolio.Manager.no.",  "Latitude",  "Longitude",    "Construction.Year",    "No.of.Structur
es",
                 "Building.Class", "Electrically.Heated",   "Cooling.Tower",    "Soft.Landscaping", "Occupant.D
ensity",
                 "Vacancy.Rate",    "Weekly.Operating.Hours")]

# perform LDA
i7.LDA <- LDA(i7.data[-1], fac=i7.data$x.2010.Qualitative.EUI, retain=1)
plot(i7.LDA,points = TRUE, labelsgroups = FALSE,
        col = c("Poor" ="#AF0000", "Fair"="#0000AF", "Good"= "#00AF00"), cex=1, pch = c(16,17,15), box=TRUE)
```




```
##############################
####    LDA iteration 8   ####
##############################

# Select outcome variable (i.e. x.2010.Qualitative.EUI) and independent variables for iteration 8 (i8)
i8.data<-data[,c("x.2010.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
                 "Portfolio.Manager.no.", "Latitude",   "Longitude",    "Construction.Year","Occupant.Density",
                 "Weekly.Operating.Hours")]

# perform LDA
i8.LDA <- LDA(i8.data[-1], fac=i8.data$x.2010.Qualitative.EUI, retain=1)
plot(i8.LDA,points = TRUE, labelsgroups = FALSE,
        col = c("Poor" ="#AF0000", "Fair"="#0000AF", "Good"= "#00AF00"), cex=1, pch = c(16,17,15), box=TRUE)
```




```
# View Results for Post Processing #
i1.LDA$CV.correct
```

```
## [1] 0.8434343
```

```
i2.LDA$CV.correct
```

```
## [1] 0.6616162
```

```
i3.LDA$CV.correct
```

```
## [1] 0.4494949
```

```
i4.LDA$CV.correct
```

```
## [1] 0.8434343
```

```
i5.LDA$CV.correct
```

```
## [1] 0.6767677
```

```
i6.LDA$CV.correct
```

```
## [1] 0.4242424
```

```
i7.LDA$CV.correct
```

```
## [1] 0.8434343
```

```
i8.LDA$CV.correct
```

```
## [1] 0.8535354
```

```
i1.LDA$CV.ce
```

```
##      Poor      Fair      Good
## 0.6129032 0.9411765 0.2857143
```

```
i2.LDA$CV.ce
```

```
##      Poor      Fair      Good
## 0.5925926 0.8080000 0.3043478
```

```
i3.LDA$CV.ce
```

```
##      Poor      Fair      Good
## 0.3200000 0.6704545 0.2333333
```

```
i4.LDA$CV.ce
```

```
##      Poor      Fair      Good
## 0.6451613 0.9411765 0.2142857
```

```
i5.LDA$CV.ce
```

```
##      Poor      Fair      Good
## 0.4814815 0.8880000 0.2173913
```

```
i6.LDA$CV.ce
```

```
##      Poor      Fair      Good
## 0.3400000 0.6363636 0.1833333
```

```
i7.LDA$CV.ce
```

```
##      Poor      Fair      Good
## 0.5806452 0.9477124 0.2857143
```

```
i8.LDA$CV.ce
```

```
##      Poor      Fair      Good
## 0.5161290 0.9607843 0.4285714
```

```
i1.LDA$CV.tab
```

```
##       classified
## actual Poor Fair Good
##   Poor   19   12    0
##   Fair    7  144    2
##   Good    0   10    4
```

```
i2.LDA$CV.tab
```

```
##       classified
## actual Poor Fair Good
##   Poor   16   11    0
##   Fair    7  101   17
##   Good    0   32   14
```

```
i3.LDA$CV.tab
```

```
##       classified
## actual Poor Fair Good
##   Poor   16   23   11
##   Fair   14   59   15
##   Good   14   32   14
```

```
i4.LDA$CV.tab
```

```
##       classified
## actual Poor Fair Good
##   Poor   20   11    0
##   Fair    6  144    3
##   Good    0   11    3
```

```
i5.LDA$CV.tab
```

```
##       classified
## actual Poor Fair Good
##   Poor   13   14    0
##   Fair    7  111    7
##   Good    2   34   10
```

```
i6.LDA$CV.tab
```

```
##       classified
## actual Poor Fair Good
##   Poor   17   23   10
##   Fair   15   56   17
##   Good   11   38   11
```

```
i7.LDA$CV.tab
```

```
##       classified
## actual Poor Fair Good
##   Poor   18   13    0
##   Fair    6  145    2
##   Good    0   10    4
```

```
i8.LDA$CV.tab
```

```
##       classified
## actual Poor Fair Good
##   Poor   16   15    0
##   Fair    6  147    0
##   Good    0    8    6
```

```
i1.LDA$LDs
```

```
##                                   LD1           LD2
## x.2010.Actual.E_C.Energy -3.236569e+00  4.268376e-01
## x.2010.Actual.Thermal    -1.819042e+00 -1.390963e+00
## Portfolio.Manager.no.     1.293701e-01 -1.057041e+00
## Gross.Floor.Area          5.052498e+04 -2.059676e+05
## Enclosed.Parking          8.551342e+03 -4.330298e+04
## Latitude                 -8.139789e-01 -8.890453e-01
## Longitude                 6.528603e+00  2.558399e+00
## Construction.Year         2.247567e+00  4.431773e+00
## No.of.Structures          2.129314e-02 -1.081149e-01
## Building.Class            5.230114e-02  1.473581e-01
## Electrically.Heated       1.872562e-02  1.496117e-01
## Cooling.Tower             1.247883e-01 -1.391289e-01
## Soft.Landscaping          2.725804e-02  4.001846e-02
## Occupant.Density         -2.533467e-02  4.777270e-01
## Vacancy.Rate             -1.376548e-01  1.827925e-01
## Weekly.Operating.Hours    5.504545e-02 -6.847589e-01
```

```
i2.LDA$LDs
```

```
##                                  LD1           LD2
## x.2010.Actual.E_C.Energy -2.031555e+00  3.887511e-01
## x.2010.Actual.Thermal    -2.019348e+00 -1.291150e+00
## Portfolio.Manager.no.     4.677587e-01 -1.340664e+00
## Gross.Floor.Area          6.824198e+04 -2.403813e+05
## Enclosed.Parking          1.567212e+04 -1.747423e+04
## Latitude                 -1.101545e+00 -4.290619e-01
## Longitude                 9.168770e+00  6.531196e-02
## Construction.Year         1.445961e+00  4.509221e+00
## No.of.Structures          2.812141e-02 -9.036107e-02
## Building.Class            1.659140e-01 -3.549794e-02
## Electrically.Heated       2.372259e-03  5.985189e-02
## Cooling.Tower             3.854198e-02 -1.630737e-02
## Soft.Landscaping          4.123068e-02  4.041914e-02
## Occupant.Density          1.757765e-01  3.358664e-01
## Vacancy.Rate             -5.848813e-01 -1.582223e+00
## Weekly.Operating.Hours   -3.153389e-01 -4.406037e-01
```

```
i3.LDA$LDs
```

```
##                                  LD1           LD2
## x.2010.Actual.E_C.Energy -7.984652e-01  2.839172e+00
## x.2010.Actual.Thermal     5.363239e-01  1.798580e+00
## Portfolio.Manager.no.     4.249802e-01  7.242304e-01
## Gross.Floor.Area          2.058130e+05  3.912249e+04
## Enclosed.Parking          3.068370e+04  6.422616e+03
## Latitude                 -2.510177e-01 -1.750321e-01
## Longitude                 6.984613e+00 -3.153267e+00
## Construction.Year         1.038280e+00 -5.367320e+00
## No.of.Structures          4.138847e-02 -1.153200e-02
## Building.Class            3.137111e-01  1.597749e-01
## Electrically.Heated      -3.264393e-02 -3.394965e-02
## Cooling.Tower            -7.381333e-02 -1.274828e-01
## Soft.Landscaping          5.537621e-02  2.213191e-01
## Occupant.Density          3.462059e-01  5.670786e-01
## Vacancy.Rate             -5.916914e-01 -6.660534e-01
## Weekly.Operating.Hours    1.692857e+00  1.684494e+00
```

```
i4.LDA$LDs
```

```
##                                  LD1           LD2
## Portfolio.Manager.no.     1.423401e-01 -1.123598e+00
## Exterior.Area             6.255374e+04 -2.621240e+05
## x.2010.Actual.Energy.Use -5.119631e+00 -1.539511e+00
## Construction.Year         2.234092e+00  4.988372e+00
## No.of.Structures          2.273790e-02 -1.141205e-01
## Building.Class            5.152425e-02  1.636337e-01
## Climate.Zone              2.758570e-01 -2.898396e-01
## Electrically.Heated       1.743286e-02  1.627166e-01
## Cooling.Tower             1.280753e-01 -1.371691e-01
## Soft.Landscaping          2.723845e-02  4.573716e-02
## Occupant.Density         -3.071066e-02  5.111035e-01
## Vacancy.Rate             -1.415879e-01  1.828336e-01
## Weekly.Operating.Hours    6.302585e-02 -7.307364e-01
```

```
i5.LDA$LDs
```

```
##                                  LD1           LD2
## Portfolio.Manager.no.     5.918823e-01 -1.466742e+00
## Exterior.Area             1.066676e+05 -2.842753e+05
## x.2010.Actual.Energy.Use -4.775105e+00 -2.553117e+00
## Construction.Year         1.587098e+00  6.016176e+00
## No.of.Structures          3.585727e-02 -1.000802e-01
## Building.Class            1.976063e-01  1.622962e-02
## Climate.Zone              3.817560e-01 -3.365586e-01
## Electrically.Heated       1.133512e-03  7.382114e-02
## Cooling.Tower             4.613010e-02 -6.057859e-03
## Soft.Landscaping          4.772621e-02  6.406637e-02
## Occupant.Density          1.988842e-01  4.725273e-01
## Vacancy.Rate             -6.487457e-01 -2.138795e+00
## Weekly.Operating.Hours   -3.613325e-01 -6.502776e-01
```

```
i6.LDA$LDs
```

```
##                                  LD1           LD2
## Portfolio.Manager.no.     5.567930e-01  6.971645e-01
## Exterior.Area             2.527574e+05 -2.764583e+03
## x.2010.Actual.Energy.Use  4.685721e-01  5.127334e+00
## Construction.Year         2.205435e-01 -6.097201e+00
## No.of.Structures          4.111950e-02 -2.180037e-02
## Building.Class            3.511490e-01  1.049335e-01
## Climate.Zone              5.195268e-01 -4.881964e-01
## Electrically.Heated      -3.930634e-02 -2.985423e-02
## Cooling.Tower            -9.697794e-02 -1.229394e-01
## Soft.Landscaping          9.282638e-02  2.295994e-01
## Occupant.Density          4.499273e-01  5.428990e-01
## Vacancy.Rate             -7.205480e-01 -5.965362e-01
## Weekly.Operating.Hours    2.026209e+00  1.465037e+00
```

```
i7.LDA$LDs
```

```
##                                  LD1         LD2
## x.2010.Actual.E_C.Energy -3.33182964  0.68058610
## x.2010.Actual.Thermal    -1.88574961 -1.31431737
## Portfolio.Manager.no.     0.12478021 -1.10925088
## Latitude                 -0.84598205 -0.86553304
## Longitude                 6.74836728  2.18486717
## Construction.Year         2.35189742  4.44681730
## No.of.Structures          0.02106952 -0.11404397
## Building.Class            0.05508607  0.14949726
## Electrically.Heated       0.02050478  0.15429531
## Cooling.Tower             0.12747063 -0.15387445
## Soft.Landscaping          0.02841249  0.03964557
## Occupant.Density         -0.02224980  0.49890214
## Vacancy.Rate             -0.14037692  0.20024436
## Weekly.Operating.Hours    0.05119495 -0.71647934
```

```
i8.LDA$LDs
```

```
##                                  LD1        LD2
## x.2010.Actual.E_C.Energy -3.42843107  1.1721853
## x.2010.Actual.Thermal    -1.95414193 -1.5043461
## Portfolio.Manager.no.     0.11965032 -1.4262686
## Latitude                 -0.87889111 -1.0268033
## Longitude                 6.97278801  2.1709711
## Construction.Year         2.45985788  5.4573947
## Occupant.Density         -0.01890464  0.6383943
## Weekly.Operating.Hours    0.04698054 -0.9185618
```

KNN

Final_KNN_Script.R

carle

Wed Oct 10 22:22:30 2018

```
##############################
####        KNN Script     ####
##############################

# Preprocessing
set.seed(1234)
setwd("C:/Users/carle/Documents/Data")
getwd()
```

```
## [1] "C:/Users/carle/Documents/Data"
```

```
library(class)
library(ggplot2)# load ggplot to allow quick plots
```

```
## Warning: package 'ggplot2' was built under R version 3.4.3
```

```
library(lattice)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
```

```
library(MASS)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
```

```
##
##     intersect, setdiff, setequal, union
```

```
# na.strings = "NA" populates every blank with "NA", which R will ignore
# stringsAsFactors = TRUE will let you read in 0's as real zeroes
# check.names = FALSE permits multiple words in Column names to be seperated by a space.
# ^ [R] prefers names to read as one word (syntactically valid): "a.b.c" rather than "a b c"
# Load Data - Script is prepared for the 2010|2015 Dataset
data <- read.csv(file.choose(), row.names = 1, check.names = FALSE, na.strings = "NA", sep = "
,", stringsAsFactors = TRUE, header = TRUE)

#Rename Columns to have 'syntactically valid' names
cnames <- cbind("x.2010.EUI","x.2015.EUI","D.EUI.percent","x.2010.Qualitative.EUI","x.2015.Qua
litative.EUI",
                "D.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2015.Actual.E_C","D.Actual.E
_C","x.2010.Actual.Thermal","x.2015.Actual.Thermal","D.Actual.Thermal",
                "x.2010.Actual.Energy.Use","D.Actual.Energy.Use","x.2015.Actual.Energy.Use", "
Portfolio.Manager","Portfolio.Manager.no.","Net.Rentable.Area",
                "Exterior.Area", "Gross.Floor.Area","Enclosed.Parking","Latitude",  "Longitude
","Construction.Year","No.of.Structures","Building.Class","Closest.Major.City","Climate.Zone",
                "Electrically.Heated","Cooling.Tower","Soft.Landscaping",   "Occupant.Density"
,"Vacancy.Rate","Weekly.Operating.Hours")
colnames(data) <-cnames
rnames <- row.names(data)

# Order Qualitative outcome variables
sapply(data, class)
```

```
##               x.2010.EUI               x.2015.EUI           D.EUI.percent
##                "numeric"                "numeric"                "numeric"
##   x.2010.Qualitative.EUI   x.2015.Qualitative.EUI       D.Qualitative.EUI
##                 "factor"                 "factor"                 "factor"
## x.2010.Actual.E_C.Energy       x.2015.Actual.E_C            D.Actual.E_C
##                "numeric"                "numeric"                "numeric"
##    x.2010.Actual.Thermal    x.2015.Actual.Thermal        D.Actual.Thermal
##                "numeric"                "numeric"                "numeric"
## x.2010.Actual.Energy.Use     D.Actual.Energy.Use x.2015.Actual.Energy.Use
##                "numeric"                "numeric"                "numeric"
##        Portfolio.Manager    Portfolio.Manager.no.       Net.Rentable.Area
##                 "factor"                "integer"                "integer"
##            Exterior.Area        Gross.Floor.Area        Enclosed.Parking
##                "integer"                "integer"                "integer"
##                 Latitude                Longitude       Construction.Year
##                "numeric"                "numeric"                "integer"
##         No.of.Structures          Building.Class      Closest.Major.City
##                "integer"                "integer"                 "factor"
##             Climate.Zone     Electrically.Heated           Cooling.Tower
##                "integer"                "integer"                "integer"
##         Soft.Landscaping        Occupant.Density            Vacancy.Rate
##                "integer"                "numeric"                "numeric"
##   Weekly.Operating.Hours
##                "numeric"
```

```
data[,c("x.2010.Qualitative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI")] <- lapply(data
[,c("x.2010.Qualitative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI")] , factor)
data[,"x.2010.Qualitative.EUI"]=ordered(data[,"x.2010.Qualitative.EUI"],levels = c("Poor","Fai
r","Good"))
data[,"x.2015.Qualitative.EUI"]=ordered(data[,"x.2015.Qualitative.EUI"],levels = c("Poor","Fai
r","Good"))
data[,"D.Qualitative.EUI"]=ordered(data[,"D.Qualitative.EUI"],levels = c("Poor","Fair","Good")
)
sapply(data[,c("x.2010.Qualitative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI")],is.orde
red)
```

```
## x.2010.Qualitative.EUI x.2015.Qualitative.EUI      D.Qualitative.EUI
##                   TRUE                   TRUE                   TRUE
```

```
#Remove outliers (as determined from PCA)
outliers<-c("256","186","289")
data<-data[!rownames(data) %in% outliers,]
rnames <- row.names(data)

##############################
####    KNN iteration 1   ####
##############################

i1.data<-data[,c("x.2010.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
                 "Portfolio.Manager.no.", "Latitude", "Longitude",    "Construction.Year",
  "Occupant.Density",
                 "Vacancy.Rate",  "Weekly.Operating.Hours")]
cnames<-c("x.2010.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
          "Portfolio.Manager.no.",  "Latitude", "Longitude",    "Construction.Year",    "Occup
ant.Density",
          "Vacancy.Rate",   "Weekly.Operating.Hours")
# Normalize
QG<-i1.data$x.2010.Qualitative.EUI
norm.data<- as.data.frame(lapply(i1.data[,-1], scale))
i1.data <-cbind(QG, norm.data)
row.names(i1.data) <- rnames
colnames(i1.data) <-cnames
##create Train and Test set
train.rows = sample(1:nrow(i1.data), (nrow(i1.data)*0.7))
i1.data.train = i1.data[train.rows,]
i1.data.test = i1.data[-train.rows,]
#View(i1.data.test)
nrow(i1.data.test)
```

```
## [1] 60
```

```
# Normalization Check
summary(i1.data)
```

```
##  x.2010.Qualitative.EUI x.2010.Actual.E_C.Energy x.2010.Actual.Thermal
```

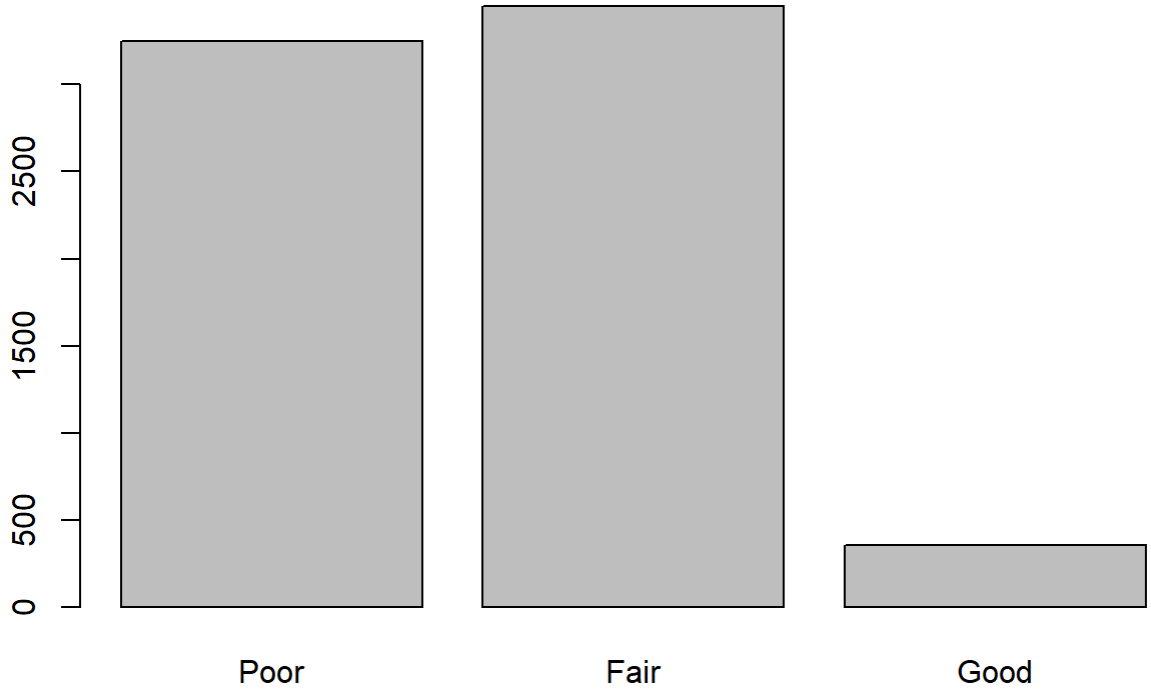
```
##  Poor: 31               Min.   :-1.3441          Min.   :-1.82286
##  Fair:153               1st Qu.:-0.6025          1st Qu.:-0.74247
##  Good: 14               Median :-0.2585          Median :-0.05318
##                         Mean   : 0.0000          Mean   : 0.00000
##                         3rd Qu.: 0.3042          3rd Qu.: 0.51796
##                         Max.   : 5.0646          Max.   : 4.45332
##  Portfolio.Manager.no.    Latitude         Longitude
##  Min.   :-0.8995       Min.   :-1.0944   Min.   :-1.4012
##  1st Qu.:-0.8995       1st Qu.:-0.9028   1st Qu.:-0.9342
##  Median :-0.5976       Median :-0.3891   Median : 0.7940
##  Mean   : 0.0000       Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.5154       3rd Qu.: 0.7341   3rd Qu.: 0.8121
##  Max.   : 1.5154       Max.   : 1.9675   Max.   : 1.6077
##  Construction.Year   Occupant.Density   Vacancy.Rate
##  Min.   :-4.183267   Min.   :-2.1132   Min.   :-0.8323
##  1st Qu.:-0.345052   1st Qu.:-0.7143   1st Qu.:-0.8323
##  Median : 0.009245   Median :-0.1129   Median :-0.3515
##  Mean   : 0.000000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.481641   3rd Qu.: 0.4928   3rd Qu.: 0.3592
##  Max.   : 1.544531   Max.   : 5.1552   Max.   : 3.3189
##  Weekly.Operating.Hours
##  Min.   :-2.0884
##  1st Qu.:-0.3092
##  Median :-0.2046
##  Mean   : 0.0000
##  3rd Qu.: 0.7374
##  Max.   : 9.0054
```

```
# rUN KNN  algorithm
k<- round(sqrt(nrow(i1.data.train)))
i1.data.pred <- knn(i1.data.train[ ,-1], i1.data.test[ ,-1], i1.data.train[,1], k=k, prob=TRUE
)
#Accuracy

Confusion_Mat = confusionMatrix(data = i1.data.pred, reference = i1.data.test[,1], dnn = c("Pr
ediction", "Reference"))
Confusion_Mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Poor Fair Good
##       Poor    2    0    0
##       Fair    5   49    4
##       Good    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.85
##                  95% CI : (0.7343, 0.929)
##     No Information Rate : 0.8167
##     P-Value [Acc > NIR] : 0.3182
```

```
##
##                   Kappa : 0.2742
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Poor Class: Fair Class: Good
## Sensitivity              0.28571      1.0000     0.00000
## Specificity              1.00000      0.1818     1.00000
## Pos Pred Value           1.00000      0.8448         NaN
## Neg Pred Value           0.91379      1.0000     0.93333
## Prevalence               0.11667      0.8167     0.06667
## Detection Rate           0.03333      0.8167     0.00000
## Detection Prevalence     0.03333      0.9667     0.00000
## Balanced Accuracy        0.64286      0.5909     0.50000
```

```
out<-capture.output(Confusion_Mat)
setwd("C:/Users/carle/Documents/Results/KNN Results")
cat(out,file="i1 KNN Confusion Matrix",sep="/n",append=TRUE)

####i1 plot###
#Seperate labels from train set
a <- i1.data$x.2010.Actual.E_C.Energy #`   #<- Customize Variables
b <- i1.data$x.2010.Actual.Thermal #<- Customize Variables
aname<-"2010 E & C Energy"
bname<-"2010 Thermal Energy"
cl <- i1.data$x.2010.Qualitative.EUI
train <- cbind(a,b)
#Populate Grid with points

test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```

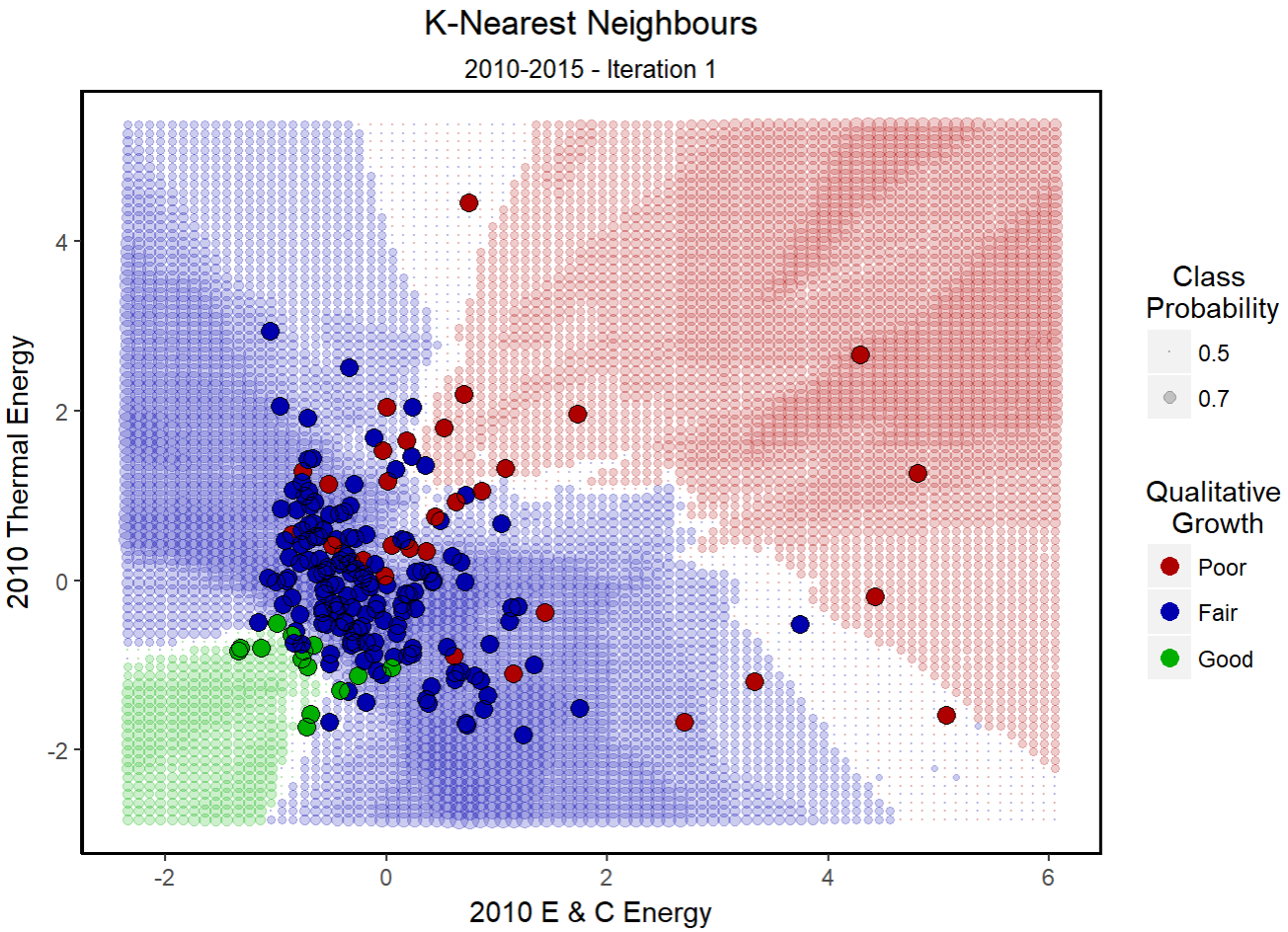



```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i1.dataf <- bind_rows(mutate(test,
                         prob=prob,
                         cls="Good",
                         prob_cls=ifelse(classif==cls,
                                         1, 0)),
                  mutate(test,
                         prob=prob,
                         cls="Fair",
                         prob_cls=ifelse(classif==cls,
                                         1, 0)),
                  mutate(test,
                         prob=prob,
                         cls="Poor",
                         prob_cls=ifelse(classif==cls,
                                         1, 0)))
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.3
```

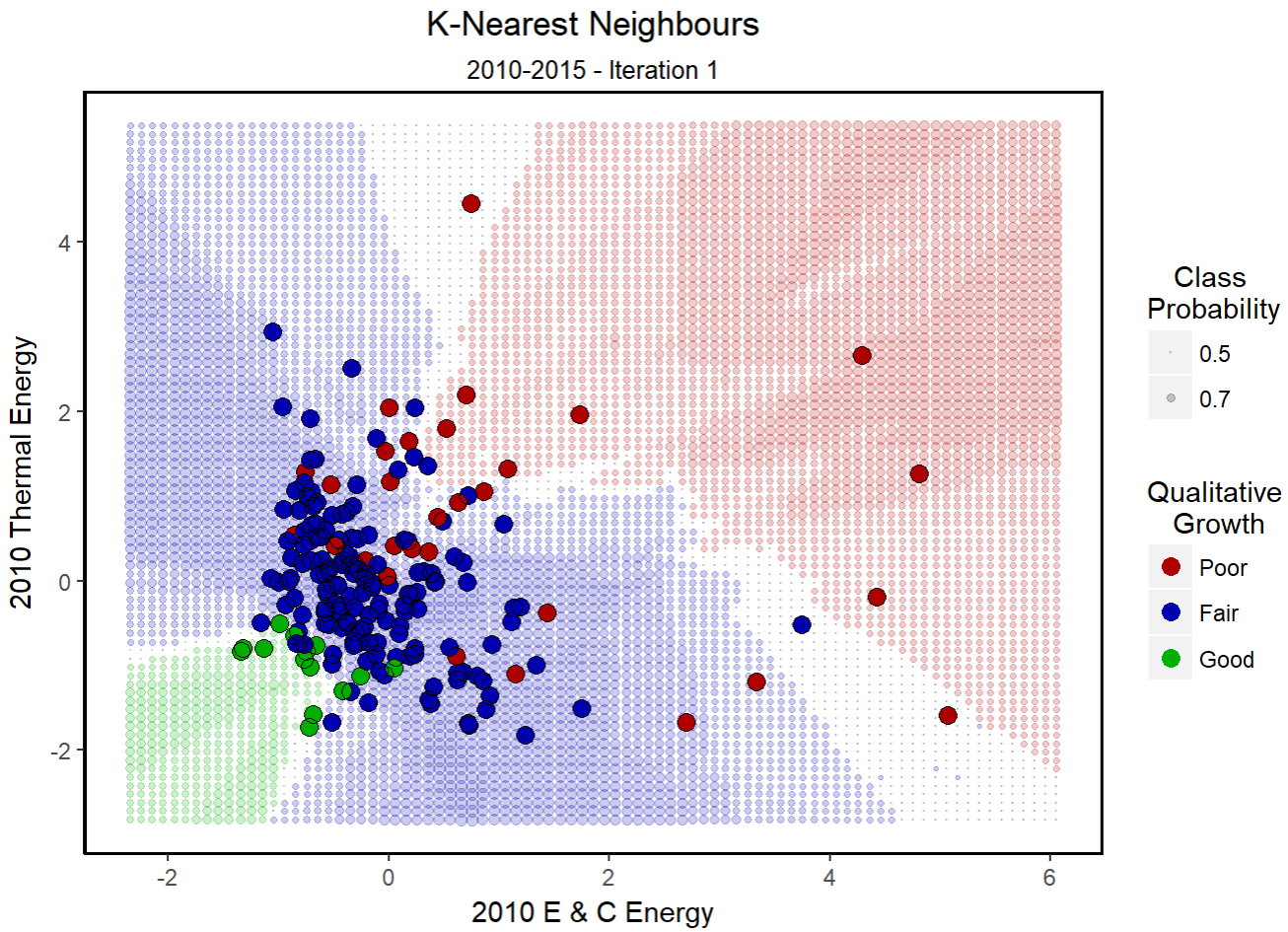
```
i1.dataf$cls =ordered(i1.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i1.dataf) +
```

```
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
            data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 1",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
        fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
           size = guide_legend(order = 1))
```



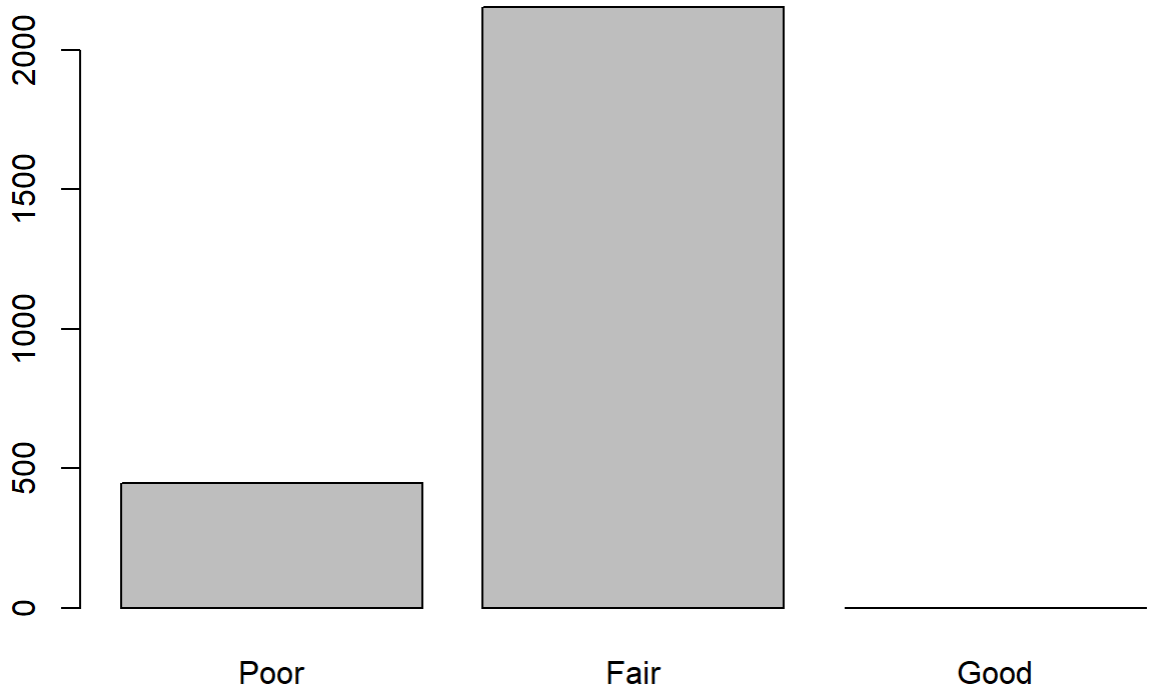

```
#Plot Class Probability for colour blind people
ggplot(i1.dataf) +
```

```
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 2), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 1",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```



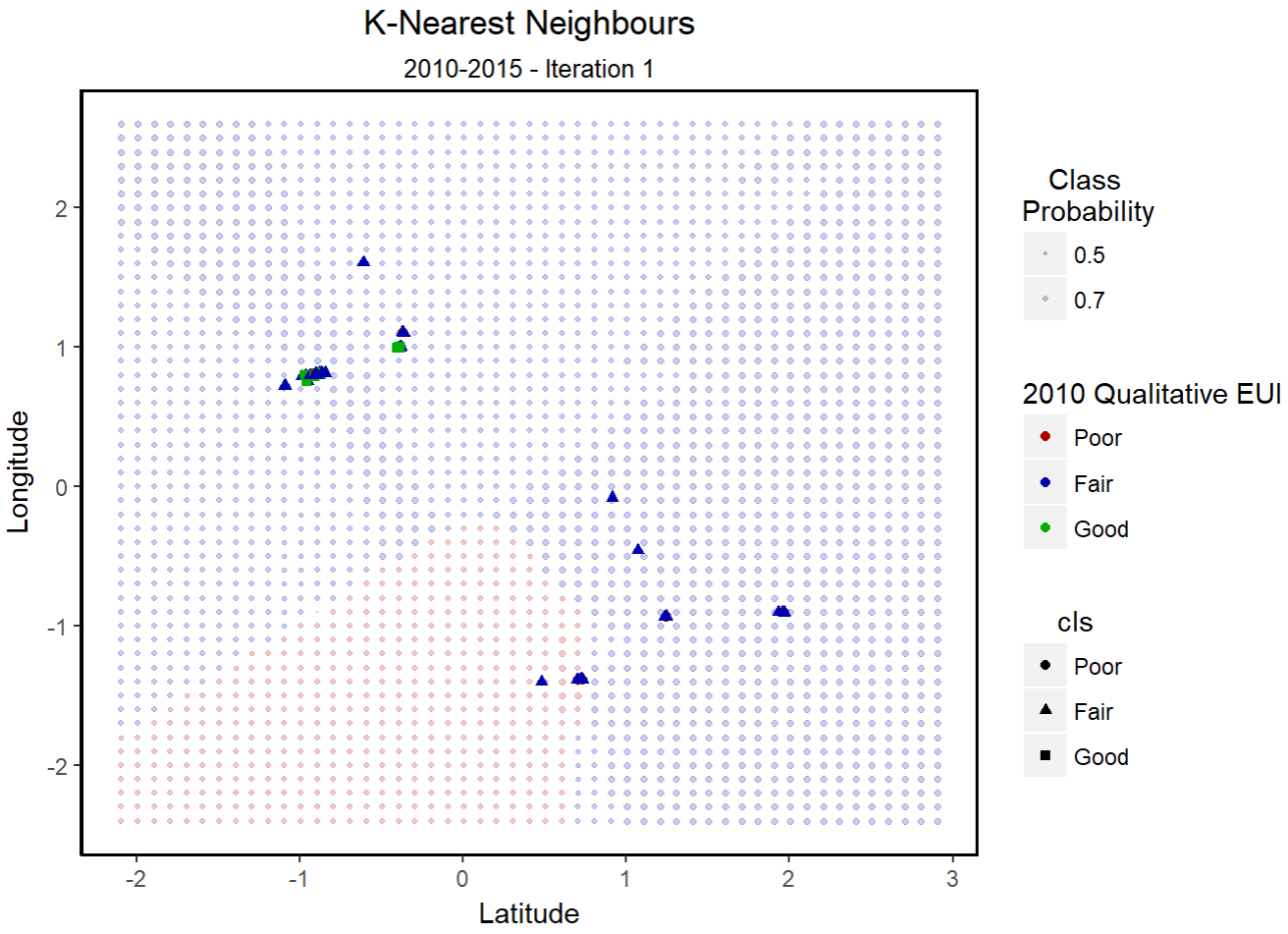

```
####v1.2 plot ####
#Seperate labels from train set
```

```
a <- i1.data$Latitude #`    #<- Customize Variables
b <- i1.data$Longitude #<- Customize Variables
aname<-"Latitude"
bname<-"Longitude"
cl <- i1.data$x.2010.Qualitative.EUI
train <- cbind(a,b)

#Populate Grid with points
test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))

#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```

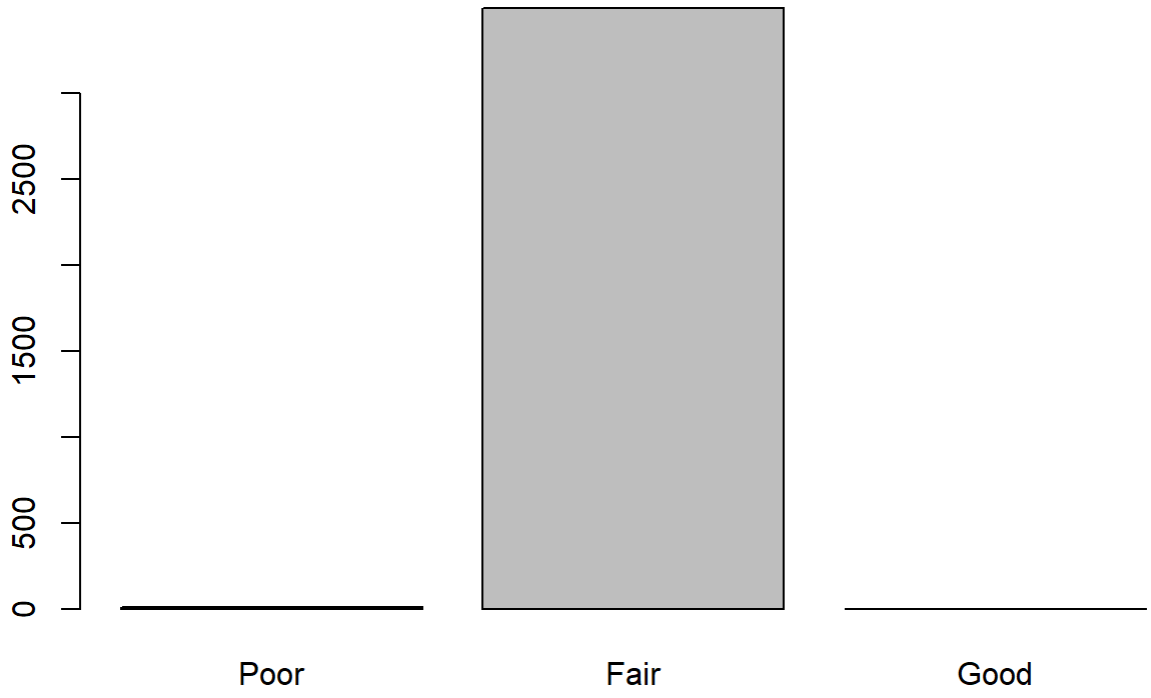



```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i1.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
```

```
                        mutate(test,
                               prob=prob,
                               cls="Fair",
                               prob_cls=ifelse(classif==cls,
                                               1, 0)),
                        mutate(test,
                               prob=prob,
                               cls="Poor",
                               prob_cls=ifelse(classif==cls,
                                               1, 0)))
i1.dataf$cls =ordered(i1.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability - Colour Blind
ggplot(i1.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 1), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls, shape=cls),
             size=1.5, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
 # geom_point(aes(x=x, y=y, shape=cls), #Black circle around point
 #             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 1",
       colour = "2010 Qualitative EUI")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```

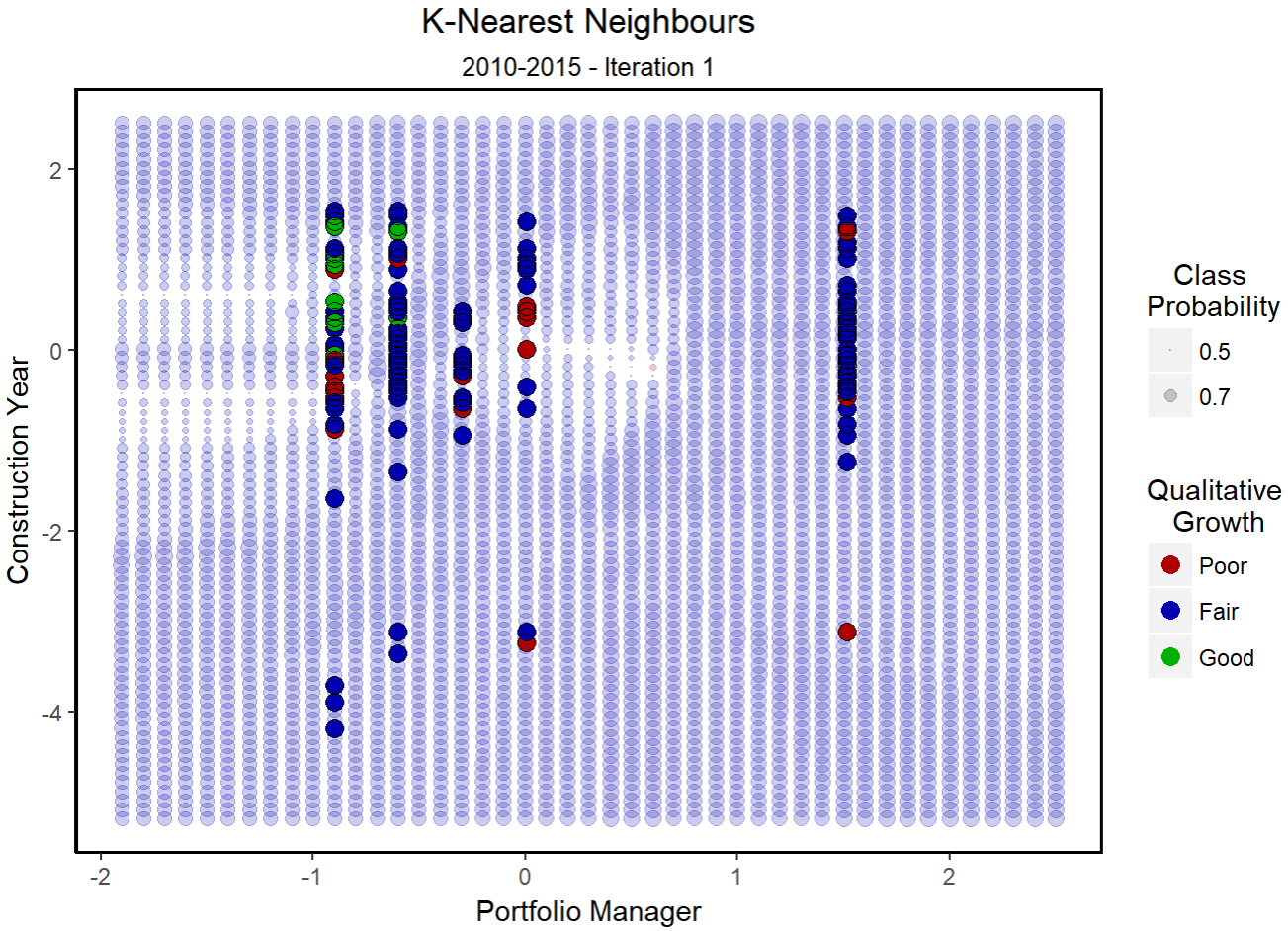



```
####v1.3 plot ####
#Seperate labels from train set
a <- i1.data$Portfolio.Manager.no. #`    #<- Customize Variables
b <- i1.data$Construction.Year #<- Customize Variables
aname<-"Portfolio Manager"
bname<-"Construction Year"
cl <- i1.data$x.2010.Qualitative.EUI
train <- cbind(a,b)
#Populate Grid with points

test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```



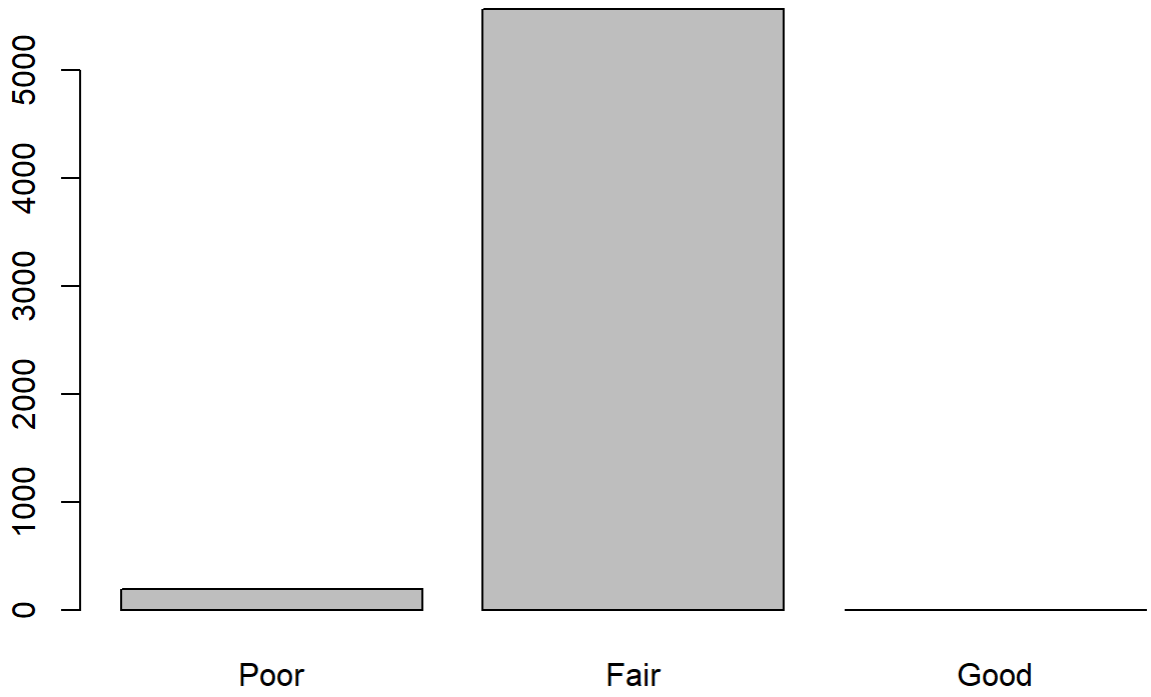

```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i1.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Poor",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)))
i1.dataf$cls =ordered(i1.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i1.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
```

```
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 1",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```



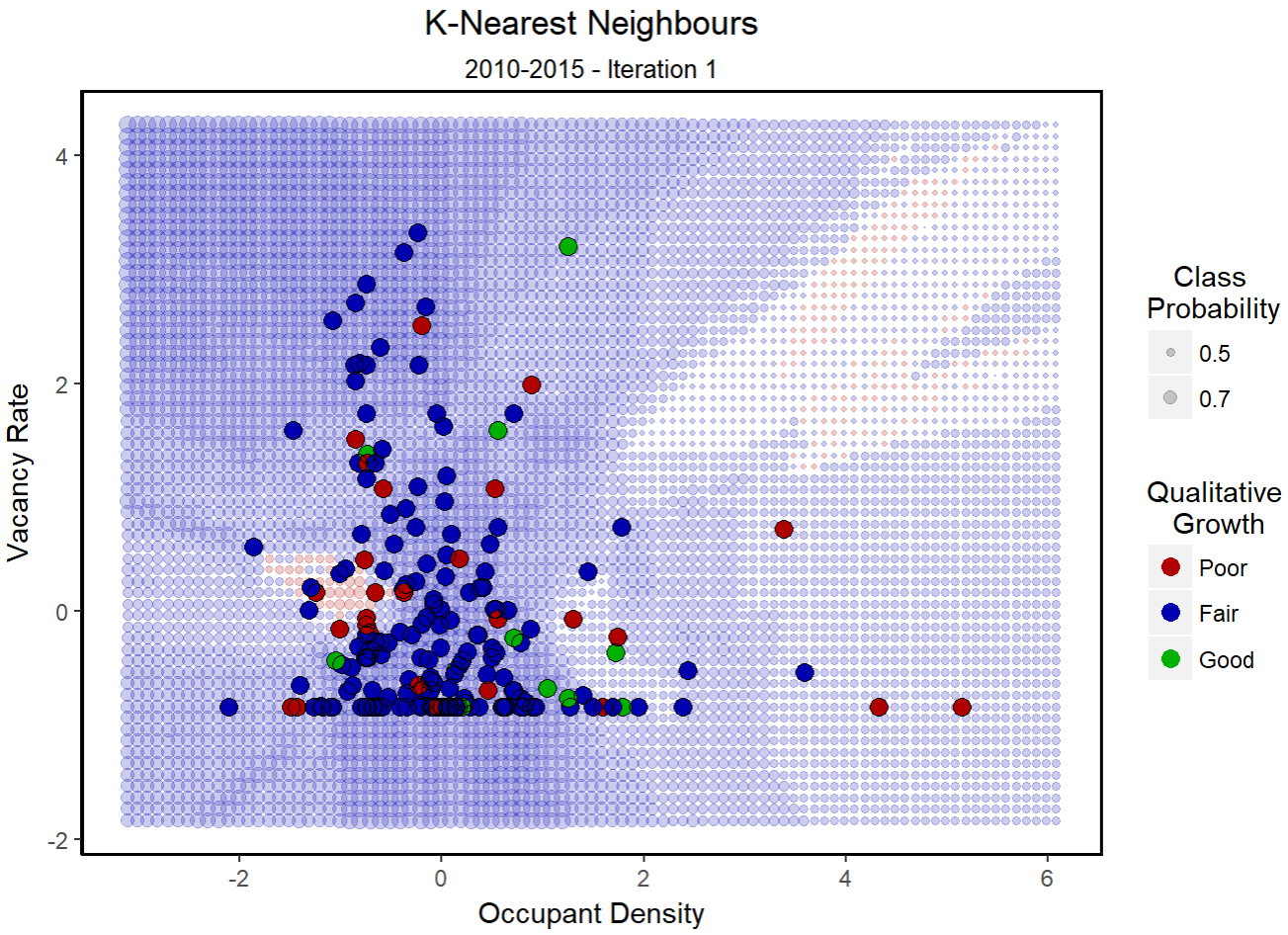

```
#### v1.4 plot ####
#Seperate labels from train set
a <- i1.data$Occupant.Density #<- Customize Variables
b <- i1.data$Vacancy.Rate #<- Customize Variables
aname<-"Occupant Density"
bname<-"Vacancy Rate"
```

```
cl <- i1.data$x.2010.Qualitative.EUI
train <- cbind(a,b)
#Populate Grid with points

test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```

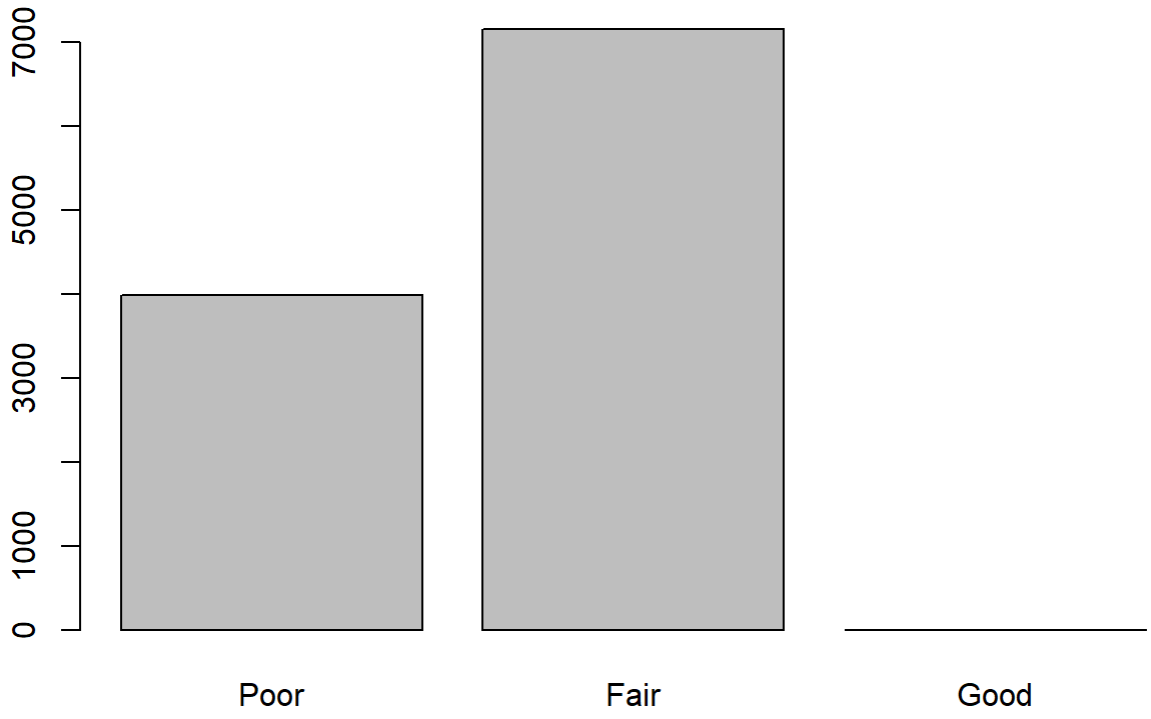



```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i1.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
```

```
                        mutate(test,
                               prob=prob,
                               cls="Poor",
                               prob_cls=ifelse(classif==cls,
                                               1, 0)))
i1.dataf$cls =ordered(i1.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i1.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 1",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```

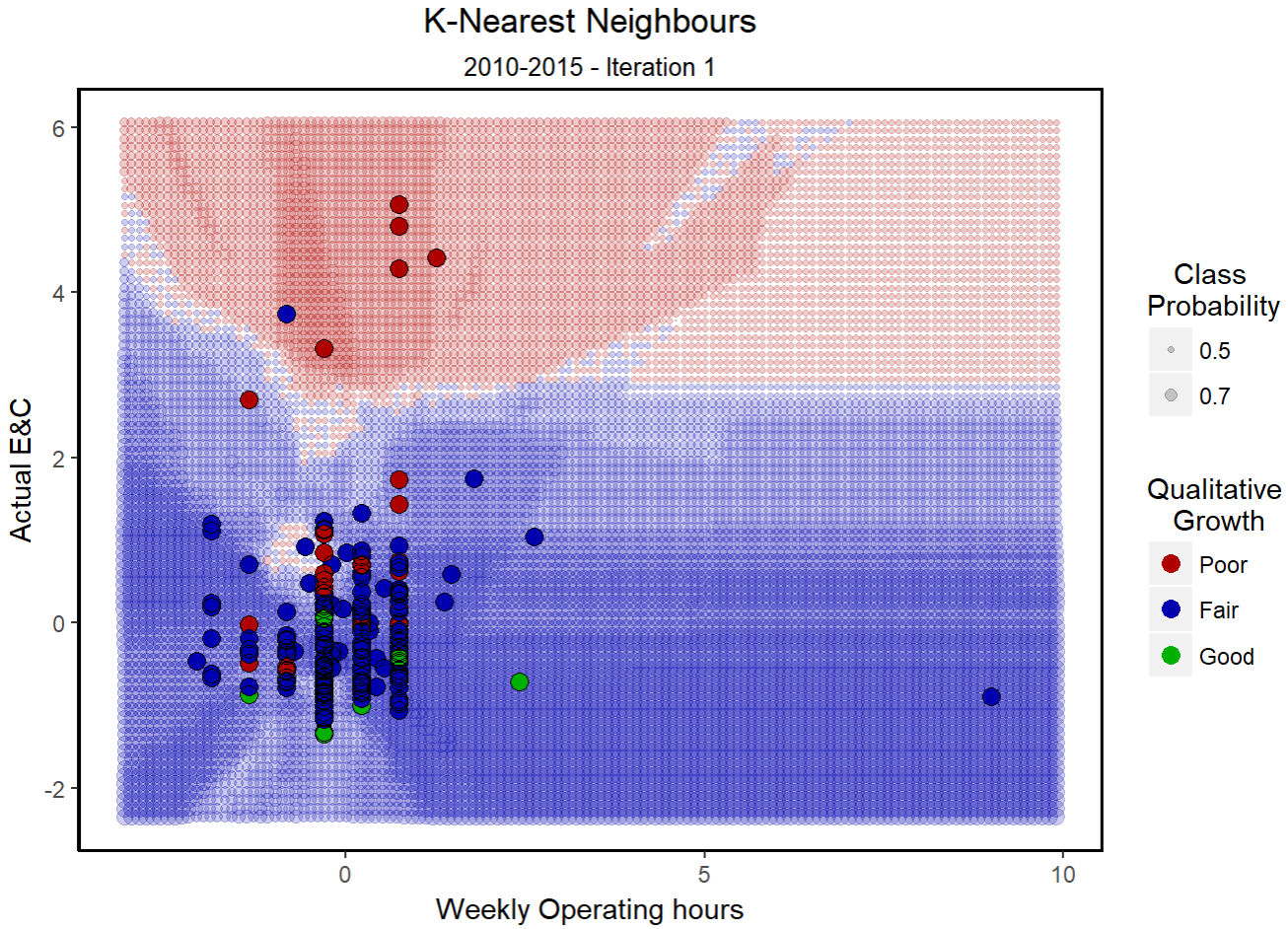



```
#### v1.5 plot ####
#Seperate labels from train set
a <- i1.data$Weekly.Operating.Hours #<- Customize Variables
b <- i1.data$x.2010.Actual.E_C.Energy #<- Customize Variables
aname<-"Weekly Operating hours"
bname<-"Actual E&C"
cl <- i1.data$x.2010.Qualitative.EUI
train <- cbind(a,b)
#Populate Grid with points

test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```



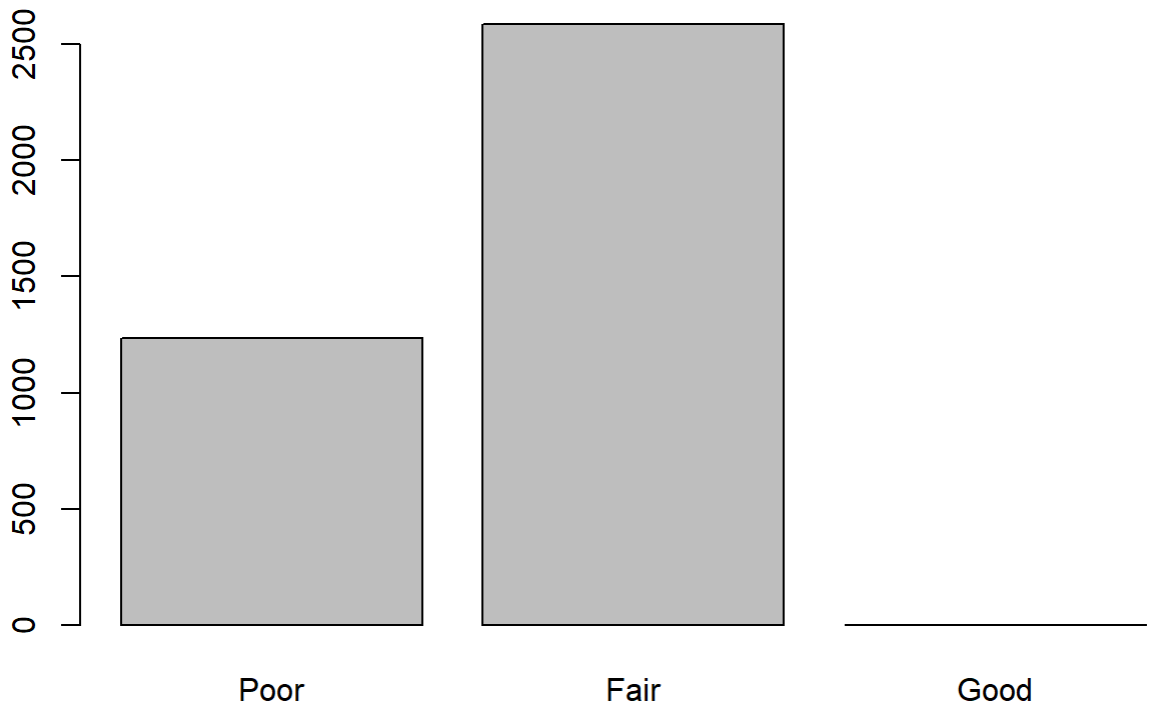

```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i1.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Poor",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)))
i1.dataf$cls =ordered(i1.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i1.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
```

```
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 1",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```



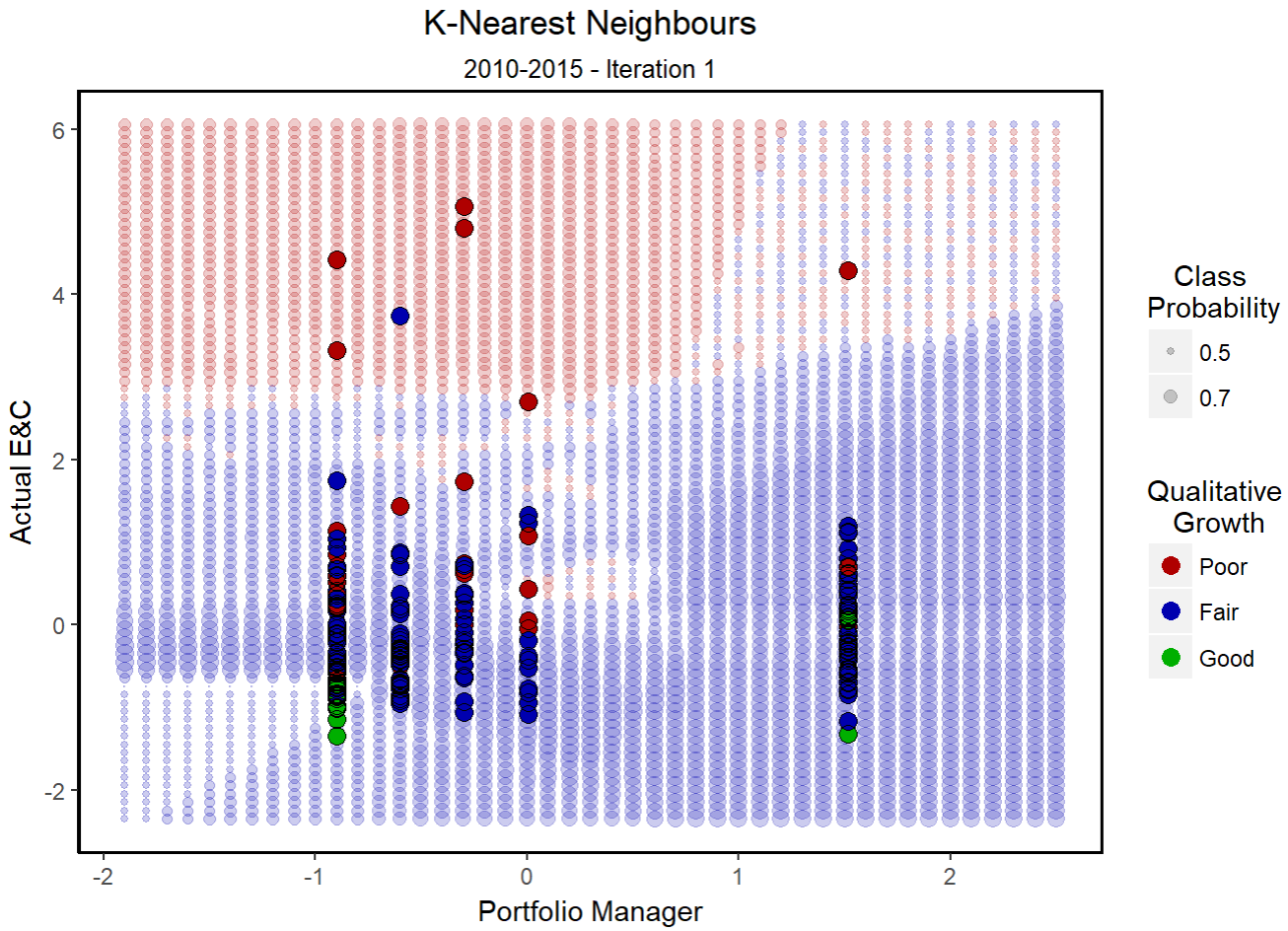

```
#### v1.6 plot ####
#Seperate labels from train set
a <- i1.data$Portfolio.Manager.no. #<- Customize Variables
b <- i1.data$x.2010.Actual.E_C.Energy #<- Customize Variables
aname<-"Portfolio Manager"
bname<-"Actual E&C"
```

```
cl <- i1.data$x.2010.Qualitative.EUI
train <- cbind(a,b)
#Populate Grid with points

test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```

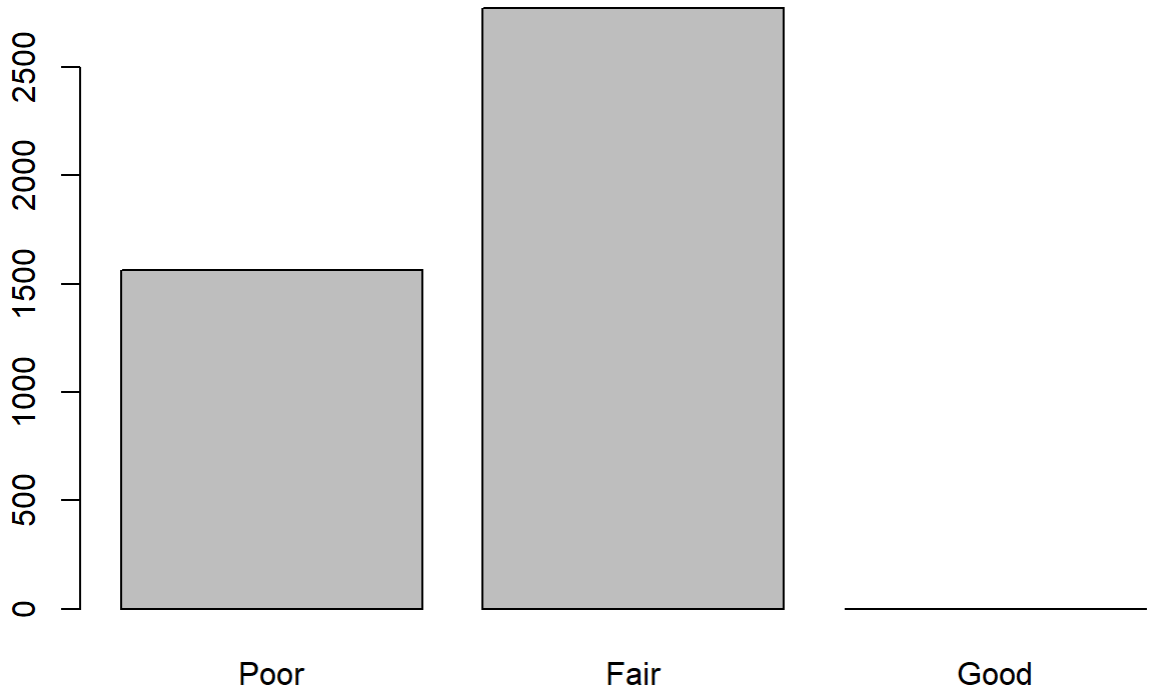



```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i1.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
```

```
                        mutate(test,
                               prob=prob,
                               cls="Poor",
                               prob_cls=ifelse(classif==cls,
                                               1, 0)))
i1.dataf$cls =ordered(i1.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i1.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 1",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```

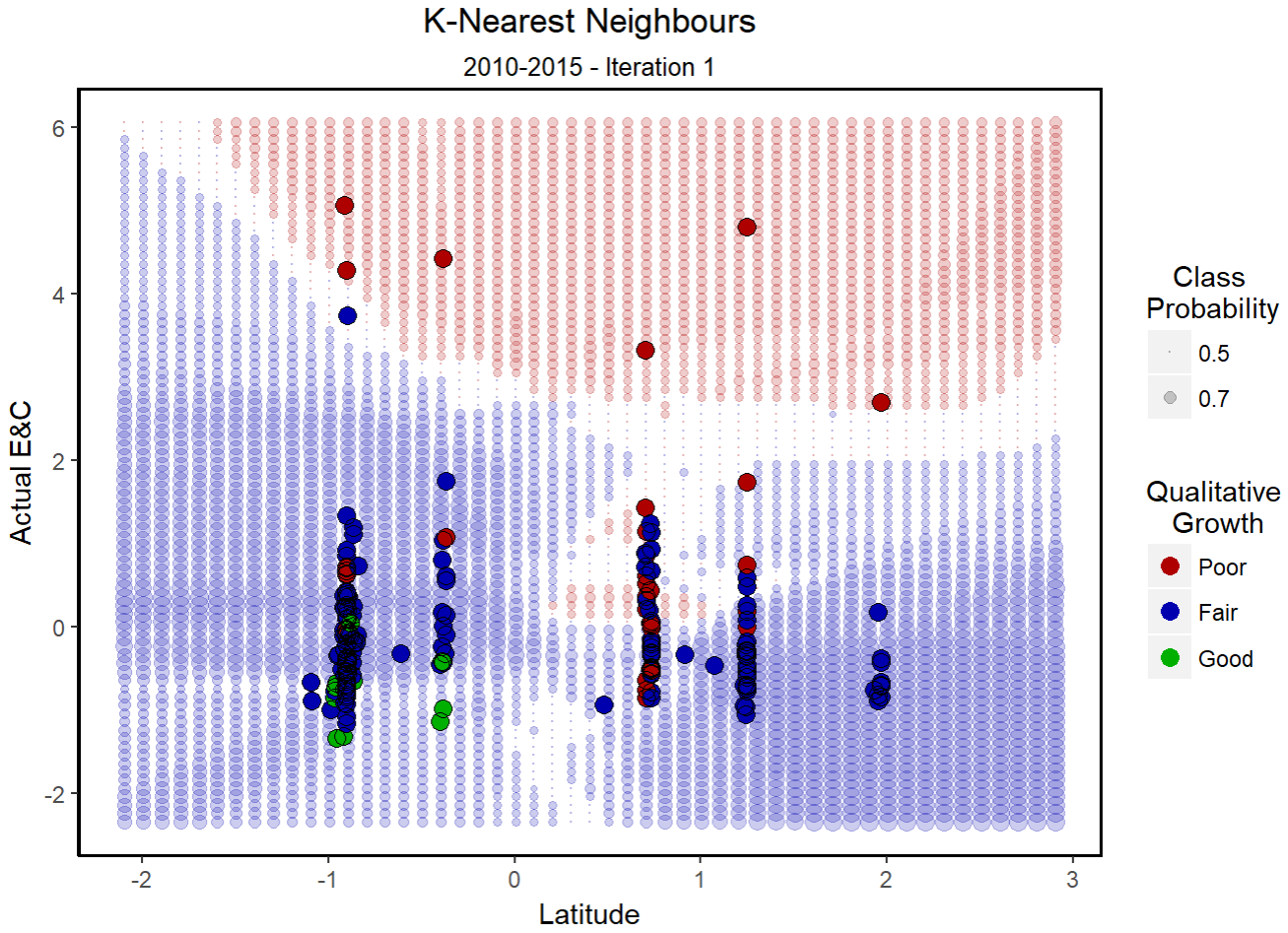



```
#### v1.7 plot ####
#Seperate labels from train set
a <- i1.data$Latitude #<- Customize Variables
b <- i1.data$x.2010.Actual.E_C.Energy #<- Customize Variables
aname<-"Latitude"
bname<-"Actual E&C"
cl <- i1.data$x.2010.Qualitative.EUI
train <- cbind(a,b)
#Populate Grid with points

test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```



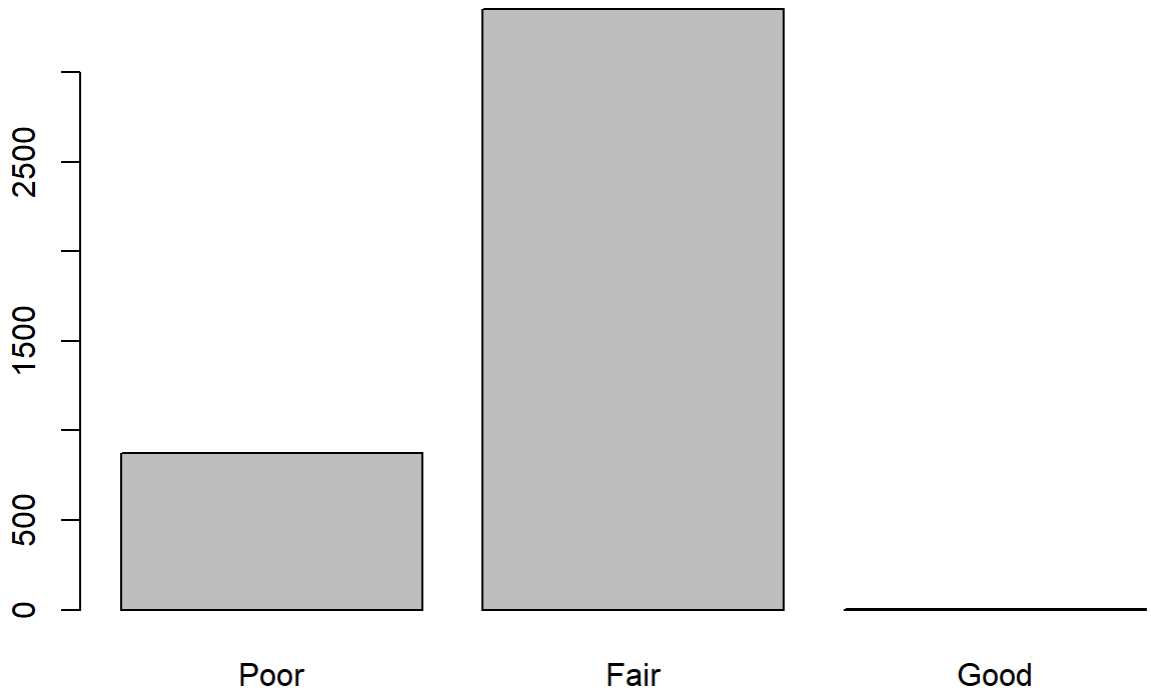

```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i1.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Poor",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)))
i1.dataf$cls =ordered(i1.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i1.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
```

```
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 1",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```



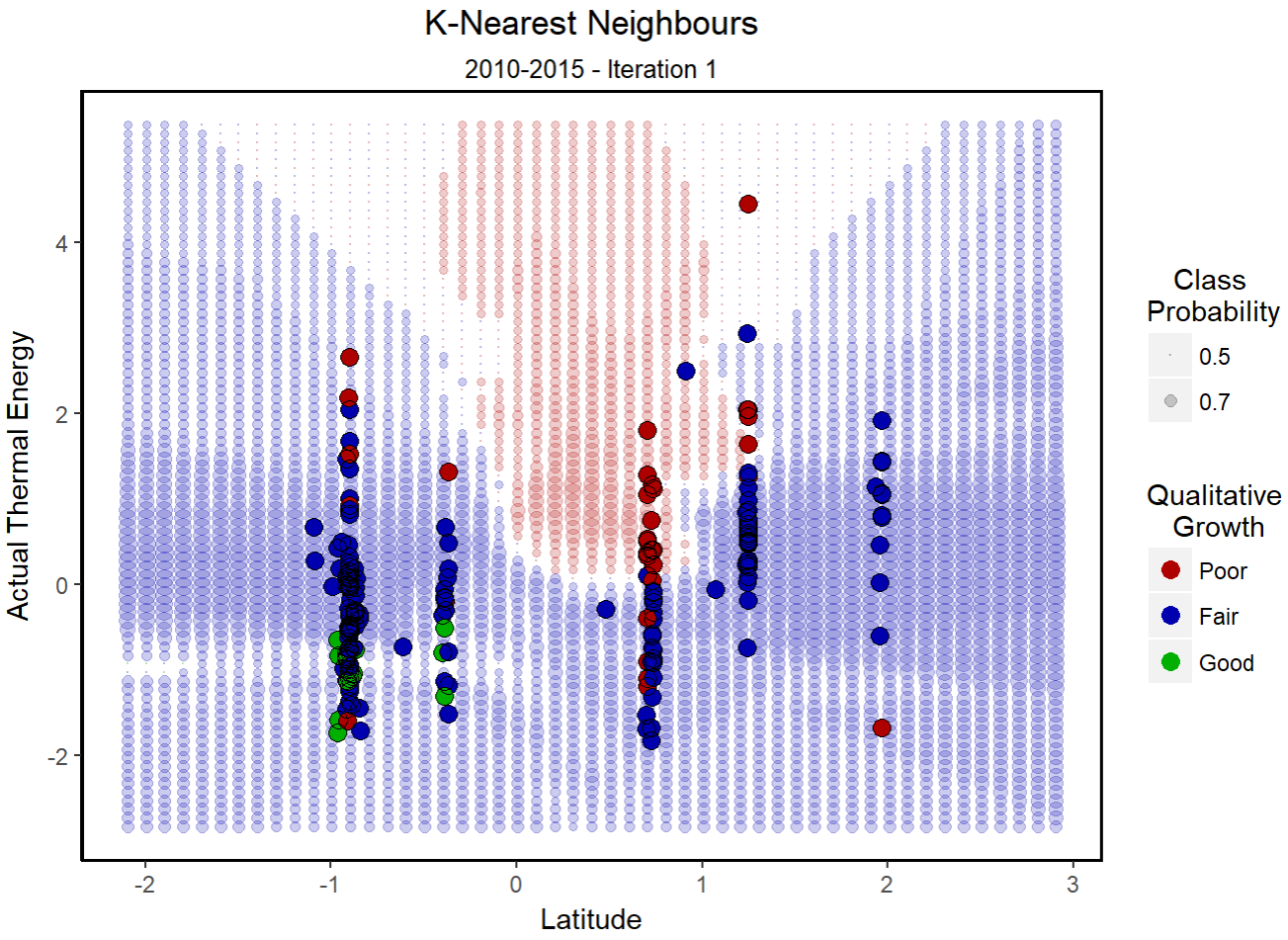

```
#### v1.8 plot ####
#Seperate labels from train set
a <- i1.data$Latitude #<- Customize Variables
b <- i1.data$x.2010.Actual.Thermal #<- Customize Variables
aname<-"Latitude"
bname<-"Actual Thermal Energy"
```

```
cl <- i1.data$x.2010.Qualitative.EUI
train <- cbind(a,b)
#Populate Grid with points

test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```




```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i1.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
```

```
                        mutate(test,
                               prob=prob,
                               cls="Poor",
                               prob_cls=ifelse(classif==cls,
                                               1, 0)))
i1.dataf$cls =ordered(i1.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i1.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 1",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```




```
################################
####    KNN iteration 2   ####
################################

i2.data<-data[,c("x.2015.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
                 "Portfolio.Manager.no.", "Latitude", "Longitude",    "Construction.Year",
  "Occupant.Density",
                 "Vacancy.Rate",  "Weekly.Operating.Hours")]
cnames<-c("x.2015.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
          "Portfolio.Manager.no.",  "Latitude", "Longitude",    "Construction.Year",    "Occup
ant.Density",
          "Vacancy.Rate",   "Weekly.Operating.Hours")
# Normalize
QG<-i2.data$x.2015.Qualitative.EUI
norm.data<- as.data.frame(lapply(i2.data[,-1], scale))
i2.data <-cbind(QG, norm.data)
row.names(i2.data) <- rnames
colnames(i2.data) <-cnames
##create Train and Test set
train.rows = sample(1:nrow(i2.data), (nrow(i2.data)*0.7))
i2.data.train = i2.data[train.rows,]
i2.data.test = i2.data[-train.rows,]
#View(i2.data.test)
nrow(i2.data.test)
```

```
## [1] 60
```

```
# Normalization Check
summary(i2.data)
```

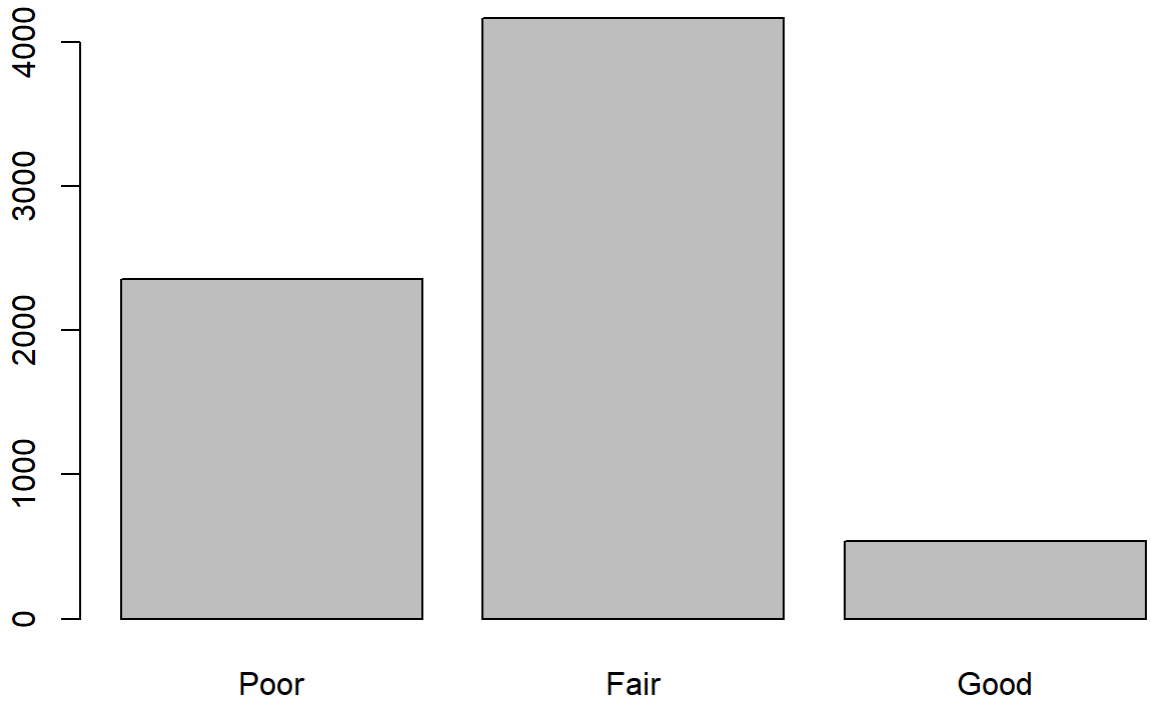
```
##  x.2015.Qualitative.EUI x.2010.Actual.E_C.Energy x.2010.Actual.Thermal
##  Poor: 27               Min.   :-1.3441          Min.   :-1.82286
##  Fair:125               1st Qu.:-0.6025          1st Qu.:-0.74247
##  Good: 46               Median :-0.2585          Median :-0.05318
##                         Mean   : 0.0000          Mean   : 0.00000
##                         3rd Qu.: 0.3042          3rd Qu.: 0.51796
##                         Max.   : 5.0646          Max.   : 4.45332
##  Portfolio.Manager.no.    Latitude         Longitude
##  Min.   :-0.8995       Min.   :-1.0944   Min.   :-1.4012
##  1st Qu.:-0.8995       1st Qu.:-0.9028   1st Qu.:-0.9342
##  Median :-0.5976       Median :-0.3891   Median : 0.7940
##  Mean   : 0.0000       Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.5154       3rd Qu.: 0.7341   3rd Qu.: 0.8121
##  Max.   : 1.5154       Max.   : 1.9675   Max.   : 1.6077
##  Construction.Year   Occupant.Density   Vacancy.Rate
##  Min.   :-4.183267   Min.   :-2.1132   Min.   :-0.8323
##  1st Qu.:-0.345052   1st Qu.:-0.7143   1st Qu.:-0.8323
##  Median : 0.009245   Median :-0.1129   Median :-0.3515
##  Mean   : 0.000000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.481641   3rd Qu.: 0.4928   3rd Qu.: 0.3592
##  Max.   : 1.544531   Max.   : 5.1552   Max.   : 3.3189
##  Weekly.Operating.Hours
##  Min.   :-2.0884
##  1st Qu.:-0.3092
##  Median :-0.2046
##  Mean   : 0.0000
##  3rd Qu.: 0.7374
##  Max.   : 9.0054
```

```
# rUN KNN  algorithm
k<- round(sqrt(nrow(i2.data.train)))
i2.data.pred <- knn(i2.data.train[ ,-1], i2.data.test[ ,-1], i2.data.train[,1], k=k, prob=TRUE
)
#Accuracy

Confusion_Mat = confusionMatrix(data = i2.data.pred, reference = i2.data.test[,1], dnn = c("Pr
ediction", "Reference"))
Confusion_Mat
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction Poor Fair Good
##       Poor    2    2    0
##       Fair    6   28    8
##       Good    0   10    4
```

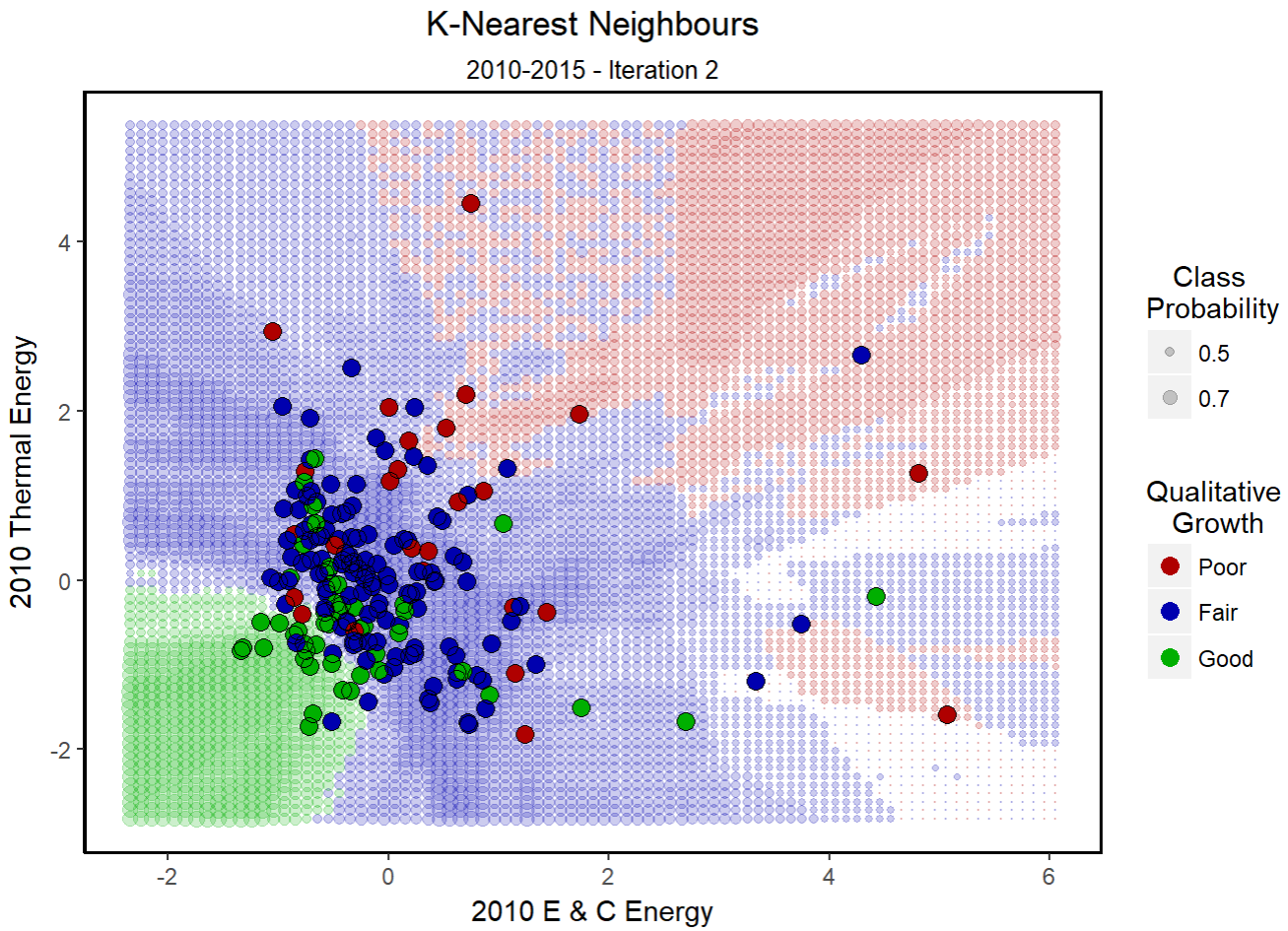
```
##
## Overall Statistics
##
##                Accuracy : 0.5667
##                  95% CI : (0.4324, 0.6941)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.9603
##
##                   Kappa : 0.093
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Poor Class: Fair Class: Good
## Sensitivity              0.25000      0.7000     0.33333
## Specificity              0.96154      0.3000     0.79167
## Pos Pred Value           0.50000      0.6667     0.28571
## Neg Pred Value           0.89286      0.3333     0.82609
## Prevalence               0.13333      0.6667     0.20000
## Detection Rate           0.03333      0.4667     0.06667
## Detection Prevalence     0.06667      0.7000     0.23333
## Balanced Accuracy        0.60577      0.5000     0.56250
```

```
out<-capture.output(Confusion_Mat)
setwd("C:/Users/carle/Documents/Results/KNN Results")
cat(out,file="i2 KNN Confusion Matrix",sep="/n",append=TRUE)

#Seperate labels from train set
a <- i2.data$x.2010.Actual.E_C.Energy #`   #<- Customize Variables
b <- i2.data$x.2010.Actual.Thermal #<- Customize Variables
aname<-"2010 E & C Energy"
bname<-"2010 Thermal Energy"
cl <- i2.data$x.2015.Qualitative.EUI
train <- cbind(a,b)
#Populate Grid with points
test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```



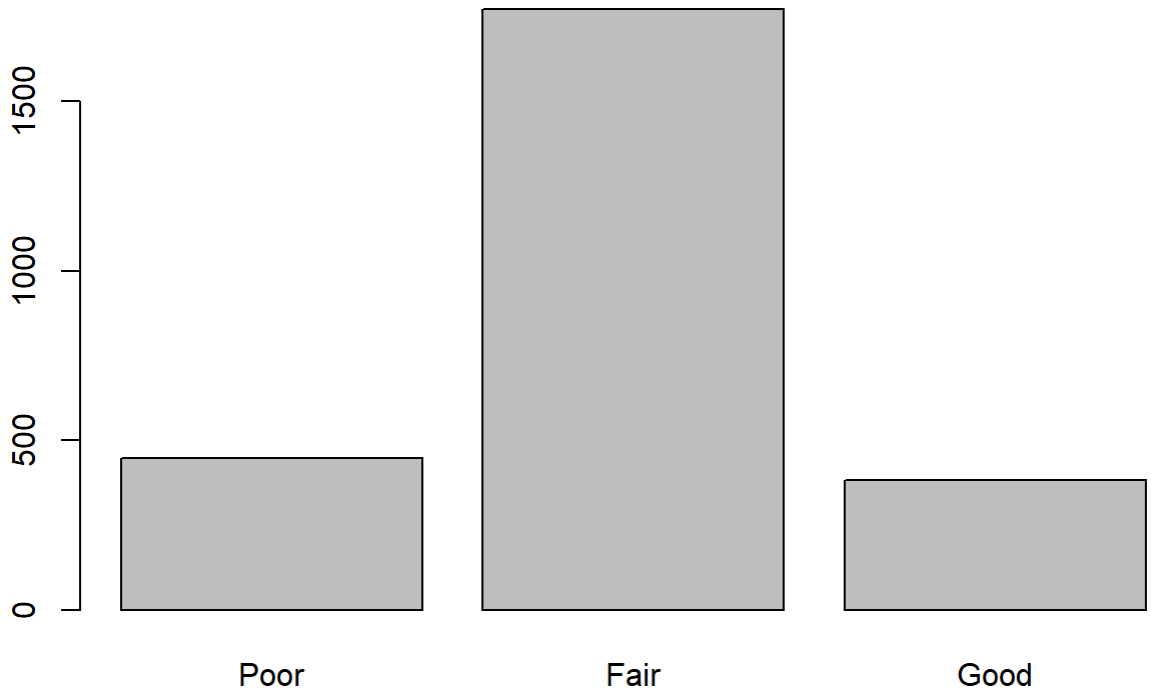

```
prob <- attr(classif, "prob")
#Data Structure for Plotting
i2.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Poor",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)))
i2.dataf$cls =ordered(i2.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Decision Boundaries


#Plot Class Probability

ggplot(i2.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
```

```
              data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 2",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```



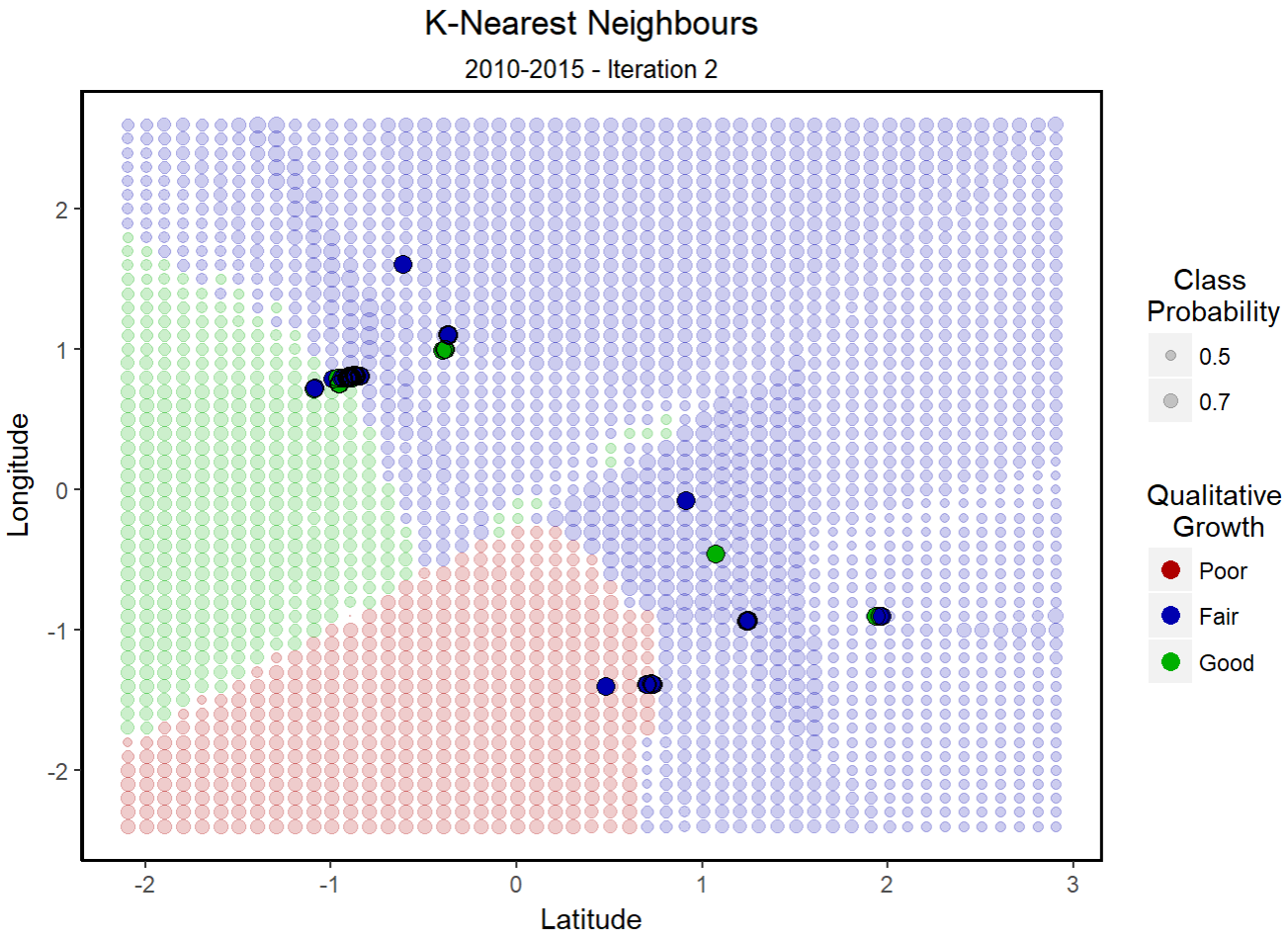

```
####v2.2 plot ####
#Seperate labels from train set
a <- i2.data$Latitude #`   #<- Customize Variables
```

```
b <- i2.data$Longitude #<- Customize Variables
aname<-"Latitude"
bname<-"Longitude"
cl <- i2.data$x.2015.Qualitative.EUI
train <- cbind(a,b)

#Populate Grid with points
test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```

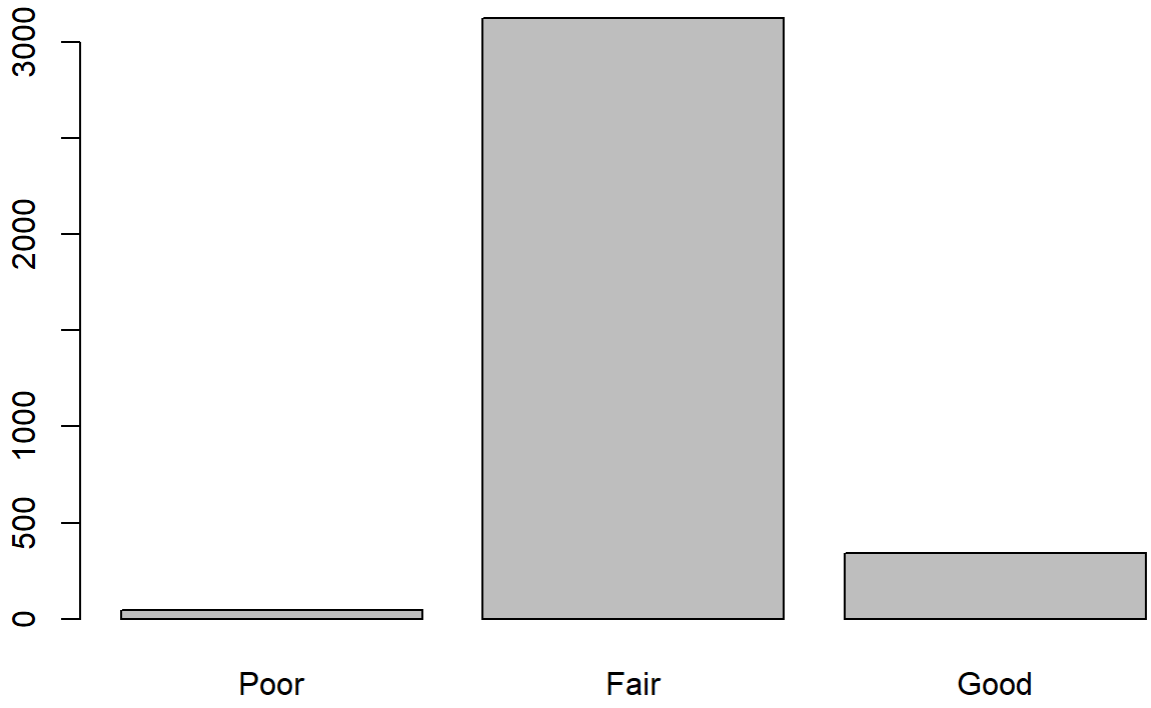



```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i2.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
```

```
                                cls="Fair",
                                prob_cls=ifelse(classif==cls,
                                                1, 0)),
                          mutate(test,
                                prob=prob,
                                cls="Poor",
                                prob_cls=ifelse(classif==cls,
                                                1, 0)))
i2.dataf$cls =ordered(i2.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i2.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 2",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```

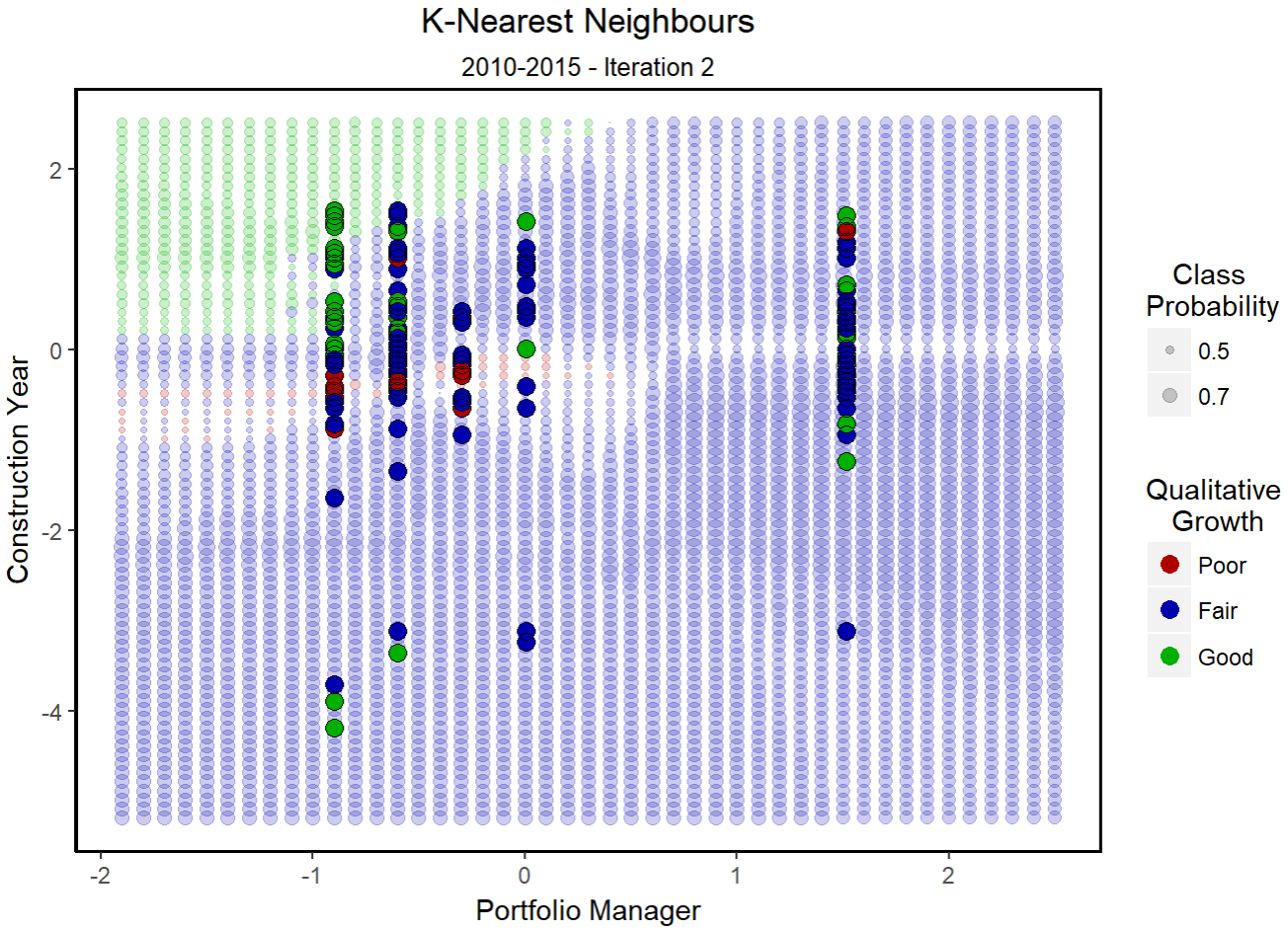



```
####v2.3 plot ####
#Seperate labels from train set
a <- i2.data$Portfolio.Manager.no. #`    #<- Customize Variables
b <- i2.data$Construction.Year #<- Customize Variables
aname<-"Portfolio Manager"
bname<-"Construction Year"
cl <- i2.data$x.2015.Qualitative.EUI
train <- cbind(a,b)
#Populate Grid with points

test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```



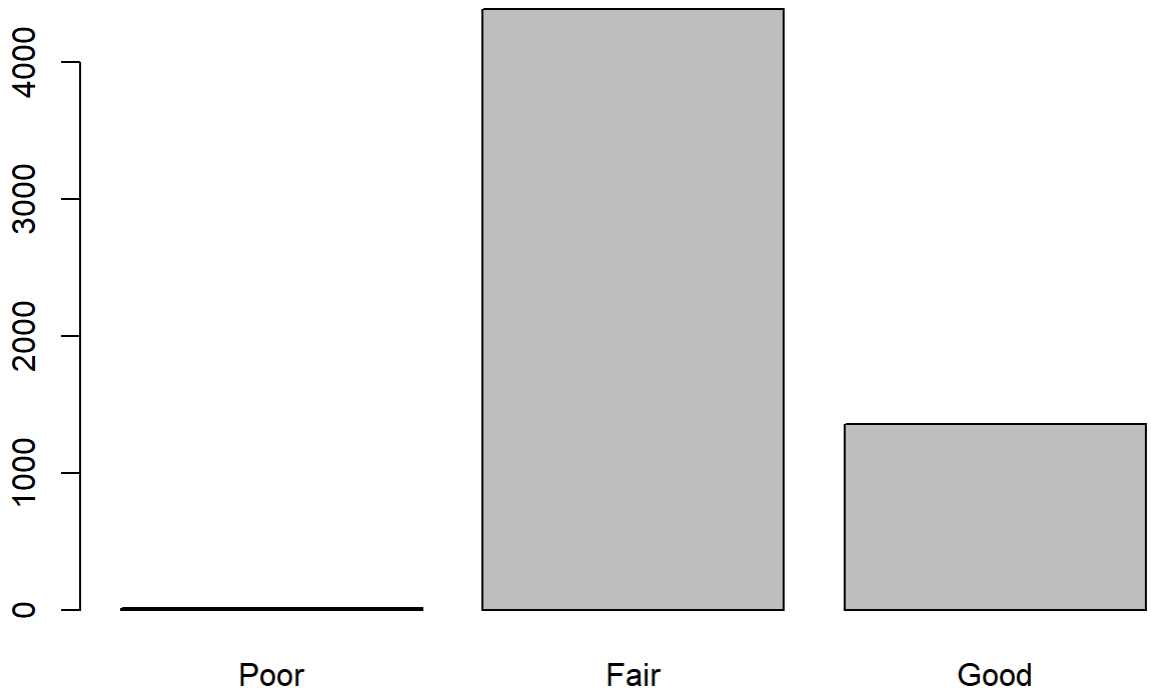

```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i2.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Poor",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)))
i2.dataf$cls =ordered(i2.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i2.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
```

```
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 2",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```



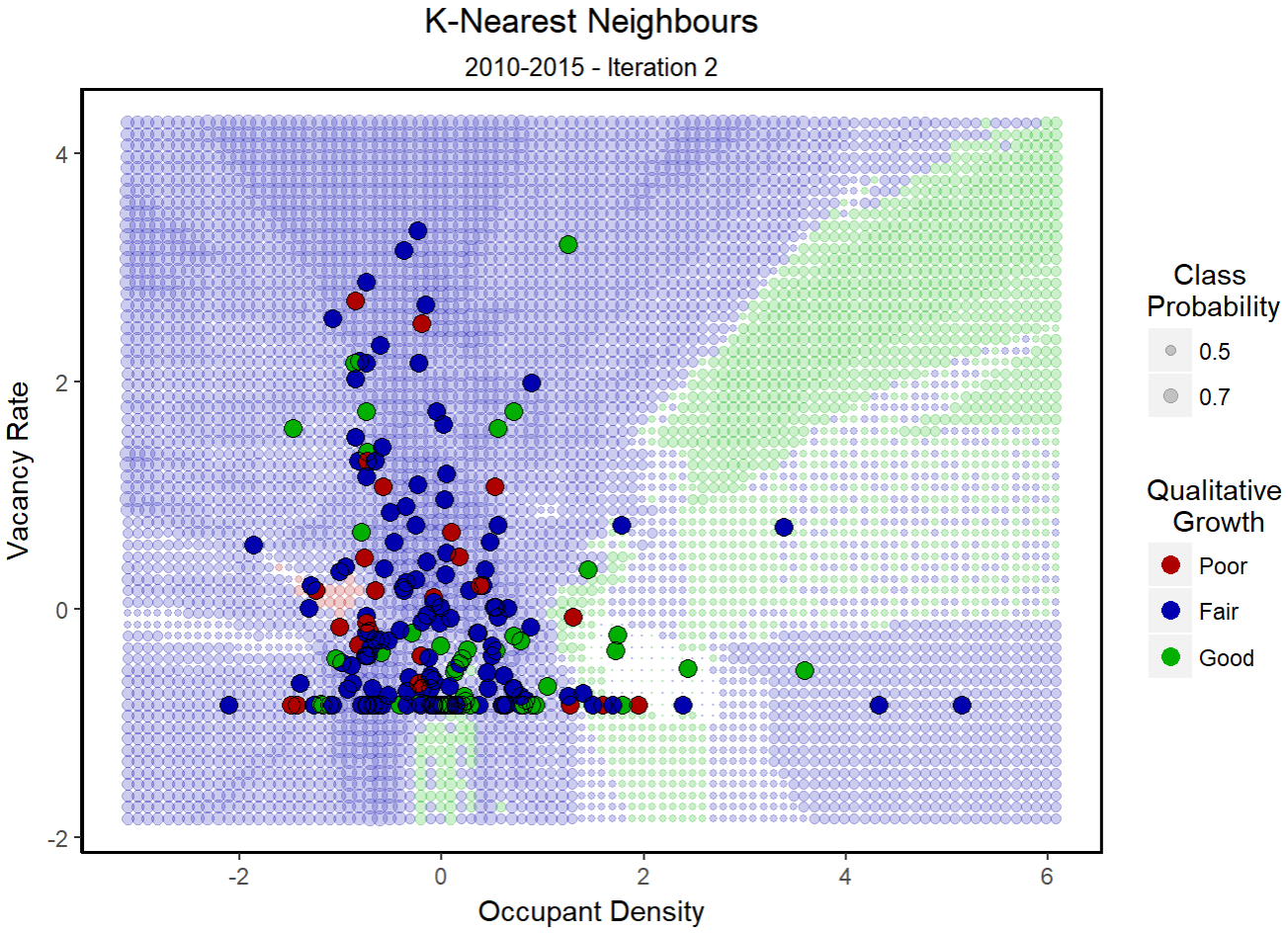

```
#### v2.4 plot ####
#Seperate labels from train set
a <- i2.data$Occupant.Density #<- Customize Variables
b <- i2.data$Vacancy.Rate #<- Customize Variables
aname<-"Occupant Density"
bname<-"Vacancy Rate"
```

```
cl <- i2.data$x.2015.Qualitative.EUI
train <- cbind(a,b)
#Populate Grid with points

test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```

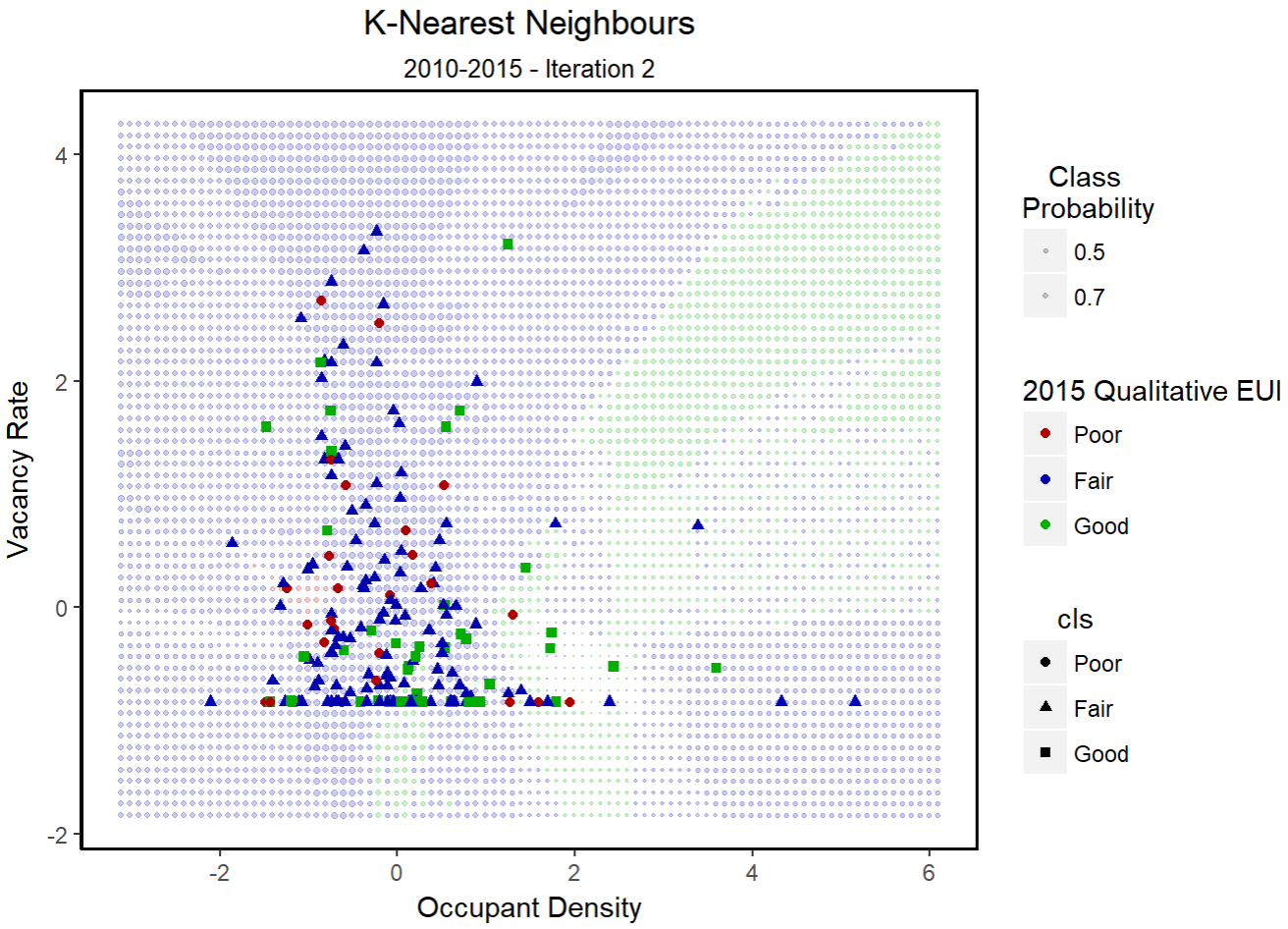



```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i2.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
```

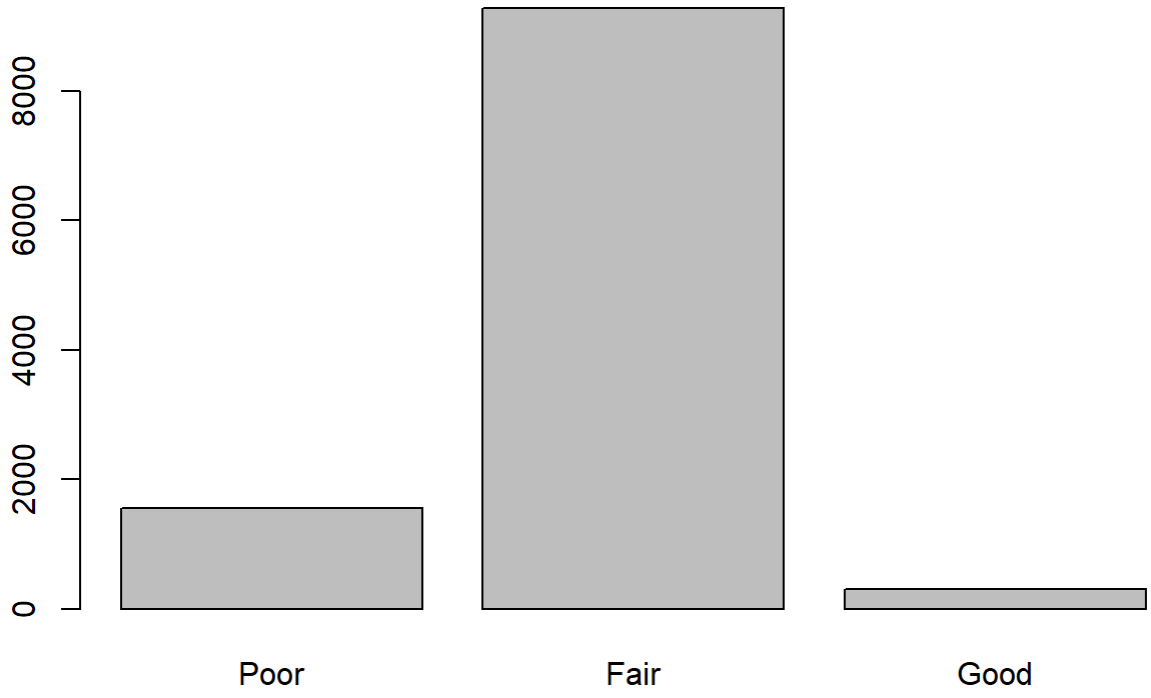
```
                        mutate(test,
                               prob=prob,
                               cls="Poor",
                               prob_cls=ifelse(classif==cls,
                                               1, 0)))
i2.dataf$cls =ordered(i2.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i2.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 2",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```




```
#Plot Class Probability - Colour Blind
ggplot(i1.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 1), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls, shape=cls),
             size=1.5, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  # geom_point(aes(x=x, y=y, shape=cls), #Black circle around point
  #            size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 2",
       colour = "2015 Qualitative EUI")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```

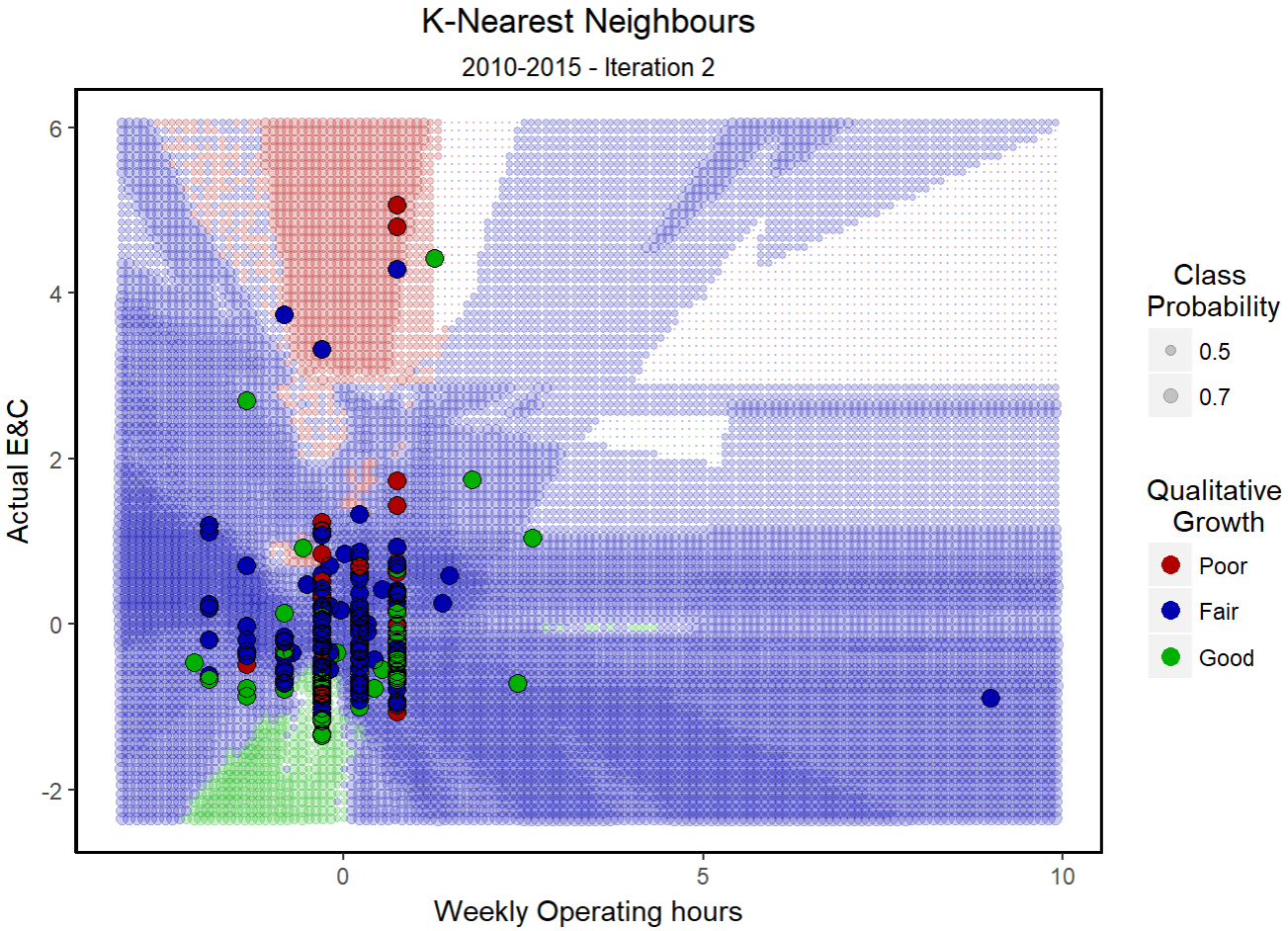



```
#### v2.5 plot ####
#Seperate labels from train set
a <- i2.data$Weekly.Operating.Hours #<- Customize Variables
b <- i2.data$x.2010.Actual.E_C.Energy #<- Customize Variables
aname<-"Weekly Operating hours"
bname<-"Actual E&C"
cl <- i2.data$x.2015.Qualitative.EUI
train <- cbind(a,b)
#Populate Grid with points

test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```



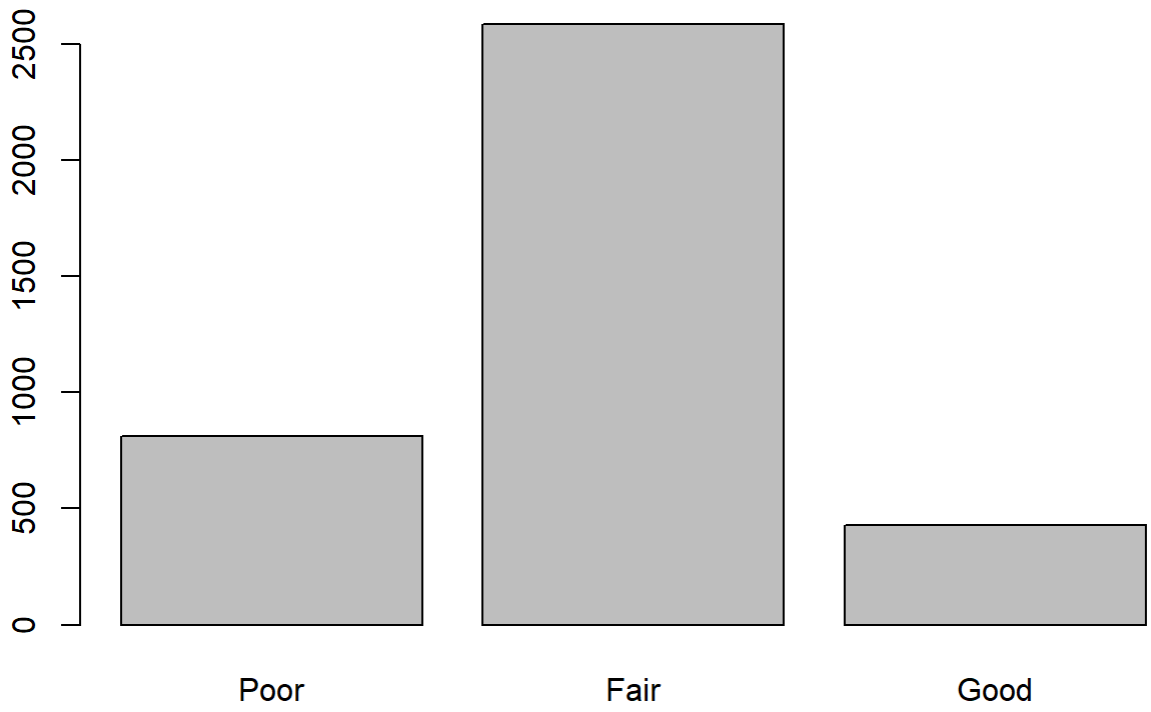

```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i2.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Poor",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)))
i2.dataf$cls =ordered(i2.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i2.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
```

```
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 2",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```



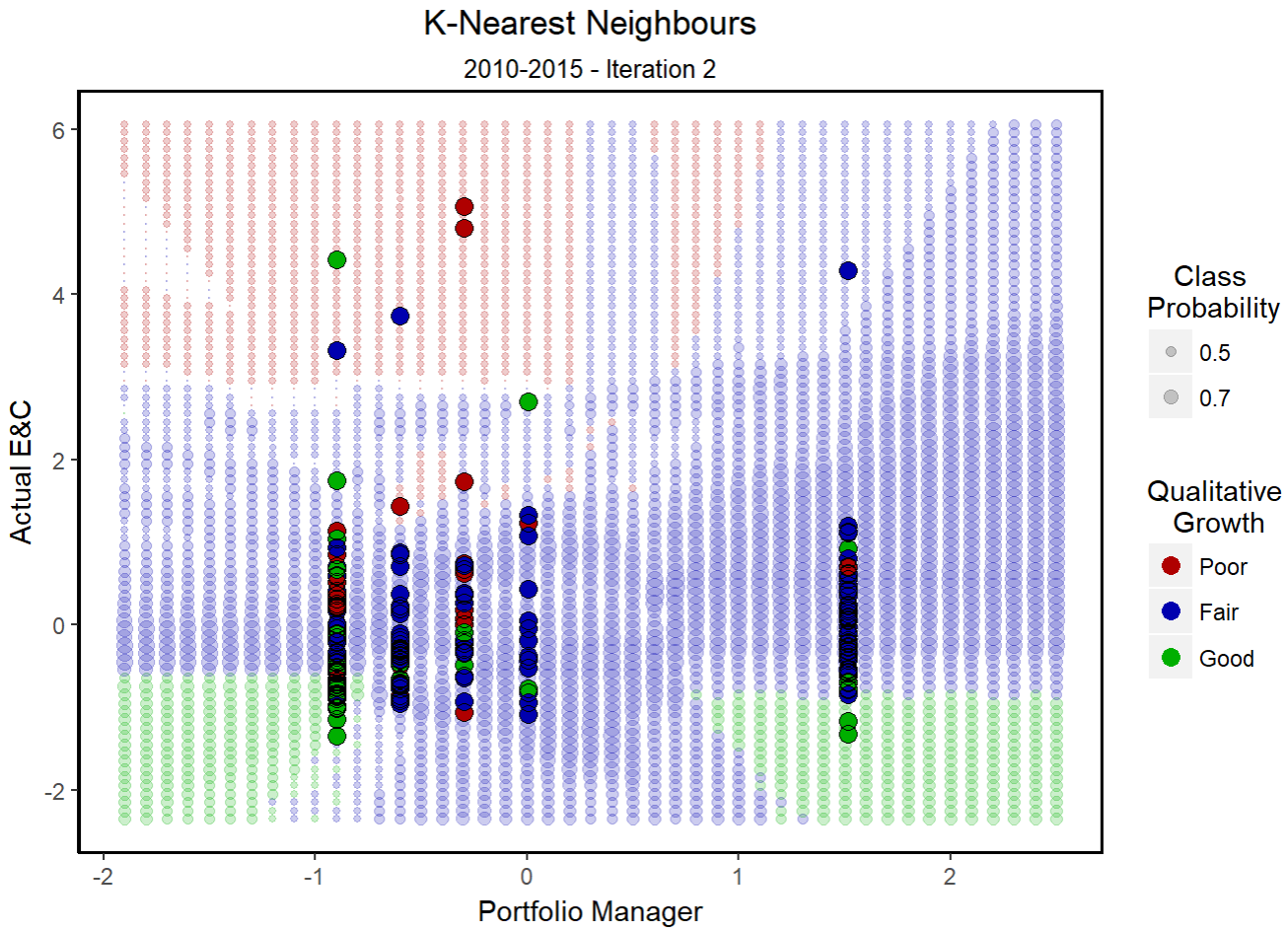

```
#### v2.6 plot ####
#Seperate labels from train set
a <- i2.data$Portfolio.Manager.no. #<- Customize Variables
b <- i2.data$x.2010.Actual.E_C.Energy #<- Customize Variables
aname<-"Portfolio Manager"
bname<-"Actual E&C"
```

```
cl <- i2.data$x.2015.Qualitative.EUI
train <- cbind(a,b)
#Populate Grid with points

test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```

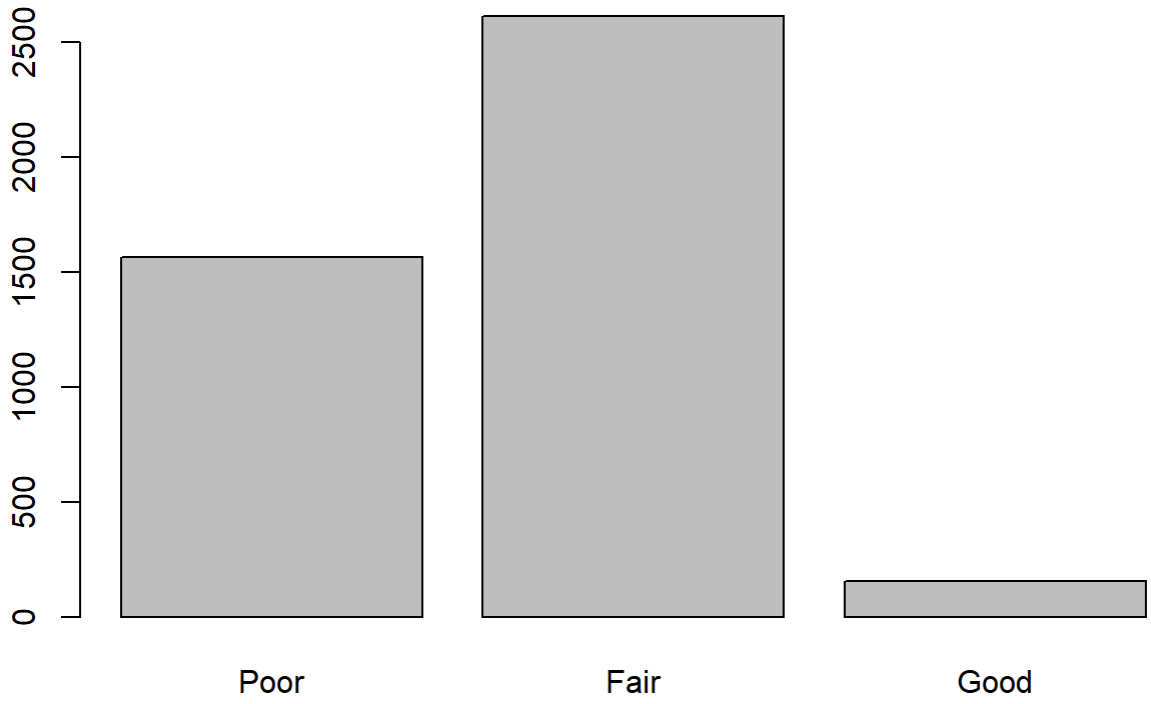



```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i2.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
```

```
                        mutate(test,
                               prob=prob,
                               cls="Poor",
                               prob_cls=ifelse(classif==cls,
                                               1, 0)))
i2.dataf$cls =ordered(i2.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i2.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 2",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```

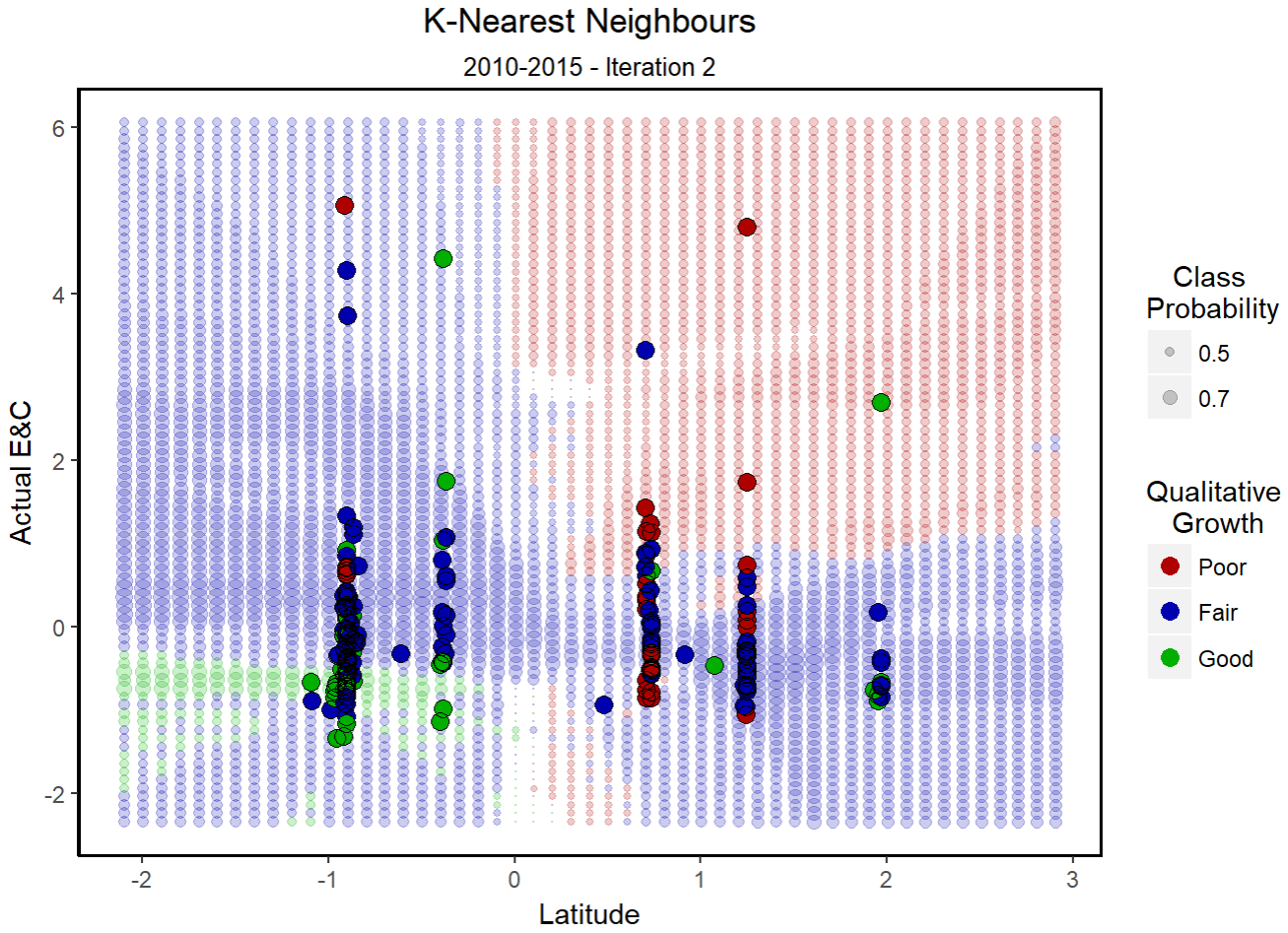



```
#### v2.7 plot ####

#Seperate labels from train set
a <- i2.data$Latitude #<- Customize Variables
b <- i2.data$x.2010.Actual.E_C.Energy #<- Customize Variables
aname<-"Latitude"
bname<-"Actual E&C"
cl <- i2.data$x.2015.Qualitative.EUI
train <- cbind(a,b)

#Populate Grid with points
test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))

#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```



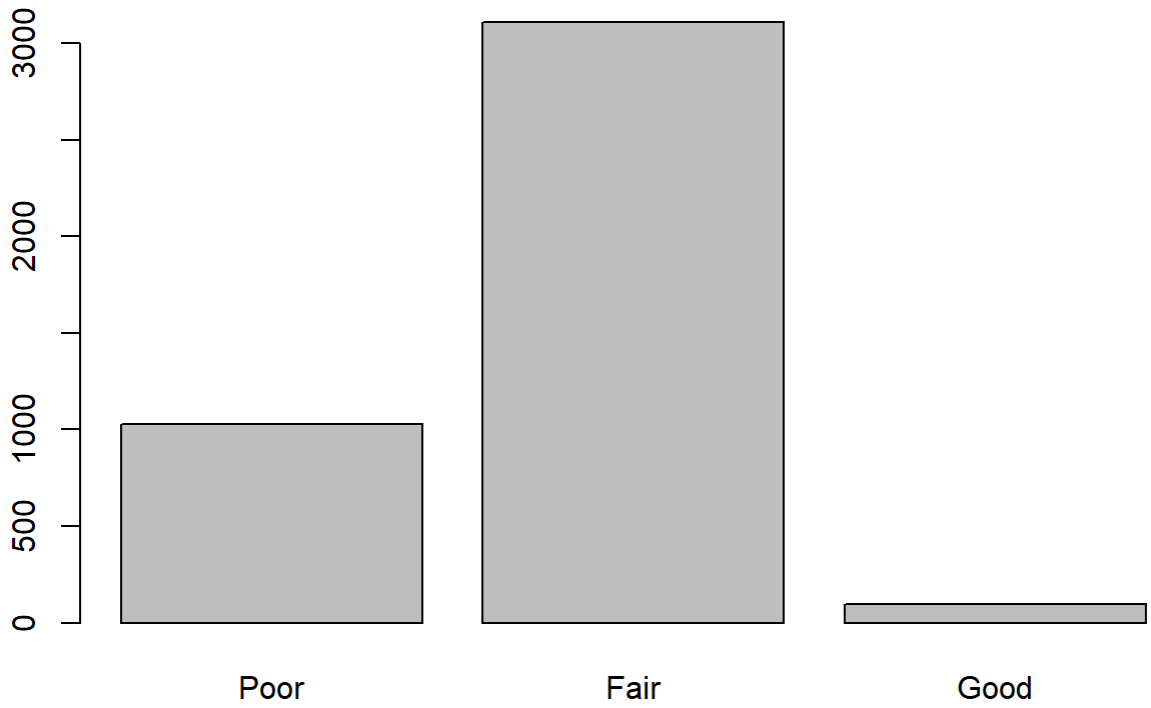

```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i2.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Poor",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)))
i2.dataf$cls =ordered(i2.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i2.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
```

```
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 2",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```



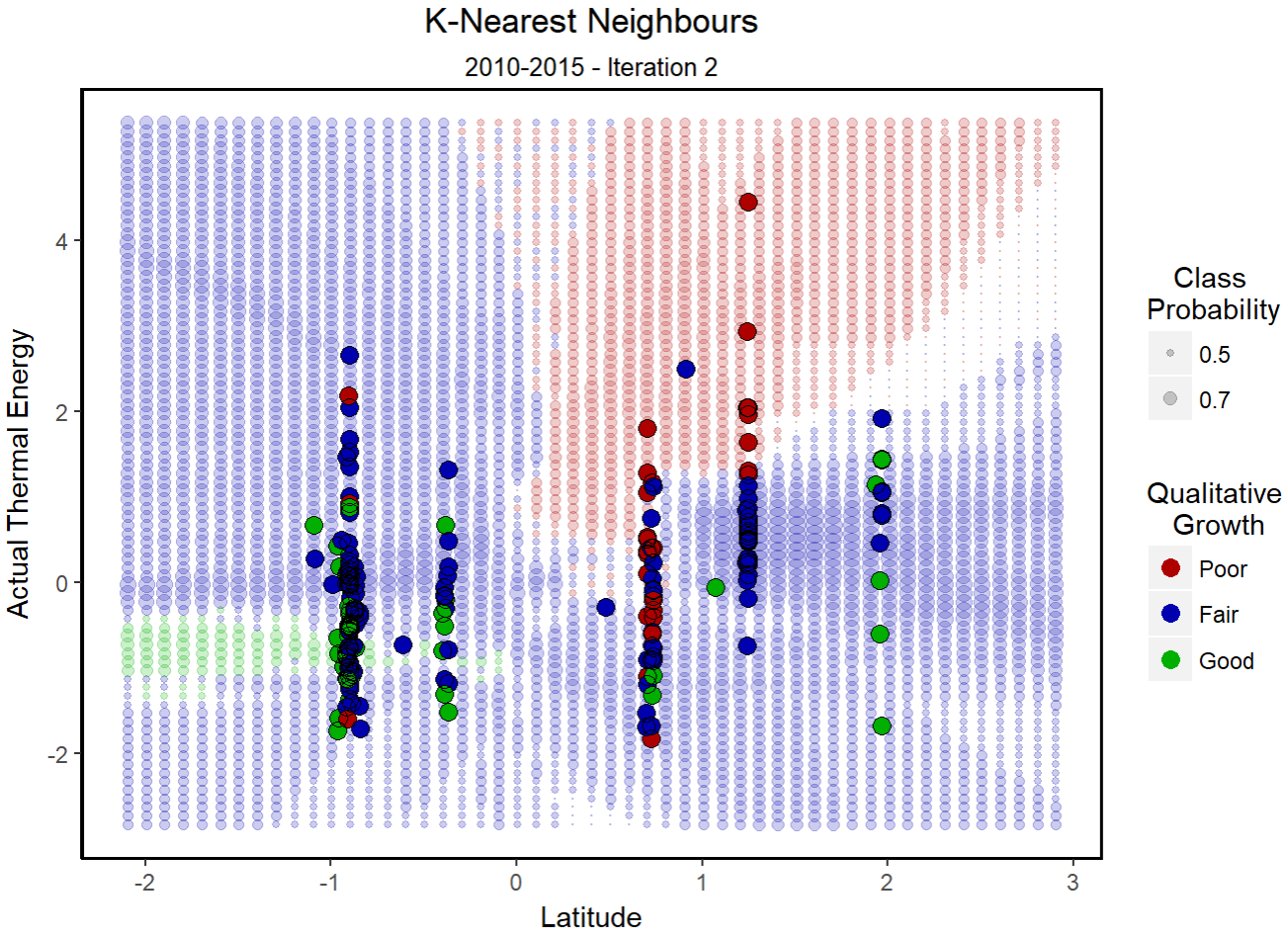

```
#### v2.8 plot ####

#Seperate labels from train set
a <- i2.data$Latitude #<- Customize Variables
b <- i2.data$x.2010.Actual.Thermal #<- Customize Variables
aname<-"Latitude"
```

```
bname<-"Actual Thermal Energy"
cl <- i2.data$x.2015.Qualitative.EUI
train <- cbind(a,b)

#Populate Grid with points
test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))

#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```




```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i2.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
```

```
                                prob_cls=ifelse(classif==cls,
                                                1, 0)),
                          mutate(test,
                                 prob=prob,
                                 cls="Poor",
                                 prob_cls=ifelse(classif==cls,
                                                 1, 0)))
i2.dataf$cls =ordered(i2.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i2.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 2",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```




```
################################
####    KNN iteration 3   ####
################################

i3.data<-data[,c("D.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
                 "Portfolio.Manager.no.", "Latitude", "Longitude",    "Construction.Year",
  "Occupant.Density",
                 "Vacancy.Rate",  "Weekly.Operating.Hours")]
cnames<-c("D.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
          "Portfolio.Manager.no.",  "Latitude", "Longitude",    "Construction.Year",    "Occup
ant.Density",
          "Vacancy.Rate",   "Weekly.Operating.Hours")
# Normalize
QG<-i3.data$D.Qualitative.EUI
norm.data<- as.data.frame(lapply(i3.data[,-1], scale))
i3.data <-cbind(QG, norm.data)
row.names(i3.data) <- rnames
colnames(i3.data) <-cnames
##create Train and Test set
train.rows = sample(1:nrow(i3.data), (nrow(i3.data)*0.7))
i3.data.train = i3.data[train.rows,]
i3.data.test = i3.data[-train.rows,]
#View(i3.data.test)
nrow(i3.data.test)
```

```
## [1] 60
```

```
# Normalization Check
summary(i3.data)
```

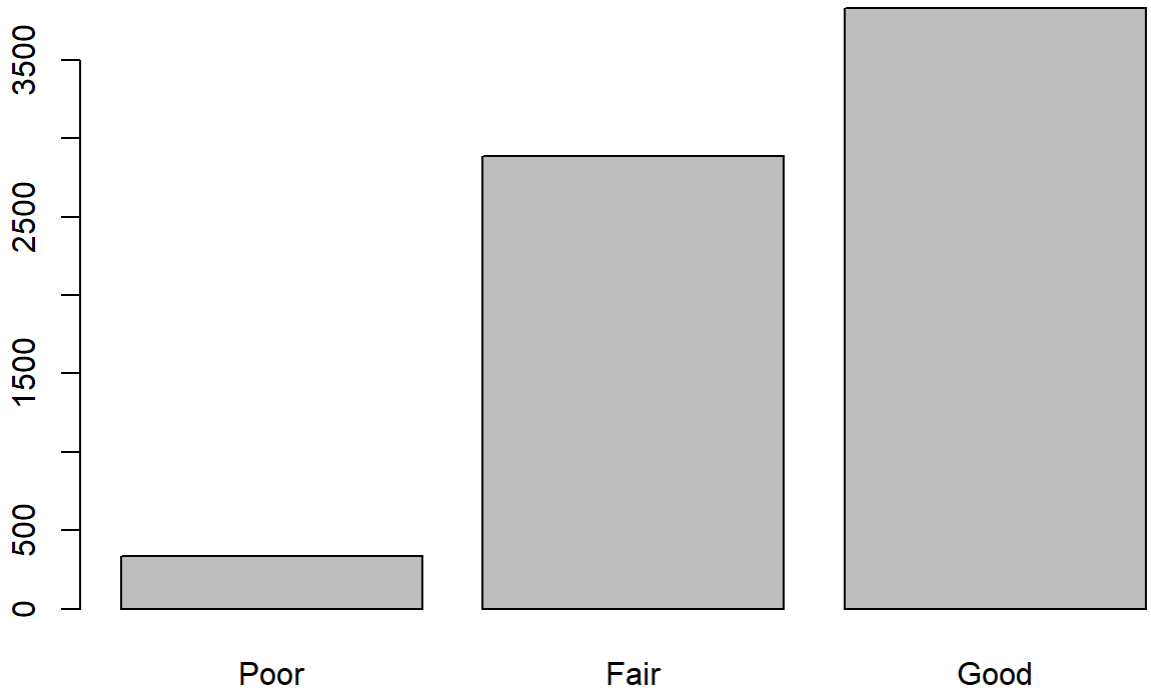
```
##  D.Qualitative.EUI x.2010.Actual.E_C.Energy x.2010.Actual.Thermal
##  Poor:50           Min.   :-1.3441          Min.   :-1.82286
##  Fair:88           1st Qu.:-0.6025          1st Qu.:-0.74247
##  Good:60           Median :-0.2585          Median :-0.05318
##                    Mean   : 0.0000          Mean   : 0.00000
##                    3rd Qu.: 0.3042          3rd Qu.: 0.51796
##                    Max.   : 5.0646          Max.   : 4.45332
##  Portfolio.Manager.no.    Latitude         Longitude
##  Min.   :-0.8995       Min.   :-1.0944   Min.   :-1.4012
##  1st Qu.:-0.8995       1st Qu.:-0.9028   1st Qu.:-0.9342
##  Median :-0.5976       Median :-0.3891   Median : 0.7940
##  Mean   : 0.0000       Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.5154       3rd Qu.: 0.7341   3rd Qu.: 0.8121
##  Max.   : 1.5154       Max.   : 1.9675   Max.   : 1.6077
##  Construction.Year   Occupant.Density   Vacancy.Rate
##  Min.   :-4.183267   Min.   :-2.1132   Min.   :-0.8323
##  1st Qu.:-0.345052   1st Qu.:-0.7143   1st Qu.:-0.8323
##  Median : 0.009245   Median :-0.1129   Median :-0.3515
##  Mean   : 0.000000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.481641   3rd Qu.: 0.4928   3rd Qu.: 0.3592
##  Max.   : 1.544531   Max.   : 5.1552   Max.   : 3.3189
##  Weekly.Operating.Hours
##  Min.   :-2.0884
##  1st Qu.:-0.3092
##  Median :-0.2046
##  Mean   : 0.0000
##  3rd Qu.: 0.7374
##  Max.   : 9.0054
```

```
# rUN KNN  algorithm
k<- round(sqrt(nrow(i3.data.train)))
i3.data.pred <- knn(i3.data.train[ ,-1], i3.data.test[ ,-1], i3.data.train[,1], k=k, prob=TRUE
)
#Accuracy

Confusion_Mat = confusionMatrix(data = i3.data.pred, reference = i3.data.test[,1], dnn = c("Pr
ediction", "Reference"))
Confusion_Mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Poor Fair Good
##       Poor    5    4    1
##       Fair   10   19   13
##       Good    2    2    4
```

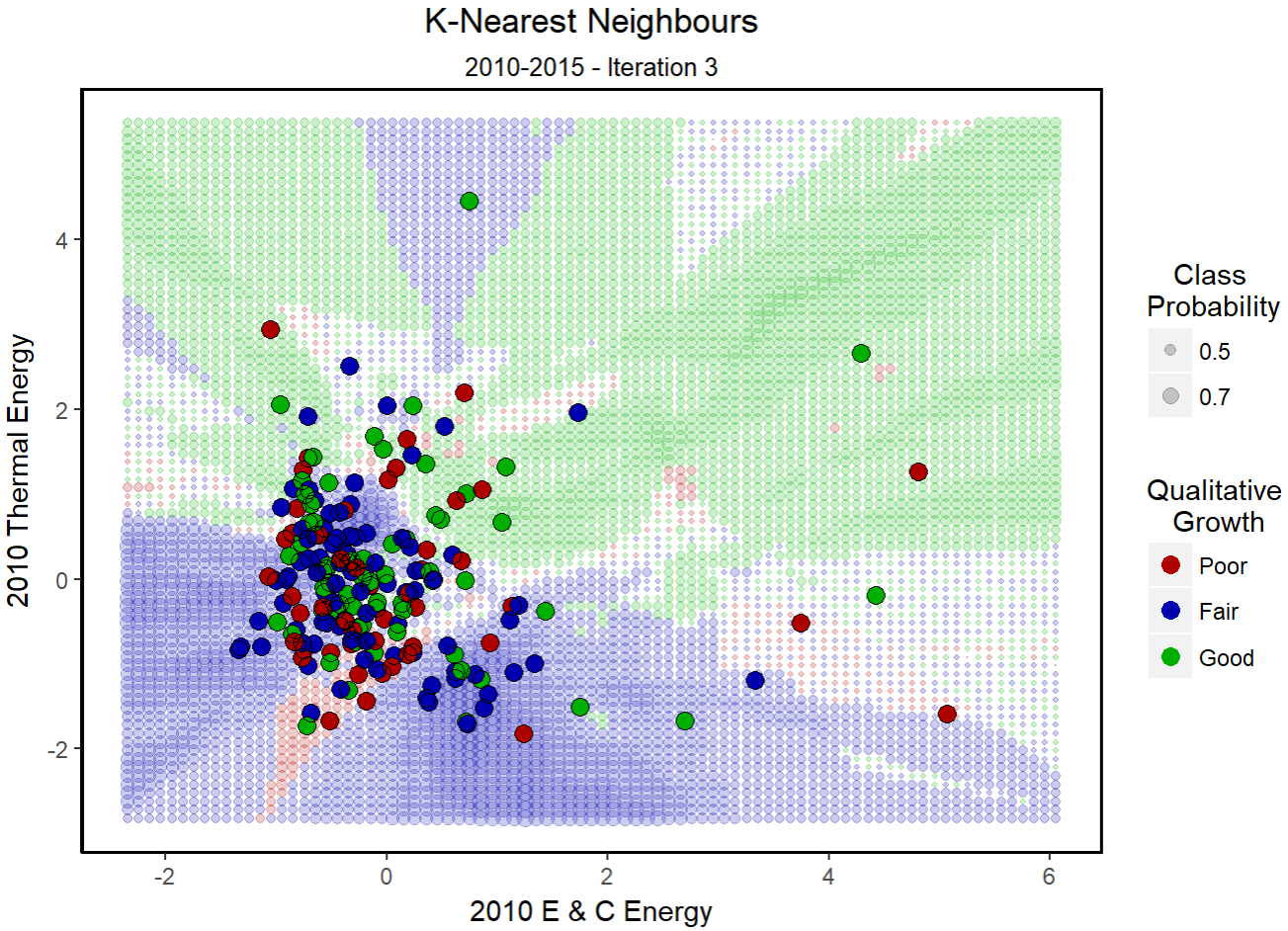
```
##
## Overall Statistics
##
##                Accuracy : 0.4667
##                  95% CI : (0.3367, 0.6)
##     No Information Rate : 0.4167
##     P-Value [Acc > NIR] : 0.25516
##
##                   Kappa : 0.1413
##  Mcnemar's Test P-Value : 0.01188
##
## Statistics by Class:
##
##                      Class: Poor Class: Fair Class: Good
## Sensitivity              0.29412      0.7600     0.22222
## Specificity              0.88372      0.3429     0.90476
## Pos Pred Value           0.50000      0.4524     0.50000
## Neg Pred Value           0.76000      0.6667     0.73077
## Prevalence               0.28333      0.4167     0.30000
## Detection Rate           0.08333      0.3167     0.06667
## Detection Prevalence     0.16667      0.7000     0.13333
## Balanced Accuracy        0.58892      0.5514     0.56349
```

```
out<-capture.output(Confusion_Mat)
setwd("C:/Users/carle/Documents/Results/KNN Results")
cat(out,file="i3 KNN Confusion Matrix",sep="/n",append=TRUE)

#Seperate labels from train set
a <- i3.data$x.2010.Actual.E_C.Energy #`   #<- Customize Variables
b <- i3.data$x.2010.Actual.Thermal #<- Customize Variables
aname<-"2010 E & C Energy"
bname<-"2010 Thermal Energy"
cl <- i3.data$D.Qualitative.EUI
train <- cbind(a,b)

#Populate Grid with points
test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))
#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```



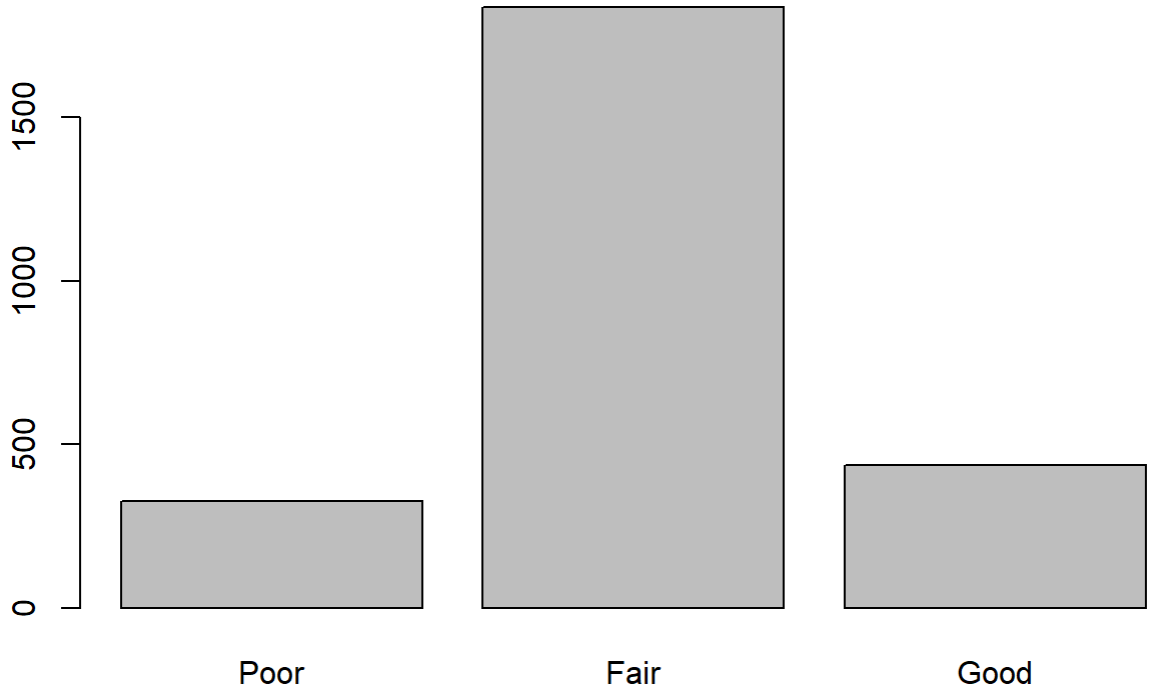

```
prob <- attr(classif, "prob")
#Data Structure for Plotting
i3.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Poor",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)))
i3.dataf$cls =ordered(i3.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability

ggplot(i3.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
```

```
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 3",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```



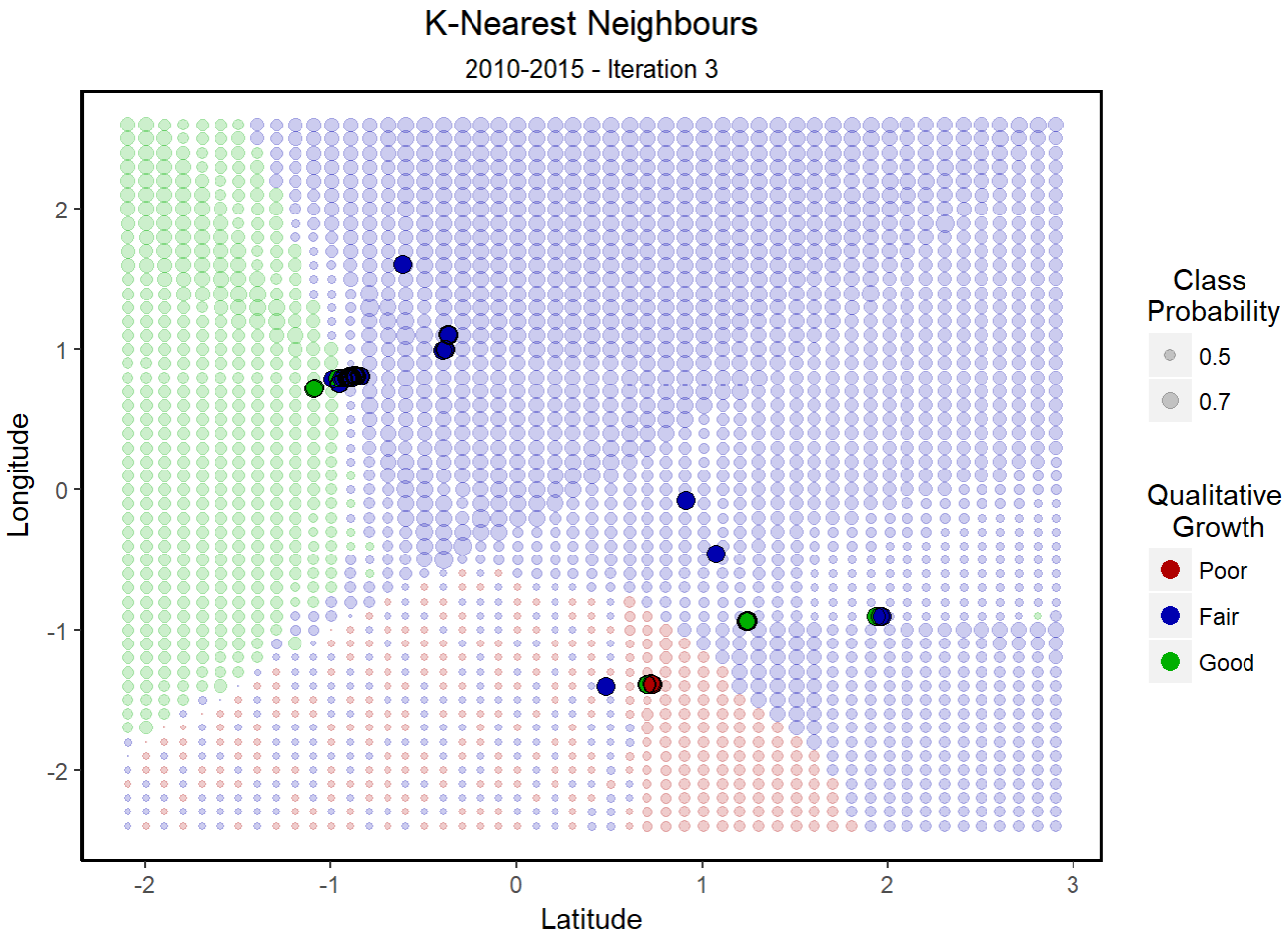

```
####v3.2 plot ####

#Seperate labels from train set
a <- i3.data$Latitude #`   #<- Customize Variables
b <- i3.data$Longitude #<- Customize Variables
aname<-"Latitude"
```

```
bname<-"Longitude"
cl <- i3.data$D.Qualitative.EUI
train <- cbind(a,b)

#Populate Grid with points
test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))

#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```

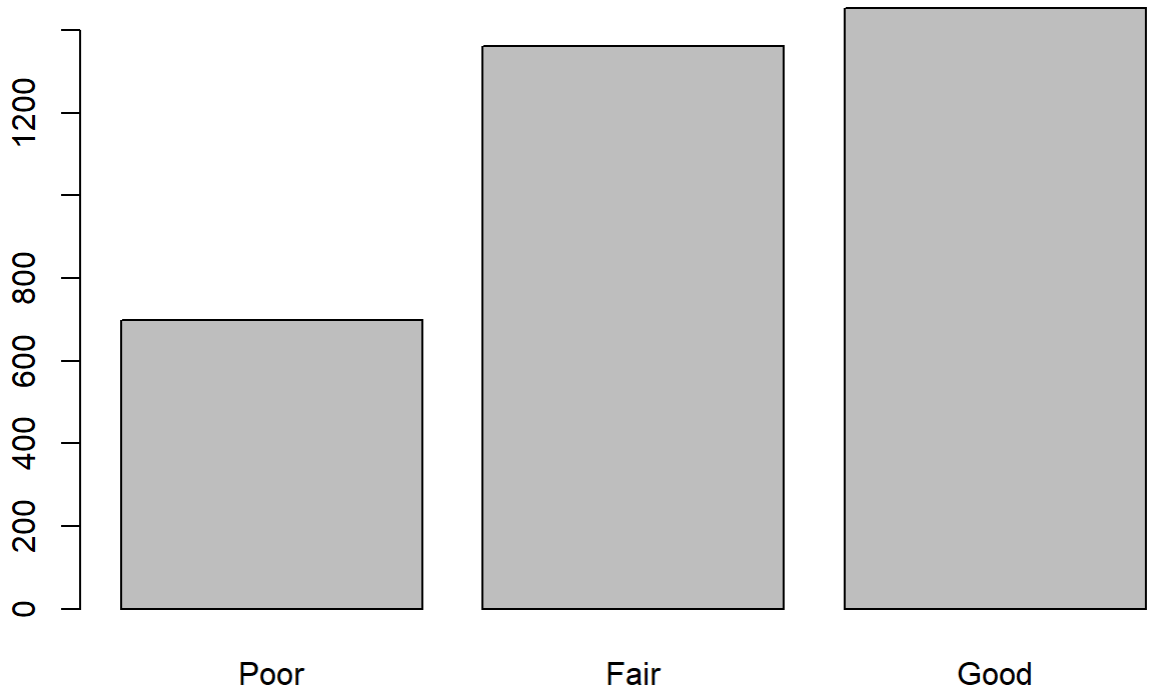



```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i3.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
```

```
                                prob_cls=ifelse(classif==cls,
                                                1, 0)),
                          mutate(test,
                                 prob=prob,
                                 cls="Poor",
                                 prob_cls=ifelse(classif==cls,
                                                 1, 0)))
i3.dataf$cls =ordered(i3.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i3.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 3",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```

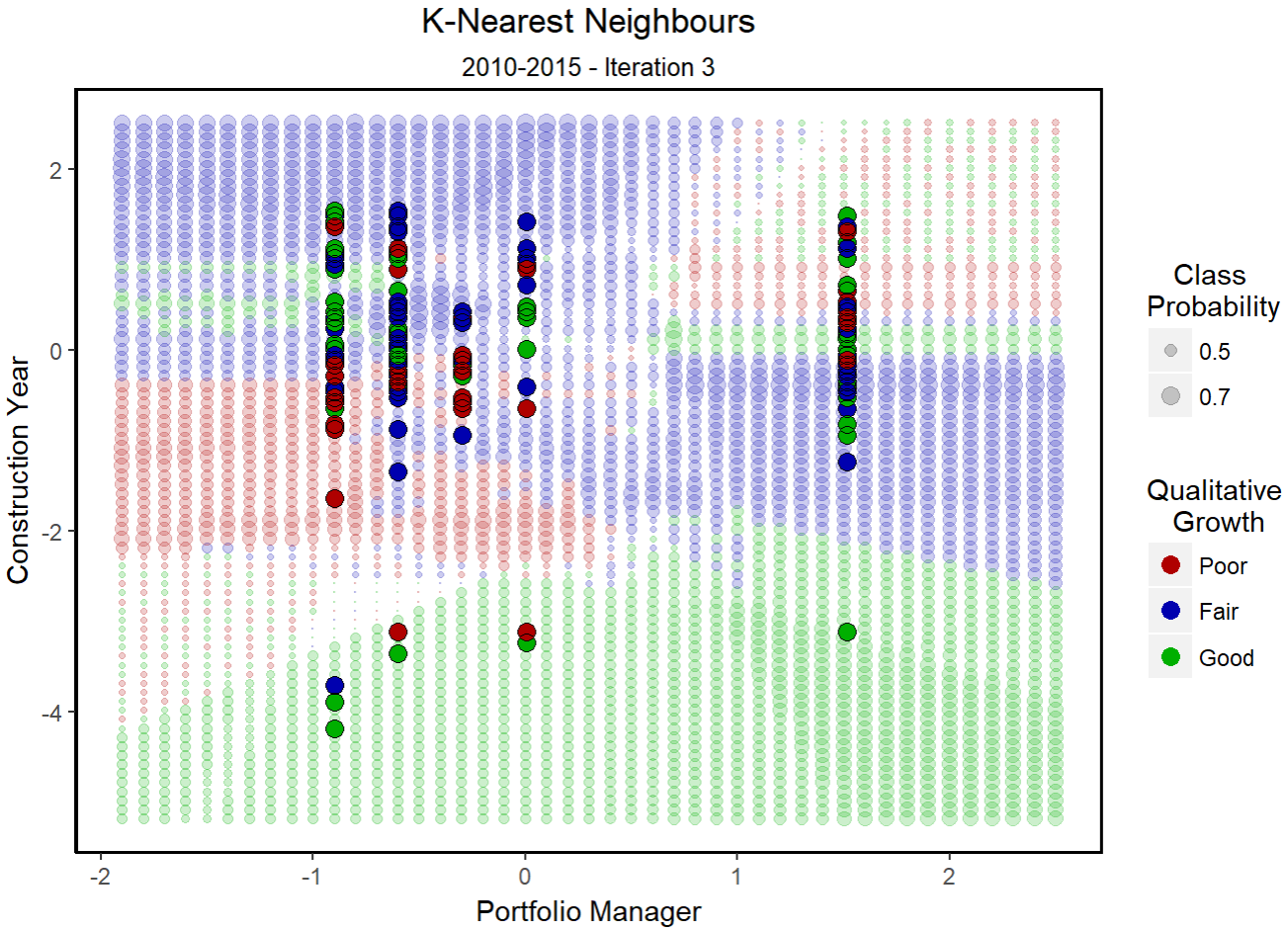



```
####v3.3 plot ####
#Seperate labels from train set
a <- i3.data$Portfolio.Manager.no. #`    #<- Customize Variables
b <- i3.data$Construction.Year #<- Customize Variables
aname<-"Portfolio Manager"
bname<-"Construction Year"
cl <- i3.data$D.Qualitative.EUI
train <- cbind(a,b)

#Populate Grid with points
test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))

#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```



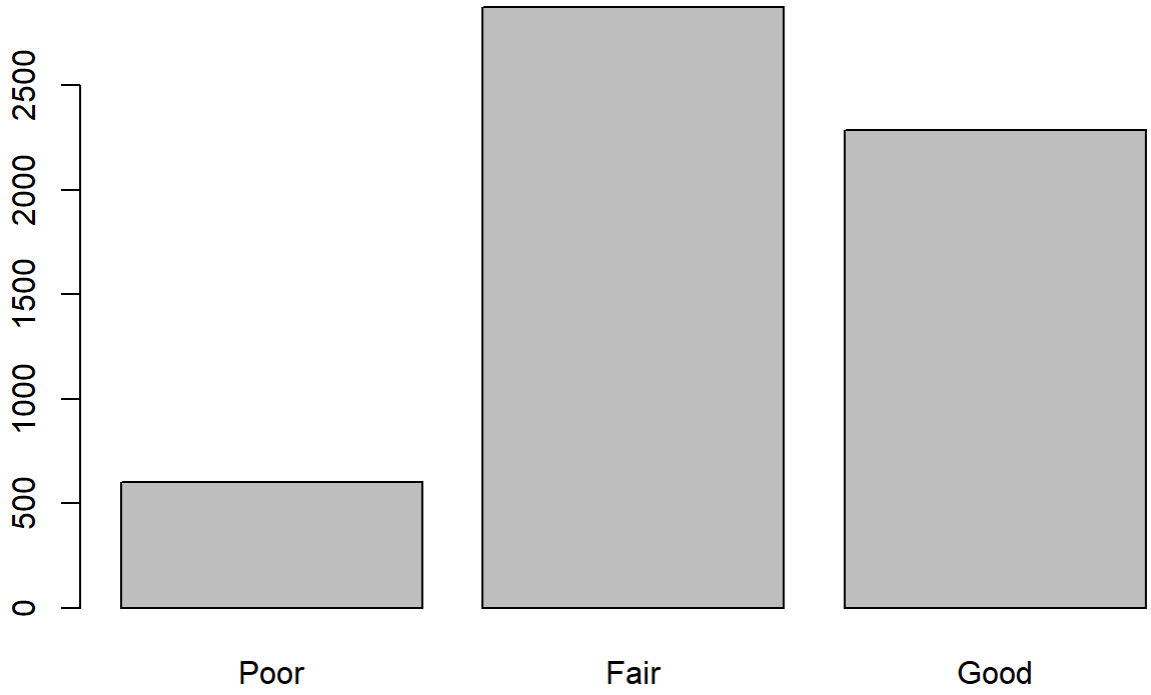

```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i3.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Poor",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)))
i3.dataf$cls =ordered(i3.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i3.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
```

```
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 3",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```



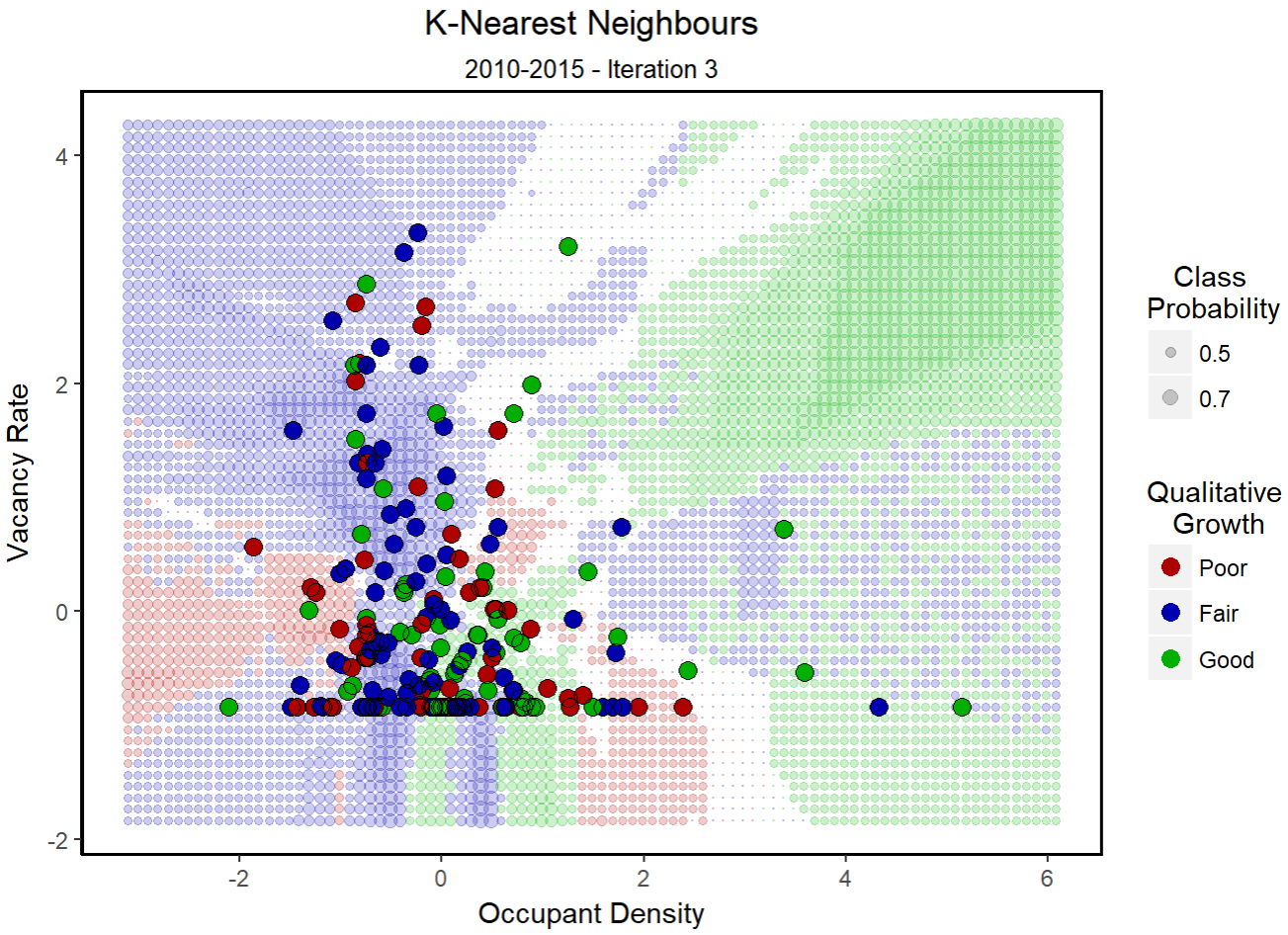

```
#### v3.4 plot ####
#Seperate labels from train set
a <- i3.data$Occupant.Density #<- Customize Variables
b <- i3.data$Vacancy.Rate #<- Customize Variables
aname<-"Occupant Density"
bname<-"Vacancy Rate"
```

```
cl <- i3.data$D.Qualitative.EUI
train <- cbind(a,b)

#Populate Grid with points
test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))

#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```

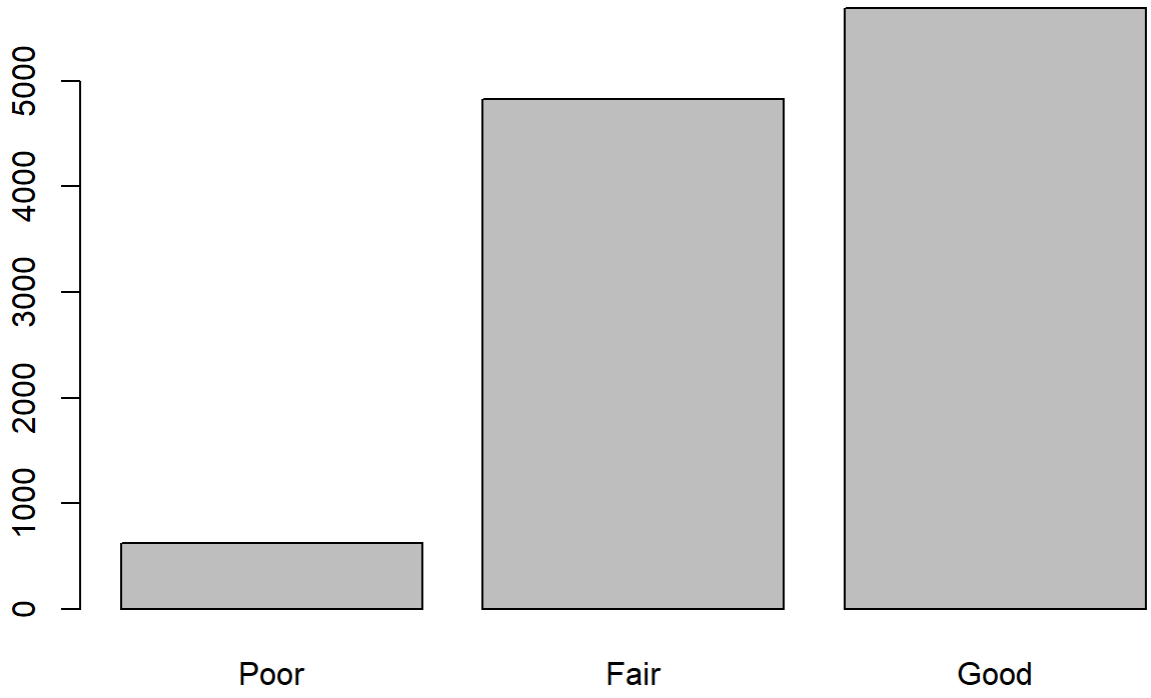



```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i3.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
```

```
                                                1, 0)),
                          mutate(test,
                                 prob=prob,
                                 cls="Poor",
                                 prob_cls=ifelse(classif==cls,
                                                 1, 0)))
i3.dataf$cls =ordered(i3.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i3.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 3",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```

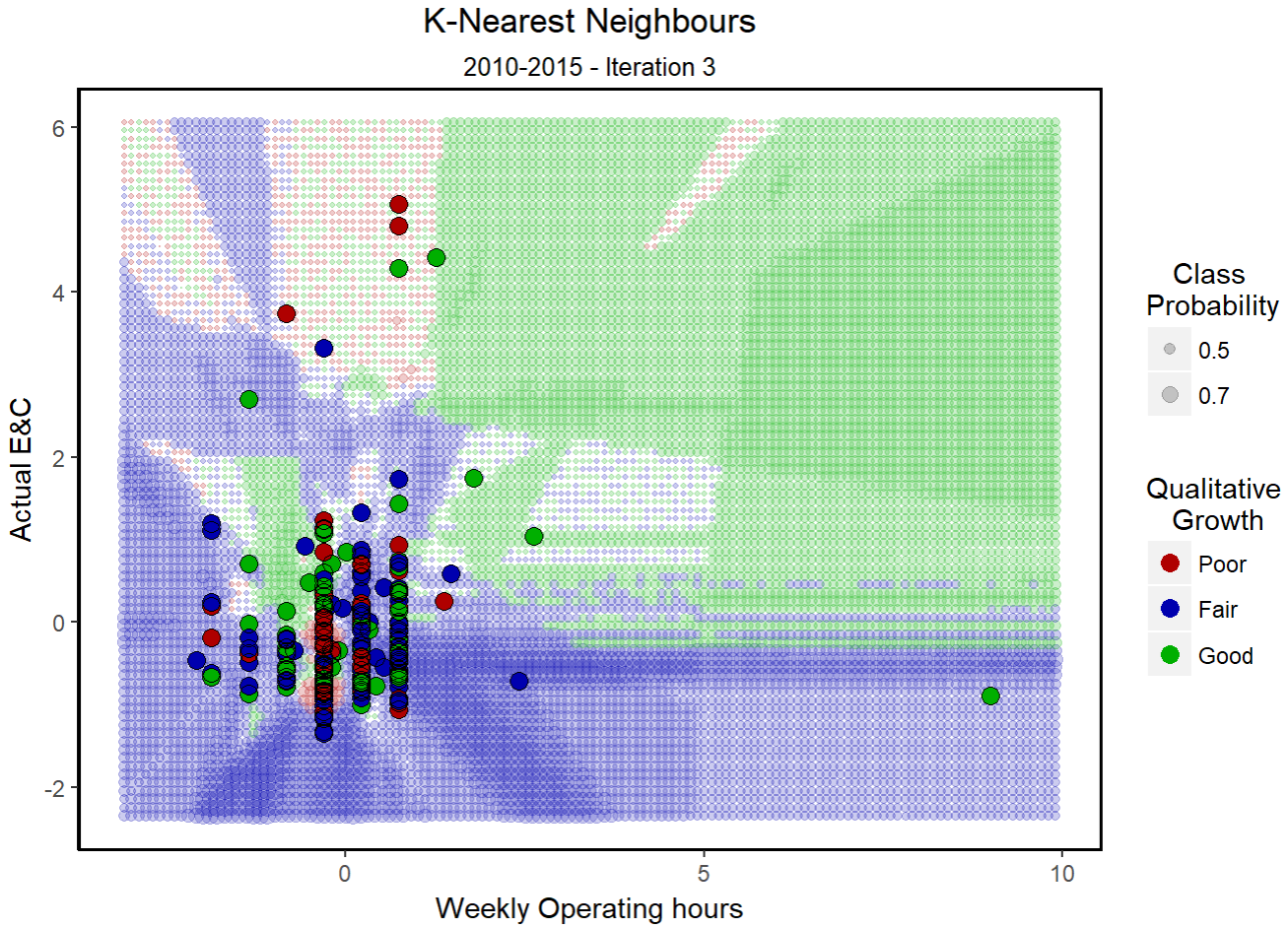



```
#### v3.5 plot ####

#Seperate labels from train set
a <- i3.data$Weekly.Operating.Hours #<- Customize Variables
b <- i3.data$x.2010.Actual.E_C.Energy #<- Customize Variables
aname<-"Weekly Operating hours"
bname<-"Actual E&C"
cl <- i3.data$D.Qualitative.EUI
train <- cbind(a,b)

#Populate Grid with points
test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))

#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```



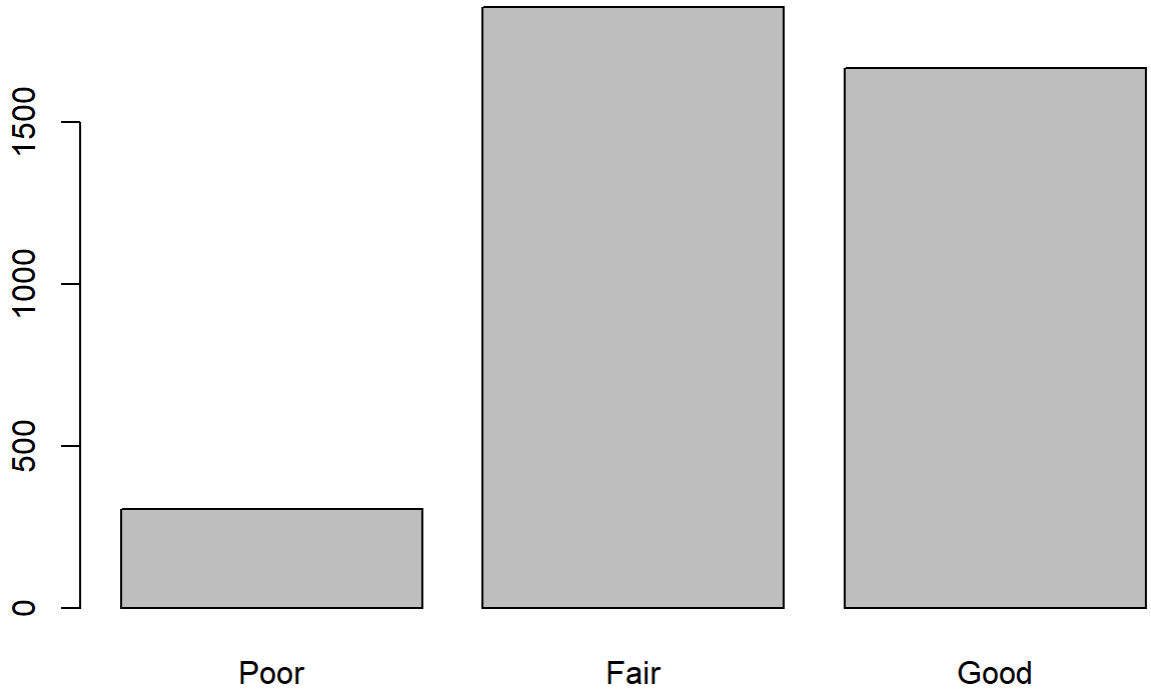

```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i3.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Poor",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)))
i3.dataf$cls =ordered(i3.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i3.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
```

```
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 3",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```



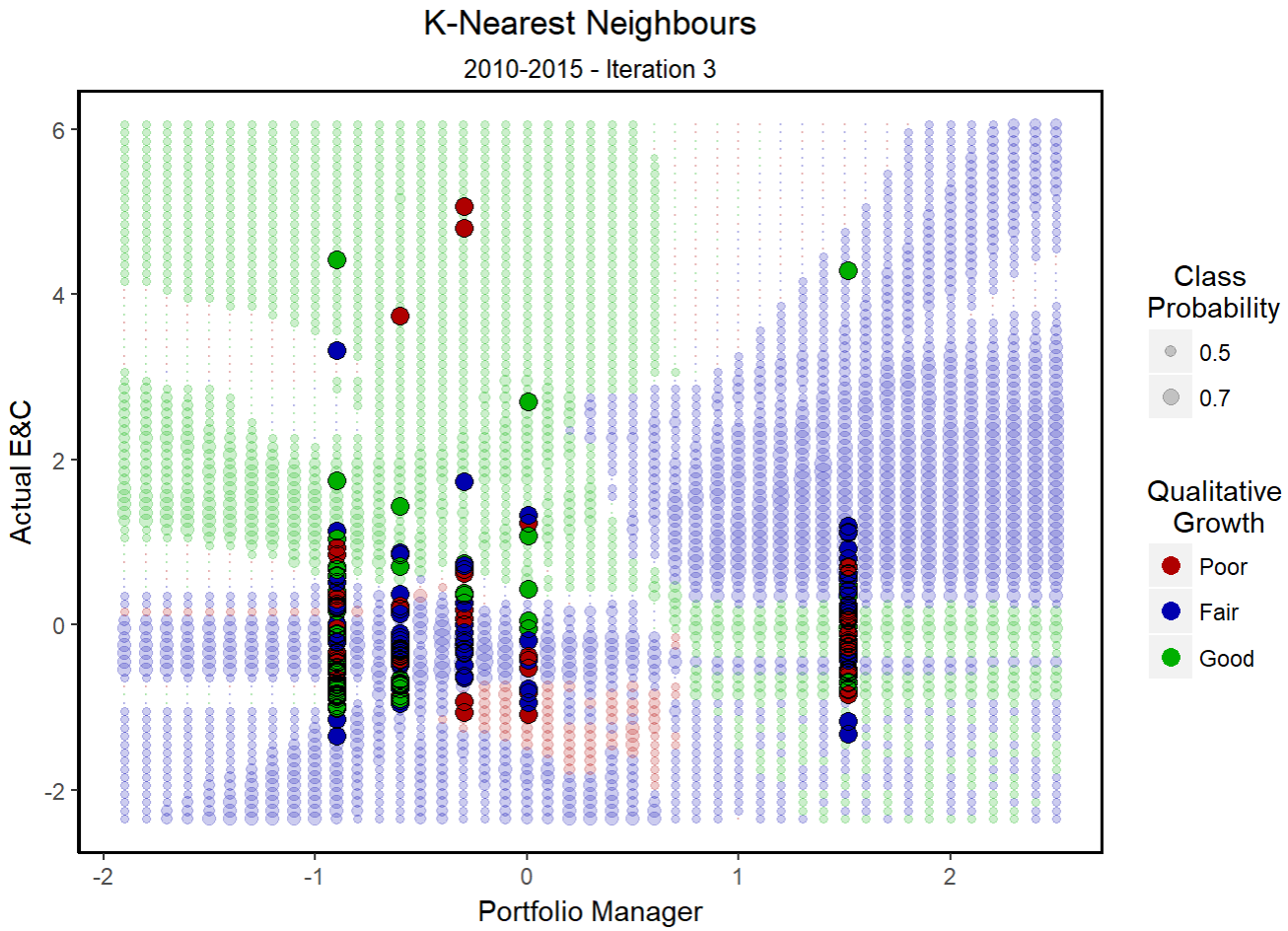

```
#### v3.6 plot ####

#Seperate labels from train set
a <- i3.data$Portfolio.Manager.no. #<- Customize Variables
b <- i3.data$x.2010.Actual.E_C.Energy #<- Customize Variables
aname<-"Portfolio Manager"
```

```
bname<-"Actual E&C"
cl <- i3.data$D.Qualitative.EUI
train <- cbind(a,b)

#Populate Grid with points
test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))

#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```

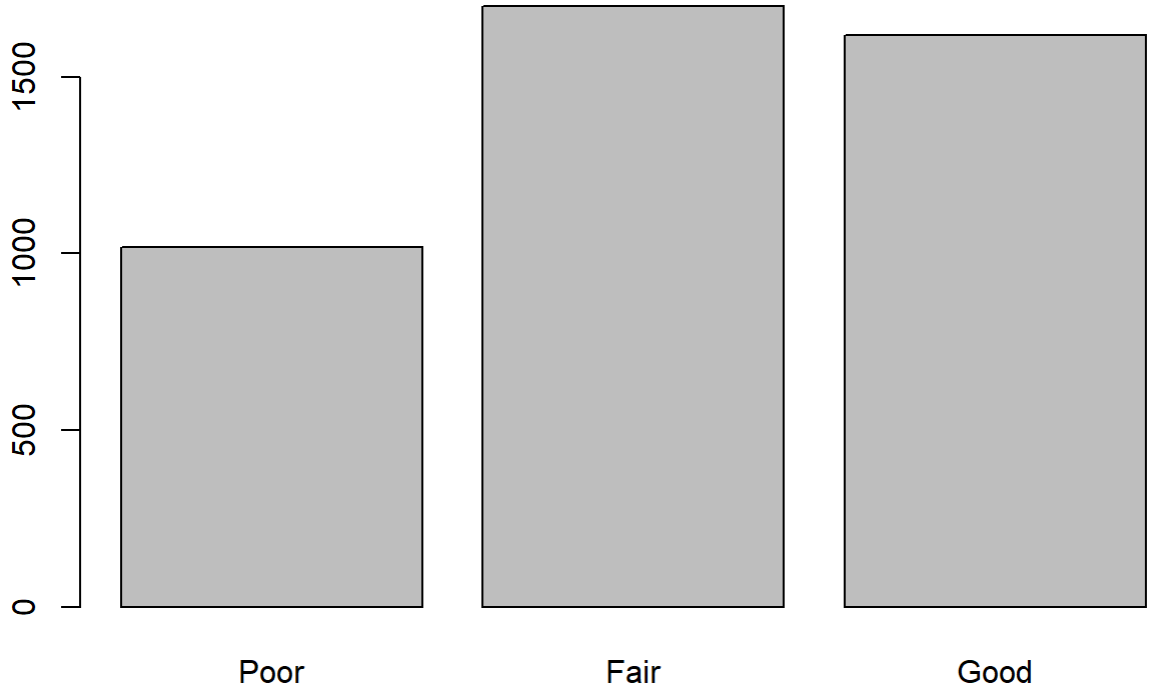



```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i3.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
```

```
                                prob_cls=ifelse(classif==cls,
                                                1, 0)),
                          mutate(test,
                                 prob=prob,
                                 cls="Poor",
                                 prob_cls=ifelse(classif==cls,
                                                 1, 0)))
i3.dataf$cls =ordered(i3.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i3.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 3",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```

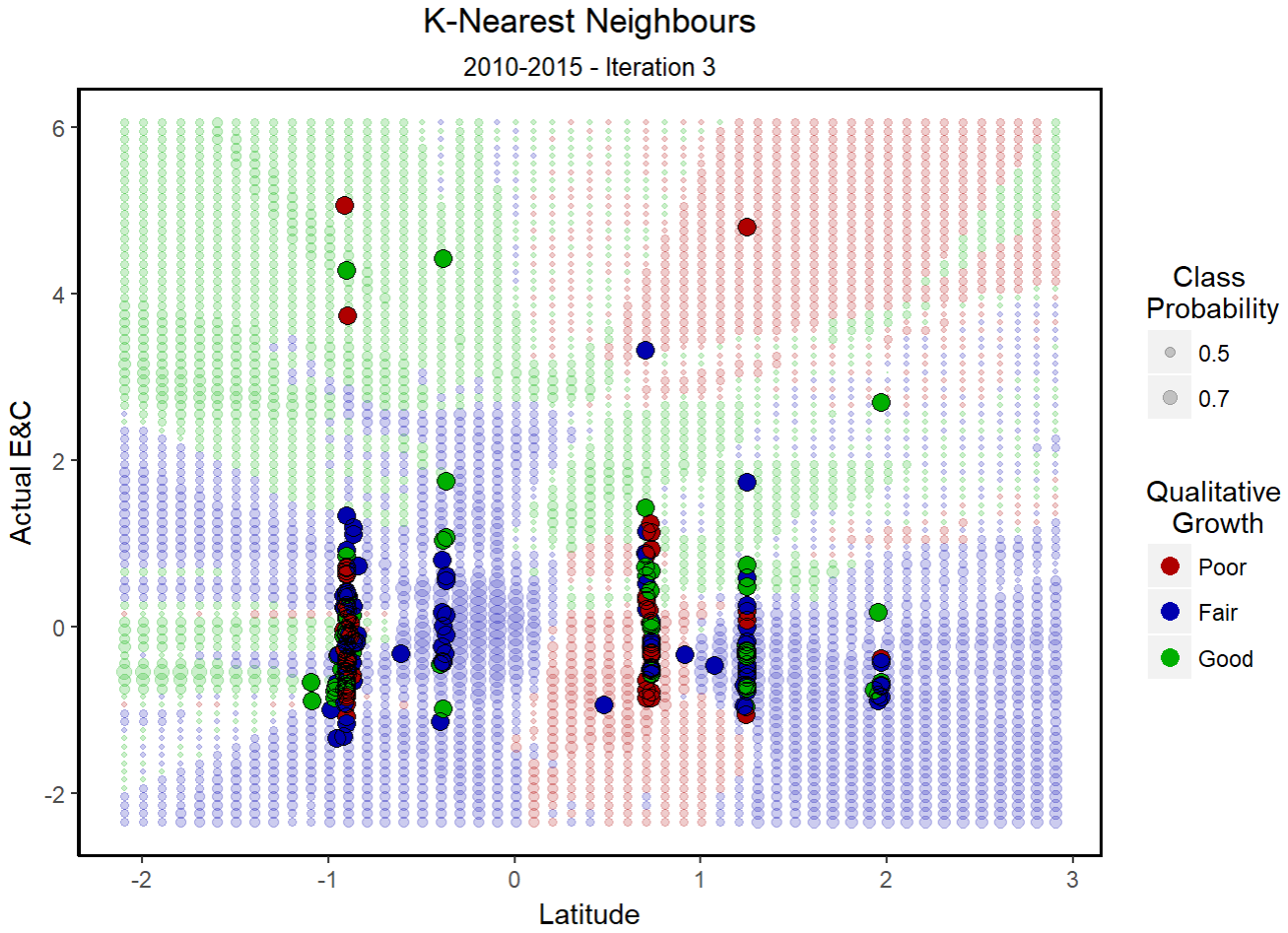



```
#### v3.7 plot ####

#Seperate labels from train set
a <- i3.data$Latitude #<- Customize Variables
b <- i3.data$x.2010.Actual.E_C.Energy #<- Customize Variables
aname<-"Latitude"
bname<-"Actual E&C"
cl <- i3.data$D.Qualitative.EUI
train <- cbind(a,b)

#Populate Grid with points
test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))

#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```



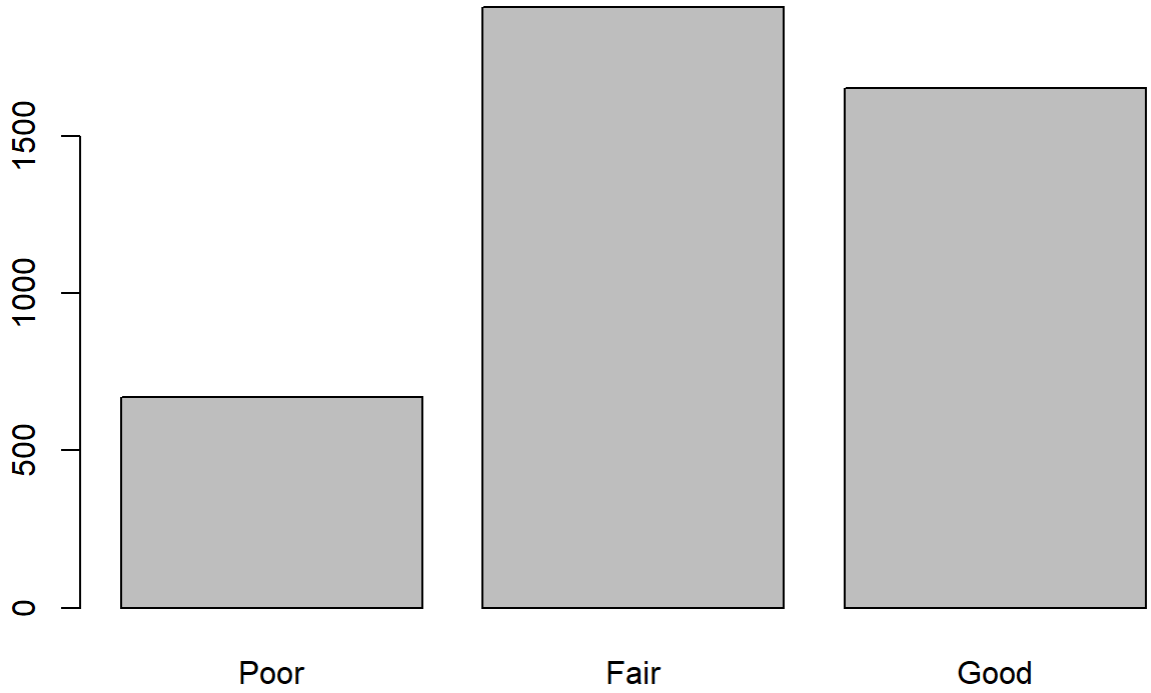

```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i3.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Poor",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)))
i3.dataf$cls =ordered(i3.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i3.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
```

```
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 3",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```



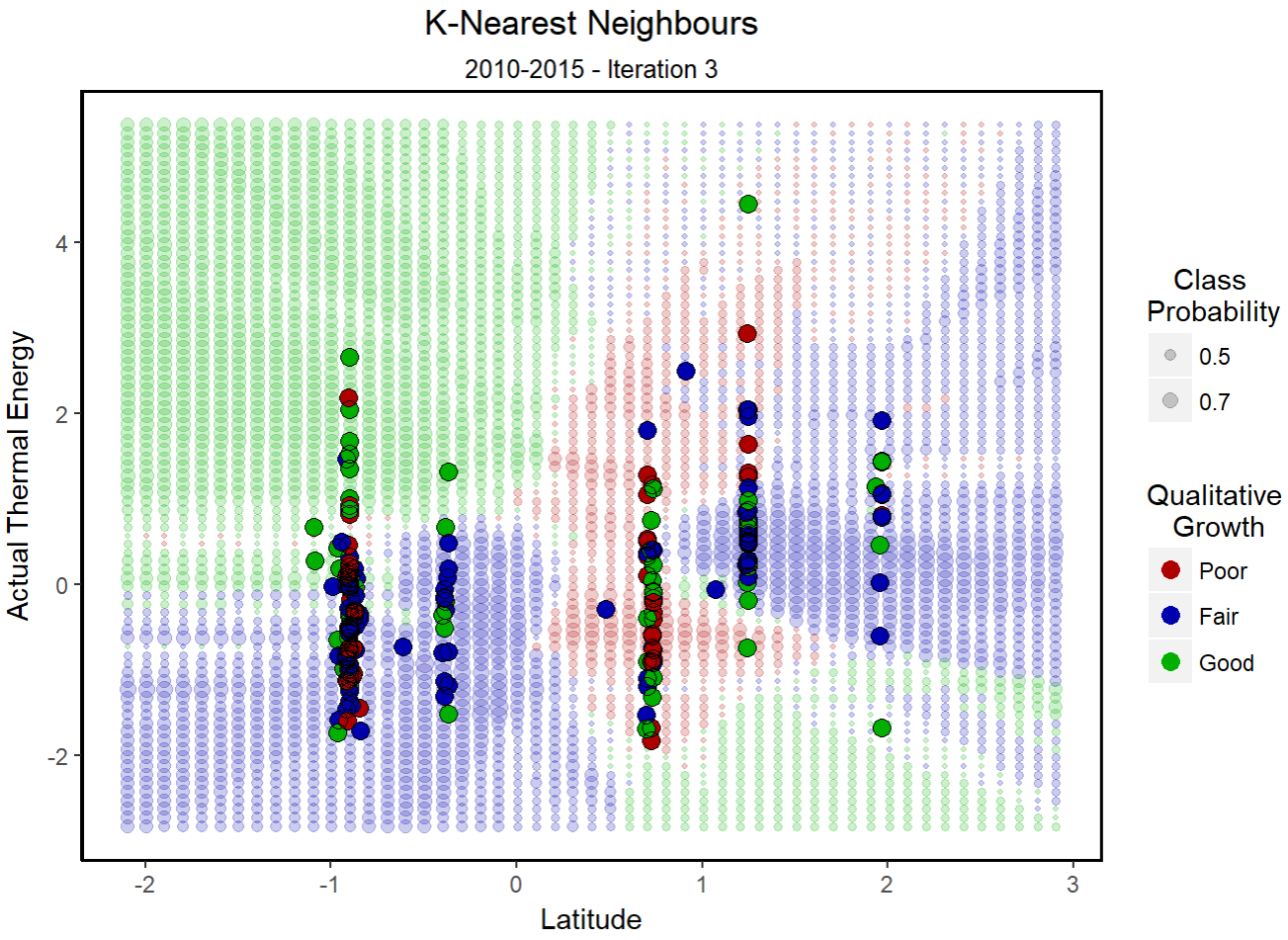

```
#### v3.8 plot ####

#Seperate labels from train set
a <- i3.data$Latitude #<- Customize Variables
b <- i3.data$x.2010.Actual.Thermal #<- Customize Variables
aname<-"Latitude"
```

```
bname<-"Actual Thermal Energy"
cl <- i3.data$D.Qualitative.EUI
train <- cbind(a,b)

#Populate Grid with points
test <- expand.grid(x=seq(min(a -1), max(a+1),
                          by=0.1),
                    y=seq(min(b-1), max(b+1),
                          by=0.1))

#Classification for that grid
k<- round(sqrt(nrow(train)))
classif <- knn(train, test, cl, k = k, prob=TRUE)
plot(classif)
```




```
prob <- attr(classif, "prob")

#Data Structure for Plotting
i3.dataf <- bind_rows(mutate(test,
                             prob=prob,
                             cls="Good",
                             prob_cls=ifelse(classif==cls,
                                             1, 0)),
                      mutate(test,
                             prob=prob,
                             cls="Fair",
```

```
                                prob_cls=ifelse(classif==cls,
                                                1, 0)),
                          mutate(test,
                                 prob=prob,
                                 cls="Poor",
                                 prob_cls=ifelse(classif==cls,
                                                 1, 0)))
i3.dataf$cls =ordered(i3.dataf$cls, levels = c("Poor","Fair","Good"))

#Plot Class Probability
ggplot(i3.dataf) +
  geom_point(aes(x=x, y=y, col=cls, size=prob),
             data = mutate(test, cls=classif), alpha = 0.2) +
  scale_size(range=c(0.1, 3), breaks=c(0.5, 0.7), name="Class \nProbability") +
  scale_color_manual(breaks=c("Poor", "Fair", "Good"), values=c("Poor" ="#AF0000", "Fair"="#00
00AF", "Good"= "#00AF00"))+
  geom_point(aes(x=x, y=y, col=cls),
             size=3, #point area
             data=data.frame(x=a, y=b, cls=cl)) +
  geom_point(aes(x=x, y=y), #Black circle around point
             size=3, shape=21, data=data.frame(x=a, y=b, cls=cl))+
  labs(x= aname, y=bname, title="K-Nearest Neighbours",
       subtitle = "2010-2015 - Iteration 3",
       colour = "Qualitative \nGrowth")+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.border = element_rect(colour = "black",
                                    fill=NA, size=1), legend.title.align=0.5)+
  guides(colour = guide_legend(order = 2),
         size = guide_legend(order = 1))
```




```
################################
####    KNN iteration 4    ####
################################

i4.data<-data[,c("x.2010.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
                 "Portfolio.Manager.no.", "Latitude", "Longitude",    "Construction.Year",
  "Occupant.Density",
                 "Vacancy.Rate",  "Weekly.Operating.Hours")]
cnames<-c("x.2010.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
          "Portfolio.Manager.no.",  "Latitude", "Longitude",    "Construction.Year",    "Occup
ant.Density",
          "Vacancy.Rate",   "Weekly.Operating.Hours")
# Normalize
QG<-i4.data$x.2010.Qualitative.EUI
norm.data<- as.data.frame(lapply(i4.data[,-1], scale))
i4.data <-cbind(QG, norm.data)
row.names(i4.data) <- rnames
colnames(i4.data) <-cnames
nrow(i4.data)
```

```
## [1] 198
```

```
# Normalization Check
summary(i4.data)
```

```
##  x.2010.Qualitative.EUI x.2010.Actual.E_C.Energy x.2010.Actual.Thermal
##  Poor: 31               Min.   :-1.3441          Min.   :-1.82286
##  Fair:153               1st Qu.:-0.6025          1st Qu.:-0.74247
##  Good: 14               Median :-0.2585          Median :-0.05318
##                         Mean   : 0.0000          Mean   : 0.00000
##                         3rd Qu.: 0.3042          3rd Qu.: 0.51796
##                         Max.   : 5.0646          Max.   : 4.45332
##  Portfolio.Manager.no.    Latitude         Longitude
##  Min.   :-0.8995       Min.   :-1.0944   Min.   :-1.4012
##  1st Qu.:-0.8995       1st Qu.:-0.9028   1st Qu.:-0.9342
##  Median :-0.5976       Median :-0.3891   Median : 0.7940
##  Mean   : 0.0000       Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.5154       3rd Qu.: 0.7341   3rd Qu.: 0.8121
##  Max.   : 1.5154       Max.   : 1.9675   Max.   : 1.6077
##  Construction.Year   Occupant.Density   Vacancy.Rate
##  Min.   :-4.183267   Min.   :-2.1132   Min.   :-0.8323
##  1st Qu.:-0.345052   1st Qu.:-0.7143   1st Qu.:-0.8323
##  Median : 0.009245   Median :-0.1129   Median :-0.3515
##  Mean   : 0.000000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.481641   3rd Qu.: 0.4928   3rd Qu.: 0.3592
##  Max.   : 1.544531   Max.   : 5.1552   Max.   : 3.3189
##  Weekly.Operating.Hours
##  Min.   :-2.0884
##  1st Qu.:-0.3092
##  Median :-0.2046
##  Mean   : 0.0000
##  3rd Qu.: 0.7374
##  Max.   : 9.0054
```

```
# rUN KNN  algorithm
k<- round(sqrt(nrow(i4.data)))
i4.data.pred <- knn.cv(i4.data[ ,-1], i4.data[,1], k=k, prob=TRUE)

#Accuracy
i4.data.pred
```

```
##   [1] Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair
##  [15] Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair
##  [29] Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair
##  [43] Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair
##  [57] Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair
##  [71] Fair Fair Fair Fair Fair Fair Fair Poor Fair Fair Fair Fair Fair Fair
##  [85] Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair
##  [99] Fair Fair Poor Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair
## [113] Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair
## [127] Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair
## [141] Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair
## [155] Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair
## [169] Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair
## [183] Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair Fair
## [197] Fair Fair
## attr(,"prob")
```

```
##   [1] 0.6428571 0.9285714 0.8571429 0.9285714 0.9285714 0.6428571 0.8571429
##   [8] 1.0000000 0.8571429 0.7142857 0.9285714 0.7857143 0.7857143 0.7142857
##  [15] 0.7857143 0.8571429 0.7857143 0.8571429 0.6428571 0.9285714 0.8571429
##  [22] 0.9285714 0.9285714 0.9285714 0.7142857 0.7857143 1.0000000 0.9285714
##  [29] 0.8571429 0.7857143 0.6428571 0.6428571 0.6428571 0.7142857 0.5714286
##  [36] 1.0000000 0.6428571 0.7142857 0.7857143 0.6428571 0.6428571 0.7857143
##  [43] 0.8571429 0.8571429 0.7142857 0.8571429 0.5714286 0.7142857 0.9285714
##  [50] 0.7142857 0.8571429 1.0000000 0.7142857 0.7142857 0.7142857 0.6428571
##  [57] 0.8571429 0.9285714 1.0000000 1.0000000 0.7142857 0.8571429 0.9285714
##  [64] 1.0000000 0.8571429 0.8571429 0.9285714 0.9285714 1.0000000 0.9285714
##  [71] 1.0000000 1.0000000 0.7857143 0.6428571 0.9285714 0.6428571 0.7857143
##  [78] 0.7857143 0.5714286 0.9285714 0.7857143 1.0000000 0.9285714 0.7142857
##  [85] 0.9285714 0.7142857 0.9285714 0.8571429 0.7142857 0.6428571 0.7142857
##  [92] 0.7857143 0.9285714 0.8571429 0.8571429 1.0000000 0.5714286 0.7857143
##  [99] 0.9285714 1.0000000 0.5000000 0.8571429 0.8571429 0.8571429 0.8571429
## [106] 0.8571429 0.7142857 0.7142857 0.7857143 0.9285714 1.0000000 0.8571429
## [113] 1.0000000 0.8571429 0.8571429 0.9285714 1.0000000 0.9285714 1.0000000
## [120] 0.8571429 1.0000000 0.8571429 0.8571429 0.8571429 0.9285714 1.0000000
## [127] 0.8571429 0.7857143 0.9285714 0.7142857 0.6428571 0.7857143 0.8571429
## [134] 0.7142857 0.8571429 0.7857143 0.7857143 0.8571429 0.9285714 0.9285714
## [141] 0.9285714 0.9285714 0.6428571 0.9285714 0.9285714 1.0000000 0.7857143
## [148] 0.9285714 0.8571429 0.7857143 0.9285714 0.8571429 0.8571429 0.9285714
## [155] 1.0000000 0.9285714 1.0000000 0.9285714 0.8571429 0.7857143 0.7857143
## [162] 0.8571429 1.0000000 0.7857143 0.9285714 0.7142857 0.8571429 1.0000000
## [169] 1.0000000 1.0000000 0.7857143 0.7142857 0.8571429 0.7857143 0.8571429
## [176] 0.7857143 1.0000000 0.9285714 0.9285714 0.7857143 0.7857143 0.7857143
## [183] 0.7857143 0.9285714 1.0000000 0.7142857 0.8571429 0.8571429 0.9285714
## [190] 0.8571429 0.8571429 1.0000000 0.8571429 0.9285714 0.7142857 0.9285714
## [197] 0.9285714 0.9285714
## Levels: Poor Fair Good
```

```
summary(i4.data.pred)
```

```
## Poor Fair Good
##    2  196    0
```

```
Confusion_Mat = confusionMatrix(data = i4.data.pred, reference = i4.data[,1], dnn = c("Predict
ion", "Reference"))
Confusion_Mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Poor Fair Good
##       Poor    2    0    0
##       Fair   29  153   14
##       Good    0    0    0
##
## Overall Statistics
##
##               Accuracy : 0.7828
```

```
##                  95% CI : (0.7188, 0.8381)
##     No Information Rate : 0.7727
##     P-Value [Acc > NIR] : 0.4053
##
##                   Kappa : 0.0699
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Poor Class: Fair Class: Good
## Sensitivity              0.06452     1.00000     0.00000
## Specificity              1.00000     0.04444     1.00000
## Pos Pred Value           1.00000     0.78061         NaN
## Neg Pred Value           0.85204     1.00000     0.92929
## Prevalence               0.15657     0.77273     0.07071
## Detection Rate           0.01010     0.77273     0.00000
## Detection Prevalence     0.01010     0.98990     0.00000
## Balanced Accuracy        0.53226     0.52222     0.50000
```

```
out<-capture.output(Confusion_Mat)
setwd("C:/Users/carle/Documents/Results/KNN Results")
cat(out,file="v4.1 KNN Confusion Matrix",sep="/n",append=TRUE)
```

MLR

Final_MLR_Script.R

carle

Wed Oct 10 22:02:24 2018

```
##############################
####        MLR Script     ####
##############################

set.seed(1234)
setwd("C:/Users/carle/Documents/Data")
getwd()
```

```
## [1] "C:/Users/carle/Documents/Data"
```

```
# Load required packages (install first if not previously done so)
library("e1071")
```

```
## Warning: package 'e1071' was built under R version 3.4.3
```

```
library(splitstackshape)
```

```
## Warning: package 'splitstackshape' was built under R version 3.4.4
```

```
library(dplyr) # To split data sets with proprtional amounts of classes
```

```
## Warning: package 'dplyr' was built under R version 3.4.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# na.strings = "NA" populates every blank with "NA", which R will ignore
# stringsAsFactors = TRUE will let you read in 0's as real zeroes
# check.names = FALSE permits multiple words in Column names to be seperated by a space.
# ^ [R] prefers names to read as one word (syntactically valid): "a.b.c" rather than "a b c"
```

```
# Load Data - Script is prepared for the 2010|2015 Dataset
data <- read.csv(file.choose(), row.names = 1, check.names = FALSE, na.strings = "NA", sep = "
,", stringsAsFactors = TRUE, header = TRUE)

#Rename columns to ensure they have 'syntactically valid' names
cnames <- cbind("x.2010.EUI","x.2015.EUI","D.EUI.percent","x.2010.Qualitative.EUI","x.2015.Qua
litative.EUI",
                "D.Qualitative.EUI","x.2010.Actual.E_C.Energy","x.2015.Actual.E_C","D.Actual.E
_C","x.2010.Actual.Thermal","x.2015.Actual.Thermal","D.Actual.Thermal",
                "x.2010.Actual.Energy.Use","D.Actual.Energy.Use","x.2015.Actual.Energy.Use", "
Portfolio.Manager","Portfolio.Manager.no.","Net.Rentable.Area",
                "Exterior.Area", "Gross.Floor.Area","Enclosed.Parking","Latitude",  "Longitude
","Construction.Year","No.of.Structures","Building.Class","Closest.Major.City","Climate.Zone",
                "Electrically.Heated","Cooling.Tower","Soft.Landscaping",   "Occupant.Density"
,"Vacancy.Rate","Weekly.Operating.Hours")
colnames(data) <-cnames
rnames <- row.names(data)

# Order Qualitative outcome variables
sapply(data, class)
```

```
##               x.2010.EUI               x.2015.EUI            D.EUI.percent
##                "numeric"                "numeric"                "numeric"
##   x.2010.Qualitative.EUI   x.2015.Qualitative.EUI        D.Qualitative.EUI
##                 "factor"                 "factor"                 "factor"
## x.2010.Actual.E_C.Energy        x.2015.Actual.E_C             D.Actual.E_C
##                "numeric"                "numeric"                "numeric"
##    x.2010.Actual.Thermal    x.2015.Actual.Thermal         D.Actual.Thermal
##                "numeric"                "numeric"                "numeric"
## x.2010.Actual.Energy.Use      D.Actual.Energy.Use x.2015.Actual.Energy.Use
##                "numeric"                "numeric"                "numeric"
##        Portfolio.Manager    Portfolio.Manager.no.        Net.Rentable.Area
##                 "factor"                "integer"                "integer"
##            Exterior.Area         Gross.Floor.Area         Enclosed.Parking
##                "integer"                "integer"                "integer"
##                 Latitude                Longitude        Construction.Year
##                "numeric"                "numeric"                "integer"
##         No.of.Structures           Building.Class       Closest.Major.City
##                "integer"                "integer"                 "factor"
##             Climate.Zone      Electrically.Heated            Cooling.Tower
##                "integer"                "integer"                "integer"
##         Soft.Landscaping         Occupant.Density             Vacancy.Rate
##                "integer"                "numeric"                "numeric"
##   Weekly.Operating.Hours
##                "numeric"
```

```
data[,c("x.2010.Qualitative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI")] <- lapply(data
[,c("x.2010.Qualitative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI")] , factor)
data[,"x.2010.Qualitative.EUI"]=ordered(data[,"x.2010.Qualitative.EUI"],levels = c("Poor","Fai
r","Good"))
data[,"x.2015.Qualitative.EUI"]=ordered(data[,"x.2015.Qualitative.EUI"],levels = c("Poor","Fai
r","Good"))
```

```
data[,"D.Qualitative.EUI"]=ordered(data[,"D.Qualitative.EUI"],levels = c("Poor","Fair","Good")
)
sapply(data[,c("x.2010.Qualitative.EUI","x.2015.Qualitative.EUI","D.Qualitative.EUI")],is.orde
red)
```

```
## x.2010.Qualitative.EUI x.2015.Qualitative.EUI      D.Qualitative.EUI
##                   TRUE                   TRUE                   TRUE
```

```
#Remove outliers (as determined from PCA)
outliers<-c("256","186","289")
data<-data[!rownames(data) %in% outliers,]

#Create functions for normalization and denormalization of the data columns
normalize <- function(x){(x-min(x))/(max(x)-min(x))}
denormalize <- function(x,y){(x)*(max(max.min[,y])-min(max.min[,y]))+min(max.min[,y])}

#Record maximums and minimums for each column in order to normalize the data and denormalize t
he model results
numerics <- unlist(lapply(data, is.numeric))
Maximums <- as.data.frame(lapply(data[,numerics], max))
Minimums <- as.data.frame(lapply(data[,numerics], min))
max.min <- rbind.data.frame(Maximums,Minimums)

##############################
####     MLR iteration 1   ####
##############################

# Select outcome variable (i.e. x.2010.EUI) and independent variables for iteration 1 (i1)
i1.data <-  data[,c("x.2010.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
                    "Portfolio.Manager.no.", "Latitude", "Longitude",    "Construction.Year",
  "Occupant.Density",
                    "Vacancy.Rate",  "Weekly.Operating.Hours")]
rnames  <-   row.names(i1.data)

# Normalize
i1.norm.data <-as.data.frame(lapply(i1.data, normalize))
row.names(i1.norm.data) <- rnames

# Create Train and Test set
train.set = as.data.frame(stratified(data, "x.2010.Qualitative.EUI", size=.7, replace = FALSE,
                 bothSets = FALSE, keep.rownames=TRUE))
train.rows = train.set$rn
i1.data.train = i1.norm.data %>% filter(row.names(i1.norm.data) %in% c(train.rows))
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.3
```

```
train = sample(1:nrow(i1.data.train), nrow(i1.data.train)*1)
i1.data.test=i1.norm.data[-train,]

# Train MLR model on iteration 1 training dataset#
i1.mlr <- lm(formula = x.2010.EUI~Portfolio.Manager.no.+Latitude+Longitude+Construction.Year+O
```

```
ccupant.Density+
               Vacancy.Rate+Weekly.Operating.Hours, data=i1.data.train)

# Define function to Classify the predicted results according to Good, Fair, Poor
classification.accuracy <- function(w,x,y,z) {

  denrom.pr.mlr <- round(as.data.frame(denormalize(x,y)),0)
  Actual.EUI <- round(w[z,y],0)
  Residuals <- Actual.EUI-denrom.pr.mlr
  Compare <- as.data.frame(cbind(Actual.EUI,denrom.pr.mlr,Residuals))
  colnames(Compare)<-c('Actual.EUI','Predicted.EUI','Residuals')
  Compare$Actual.Class = ifelse((Compare$Actual.EUI >= 40 ),"Poor",
                                ifelse((Compare$Actual.EUI > 20 & Compare$Actual.EUI <40),"Fai
r",
                                       ifelse((Compare$Actual.EUI <=20),"Good",0)))
  Compare$Predicted.Class = ifelse((Compare$Predicted.EUI >= 40 ),"Poor",
                                   ifelse((Compare$Predicted.EUI > 20 & Compare$Predicted.EUI
<40),"Fair",
                                          ifelse((Compare$Predicted.EUI >=20),"Good",0)))
  Compare$Correct = ifelse((Compare$Actual.EUI >= 40 & Compare$Predicted.EUI >= 40 ),1,
                           ifelse((Compare$Actual.EUI > 20&Compare$Actual.EUI <40 & Compare$Pr
edicted.EUI > 20 & Compare$Predicted.EUI <40),1,
                                  ifelse((Compare$Actual.EUI <20 & Compare$Predicted.EUI <20),
1,0)))
  Compare$Poor.Correct = ifelse((Compare$Actual.EUI >= 40 & Compare$Predicted.EUI >= 40 ),1,0)
  Compare$Fair.Correct = ifelse((Compare$Actual.EUI > 20&Compare$Actual.EUI <40 & Compare$Pred
icted.EUI > 20 & Compare$Predicted.EUI <40),1,0)
  Compare$Good.Correct = ifelse((Compare$Actual.EUI <20 & Compare$Predicted.EUI <20),1,0)
  Class.Accuracy<- round((sum(Compare$Correct))/(nrow(Compare))*100,2)
  Poor.Accuracy<- round((sum(Compare$Poor.Correct))/(sum(Compare$Actual.Class=="Poor"))*100,2)
  Fair.Accuracy<- round((sum(Compare$Fair.Correct))/(sum(Compare$Actual.Class=="Fair"))*100,2)
  Good.Accuracy<- round((sum(Compare$Good.Correct))/(sum(Compare$Actual.Class=="Good"))*100,2)
  Accuracy.Rates <- c("Class.Accuracy" = Class.Accuracy, "Poor.Accuracy" = Poor.Accuracy,"Fair
.Accuracy" = Fair.Accuracy,"Good.Accuracy" = Good.Accuracy)
  Results=list("AccuracyRates" = Accuracy.Rates,"Compare.Results" = Compare)
  return(Results)
}

# Classify the predicted results of the when applied to the training dataset
i1.train.Results =classification.accuracy(w=i1.data, x=i1.mlr$fitted.values, y="x.2010.EUI",z=
train)

# Predict and Classify the results of the model when applied to the test dataset
pred.mlr <- (predict(i1.mlr,i1.data.test[,-1]))
i1.test.Results =classification.accuracy(w=i1.data,x=pred.mlr, y="x.2010.EUI",z=-train)

# View and Record normalized Prediction Results
i1.train.Results$Compare.Results
```

```
##     Actual.EUI Predicted.EUI Residuals Actual.Class Predicted.Class
## 1           25            33        -8         Fair            Fair
## 2           20            28        -8         Good            Fair
## 3           24            27        -3         Fair            Fair
```

```
## 4           27           26          1         Fair           Fair
## 5           25           27         -2         Fair           Fair
## 6           29           33         -4         Fair           Fair
## 7           37           34          3         Fair           Fair
## 8           26           36        -10         Fair           Fair
## 9           31           36         -5         Fair           Fair
## 10          45           39          6         Poor           Fair
## 11          40           39          1         Poor           Fair
## 12          38           38          0         Fair           Fair
## 13          28           25          3         Fair           Fair
## 14          33           25          8         Fair           Fair
## 15          39           29         10         Fair           Fair
## 16          44           28         16         Poor           Fair
## 17          26           41        -15         Fair           Poor
## 18          56           30         26         Poor           Fair
## 19          22           32        -10         Fair           Fair
## 20          17           26         -9         Good           Fair
## 21          23           25         -2         Fair           Fair
## 22          25           23          2         Fair           Fair
## 23          31           25          6         Fair           Fair
## 24          32           40         -8         Fair           Poor
## 25          48           34         14         Poor           Fair
## 26          28           37         -9         Fair           Fair
## 27          43           39          4         Poor           Fair
## 28          27           37        -10         Fair           Fair
## 29          29           41        -12         Fair           Poor
## 30          30           40        -10         Fair           Poor
## 31          28           24          4         Fair           Fair
## 32          36           24         12         Fair           Fair
## 33          21           35        -14         Fair           Fair
## 34          36           25         11         Fair           Fair
## 35          25           22          3         Fair           Fair
## 36          54           25         29         Poor           Fair
## 37          64           25         39         Poor           Fair
## 38          20           40        -20         Good           Poor
## 39          23           27         -4         Fair           Fair
## 40          29           35         -6         Fair           Fair
## 41          16           29        -13         Good           Fair
## 42          12           27        -15         Good           Fair
## 43          16           25         -9         Good           Fair
## 44          21           34        -13         Fair           Fair
## 45          27           27          0         Fair           Fair
## 46          24           26         -2         Fair           Fair
## 47          26           27         -1         Fair           Fair
## 48          42           26         16         Poor           Fair
## 49          15           25        -10         Good           Fair
## 50          22           36        -14         Fair           Fair
## 51          34           36         -2         Fair           Fair
## 52          46           36         10         Poor           Fair
## 53          19           36        -17         Good           Fair
## 54          23           36        -13         Fair           Fair
## 55          36           35          1         Fair           Fair
## 56          27           35         -8         Fair           Fair
## 57          30           35         -5         Fair           Fair
```

```
## 58          34          36         -2         Fair          Fair
## 59          27          36         -9         Fair          Fair
## 60          23          37        -14         Fair          Fair
## 61          26          26          0         Fair          Fair
## 62          18          25         -7         Good          Fair
## 63          38          26         12         Fair          Fair
## 64          25          33         -8         Fair          Fair
## 65          26          25          1         Fair          Fair
## 66          46          25         21         Poor          Fair
## 67          44          29         15         Poor          Fair
## 68          27          37        -10         Fair          Fair
## 69          29          23          6         Fair          Fair
## 70          30          28          2         Fair          Fair
## 71          47          35         12         Poor          Fair
## 72          40          28         12         Poor          Fair
## 73          44          27         17         Poor          Fair
## 74          30          25          5         Fair          Fair
## 75          35          26          9         Fair          Fair
## 76          50          39         11         Poor          Fair
## 77          21          39        -18         Fair          Fair
## 78          60          40         20         Poor          Poor
## 79          45          40          5         Poor          Poor
## 80          28          26          2         Fair          Fair
## 81          33          26          7         Fair          Fair
## 82          38          25         13         Fair          Fair
## 83          34          26          8         Fair          Fair
## 84          28          25          3         Fair          Fair
## 85          24          26         -2         Fair          Fair
## 86          25          22          3         Fair          Fair
## 87          20          32        -12         Good          Fair
## 88          57          32         25         Poor          Fair
## 89          27          38        -11         Fair          Fair
## 90          29          36         -7         Fair          Fair
## 91          38          39         -1         Fair          Fair
## 92          31          35         -4         Fair          Fair
## 93          19          30        -11         Good          Fair
## 94          29          26          3         Fair          Fair
## 95          28          24          4         Fair          Fair
## 96          30          26          4         Fair          Fair
## 97          27          26          1         Fair          Fair
## 98          21          25         -4         Fair          Fair
## 99          21          28         -7         Fair          Fair
## 100         29          28          1         Fair          Fair
## 101         25          29         -4         Fair          Fair
## 102         34          27          7         Fair          Fair
## 103         29          30         -1         Fair          Fair
## 104         28          25          3         Fair          Fair
## 105         22          22          0         Fair          Fair
## 106         37          39         -2         Fair          Fair
## 107         31          26          5         Fair          Fair
## 108         65          24         41         Poor          Fair
## 109         22          27         -5         Fair          Fair
## 110         40          25         15         Poor          Fair
## 111         26          26          0         Fair          Fair
```

```
## 112         42            37         5         Poor            Fair
## 113         40            38         2         Poor            Fair
## 114         28            39       -11         Fair            Fair
## 115         20            36       -16         Good            Fair
## 116         30            36        -6         Fair            Fair
## 117         42            35         7         Poor            Fair
## 118         16            36       -20         Good            Fair
## 119         37            33         4         Fair            Fair
## 120         24            33        -9         Fair            Fair
## 121         46            34        12         Poor            Fair
## 122         23            33       -10         Fair            Fair
## 123         33            29         4         Fair            Fair
## 124         26            33        -7         Fair            Fair
## 125         21            32       -11         Fair            Fair
## 126         33            25         8         Fair            Fair
## 127         51            22        29         Poor            Fair
## 128         33            24         9         Fair            Fair
## 129         24            37       -13         Fair            Fair
## 130         32            37        -5         Fair            Fair
## 131         19            38       -19         Good            Fair
## 132         28            24         4         Fair            Fair
## 133         22            27        -5         Fair            Fair
## 134         32            25         7         Fair            Fair
## 135         30            26         4         Fair            Fair
## 136         26            20         6         Fair            Good
## 137         26            27        -1         Fair            Fair
## 138         36            40        -4         Fair            Poor
## 139         29            32        -3         Fair            Fair
##     Correct Poor.Correct Fair.Correct Good.Correct
## 1         1            0            1            0
## 2         0            0            0            0
## 3         1            0            1            0
## 4         1            0            1            0
## 5         1            0            1            0
## 6         1            0            1            0
## 7         1            0            1            0
## 8         1            0            1            0
## 9         1            0            1            0
## 10        0            0            0            0
## 11        0            0            0            0
## 12        1            0            1            0
## 13        1            0            1            0
## 14        1            0            1            0
## 15        1            0            1            0
## 16        0            0            0            0
## 17        0            0            0            0
## 18        0            0            0            0
## 19        1            0            1            0
## 20        0            0            0            0
## 21        1            0            1            0
## 22        1            0            1            0
## 23        1            0            1            0
## 24        0            0            0            0
## 25        0            0            0            0
```

```
## 26        1              0              1              0
## 27        0              0              0              0
## 28        1              0              1              0
## 29        0              0              0              0
## 30        0              0              0              0
## 31        1              0              1              0
## 32        1              0              1              0
## 33        1              0              1              0
## 34        1              0              1              0
## 35        1              0              1              0
## 36        0              0              0              0
## 37        0              0              0              0
## 38        0              0              0              0
## 39        1              0              1              0
## 40        1              0              1              0
## 41        0              0              0              0
## 42        0              0              0              0
## 43        0              0              0              0
## 44        1              0              1              0
## 45        1              0              1              0
## 46        1              0              1              0
## 47        1              0              1              0
## 48        0              0              0              0
## 49        0              0              0              0
## 50        1              0              1              0
## 51        1              0              1              0
## 52        0              0              0              0
## 53        0              0              0              0
## 54        1              0              1              0
## 55        1              0              1              0
## 56        1              0              1              0
## 57        1              0              1              0
## 58        1              0              1              0
## 59        1              0              1              0
## 60        1              0              1              0
## 61        1              0              1              0
## 62        0              0              0              0
## 63        1              0              1              0
## 64        1              0              1              0
## 65        1              0              1              0
## 66        0              0              0              0
## 67        0              0              0              0
## 68        1              0              1              0
## 69        1              0              1              0
## 70        1              0              1              0
## 71        0              0              0              0
## 72        0              0              0              0
## 73        0              0              0              0
## 74        1              0              1              0
## 75        1              0              1              0
## 76        0              0              0              0
## 77        1              0              1              0
## 78        1              1              0              0
## 79        1              1              0              0
```

```
## 80        1             0             1             0
## 81        1             0             1             0
## 82        1             0             1             0
## 83        1             0             1             0
## 84        1             0             1             0
## 85        1             0             1             0
## 86        1             0             1             0
## 87        0             0             0             0
## 88        0             0             0             0
## 89        1             0             1             0
## 90        1             0             1             0
## 91        1             0             1             0
## 92        1             0             1             0
## 93        0             0             0             0
## 94        1             0             1             0
## 95        1             0             1             0
## 96        1             0             1             0
## 97        1             0             1             0
## 98        1             0             1             0
## 99        1             0             1             0
## 100       1             0             1             0
## 101       1             0             1             0
## 102       1             0             1             0
## 103       1             0             1             0
## 104       1             0             1             0
## 105       1             0             1             0
## 106       1             0             1             0
## 107       1             0             1             0
## 108       0             0             0             0
## 109       1             0             1             0
## 110       0             0             0             0
## 111       1             0             1             0
## 112       0             0             0             0
## 113       0             0             0             0
## 114       1             0             1             0
## 115       0             0             0             0
## 116       1             0             1             0
## 117       0             0             0             0
## 118       0             0             0             0
## 119       1             0             1             0
## 120       1             0             1             0
## 121       0             0             0             0
## 122       1             0             1             0
## 123       1             0             1             0
## 124       1             0             1             0
## 125       1             0             1             0
## 126       1             0             1             0
## 127       0             0             0             0
## 128       1             0             1             0
## 129       1             0             1             0
## 130       1             0             1             0
## 131       0             0             0             0
## 132       1             0             1             0
## 133       1             0             1             0
```

```
## 134       1            0            1            0
## 135       1            0            1            0
## 136       0            0            0            0
## 137       1            0            1            0
## 138       0            0            0            0
## 139       1            0            1            0
```

```
i1.test.Results$Compare.Results
```

```
##      Actual.EUI Predicted.EUI Residuals Actual.Class Predicted.Class
## 506          23            28        -5         Fair            Fair
## 507          30            28         2         Fair            Fair
## 508          41            29        12         Poor            Fair
## 509          57            27        30         Poor            Fair
## 512          24            30        -6         Fair            Fair
## 513          27            30        -3         Fair            Fair
## 514          37            28         9         Fair            Fair
## 516          26            27        -1         Fair            Fair
## 517          27            27         0         Fair            Fair
## 518          23            26        -3         Fair            Fair
## 535          24            25        -1         Fair            Fair
## 543          43            22        21         Poor            Fair
## 547          31            39        -8         Fair            Fair
## 559          22            26        -4         Fair            Fair
## 560          19            24        -5         Good            Fair
## 562          23            27        -4         Fair            Fair
## 565          17            25        -8         Good            Fair
## 569          26            26         0         Fair            Fair
## 570          20            26        -6         Good            Fair
## 582          36            37        -1         Fair            Fair
## 635          45            38         7         Poor            Fair
## 636          34            39        -5         Fair            Fair
## 637          26            35        -9         Fair            Fair
## 638          25            39       -14         Fair            Fair
## 639          54            39        15         Poor            Fair
## 640          27            35        -8         Fair            Fair
## 641          37            36         1         Fair            Fair
## 643          27            36        -9         Fair            Fair
## 644          30            35        -5         Fair            Fair
## 645          27            33        -6         Fair            Fair
## 647          27            33        -6         Fair            Fair
## 648          29            36        -7         Fair            Fair
## 649          26            33        -7         Fair            Fair
## 650          27            33        -6         Fair            Fair
## 651          31            34        -3         Fair            Fair
## 652          26            32        -6         Fair            Fair
## 653          23            33       -10         Fair            Fair
## 654          28            33        -5         Fair            Fair
## 655          21            29        -8         Fair            Fair
## 658          28            33        -5         Fair            Fair
## 659          26            32        -6         Fair            Fair
## 662          26            25         1         Fair            Fair
## 663          18            22        -4         Good            Fair
```

```
## 664          31            24          7         Fair            Fair
## 667          34            37         -3         Fair            Fair
## 668          34            37         -3         Fair            Fair
## 671          33            38         -5         Fair            Fair
## 673          30            39         -9         Fair            Fair
## 675          34            36         -2         Fair            Fair
## 682          32            24          8         Fair            Fair
## 686          28            27          1         Fair            Fair
## 689          20            25         -5         Good            Fair
## 695          27            26          1         Fair            Fair
## 699          42            25         17         Poor            Fair
## 722          25            26         -1         Fair            Fair
## 861          25            20          5         Fair            Good
## 965          30            27          3         Fair            Fair
## 1282         29            40        -11         Fair            Poor
## 1286         23            32         -9         Fair            Fair
##      Correct Poor.Correct Fair.Correct Good.Correct
## 506        1            0            1            0
## 507        1            0            1            0
## 508        0            0            0            0
## 509        0            0            0            0
## 512        1            0            1            0
## 513        1            0            1            0
## 514        1            0            1            0
## 516        1            0            1            0
## 517        1            0            1            0
## 518        1            0            1            0
## 535        1            0            1            0
## 543        0            0            0            0
## 547        1            0            1            0
## 559        1            0            1            0
## 560        0            0            0            0
## 562        1            0            1            0
## 565        0            0            0            0
## 569        1            0            1            0
## 570        0            0            0            0
## 582        1            0            1            0
## 635        0            0            0            0
## 636        1            0            1            0
## 637        1            0            1            0
## 638        1            0            1            0
## 639        0            0            0            0
## 640        1            0            1            0
## 641        1            0            1            0
## 643        1            0            1            0
## 644        1            0            1            0
## 645        1            0            1            0
## 647        1            0            1            0
## 648        1            0            1            0
## 649        1            0            1            0
## 650        1            0            1            0
## 651        1            0            1            0
## 652        1            0            1            0
## 653        1            0            1            0
```

```
## 654        1             0             1             0
## 655        1             0             1             0
## 658        1             0             1             0
## 659        1             0             1             0
## 662        1             0             1             0
## 663        0             0             0             0
## 664        1             0             1             0
## 667        1             0             1             0
## 668        1             0             1             0
## 671        1             0             1             0
## 673        1             0             1             0
## 675        1             0             1             0
## 682        1             0             1             0
## 686        1             0             1             0
## 689        0             0             0             0
## 695        1             0             1             0
## 699        0             0             0             0
## 722        1             0             1             0
## 861        0             0             0             0
## 965        1             0             1             0
## 1282       0             0             0             0
## 1286       1             0             1             0
```

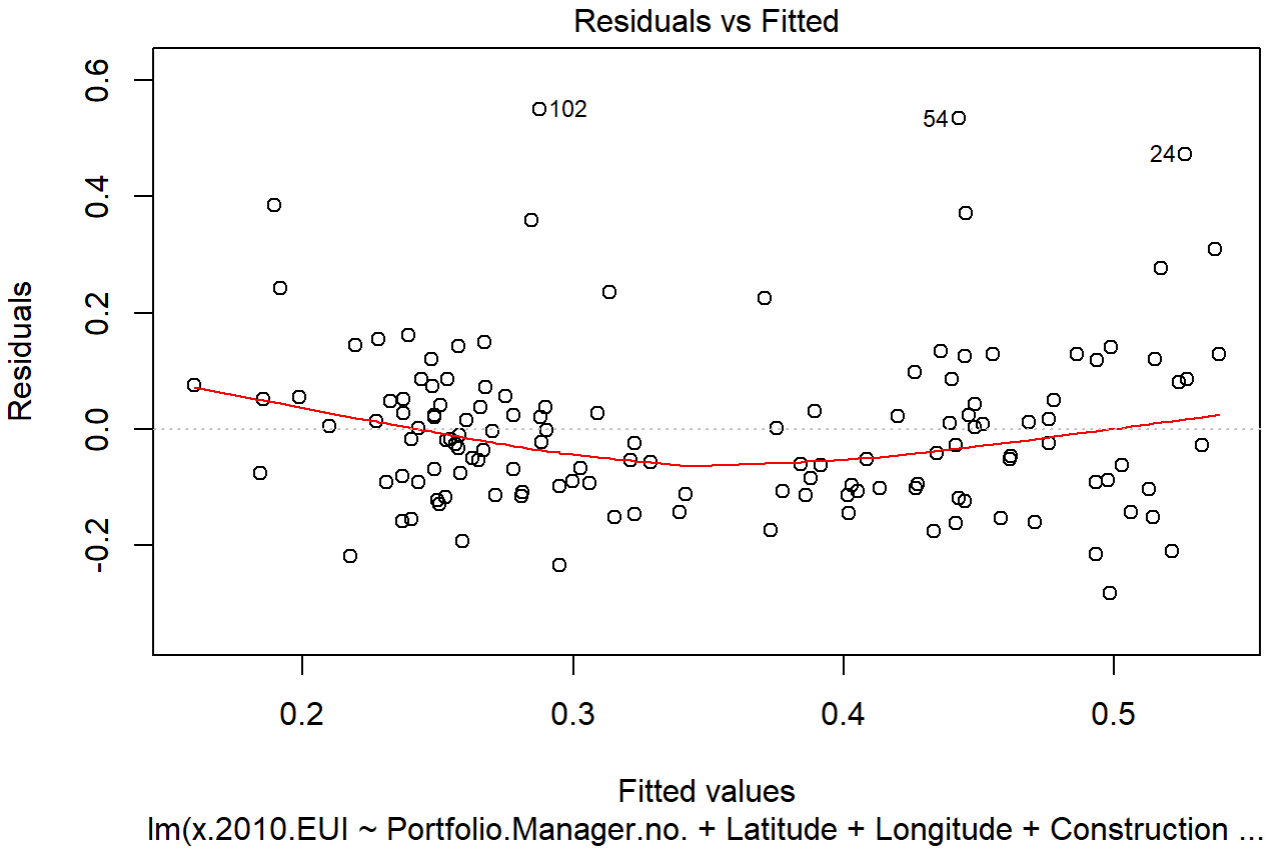
```
i1.train.Results$AccuracyRates
```

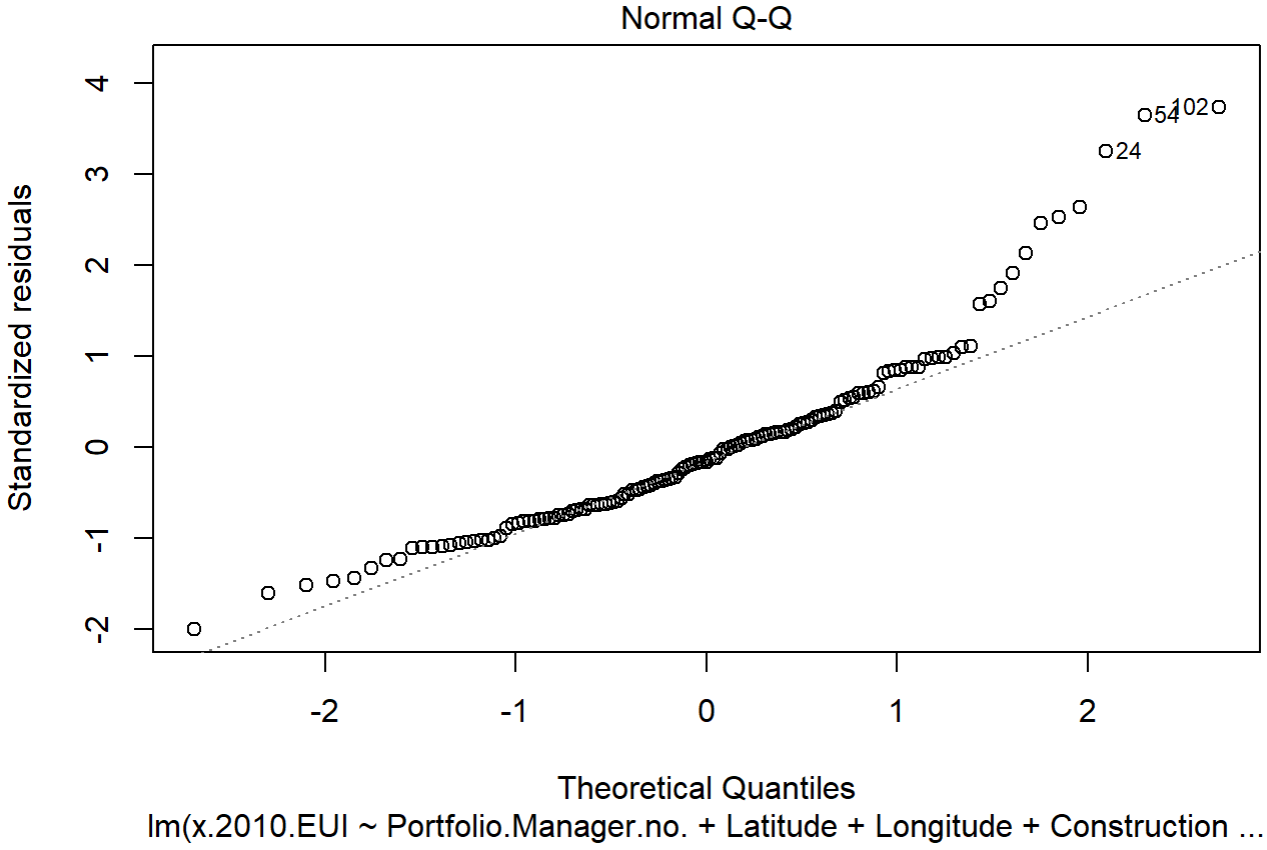
```
## Class.Accuracy  Poor.Accuracy  Fair.Accuracy  Good.Accuracy
##          68.35           7.69          93.94           0.00
```

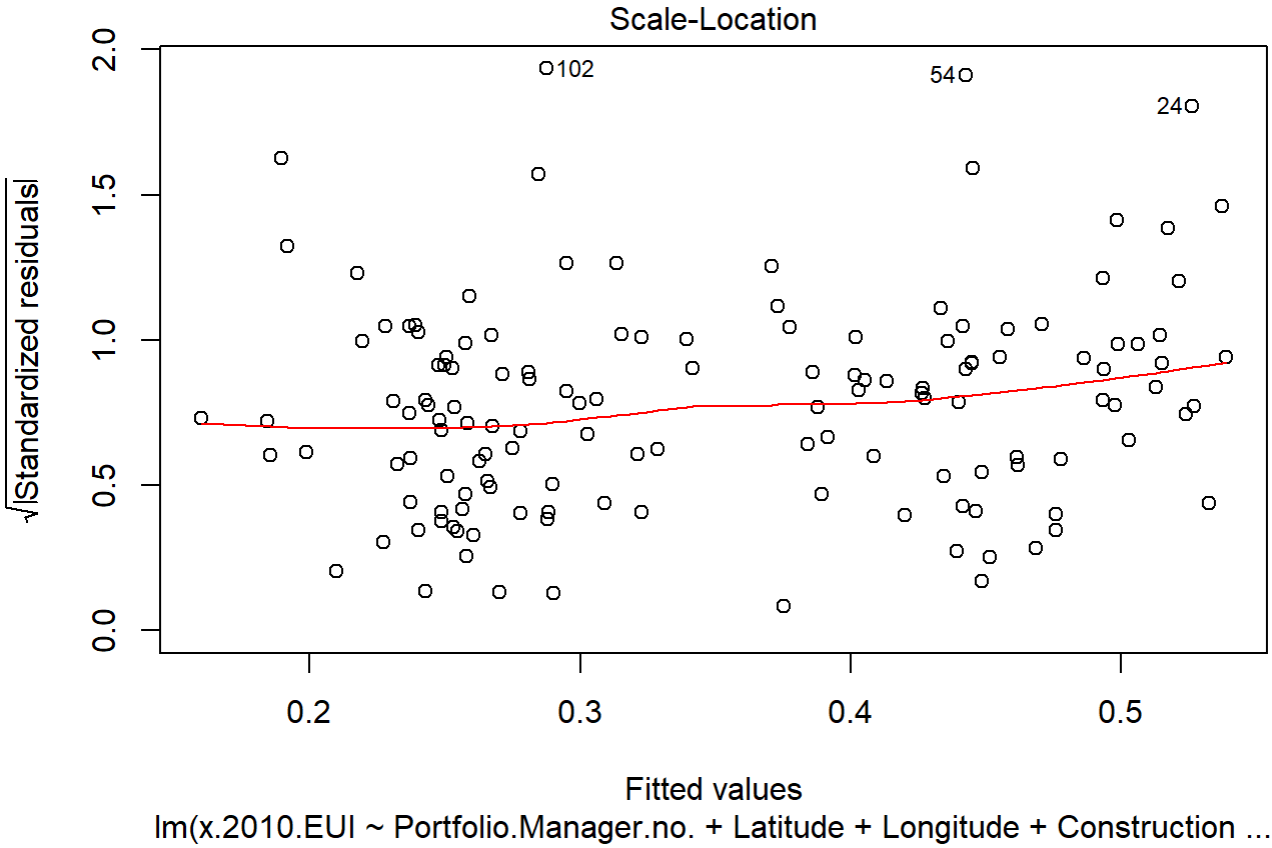
```
i1.test.Results$AccuracyRates
```

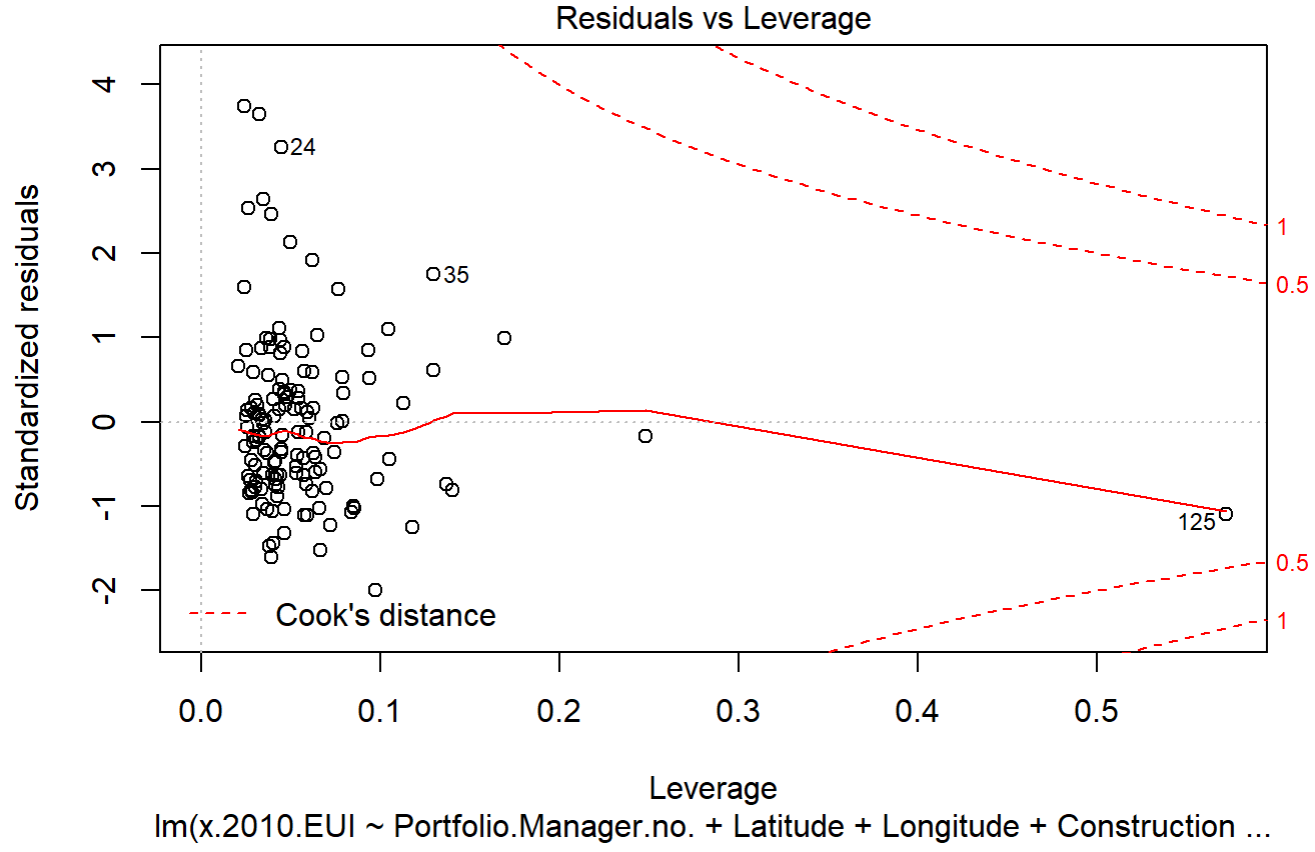
```
## Class.Accuracy  Poor.Accuracy  Fair.Accuracy  Good.Accuracy
##          77.97           0.00          95.83           0.00
```

```
write.csv(i1.train.Results$Compare.Results, file ="i1 - Train Results Summary")
write.csv(i1.test.Results$Compare.Results, file ="i1 - Test Results Summary")
plot(i1.mlr)
```


Residuals vs Fitted
Residuals
Fitted values
lm(x.2010.EUI ~ Portfolio.Manager.no. + Latitude + Longitude + Construction ...
102
54
24
0.6
0.4
0.2
0.0
-0.2
0.2
0.3
0.4
0.5


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(x.2010.EUI ~ Portfolio.Manager.no. + Latitude + Longitude + Construction ...
102
54
24
4
3
2
1
0
-1
-2
-2
-1
0
1
2


Scale-Location
102
54
24
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
0.2
0.3
0.4
0.5
Fitted values
lm(x.2010.EUI ~ Portfolio.Manager.no. + Latitude + Longitude + Construction ...


Residuals vs Leverage
24
35
125
Standardized residuals
4
3
2
1
0
-1
-2
Cook's distance
1
0.5
0.5
1
0.0
0.1
0.2
0.3
0.4
0.5
Leverage
lm(x.2010.EUI ~ Portfolio.Manager.no. + Latitude + Longitude + Construction ...

```
# Dernormalize, Plot and Record Prediction Results
denorm.train=denormalize(x=i1.mlr$fitted.values, y="x.2010.EUI")
Actual.train = denormalize(x=i1.data.train[,1],y="x.2010.EUI")

jpeg(filename="MLR, Actual vs. Predicted 2010 EUI (Train, 2010-2015 Dataset).jpg")
plot( Actual.train, denorm.train, main="Actual vs. Predicted 2010 EUI (Train, 2010-2015 Datase
t)", xlab="Actual Values", ylab="Predicted Values")
dev.off()
```

```
## png
##   2
```

```
denorm.test=denormalize(x=pred.mlr,y="x.2010.EUI")
Actual.test = denormalize(x=i1.data.test[,1],y="x.2010.EUI")

jpeg(filename="MLR, Actual vs. Predicted 2010 EUI (Test, 2010-2015 Dataset).jpg")
plot( Actual.test, denorm.test, main="Actual vs. Predicted 2010 EUI (Test, 2010-2015 Dataset)"
, xlab="Actual Values", ylab="Predicted Values")
dev.off()
```

```
## png
##   2
```

```
##############################
####     MLR iteration 2   ####
##############################

# Select outcome variable (i.e. x.2015.EUI) and independent variables for iteration 2 (i2)
i2.data<-data[,c("x.2015.EUI","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
                 "Portfolio.Manager.no.",   "Latitude", "Longitude",    "Construction.Year",
  "Occupant.Density",
                 "Vacancy.Rate",    "Weekly.Operating.Hours")]
rnames <- row.names(i2.data)

# Normalize
i2.norm.data <-as.data.frame(lapply(i2.data, normalize))
row.names(i2.norm.data) <- rnames

# Create Train and Test set
train.set = as.data.frame(stratified(data, "x.2015.Qualitative.EUI", size=.7, replace = FALSE,
                                     bothSets = FALSE, keep.rownames=TRUE))
train.rows = train.set$rn
i2.data.train = i2.norm.data %>% filter(row.names(i2.norm.data) %in% c(train.rows))
i2.train = sample(1:nrow(i2.data.train), nrow(i2.data.train)*1)
i2.data.test=i2.norm.data[-i2.train,]

# Train MLR model on iteration 2 training dataset
i2.mlr <- lm(formula = x.2015.EUI~Portfolio.Manager.no.+Latitude+Longitude+Construction.Year+O
ccupant.Density+
               Vacancy.Rate+Weekly.Operating.Hours, data=i2.data.train)
# Classify the predicted results of the when applied to the training dataset
```

```
i2.train.Results =classification.accuracy(w=i2.data,x=i2.mlr$fitted.values, y="x.2015.EUI",z=i
2.train)

# Predict and Classify the results of the model when applied to the test dataset
pred.mlr <- (predict(i2.mlr,i2.data.test[,-1]))
i2.test.Results =classification.accuracy(w=i2.data,x=pred.mlr, y="x.2015.EUI",z=-i2.train)

# View and Record normalized Prediction Results
i2.train.Results$Compare.Results
```

```
##     Actual.EUI Predicted.EUI Residuals Actual.Class Predicted.Class
## 1           30            27         3         Fair            Fair
## 2           17            32       -15         Good            Fair
## 3           24            23         1         Fair            Fair
## 4           20            21        -1         Good            Fair
## 5           20            31       -11         Good            Fair
## 6           12            33       -21         Good            Fair
## 7           60            43        17         Poor            Poor
## 8           12            41       -29         Good            Poor
## 9           16            38       -22         Good            Fair
## 10          26            38       -12         Fair            Fair
## 11          20            23        -3         Good            Fair
## 12          15            29       -14         Good            Fair
## 13          26            28        -2         Fair            Fair
## 14          52            29        23         Poor            Fair
## 15          23            26        -3         Fair            Fair
## 16          40            43        -3         Poor            Poor
## 17          22            24        -2         Fair            Fair
## 18          17            20        -3         Good            Good
## 19          26            21         5         Fair            Fair
## 20          17            21        -4         Good            Fair
## 21          19            41       -22         Good            Poor
## 22          19            44       -25         Good            Poor
## 23          17            39       -22         Good            Fair
## 24          24            37       -13         Fair            Fair
## 25          16            41       -25         Good            Poor
## 26          32            21        11         Fair            Fair
## 27          19            20        -1         Good            Good
## 28          23            20         3         Fair            Good
## 29          25            18         7         Fair               0
## 30          51            22        29         Poor            Fair
## 31          23            21         2         Fair            Fair
## 32          25            22         3         Fair            Fair
## 33          22            41       -19         Fair            Poor
## 34          31            22         9         Fair            Fair
## 35          34            32         2         Fair            Fair
## 36          21            17         4         Fair               0
## 37          26            23         3         Fair            Fair
## 38          40            19        21         Poor               0
## 39          22            31        -9         Fair            Fair
## 40          23            24        -1         Fair            Fair
## 41          24            21         3         Fair            Fair
## 42          21            20         1         Fair            Good
```

```
## 43          26          19           7         Fair             0
## 44          19          22          -3         Good          Fair
## 45          21          18           3         Fair             0
## 46          73          22          51         Poor          Fair
## 47          24          20           4         Fair          Good
## 48          28          23           5         Fair          Fair
## 49          38          36           2         Fair          Fair
## 50          14          34         -20         Good          Fair
## 51          74          34          40         Poor          Fair
## 52          34          35          -1         Fair          Fair
## 53          25          34          -9         Fair          Fair
## 54          25          32          -7         Fair          Fair
## 55          22          34         -12         Fair          Fair
## 56          33          36          -3         Fair          Fair
## 57          22          21           1         Fair          Fair
## 58          18          33         -15         Good          Fair
## 59          27          21           6         Fair          Fair
## 60          42          20          22         Poor          Good
## 61          29          27           2         Fair          Fair
## 62          35          36          -1         Fair          Fair
## 63          34          15          19         Fair             0
## 64          57          25          32         Poor          Fair
## 65          41          43          -2         Poor          Poor
## 66          22          25          -3         Fair          Fair
## 67          18          25          -7         Good          Fair
## 68          24          23           1         Fair          Fair
## 69          14          20          -6         Good          Good
## 70          52          22          30         Poor          Fair
## 71          52          41          11         Poor          Poor
## 72          16          38         -22         Good          Fair
## 73          23          41         -18         Fair          Poor
## 74          30          48         -18         Fair          Poor
## 75          31          21          10         Fair          Fair
## 76          19          21          -2         Good          Fair
## 77          27          23           4         Fair          Fair
## 78          53          21          32         Poor          Fair
## 79          27          30          -3         Fair          Fair
## 80          51          20          31         Poor          Good
## 81          19          22          -3         Good          Fair
## 82          13          21          -8         Good          Fair
## 83          23          24          -1         Fair          Fair
## 84          27          18           9         Fair             0
## 85          16          15           1         Good             0
## 86          21          22          -1         Fair          Fair
## 87          22          25          -3         Fair          Fair
## 88          20          26          -6         Good          Fair
## 89          29          31          -2         Fair          Fair
## 90          21          26          -5         Fair          Fair
## 91          33          41          -8         Fair          Poor
## 92          21          25          -4         Fair          Fair
## 93          15          22          -7         Good          Fair
## 94          18          22          -4         Good          Fair
## 95          33          22          11         Fair          Fair
## 96          35          20          15         Fair          Good
```

```
## 97          59            39        20         Poor            Fair
## 98          17            26        -9         Good            Fair
## 99          27            27         0         Fair            Fair
## 100         18            25        -7         Good            Fair
## 101         28            27         1         Fair            Fair
## 102         15            30       -15         Good            Fair
## 103         26            29        -3         Fair            Fair
## 104         24            28        -4         Fair            Fair
## 105         19            23        -4         Good            Fair
## 106         22            25        -3         Fair            Fair
## 107         23            20         3         Fair            Good
## 108         25            22         3         Fair            Fair
## 109         18            18         0         Good               0
## 110         24            22         2         Fair            Fair
## 111         23            22         1         Fair            Fair
## 112         25            38       -13         Fair            Fair
## 113         23            40       -17         Fair            Poor
## 114         34            40        -6         Fair            Poor
## 115         40            35         5         Poor            Fair
## 116         24            34       -10         Fair            Fair
## 117         14            29       -15         Good            Fair
## 118         15            29       -14         Good            Fair
## 119         23            30        -7         Fair            Fair
## 120         60            30        30         Poor            Fair
## 121         20            27        -7         Good            Fair
## 122         26            29        -3         Fair            Fair
## 123         31            28         3         Fair            Fair
## 124         19            24        -5         Good            Fair
## 125         18            33       -15         Good            Fair
## 126         43            31        12         Poor            Fair
## 127         22            21         1         Fair            Fair
## 128         22            15         7         Fair               0
## 129         68            18        50         Poor               0
## 130         24            35       -11         Fair            Fair
## 131         30            35        -5         Fair            Fair
## 132         35            38        -3         Fair            Fair
## 133         25            40       -15         Fair            Poor
## 134         22            35       -13         Fair            Fair
## 135         27            21         6         Fair            Fair
## 136         41            20        21         Poor            Good
## 137         37            15        22         Fair               0
## 138         19            23        -4         Good            Fair
## 139         26            30        -4         Fair            Fair
##     Correct Poor.Correct Fair.Correct Good.Correct
## 1         1            0            1            0
## 2         0            0            0            0
## 3         1            0            1            0
## 4         0            0            0            0
## 5         0            0            0            0
## 6         0            0            0            0
## 7         1            1            0            0
## 8         0            0            0            0
## 9         0            0            0            0
## 10        1            0            1            0
```

```
## 11        0             0             0             0
## 12        0             0             0             0
## 13        1             0             1             0
## 14        0             0             0             0
## 15        1             0             1             0
## 16        1             1             0             0
## 17        1             0             1             0
## 18        0             0             0             0
## 19        1             0             1             0
## 20        0             0             0             0
## 21        0             0             0             0
## 22        0             0             0             0
## 23        0             0             0             0
## 24        1             0             1             0
## 25        0             0             0             0
## 26        1             0             1             0
## 27        0             0             0             0
## 28        0             0             0             0
## 29        0             0             0             0
## 30        0             0             0             0
## 31        1             0             1             0
## 32        1             0             1             0
## 33        0             0             0             0
## 34        1             0             1             0
## 35        1             0             1             0
## 36        0             0             0             0
## 37        1             0             1             0
## 38        0             0             0             0
## 39        1             0             1             0
## 40        1             0             1             0
## 41        1             0             1             0
## 42        0             0             0             0
## 43        0             0             0             0
## 44        0             0             0             0
## 45        0             0             0             0
## 46        0             0             0             0
## 47        0             0             0             0
## 48        1             0             1             0
## 49        1             0             1             0
## 50        0             0             0             0
## 51        0             0             0             0
## 52        1             0             1             0
## 53        1             0             1             0
## 54        1             0             1             0
## 55        1             0             1             0
## 56        1             0             1             0
## 57        1             0             1             0
## 58        0             0             0             0
## 59        1             0             1             0
## 60        0             0             0             0
## 61        1             0             1             0
## 62        1             0             1             0
## 63        0             0             0             0
## 64        0             0             0             0
```

```
## 65        1             1             0             0
## 66        1             0             1             0
## 67        0             0             0             0
## 68        1             0             1             0
## 69        0             0             0             0
## 70        0             0             0             0
## 71        1             1             0             0
## 72        0             0             0             0
## 73        0             0             0             0
## 74        0             0             0             0
## 75        1             0             1             0
## 76        0             0             0             0
## 77        1             0             1             0
## 78        0             0             0             0
## 79        1             0             1             0
## 80        0             0             0             0
## 81        0             0             0             0
## 82        0             0             0             0
## 83        1             0             1             0
## 84        0             0             0             0
## 85        1             0             0             1
## 86        1             0             1             0
## 87        1             0             1             0
## 88        0             0             0             0
## 89        1             0             1             0
## 90        1             0             1             0
## 91        0             0             0             0
## 92        1             0             1             0
## 93        0             0             0             0
## 94        0             0             0             0
## 95        1             0             1             0
## 96        0             0             0             0
## 97        0             0             0             0
## 98        0             0             0             0
## 99        1             0             1             0
## 100       0             0             0             0
## 101       1             0             1             0
## 102       0             0             0             0
## 103       1             0             1             0
## 104       1             0             1             0
## 105       0             0             0             0
## 106       1             0             1             0
## 107       0             0             0             0
## 108       1             0             1             0
## 109       1             0             0             1
## 110       1             0             1             0
## 111       1             0             1             0
## 112       1             0             1             0
## 113       0             0             0             0
## 114       0             0             0             0
## 115       0             0             0             0
## 116       1             0             1             0
## 117       0             0             0             0
## 118       0             0             0             0
```

```
## 119       1            0            1            0
## 120       0            0            0            0
## 121       0            0            0            0
## 122       1            0            1            0
## 123       1            0            1            0
## 124       0            0            0            0
## 125       0            0            0            0
## 126       0            0            0            0
## 127       1            0            1            0
## 128       0            0            0            0
## 129       0            0            0            0
## 130       1            0            1            0
## 131       1            0            1            0
## 132       1            0            1            0
## 133       0            0            0            0
## 134       1            0            1            0
## 135       1            0            1            0
## 136       0            0            0            0
## 137       0            0            0            0
## 138       0            0            0            0
## 139       1            0            1            0
```

```
i2.test.Results$Compare.Results
```

```
##      Actual.EUI Predicted.EUI Residuals Actual.Class Predicted.Class
## 506          22            27        -5         Fair            Fair
## 507          34            25         9         Fair            Fair
## 508          45            27        18         Poor            Fair
## 509          70            25        45         Poor            Fair
## 512          19            30       -11         Good            Fair
## 513          27            29        -2         Fair            Fair
## 514          28            28         0         Fair            Fair
## 516          24            23         1         Fair            Fair
## 517          26            25         1         Fair            Fair
## 518          24            22         2         Fair            Fair
## 535          21            20         1         Fair            Good
## 543          25            16         9         Fair               0
## 547          36            39        -3         Fair            Fair
## 559          22            22         0         Fair            Fair
## 560          23            18         5         Fair               0
## 562          18            22        -4         Good            Fair
## 565          18            19        -1         Good               0
## 569          23            22         1         Fair            Fair
## 570          39            22        17         Fair            Fair
## 582          19            37       -18         Good            Fair
## 635          41            38         3         Poor            Fair
## 636          45            40         5         Poor            Poor
## 637          39            33         6         Fair            Fair
## 638          19            39       -20         Good            Fair
## 639          41            40         1         Poor            Poor
## 640          24            32        -8         Fair            Fair
## 641          35            35         0         Fair            Fair
## 643          21            34       -13         Fair            Fair
```

```
## 644          20            32       -12          Good            Fair
## 645          27            29        -2          Fair            Fair
## 647          26            29        -3          Fair            Fair
## 648          27            35        -8          Fair            Fair
## 649          26            30        -4          Fair            Fair
## 650          29            28         1          Fair            Fair
## 651          28            30        -2          Fair            Fair
## 652          20            27        -7          Good            Fair
## 653          23            29        -6          Fair            Fair
## 654          25            28        -3          Fair            Fair
## 655          17            24        -7          Good            Fair
## 658          19            33       -14          Good            Fair
## 659          21            31       -10          Fair            Fair
## 662          23            21         2          Fair            Fair
## 663          17            15         2          Good               0
## 664          27            18         9          Fair               0
## 667          27            35        -8          Fair            Fair
## 668          23            35       -12          Fair            Fair
## 671          31            38        -7          Fair            Fair
## 673          55            40        15          Poor            Poor
## 675          22            35       -13          Fair            Fair
## 682          31            21        10          Fair            Fair
## 686          29            25         4          Fair            Fair
## 689          17            20        -3          Good            Good
## 695          25            23         2          Fair            Fair
## 699          45            21        24          Poor            Fair
## 722          22            21         1          Fair            Fair
## 861          21            15         6          Fair               0
## 965          30            23         7          Fair            Fair
## 1282         29            41       -12          Fair            Poor
## 1286         22            30        -8          Fair            Fair
##      Correct Poor.Correct Fair.Correct Good.Correct
## 506        1            0            1            0
## 507        1            0            1            0
## 508        0            0            0            0
## 509        0            0            0            0
## 512        0            0            0            0
## 513        1            0            1            0
## 514        1            0            1            0
## 516        1            0            1            0
## 517        1            0            1            0
## 518        1            0            1            0
## 535        0            0            0            0
## 543        0            0            0            0
## 547        1            0            1            0
## 559        1            0            1            0
## 560        0            0            0            0
## 562        0            0            0            0
## 565        1            0            0            1
## 569        1            0            1            0
## 570        1            0            1            0
## 582        0            0            0            0
## 635        0            0            0            0
## 636        1            1            0            0
```

```
## 637        1             0             1             0
## 638        0             0             0             0
## 639        1             1             0             0
## 640        1             0             1             0
## 641        1             0             1             0
## 643        1             0             1             0
## 644        0             0             0             0
## 645        1             0             1             0
## 647        1             0             1             0
## 648        1             0             1             0
## 649        1             0             1             0
## 650        1             0             1             0
## 651        1             0             1             0
## 652        0             0             0             0
## 653        1             0             1             0
## 654        1             0             1             0
## 655        0             0             0             0
## 658        0             0             0             0
## 659        1             0             1             0
## 662        1             0             1             0
## 663        1             0             0             1
## 664        0             0             0             0
## 667        1             0             1             0
## 668        1             0             1             0
## 671        1             0             1             0
## 673        1             1             0             0
## 675        1             0             1             0
## 682        1             0             1             0
## 686        1             0             1             0
## 689        0             0             0             0
## 695        1             0             1             0
## 699        0             0             0             0
## 722        1             0             1             0
## 861        0             0             0             0
## 965        1             0             1             0
## 1282       0             0             0             0
## 1286       1             0             1             0
```

```
i2.train.Results$AccuracyRates
```

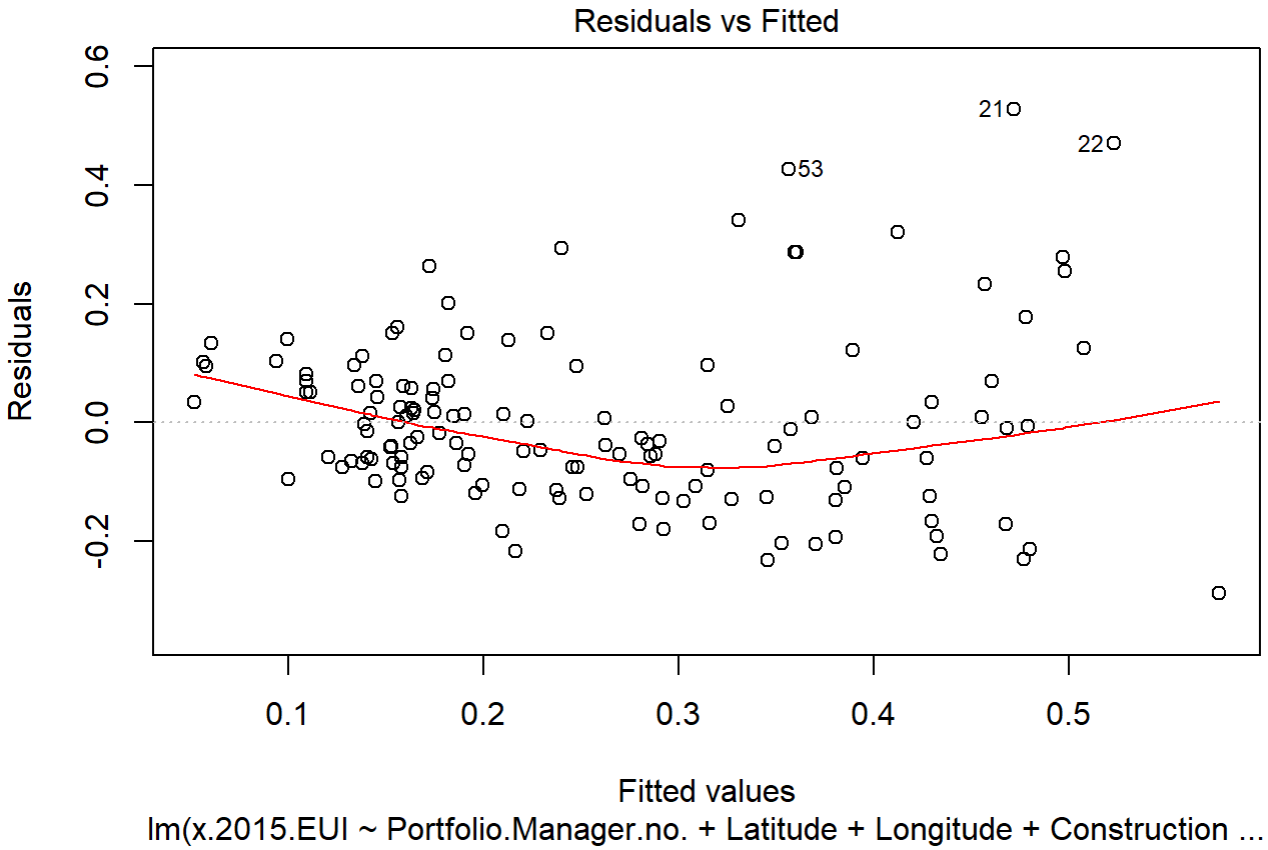
```
## Class.Accuracy  Poor.Accuracy  Fair.Accuracy  Good.Accuracy
##          47.48          20.00          75.00           5.13
```

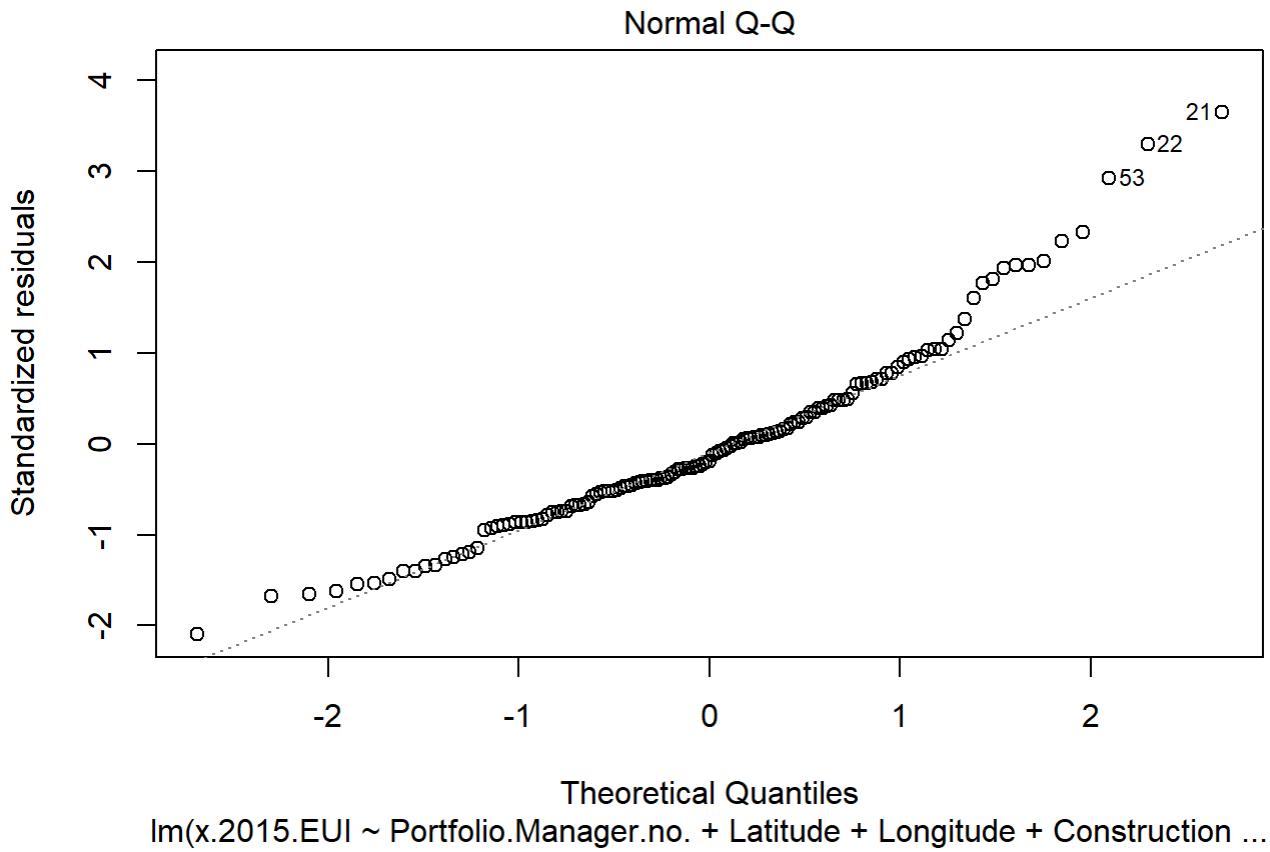
```
i2.test.Results$AccuracyRates
```

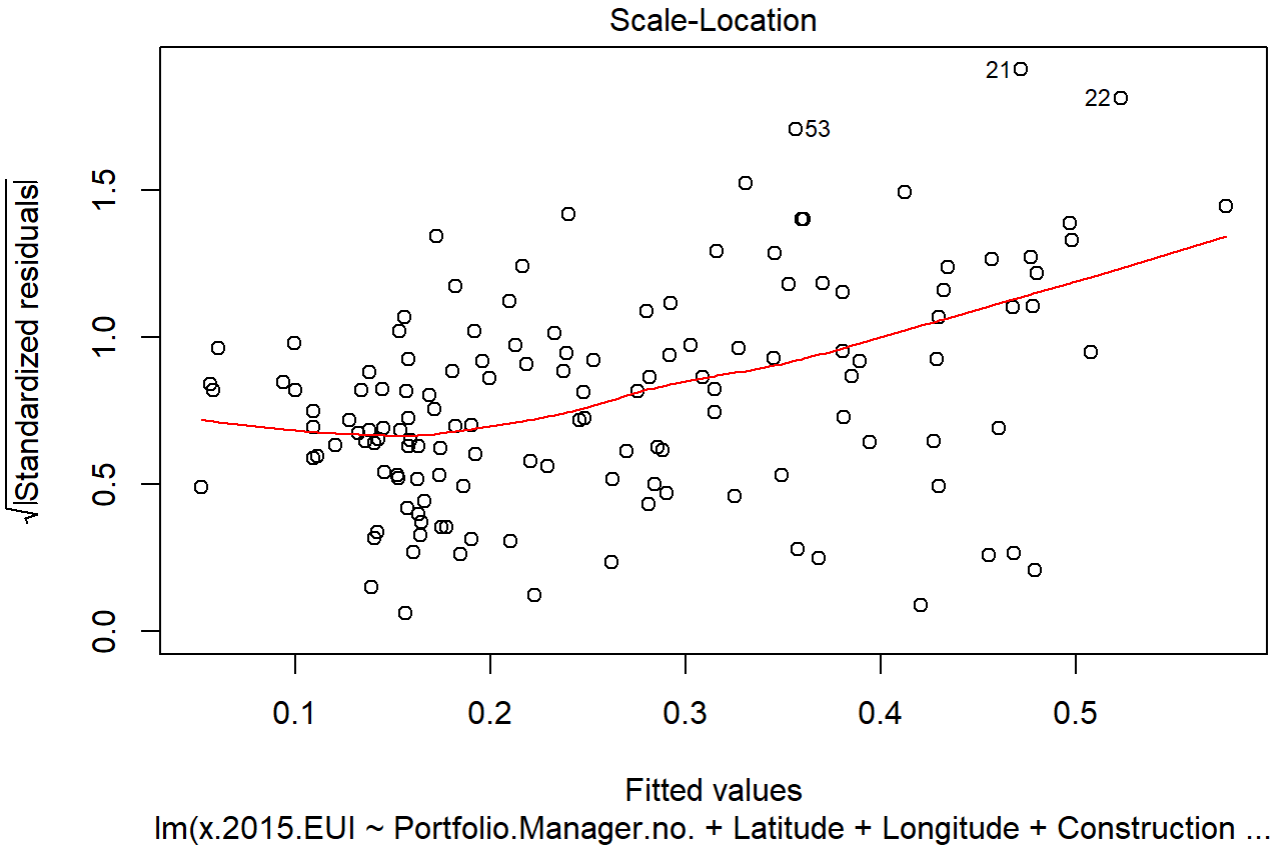
```
## Class.Accuracy  Poor.Accuracy  Fair.Accuracy  Good.Accuracy
##          67.80          42.86          85.37          18.18
```

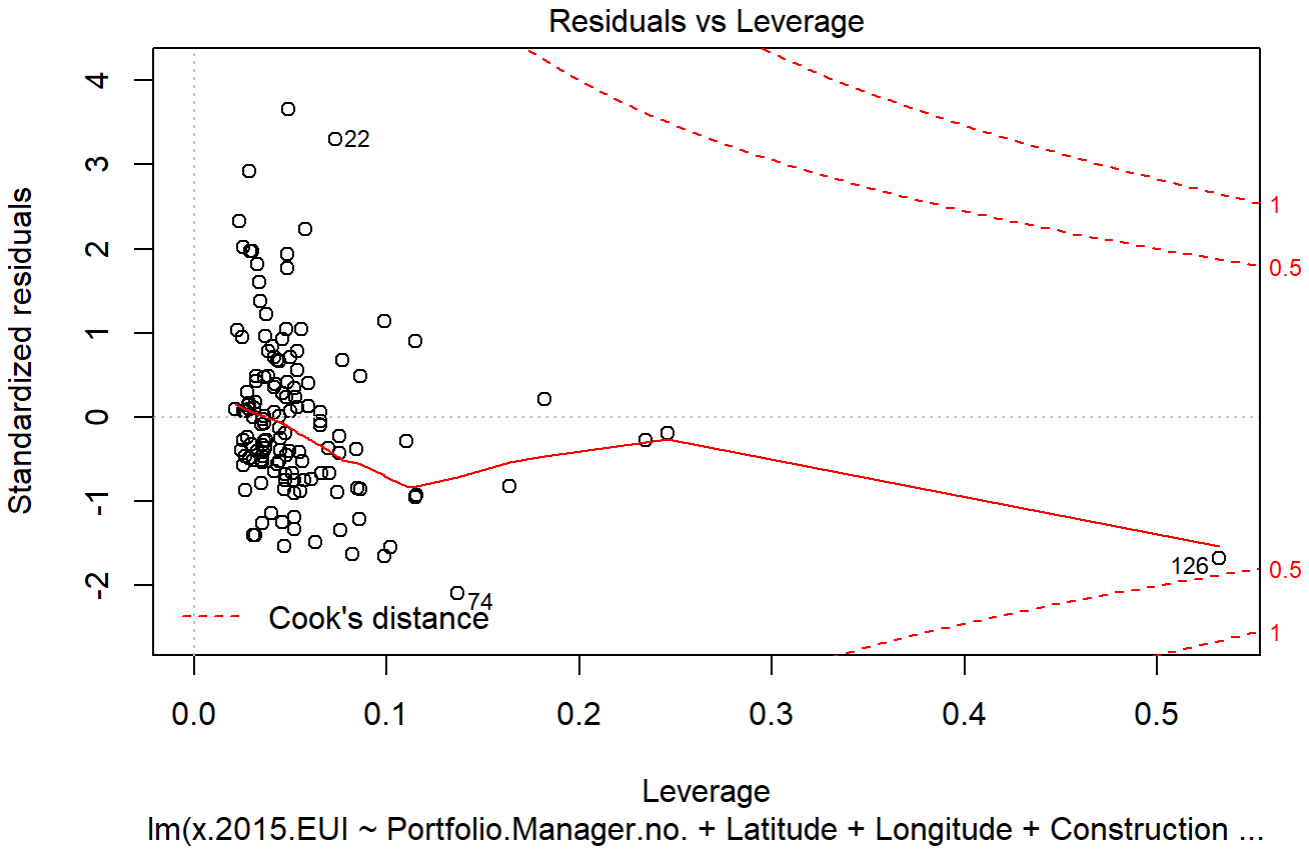
```
write.csv(i2.train.Results$Compare.Results, file ="i2 - Train Results Summary")
write.csv(i2.test.Results$Compare.Results, file ="i2 - Test Results Summary")
```

```
plot(i2.mlr)
```


Residuals vs Fitted
Residuals
Fitted values
lm(x.2015.EUI ~ Portfolio.Manager.no. + Latitude + Longitude + Construction ...
21
22
53
0.6
0.4
0.2
0.0
-0.2
0.1
0.2
0.3
0.4
0.5


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(x.2015.EUI ~ Portfolio.Manager.no. + Latitude + Longitude + Construction ...
21
22
53


Scale-Location
√|Standardized residuals|
Fitted values
lm(x.2015.EUI ~ Portfolio.Manager.no. + Latitude + Longitude + Construction ...
21
22
53




```
# Dernormalize, Plot and Record Prediction Results
denorm.train=denormalize(x=i2.mlr$fitted.values, y="x.2015.EUI")
Actual.train = denormalize(x=i2.data.train[,1],y="x.2015.EUI")

jpeg(filename="MLR, Actual vs. Predicted 2015 EUI (Train, 2010-2015 Dataset).jpg")
plot( Actual.train, denorm.train, main="Actual vs. Predicted 2010 EUI (Train, 2010-2015 Datase
t)", xlab="Actual Values", ylab="Predicted Values")
dev.off()
```

```
## png
##   2
```

```
denorm.test=denormalize(x=pred.mlr,y="x.2015.EUI")
Actual.test = denormalize(x=i2.data.test[,1],y="x.2015.EUI")

jpeg(filename="MLR, Actual vs. Predicted 2015 EUI (Test, 2010-2015 Dataset).jpg")
plot( Actual.test, denorm.test, main="Actual vs. Predicted 2015 EUI (Test, 2010-2015 Dataset)"
, xlab="Actual Values", ylab="Predicted Values")
dev.off()
```

```
## png
##   2
```

```
##############################
####     MLR iteration 3   ####
##############################

# Select outcome variable (i.e. D.EUI.percent) and independent variables for iteration 3 (i3)
i3.data <- data[,c("D.EUI.percent","x.2010.Actual.E_C.Energy","x.2010.Actual.Thermal",
                   "Portfolio.Manager.no.", "Latitude", "Longitude",    "Construction.Year",
  "Occupant.Density",
                   "Vacancy.Rate",  "Weekly.Operating.Hours")]
rnames  <- row.names(i3.data)

# Normalize
i3.norm.data <-as.data.frame(lapply(i3.data, normalize))
row.names(i3.norm.data) <- rnames

##create Train and Test set
train.set = as.data.frame(stratified(data, "D.Qualitative.EUI", size=.7, replace = FALSE,
                                     bothSets = FALSE, keep.rownames=TRUE))
train.rows = train.set$rn
i3.data.train = i3.norm.data %>% filter(row.names(i3.norm.data) %in% c(train.rows))
i3.train = sample(1:nrow(i3.data.train), nrow(i3.data.train)*1)
i3.data.test=i3.norm.data[-i3.train,]

# Define function to Classify the predicted results according to Good, Fair, Poor for D.EUI.pe
rcent [EUI percent Change]
D.classification.accuracy <- function(w,x,y,z) {
  denrom.pr.mlr <- round(as.data.frame(denormalize(x,y)),2)
  Actual.EUI <- round(w[z,y],2)
  Residuals <- Actual.EUI-denrom.pr.mlr
  Compare <- as.data.frame(cbind(Actual.EUI,denrom.pr.mlr,Residuals))
  colnames(Compare)<-c('Actual.EUI','Predicted.EUI','Residuals')
  Compare$Actual.Class = ifelse((Compare$Actual.EUI <= 0 ),"Poor",
                                ifelse((Compare$Actual.EUI > 0 & Compare$Actual.EUI <.20),"Fai
r",
                                       ifelse((Compare$Actual.EUI >=.20),"Good",0)))
  Compare$Predicted.Class = ifelse((Compare$Predicted.EUI <= 0 ),"Poor",
                                   ifelse((Compare$Predicted.EUI > 0 & Compare$Predicted.EUI <
.20),"Fair",
                                          ifelse((Compare$Predicted.EUI >=.20),"Good",0)))
  Compare$Correct = ifelse((Compare$Actual.EUI <= 0 & Compare$Predicted.EUI <= 0 ),1,
                           ifelse((Compare$Actual.EUI > 0&Compare$Actual.EUI <.20 & Compare$Pr
edicted.EUI > 0 & Compare$Predicted.EUI <.20),1,
                                  ifelse((Compare$Actual.EUI >=.20 & Compare$Predicted.EUI >=.
20),1,0)))
  Compare$Poor.Correct = ifelse((Compare$Actual.EUI <= 0 & Compare$Predicted.EUI <= 0 ),1,0)
  Compare$Fair.Correct = ifelse((Compare$Actual.EUI > 0&Compare$Actual.EUI <.20 & Compare$Pred
icted.EUI > 0 & Compare$Predicted.EUI <.20),1,0)
  Compare$Good.Correct = ifelse((Compare$Actual.EUI >=.20 & Compare$Predicted.EUI >.20),1,0)
  Class.Accuracy<- round((sum(Compare$Correct))/(nrow(Compare))*100,2)
  Poor.Accuracy<- round((sum(Compare$Poor.Correct))/(sum(Compare$Actual.Class=="Poor"))*100,2)
  Fair.Accuracy<- round((sum(Compare$Fair.Correct))/(sum(Compare$Actual.Class=="Fair"))*100,2)
  Good.Accuracy<- round((sum(Compare$Good.Correct))/(sum(Compare$Actual.Class=="Good"))*100,2)
  Accuracy.Rates <- c("Class.Accuracy" = Class.Accuracy, "Poor.Accuracy" = Poor.Accuracy,"Fair
.Accuracy" = Fair.Accuracy,"Good.Accuracy" = Good.Accuracy)
```

```
  Results=list("AccuracyRates" = Accuracy.Rates,"Compare.Results" = Compare)
  return(Results)
}

# Train MLR model on iteration 3 training dataset
i3.mlr <- lm(formula = D.EUI.percent~Portfolio.Manager.no.+Latitude+Longitude+Construction.Yea
r+Occupant.Density+
               Vacancy.Rate+Weekly.Operating.Hours, data=i3.data.train)

# Classify the predicted results of the when applied to the training dataset
denormalize(x=i3.mlr$fitted.values, y="D.EUI.percent")
```

```
##            1            2            3            4            5
## -0.144555915 -0.065811936 -0.092476065 -0.042685958 -0.185293053
##            6            7            8            9           10
## -0.062765670 -0.059470244 -0.037117707 -0.080378191 -0.093168362
##           11           12           13           14           15
## -0.093219611 -0.019593839  0.025566417 -0.022570436 -0.036184864
##           16           17           18           19           20
## -0.320051424 -0.123281023 -0.109135210  0.002078785  0.004328793
##           21           22           23           24           25
## -0.057644621  0.093216901 -0.196300068 -0.175574674 -0.206421881
##           26           27           28           29           30
## -0.068154117 -0.106360596  0.060260436 -0.166417634  0.070499075
##           31           32           33           34           35
## -0.056615318 -0.084189620 -0.109083741  0.111058045 -0.069830773
##           36           37           38           39           40
## -0.122880976 -0.196888009 -0.202799736 -0.142458774 -0.399027779
##           41           42           43           44           45
## -0.232792046  0.023575892 -0.183593988 -0.118170037 -0.222855354
##           46           47           48           49           50
## -0.093012212 -0.042412248 -0.119913556 -0.156860740 -0.169569576
##           51           52           53           54           55
## -0.171492552 -0.245271435 -0.096644791 -0.144261887 -0.176208922
##           56           57           58           59           60
## -0.062007433 -0.080954991 -0.056287941 -0.086993592 -0.124027780
##           61           62           63           64           65
## -0.106490427 -0.086211106 -0.084252631 -0.024142490 -0.099107749
##           66           67           68           69           70
##  0.030145174 -0.197527112 -0.116984732 -0.075363198 -0.044045923
##           71           72           73           74           75
## -0.044311761  0.079573434 -0.024476836 -0.128790448 -0.128470931
##           76           77           78           79           80
## -0.169461699 -0.162859032 -0.022777152  0.035941110  0.033678036
##           81           82           83           84           85
## -0.152588909 -0.115709915 -0.228314879 -0.132245646 -0.091658281
##           86           87           88           89           90
## -0.116553789 -0.105890010 -0.023987545 -0.192536183 -0.346244592
##           91           92           93           94           95
## -0.429346444 -0.120713431 -0.062701049  0.036375628 -0.202790584
##           96           97           98           99          100
## -0.260926949 -0.209649409 -0.131642589 -0.067307823 -0.004306179
##          101          102          103          104          105
```

```
## -0.011840610  0.041490720 -0.047152957 -0.041415073 -0.067188544
##          106          107          108          109          110
## -0.061941600 -0.058485277 -0.029223321 -0.032558319 -0.078166415
##          111          112          113          114          115
## -0.071838518 -0.152847815 -0.234885585  0.008773670 -0.133355465
##          116          117          118          119          120
## -0.209099958 -0.177632074 -0.188232604 -0.166719236 -0.063318474
##          121          122          123          124          125
## -0.016476315  0.043656009 -0.085988230 -0.038314961 -0.060056795
##          126          127          128          129          130
## -0.096073710 -0.158452785 -0.144771715 -0.171206681 -0.138499856
##          131          132          133          134          135
## -0.127661419 -0.224966011 -0.044178204  0.039759245 -0.147682677
##          136          137          138          139
##  0.023072490 -0.054127129 -0.114005549 -0.142559640
```

```
i3.train.Results =D.classification.accuracy(w=i3.data,x=i3.mlr$fitted.values, y="D.EUI.percent
",z=i3.train)

# Predict and Classify the results of the model when applied to the test dataset
pred.mlr <- (predict(i3.mlr,i3.data.test[,-1]))
i3.test.Results =classification.accuracy(w=i3.data,x=pred.mlr, y="D.EUI.percent",z=-i3.train)

# View and Record normalized Prediction Results
i3.train.Results$Compare.Results
```

```
##     Actual.EUI Predicted.EUI Residuals Actual.Class Predicted.Class
## 1        -0.17         -0.14     -0.03         Poor            Poor
## 2        -0.22         -0.07     -0.15         Poor            Poor
## 3         0.06         -0.09      0.15         Fair            Poor
## 4         0.18         -0.04      0.22         Fair            Poor
## 5        -0.05         -0.19      0.14         Poor            Poor
## 6        -0.31         -0.06     -0.25         Poor            Poor
## 7        -0.22         -0.06     -0.16         Poor            Poor
## 8        -0.07         -0.04     -0.03         Poor            Poor
## 9        -0.29         -0.08     -0.21         Poor            Poor
## 10       -0.23         -0.09     -0.14         Poor            Poor
## 11        0.28         -0.09      0.37         Good            Poor
## 12       -0.22         -0.02     -0.20         Poor            Poor
## 13       -0.06          0.03     -0.09         Poor            Fair
## 14       -0.05         -0.02     -0.03         Poor            Poor
## 15       -0.10         -0.04     -0.06         Poor            Poor
## 16       -0.29         -0.32      0.03         Poor            Poor
## 17       -0.13         -0.12     -0.01         Poor            Poor
## 18       -0.04         -0.11      0.07         Poor            Poor
## 19        0.14          0.00      0.14         Fair            Poor
## 20       -0.05          0.00     -0.05         Poor            Poor
## 21        0.04         -0.06      0.10         Fair            Poor
## 22        0.18          0.09      0.09         Fair            Fair
## 23       -0.30         -0.20     -0.10         Poor            Poor
## 24       -0.33         -0.18     -0.15         Poor            Poor
## 25       -0.20         -0.21      0.01         Poor            Poor
## 26       -0.25         -0.07     -0.18         Poor            Poor
```

```
## 27       -0.25         -0.11     -0.14         Poor            Poor
## 28       -0.59          0.06     -0.65         Poor            Fair
## 29       -0.11         -0.17      0.06         Poor            Poor
## 30       -0.25          0.07     -0.32         Poor            Fair
## 31       -0.38         -0.06     -0.32         Poor            Poor
## 32       -0.23         -0.08     -0.15         Poor            Poor
## 33        0.47         -0.11      0.58         Good            Poor
## 34       -0.22          0.11     -0.33         Poor            Fair
## 35       -0.33         -0.07     -0.26         Poor            Poor
## 36        0.37         -0.12      0.49         Good            Poor
## 37       -0.16         -0.20      0.04         Poor            Poor
## 38        0.04         -0.20      0.24         Fair            Poor
## 39       -0.22         -0.14     -0.08         Poor            Poor
## 40       -0.04         -0.40      0.36         Poor            Poor
## 41       -0.21         -0.23      0.02         Poor            Poor
## 42       -0.36          0.02     -0.38         Poor            Fair
## 43       -0.24         -0.18     -0.06         Poor            Poor
## 44        0.00         -0.12      0.12         Poor            Poor
## 45        0.05         -0.22      0.27         Fair            Poor
## 46        0.23         -0.09      0.32         Good            Poor
## 47       -0.15         -0.04     -0.11         Poor            Poor
## 48       -0.26         -0.12     -0.14         Poor            Poor
## 49        0.41         -0.16      0.57         Good            Poor
## 50       -0.04         -0.17      0.13         Poor            Poor
## 51       -0.31         -0.17     -0.14         Poor            Poor
## 52       -0.08         -0.25      0.17         Poor            Poor
## 53       -0.04         -0.10      0.06         Poor            Poor
## 54       -0.22         -0.14     -0.08         Poor            Poor
## 55       -0.38         -0.18     -0.20         Poor            Poor
## 56       -0.01         -0.06      0.05         Poor            Poor
## 57       -0.24         -0.08     -0.16         Poor            Poor
## 58       -0.03         -0.06      0.03         Poor            Poor
## 59       -0.26         -0.09     -0.17         Poor            Poor
## 60       -0.07         -0.12      0.05         Poor            Poor
## 61       -0.01         -0.11      0.10         Poor            Poor
## 62        0.02         -0.09      0.11         Fair            Poor
## 63       -0.41         -0.08     -0.33         Poor            Poor
## 64        0.28         -0.02      0.30         Good            Poor
## 65        0.40         -0.10      0.50         Good            Poor
## 66       -0.17          0.03     -0.20         Poor            Fair
## 67       -0.10         -0.20      0.10         Poor            Poor
## 68       -0.13         -0.12     -0.01         Poor            Poor
## 69        0.11         -0.08      0.19         Fair            Poor
## 70       -0.14         -0.04     -0.10         Poor            Poor
## 71       -0.25         -0.04     -0.21         Poor            Poor
## 72       -0.40          0.08     -0.48         Poor            Fair
## 73       -0.11         -0.02     -0.09         Poor            Poor
## 74       -0.35         -0.13     -0.22         Poor            Poor
## 75       -0.16         -0.13     -0.03         Poor            Poor
## 76       -0.14         -0.17      0.03         Poor            Poor
## 77        0.43         -0.16      0.59         Good            Poor
## 78       -0.12         -0.02     -0.10         Poor            Poor
## 79       -0.11          0.04     -0.15         Poor            Fair
## 80       -0.34          0.03     -0.37         Poor            Fair
```

```
## 81       -0.09         -0.15      0.06         Poor            Poor
## 82       -0.27         -0.12     -0.15         Poor            Poor
## 83       -0.14         -0.23      0.09         Poor            Poor
## 84       -0.58         -0.13     -0.45         Poor            Poor
## 85       -0.40         -0.09     -0.31         Poor            Poor
## 86       -0.20         -0.12     -0.08         Poor            Poor
## 87       -0.07         -0.11      0.04         Poor            Poor
## 88        0.64         -0.02      0.66         Good            Poor
## 89       -0.49         -0.19     -0.30         Poor            Poor
## 90       -0.26         -0.35      0.09         Poor            Poor
## 91        0.04         -0.43      0.47         Fair            Poor
## 92        0.03         -0.12      0.15         Fair            Poor
## 93       -0.03         -0.06      0.03         Poor            Poor
## 94       -0.21          0.04     -0.25         Poor            Fair
## 95        0.16         -0.20      0.36         Fair            Poor
## 96        0.13         -0.26      0.39         Fair            Poor
## 97       -0.06         -0.21      0.15         Poor            Poor
## 98       -0.16         -0.13     -0.03         Poor            Poor
## 99        0.02         -0.07      0.09         Fair            Poor
## 100       0.01          0.00      0.01         Fair            Poor
## 101      -0.60         -0.01     -0.59         Poor            Poor
## 102      -0.24          0.04     -0.28         Poor            Fair
## 103      -0.09         -0.05     -0.04         Poor            Poor
## 104      -0.15         -0.04     -0.11         Poor            Poor
## 105      -0.27         -0.07     -0.20         Poor            Poor
## 106       0.01         -0.06      0.07         Fair            Poor
## 107      -0.05         -0.06      0.01         Poor            Poor
## 108      -0.17         -0.03     -0.14         Poor            Poor
## 109      -0.35         -0.03     -0.32         Poor            Poor
## 110       0.09         -0.08      0.17         Fair            Poor
## 111      -0.32         -0.07     -0.25         Poor            Poor
## 112      -0.12         -0.15      0.03         Poor            Poor
## 113      -0.24         -0.23     -0.01         Poor            Poor
## 114      -0.14          0.01     -0.15         Poor            Fair
## 115      -0.20         -0.13     -0.07         Poor            Poor
## 116       0.17         -0.21      0.38         Fair            Poor
## 117      -0.16         -0.18      0.02         Poor            Poor
## 118      -0.19         -0.19      0.00         Poor            Poor
## 119      -0.26         -0.17     -0.09         Poor            Poor
## 120      -0.27         -0.06     -0.21         Poor            Poor
## 121      -0.42         -0.02     -0.40         Poor            Poor
## 122      -0.20          0.04     -0.24         Poor            Fair
## 123      -0.20         -0.09     -0.11         Poor            Poor
## 124      -0.61         -0.04     -0.57         Poor            Poor
## 125      -0.05         -0.06      0.01         Poor            Poor
## 126      -0.04         -0.10      0.06         Poor            Poor
## 127      -0.14         -0.16      0.02         Poor            Poor
## 128       0.34         -0.14      0.48         Good            Poor
## 129      -0.19         -0.17     -0.02         Poor            Poor
## 130      -0.21         -0.14     -0.07         Poor            Poor
## 131       0.05         -0.13      0.18         Fair            Poor
## 132      -0.05         -0.22      0.17         Poor            Poor
## 133      -0.20         -0.04     -0.16         Poor            Poor
## 134      -0.05          0.04     -0.09         Poor            Fair
```

```
## 135       0.01         -0.15      0.16         Fair            Poor
## 136      -0.03          0.02     -0.05         Poor            Fair
## 137      -0.06         -0.05     -0.01         Poor            Poor
## 138      -0.11         -0.11      0.00         Poor            Poor
## 139      -0.11         -0.14      0.03         Poor            Poor
##     Correct Poor.Correct Fair.Correct Good.Correct
## 1         1            1            0            0
## 2         1            1            0            0
## 3         0            0            0            0
## 4         0            0            0            0
## 5         1            1            0            0
## 6         1            1            0            0
## 7         1            1            0            0
## 8         1            1            0            0
## 9         1            1            0            0
## 10        1            1            0            0
## 11        0            0            0            0
## 12        1            1            0            0
## 13        0            0            0            0
## 14        1            1            0            0
## 15        1            1            0            0
## 16        1            1            0            0
## 17        1            1            0            0
## 18        1            1            0            0
## 19        0            0            0            0
## 20        1            1            0            0
## 21        0            0            0            0
## 22        1            0            1            0
## 23        1            1            0            0
## 24        1            1            0            0
## 25        1            1            0            0
## 26        1            1            0            0
## 27        1            1            0            0
## 28        0            0            0            0
## 29        1            1            0            0
## 30        0            0            0            0
## 31        1            1            0            0
## 32        1            1            0            0
## 33        0            0            0            0
## 34        0            0            0            0
## 35        1            1            0            0
## 36        0            0            0            0
## 37        1            1            0            0
## 38        0            0            0            0
## 39        1            1            0            0
## 40        1            1            0            0
## 41        1            1            0            0
## 42        0            0            0            0
## 43        1            1            0            0
## 44        1            1            0            0
## 45        0            0            0            0
## 46        0            0            0            0
## 47        1            1            0            0
## 48        1            1            0            0
```

```
## 49        0             0             0             0
## 50        1             1             0             0
## 51        1             1             0             0
## 52        1             1             0             0
## 53        1             1             0             0
## 54        1             1             0             0
## 55        1             1             0             0
## 56        1             1             0             0
## 57        1             1             0             0
## 58        1             1             0             0
## 59        1             1             0             0
## 60        1             1             0             0
## 61        1             1             0             0
## 62        0             0             0             0
## 63        1             1             0             0
## 64        0             0             0             0
## 65        0             0             0             0
## 66        0             0             0             0
## 67        1             1             0             0
## 68        1             1             0             0
## 69        0             0             0             0
## 70        1             1             0             0
## 71        1             1             0             0
## 72        0             0             0             0
## 73        1             1             0             0
## 74        1             1             0             0
## 75        1             1             0             0
## 76        1             1             0             0
## 77        0             0             0             0
## 78        1             1             0             0
## 79        0             0             0             0
## 80        0             0             0             0
## 81        1             1             0             0
## 82        1             1             0             0
## 83        1             1             0             0
## 84        1             1             0             0
## 85        1             1             0             0
## 86        1             1             0             0
## 87        1             1             0             0
## 88        0             0             0             0
## 89        1             1             0             0
## 90        1             1             0             0
## 91        0             0             0             0
## 92        0             0             0             0
## 93        1             1             0             0
## 94        0             0             0             0
## 95        0             0             0             0
## 96        0             0             0             0
## 97        1             1             0             0
## 98        1             1             0             0
## 99        0             0             0             0
## 100       0             0             0             0
## 101       1             1             0             0
## 102       0             0             0             0
```

```
## 103       1             1             0             0
## 104       1             1             0             0
## 105       1             1             0             0
## 106       0             0             0             0
## 107       1             1             0             0
## 108       1             1             0             0
## 109       1             1             0             0
## 110       0             0             0             0
## 111       1             1             0             0
## 112       1             1             0             0
## 113       1             1             0             0
## 114       0             0             0             0
## 115       1             1             0             0
## 116       0             0             0             0
## 117       1             1             0             0
## 118       1             1             0             0
## 119       1             1             0             0
## 120       1             1             0             0
## 121       1             1             0             0
## 122       0             0             0             0
## 123       1             1             0             0
## 124       1             1             0             0
## 125       1             1             0             0
## 126       1             1             0             0
## 127       1             1             0             0
## 128       0             0             0             0
## 129       1             1             0             0
## 130       1             1             0             0
## 131       0             0             0             0
## 132       1             1             0             0
## 133       1             1             0             0
## 134       0             0             0             0
## 135       0             0             0             0
## 136       0             0             0             0
## 137       1             1             0             0
## 138       1             1             0             0
## 139       1             1             0             0
```

```
i3.test.Results$Compare.Results
```

```
##      Actual.EUI Predicted.EUI Residuals Actual.Class Predicted.Class
## 506           0             0         0         Good               0
## 507           0             0         0         Good               0
## 508           0             0         0         Good               0
## 509           0             0         0         Good               0
## 512           0             0         0         Good               0
## 513           0             0         0         Good               0
## 514           0             0         0         Good               0
## 516           0             0         0         Good               0
## 517           0             0         0         Good               0
## 518           0             0         0         Good               0
## 535           0             0         0         Good               0
```

```
## 543          0             0         0         Good                0
## 547          0             0         0         Good                0
## 559          0             0         0         Good                0
## 560          0             0         0         Good                0
## 562          0             0         0         Good                0
## 565          0             0         0         Good                0
## 569          0             0         0         Good                0
## 570          1             0         1         Good                0
## 582          0             0         0         Good                0
## 635          0             0         0         Good                0
## 636          0             0         0         Good                0
## 637          1             0         1         Good                0
## 638          0             0         0         Good                0
## 639          0             0         0         Good                0
## 640          0             0         0         Good                0
## 641          0             0         0         Good                0
## 643          0             0         0         Good                0
## 644          0             0         0         Good                0
## 645          0             0         0         Good                0
## 647          0             0         0         Good                0
## 648          0             0         0         Good                0
## 649          0             0         0         Good                0
## 650          0             0         0         Good                0
## 651          0             0         0         Good                0
## 652          0             0         0         Good                0
## 653          0             0         0         Good                0
## 654          0             0         0         Good                0
## 655          0             0         0         Good                0
## 658          0             0         0         Good                0
## 659          0             0         0         Good                0
## 662          0             0         0         Good                0
## 663          0             0         0         Good                0
## 664          0             0         0         Good                0
## 667          0             0         0         Good                0
## 668          0             0         0         Good                0
## 671          0             0         0         Good                0
## 673          1             0         1         Good                0
## 675          0             0         0         Good                0
## 682          0             0         0         Good                0
## 686          0             0         0         Good                0
## 689          0             0         0         Good                0
## 695          0             0         0         Good                0
## 699          0             0         0         Good                0
## 722          0             0         0         Good                0
## 861          0             0         0         Good                0
## 965          0             0         0         Good                0
## 1282         0             0         0         Good                0
## 1286         0             0         0         Good                0
##      Correct Poor.Correct Fair.Correct Good.Correct
## 506        1            0            0            1
## 507        1            0            0            1
## 508        1            0            0            1
## 509        1            0            0            1
## 512        1            0            0            1
```

```
## 513        1             0             0             1
## 514        1             0             0             1
## 516        1             0             0             1
## 517        1             0             0             1
## 518        1             0             0             1
## 535        1             0             0             1
## 543        1             0             0             1
## 547        1             0             0             1
## 559        1             0             0             1
## 560        1             0             0             1
## 562        1             0             0             1
## 565        1             0             0             1
## 569        1             0             0             1
## 570        1             0             0             1
## 582        1             0             0             1
## 635        1             0             0             1
## 636        1             0             0             1
## 637        1             0             0             1
## 638        1             0             0             1
## 639        1             0             0             1
## 640        1             0             0             1
## 641        1             0             0             1
## 643        1             0             0             1
## 644        1             0             0             1
## 645        1             0             0             1
## 647        1             0             0             1
## 648        1             0             0             1
## 649        1             0             0             1
## 650        1             0             0             1
## 651        1             0             0             1
## 652        1             0             0             1
## 653        1             0             0             1
## 654        1             0             0             1
## 655        1             0             0             1
## 658        1             0             0             1
## 659        1             0             0             1
## 662        1             0             0             1
## 663        1             0             0             1
## 664        1             0             0             1
## 667        1             0             0             1
## 668        1             0             0             1
## 671        1             0             0             1
## 673        1             0             0             1
## 675        1             0             0             1
## 682        1             0             0             1
## 686        1             0             0             1
## 689        1             0             0             1
## 695        1             0             0             1
## 699        1             0             0             1
## 722        1             0             0             1
## 861        1             0             0             1
## 965        1             0             0             1
## 1282       1             0             0             1
## 1286       1             0             0             1
```

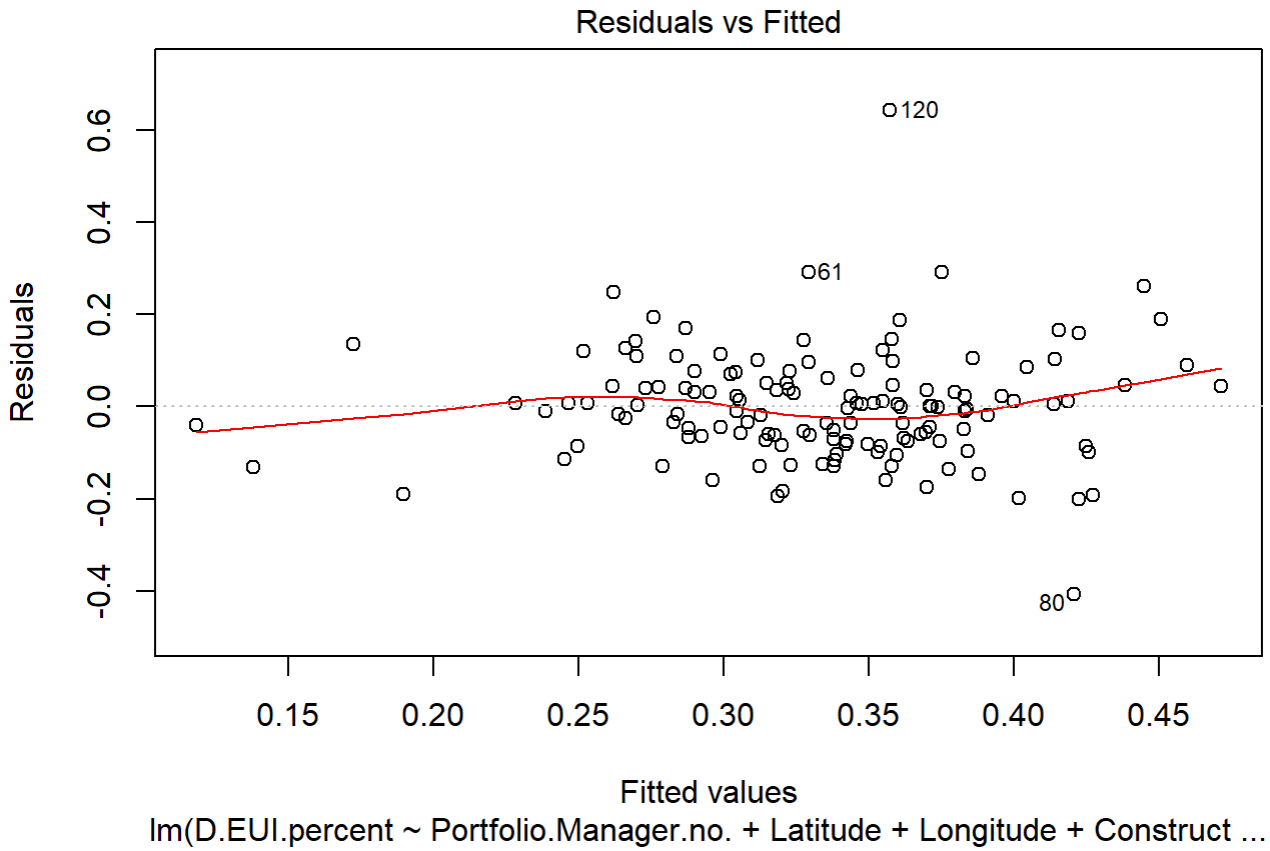
```
i3.train.Results$AccuracyRates
```

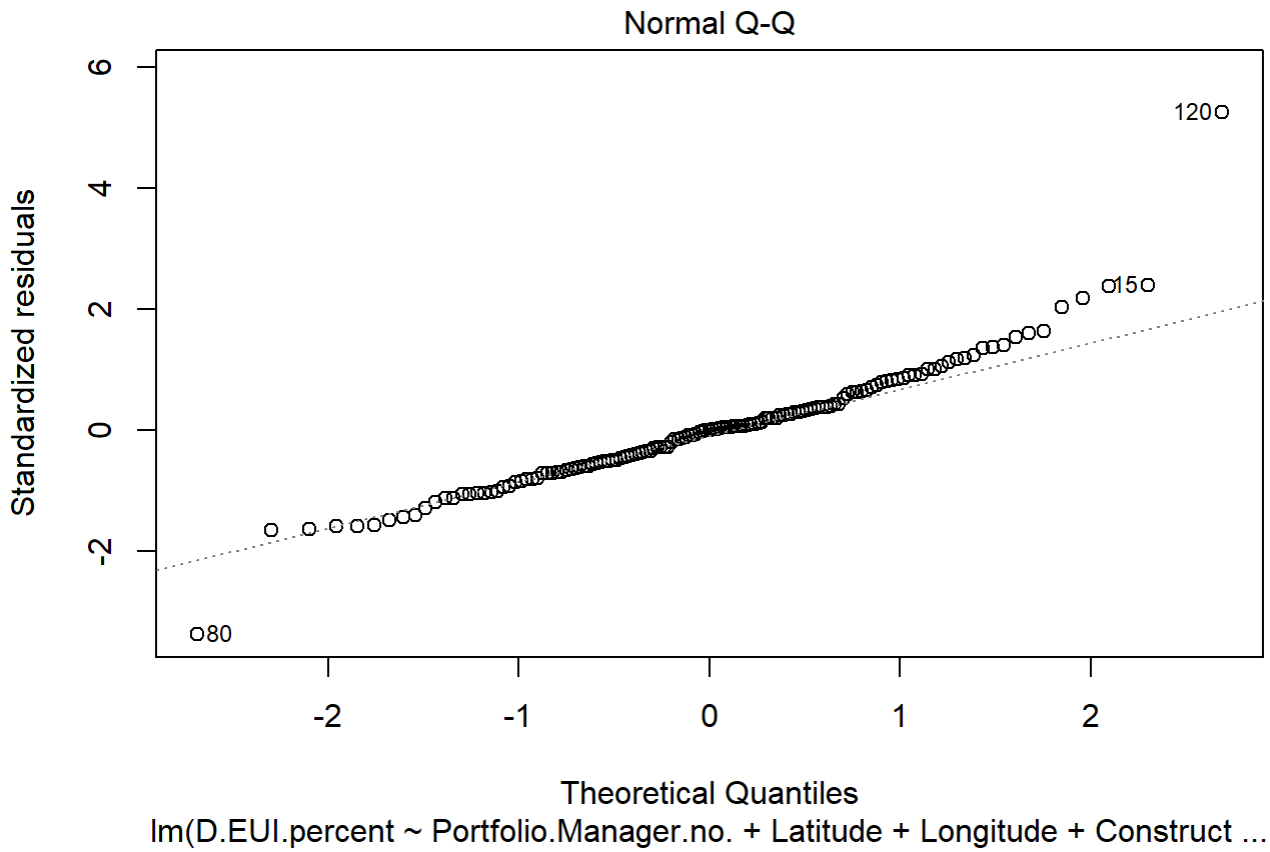
```
## Class.Accuracy  Poor.Accuracy  Fair.Accuracy  Good.Accuracy
##          68.35          86.24           5.00           0.00
```

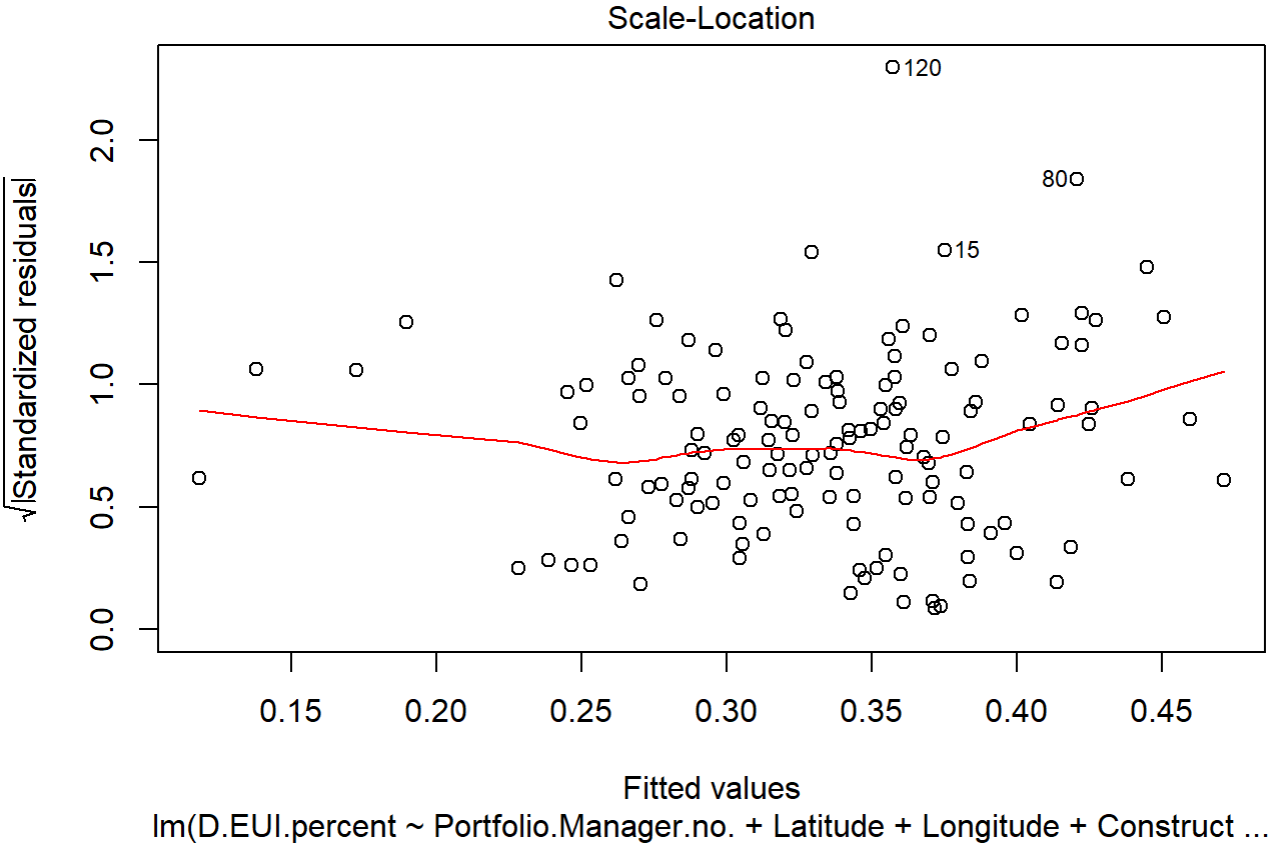
```
i3.test.Results$AccuracyRates
```

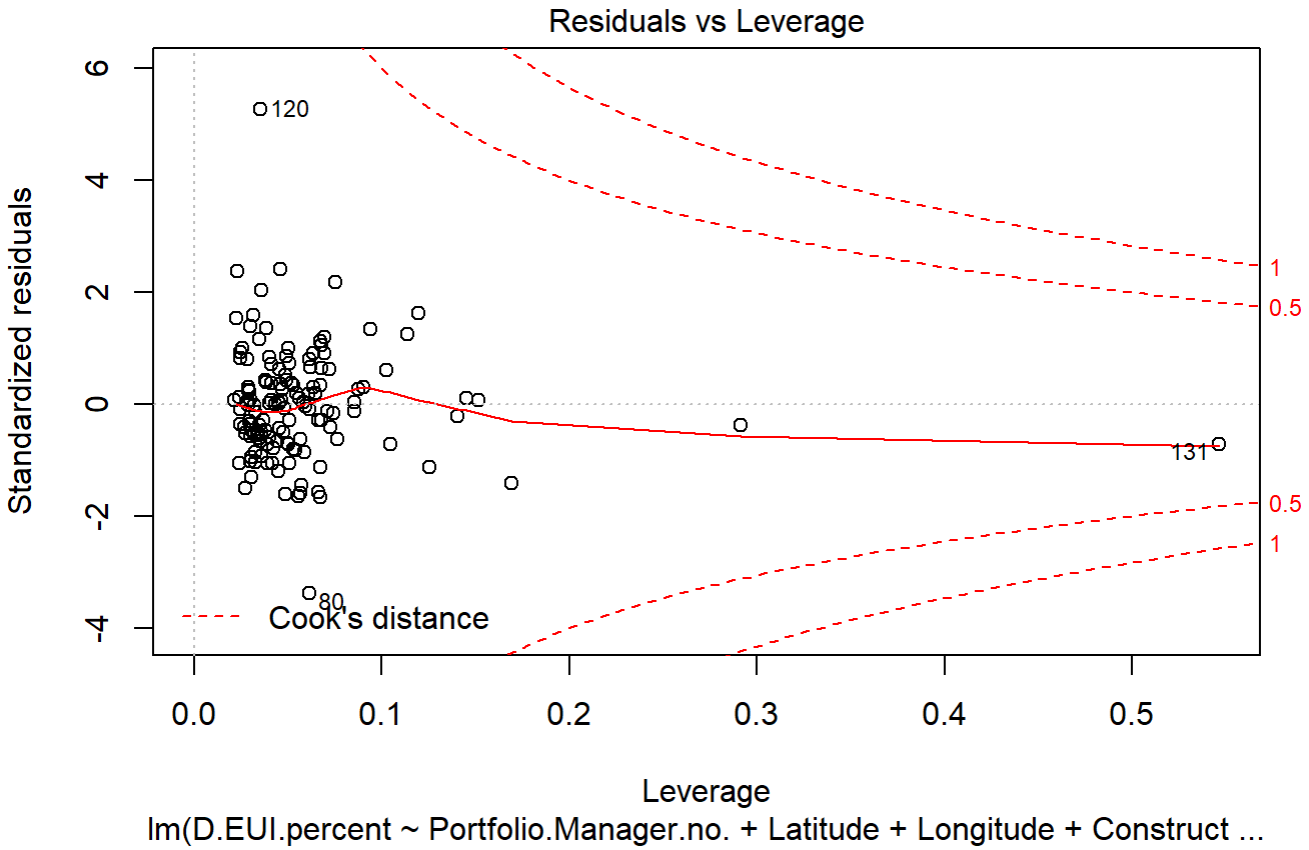
```
## Class.Accuracy  Poor.Accuracy  Fair.Accuracy  Good.Accuracy
##            100            NaN            NaN            100
```

```
write.csv(i3.train.Results$Compare.Results, file ="i3 - Train Results Summary")
write.csv(i3.test.Results$Compare.Results, file ="i3 - Test Results Summary")
plot(i3.mlr)
```




Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(D.EUI.percent ~ Portfolio.Manager.no. + Latitude + Longitude + Construct ...
120
15
80
6
4
2
0
-2
-2
-1
0
1
2


Scale-Location
√|Standardized residuals|
Fitted values
lm(D.EUI.percent ~ Portfolio.Manager.no. + Latitude + Longitude + Construct ...
120
80
15
2.0
1.5
1.0
0.5
0.0
0.15
0.20
0.25
0.30
0.35
0.40
0.45




```
# Dernormalize, Plot and Record Prediction Results
denorm.train=denormalize(x=i3.mlr$fitted.values, y="D.EUI.percent")
Actual.train = denormalize(x=i3.data.train[,1],y="D.EUI.percent")

jpeg(filename="MLR, Actual vs. Predicted EUI Change (Train, 2010-2015 Dataset).jpg")
plot( Actual.train, denorm.train, main="Actual vs. Predicted EUI Change (Train, 2010-2015 Data
set)", xlab="Actual Values", ylab="Predicted Values")
dev.off()
```

```
## png
##   2
```

```
denorm.test=denormalize(x=pred.mlr,y="D.EUI.percent")
Actual.test = denormalize(x=i3.data.test[,1],y="D.EUI.percent")

jpeg(filename="MLR, Actual vs. Predicted EUI Change (Test, 2010-2015 Dataset).jpg")
plot( Actual.test, denorm.test, main="Actual vs. Predicted EUI Change (Test, 2010-2015 Dataset
)", xlab="Actual Values", ylab="Predicted Values")
dev.off()
```

```
## png
##   2
```

Decision Trees

Final_Random_Forest_Script_-_Current_Year.R

carle

Wed Oct 10 23:28:25 2018

```
########################################
####        Decision Trees Script     ####
########################################

####Load Packages####
library(ranger)
```

```
## Warning: package 'ranger' was built under R version 3.4.3
```

```
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.4.3
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.4.3
```

```
####Load and PreProcess Data ####
set.seed(1234)
setwd("C:/Users/carle/Documents/Data")
getwd()
```

```
## [1] "C:/Users/carle/Documents/Data"
```

```
# na.strings = "NA" populates every blank with "NA", which R will ignore
# stringsAsFactors = TRUE will let you read in 0's as real zeroes
# sep = "," is used for csv files
# check.names = FALSE permits multiple words in Column names to be seperated by a space.
# ^ [R] prefers names to read as one word (syntactically valid): "a.b.c" rather than "a b c"
# Load Data - Script is prepared for any Individual Year Dataset (2010 is used as example)
data <- read.csv(file.choose(), row.names = 1, check.names = FALSE, na.strings = "NA", sep = "
,", stringsAsFactors = TRUE, header = TRUE)
#Rename Columns to have 'syntactically valid' names
cnames <- cbind("EUI","Qualitative.EUI","Actual.E_C.Energy","Actual.Thermal.Energy", "Actual.T
otal.Energy",
                "Portfolio.Manager","Net.Rentable.Area","Exterior.Area","Gross.Floor.Area", "E
nclosed.Parking.Area",
                "Latitude","Longitude","Construction.Year","No.of.Structures",
```

```
                "Building.Class","Closest.Major.City","Climate.Zone","Electrically.Heated","Co
oling.Tower","Soft.Landscaping",
                "Occupant.Density","Vacancy.Rate","Weekly.Operating.Hours")
colnames(data) <- cnames

# Order the factor variable Qualitative.EUI
data[,"Qualitative.EUI"]=ordered(data[,"Qualitative.EUI"], levels = c("Poor","Fair","Good"))

# Sanity Check
is.ordered(data[,"Qualitative.EUI"]) #<- Should Say True
```

```
## [1] TRUE
```

```
#Remove rows with missing data
data<-data[complete.cases(data), ]
#check.is.na=is.na(data) to confirm that there are no missing data
#View(check.is.na)

#Functions
cb <- function(df, sep="\t", dec=",", max.size=(200*1000)){
  # Copy a data.frame to clipboard
  write.table(df, paste0("clipboard-", formatC(max.size, format="f", digits=0)), sep=sep, row.
names=FALSE, dec=dec)
}
#Function to round value to the nearest increment of x (i.e. 5)
mround <- function(x,base){
  base*round(x/base)
}

#### Define Model Varibles ####

#Select Independant variables
selected.indep.variables <- c("Latitude","Longitude","Construction.Year","No.of.Structures",
                        "Building.Class","Climate.Zone","Electrically.Heated","Cooling.Tower",
"Soft.Landscaping",
                        "Occupant.Density","Vacancy.Rate","Weekly.Operating.Hours")
# Edit to include chosen dependant variable: "EUI","Qualitative.EUI","Actual.E_C.Energy","Actu
al.Thermal.Energy", "Actual.Total.Energy"

#### Make Test/Training Sets####

# assign rows to either the training or test sets
data_ind <- sample(2, nrow(data), replace=TRUE, prob=c(0.67, 0.33))
# move all samples for the rows associated with ind=1 to the training set
data.train <- data[data_ind==1,selected.indep.variables]
# move all samples for the rows associated with ind=2 to the test set
data.test <- data[data_ind==2,selected.indep.variables]

####Classification Tree####
# now make a training label set (i.e. Qualitative.EUI=2)
data.train.Labels = data[data_ind==1, 2]
data.test.Labels = data[data_ind==2, 2]
```

```
data.trainset = cbind(data.train,data.train.Labels)
data.testset = cbind(data.test,data.test.Labels)
####Classification Tree (Change importance = none or impurity to see if accuraccy improves)
data.rfmodel <-ranger(data.train.Labels ~ ., data = data.trainset, min.node.size=1, importance
 = "impurity", write.forest = TRUE, classification = TRUE)
data.rfpred <- predict(data.rfmodel, data.test)
data.rfpred$predictions
```

```
##   [1] Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor
##  [15] Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor
##  [29] Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor
##  [43] Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor
##  [57] Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor
##  [71] Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor
##  [85] Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor Poor
##  [99] Poor Poor Poor Poor Poor
## Levels: Poor
```

```
Confusion_Matrix <- table(data.rfpred$predictions,data.test.Labels)
Confusion_Matrix
```

```
##       data.test.Labels
##        Poor Fair Good
##   Poor  103    0    0
```

```
Accuracy <- sum(diag(Confusion_Matrix))/sum(Confusion_Matrix)
Accuracy
```

```
## [1] 1
```

```
#Export Results

Confusion_Matrix = as.data.frame(Confusion_Matrix)
Confusion_Matrix
```

```
##   Var1 data.test.Labels Freq
## 1 Poor             Poor  103
## 2 Poor             Fair    0
## 3 Poor             Good    0
```

```
cb(Confusion_Matrix)

####Preprocess for regression####
#Select Independant variables
selected.indep.variables <- c("Latitude","Longitude","Construction.Year","No.of.Structures",
                              "Building.Class","Climate.Zone","Electrically.Heated","Cooling.T
ower","Soft.Landscaping",
                               "Occupant.Density","Vacancy.Rate","Weekly.Operating.Hours")
####For Regression-Run EUI code first as there are non-repeated formulas located in this secti
```

```
on####
####EUI####

#Use the following section code to copy the confusion matrix for predicted current year EUI.

# Make Test/Training Sets#
# Now make a training label set. Ensure the selected column number matches the dependent varia
ble
#(i.e. EUI=1, Actual.E_C.Energy = 3, Actual.Thermal.Energy=4, Actual.Total.Energy=5)
data.train.depvar = data[data_ind==1, 1]
data.test.depvar = data[data_ind==2, 1]
data.trainset = cbind(data.train,data.train.depvar)
data.testset = cbind(data.test,data.test.depvar)

# Regression tres
data.rfmodel <-ranger(data.train.depvar ~ ., data = data.trainset, importance = "impurity", wr
ite.forest = TRUE, classification = FALSE)
data.rfpred <- predict(data.rfmodel, data.test)
data.rfpred$predictions
```

```
##   [1] 27.27198 49.36602 49.26632 26.35852 27.11643 33.32748 40.42884
##   [8] 29.72342 38.57760 27.93303 32.08746 34.50441 36.37883 30.71715
##  [15] 32.29404 27.46828 36.14979 32.77126 23.54010 37.12195 34.74769
##  [22] 30.94427 37.86891 32.61697 39.53855 31.41981 30.75221 35.07187
##  [29] 33.05710 38.33182 29.32390 28.03445 25.83645 28.34854 24.86424
##  [36] 27.75925 31.93573 32.01255 32.31153 27.05812 32.36435 32.66612
##  [43] 25.21720 28.23750 21.78846 25.71059 32.60505 32.14321 25.47897
##  [50] 34.66026 23.09983 27.90339 33.55228 24.02308 23.95684 29.70676
##  [57] 24.83782 27.65297 32.61314 32.29878 24.03570 22.69178 26.81455
##  [64] 30.55026 29.65280 23.01953 25.08536 29.34254 25.73221 28.86463
##  [71] 29.87179 24.83106 26.51262 27.06159 30.66626 28.00583 25.87228
##  [78] 26.81175 29.17436 27.09036 28.58266 29.58464 26.11352 27.29604
##  [85] 23.30050 25.97616 27.65434 24.76239 27.67524 26.61825 26.14570
##  [92] 26.56392 23.58858 22.93278 30.34469 29.77009 27.08282 26.70398
##  [99] 24.00321 26.37028 22.94512 26.81134 26.25944
```

```
#Function to round value to the nearest increment of x (i.e. 5)
mround <- function(x,base){
  base*round(x/base)
}

Confusion_Matrix <- table(mround(data.rfpred$predictions,1),mround(data.test.depvar,1))
Confusion_Matrix
```

```
##
##      11 15 16 17 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
##   22  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
##   23  0  1  0  0  0  1  1  0  0  0  1  1  1  0  0  0  0  0  0  0  0  0  0
##   24  0  0  0  1  0  1  0  0  0  0  0  1  2  0  0  0  0  0  0  0  1  0  0
##   25  0  0  0  0  0  0  1  0  0  1  1  0  2  0  2  0  0  0  0  0  0  0  0
##   26  1  0  1  0  0  1  2  1  0  1  0  0  0  1  0  1  0  0  0  0  0  0  0
##   27  0  1  0  2  0  2  1  2  0  2  0  1  0  0  1  0  0  0  0  0  0  0  1
```

```
##   28  0  0  0  0  0  1  1  0  1  0  0  1  1  1  1  2  0  0  0  0  0  0  0
##   29  0  0  0  0  0  0  1  1  0  1  1  0  0  0  0  1  0  0  0  0  0  0  0
##   30  0  0  0  0  2  0  1  0  0  1  1  0  1  0  0  0  0  0  0  0  0  0  0
##   31  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  2  0  0  1  0  0  1
##   32  0  0  0  0  0  0  0  0  0  0  0  1  1  0  4  0  0  0  0  0  0  0  1
##   33  0  0  0  0  0  0  0  0  0  0  0  1  0  1  1  1  0  1  0  0  1  0  0
##   34  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
##   35  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  1  0  0  0
##   36  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1
##   37  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
##   38  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0
##   39  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   40  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
##   49  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##
##       38 39 40 41 42 43 44 46 48 50 57
##   22  0  0  0  0  0  0  0  0  0  0  0
##   23  0  0  0  0  0  0  0  0  0  0  0
##   24  0  0  0  0  0  0  0  0  0  0  0
##   25  0  0  0  0  0  0  0  0  0  0  0
##   26  0  0  0  0  0  0  0  1  0  0  0
##   27  0  0  0  0  0  0  1  0  0  0  1
##   28  0  0  1  0  0  0  0  0  0  0  0
##   29  0  0  0  0  0  0  0  0  0  0  0
##   30  0  0  0  1  0  0  0  0  0  0  0
##   31  0  0  0  0  0  0  0  0  0  0  0
##   32  0  1  0  0  0  0  0  0  0  0  0
##   33  0  0  0  0  0  1  0  0  0  0  0
##   34  0  0  0  0  0  0  0  0  0  0  0
##   35  1  0  0  0  0  0  0  0  0  0  0
##   36  0  0  0  0  0  0  0  0  0  0  0
##   37  0  0  0  0  0  0  0  0  0  0  0
##   38  0  0  0  0  0  0  0  0  0  0  0
##   39  0  0  0  1  0  0  0  0  0  0  0
##   40  0  0  0  0  1  0  0  0  0  0  0
##   49  0  0  0  0  0  0  0  0  1  1  0
```

```
Accuracy <- sum(diag(Confusion_Matrix))/sum(Confusion_Matrix)
Accuracy
```

```
## [1] 0.06796117
```

```
Confusion_Matrix = as.data.frame(Confusion_Matrix)
Confusion_Matrix
```

```
##     Var1 Var2 Freq
## 1     22   11    0
## 2     23   11    0
## 3     24   11    0
## 4     25   11    0
## 5     26   11    1
```

```
## 6     27   11    0
## 7     28   11    0
## 8     29   11    0
## 9     30   11    0
## 10    31   11    0
## 11    32   11    0
## 12    33   11    0
## 13    34   11    0
## 14    35   11    0
## 15    36   11    0
## 16    37   11    0
## 17    38   11    0
## 18    39   11    0
## 19    40   11    0
## 20    49   11    0
## 21    22   15    0
## 22    23   15    1
## 23    24   15    0
## 24    25   15    0
## 25    26   15    0
## 26    27   15    1
## 27    28   15    0
## 28    29   15    0
## 29    30   15    0
## 30    31   15    0
## 31    32   15    0
## 32    33   15    0
## 33    34   15    0
## 34    35   15    0
## 35    36   15    0
## 36    37   15    0
## 37    38   15    0
## 38    39   15    0
## 39    40   15    0
## 40    49   15    0
## 41    22   16    0
## 42    23   16    0
## 43    24   16    0
## 44    25   16    0
## 45    26   16    1
## 46    27   16    0
## 47    28   16    0
## 48    29   16    0
## 49    30   16    0
## 50    31   16    0
## 51    32   16    0
## 52    33   16    0
## 53    34   16    0
## 54    35   16    0
## 55    36   16    0
## 56    37   16    0
## 57    38   16    0
## 58    39   16    0
## 59    40   16    0
```

```
## 60    49   16    0
## 61    22   17    0
## 62    23   17    0
## 63    24   17    1
## 64    25   17    0
## 65    26   17    0
## 66    27   17    2
## 67    28   17    0
## 68    29   17    0
## 69    30   17    0
## 70    31   17    0
## 71    32   17    0
## 72    33   17    0
## 73    34   17    0
## 74    35   17    0
## 75    36   17    0
## 76    37   17    0
## 77    38   17    0
## 78    39   17    0
## 79    40   17    0
## 80    49   17    0
## 81    22   19    0
## 82    23   19    0
## 83    24   19    0
## 84    25   19    0
## 85    26   19    0
## 86    27   19    0
## 87    28   19    0
## 88    29   19    0
## 89    30   19    2
## 90    31   19    0
## 91    32   19    0
## 92    33   19    0
## 93    34   19    0
## 94    35   19    0
## 95    36   19    0
## 96    37   19    0
## 97    38   19    0
## 98    39   19    0
## 99    40   19    0
## 100   49   19    0
## 101   22   20    0
## 102   23   20    1
## 103   24   20    1
## 104   25   20    0
## 105   26   20    1
## 106   27   20    2
## 107   28   20    1
## 108   29   20    0
## 109   30   20    0
## 110   31   20    0
## 111   32   20    0
## 112   33   20    0
## 113   34   20    0
```

```
## 114   35   20    0
## 115   36   20    0
## 116   37   20    0
## 117   38   20    0
## 118   39   20    0
## 119   40   20    0
## 120   49   20    0
## 121   22   21    0
## 122   23   21    1
## 123   24   21    0
## 124   25   21    1
## 125   26   21    2
## 126   27   21    1
## 127   28   21    1
## 128   29   21    1
## 129   30   21    1
## 130   31   21    0
## 131   32   21    0
## 132   33   21    0
## 133   34   21    0
## 134   35   21    0
## 135   36   21    0
## 136   37   21    0
## 137   38   21    0
## 138   39   21    0
## 139   40   21    0
## 140   49   21    0
## 141   22   22    0
## 142   23   22    0
## 143   24   22    0
## 144   25   22    0
## 145   26   22    1
## 146   27   22    2
## 147   28   22    0
## 148   29   22    1
## 149   30   22    0
## 150   31   22    0
## 151   32   22    0
## 152   33   22    0
## 153   34   22    0
## 154   35   22    0
## 155   36   22    0
## 156   37   22    0
## 157   38   22    0
## 158   39   22    0
## 159   40   22    0
## 160   49   22    0
## 161   22   23    0
## 162   23   23    0
## 163   24   23    0
## 164   25   23    0
## 165   26   23    0
## 166   27   23    0
## 167   28   23    1
```

```
## 168   29   23    0
## 169   30   23    0
## 170   31   23    1
## 171   32   23    0
## 172   33   23    0
## 173   34   23    0
## 174   35   23    0
## 175   36   23    0
## 176   37   23    0
## 177   38   23    0
## 178   39   23    0
## 179   40   23    0
## 180   49   23    0
## 181   22   24    0
## 182   23   24    0
## 183   24   24    0
## 184   25   24    1
## 185   26   24    1
## 186   27   24    2
## 187   28   24    0
## 188   29   24    1
## 189   30   24    1
## 190   31   24    0
## 191   32   24    0
## 192   33   24    0
## 193   34   24    0
## 194   35   24    0
## 195   36   24    0
## 196   37   24    0
## 197   38   24    0
## 198   39   24    0
## 199   40   24    0
## 200   49   24    0
## 201   22   25    0
## 202   23   25    1
## 203   24   25    0
## 204   25   25    1
## 205   26   25    0
## 206   27   25    0
## 207   28   25    0
## 208   29   25    1
## 209   30   25    1
## 210   31   25    0
## 211   32   25    0
## 212   33   25    0
## 213   34   25    0
## 214   35   25    0
## 215   36   25    0
## 216   37   25    0
## 217   38   25    0
## 218   39   25    0
## 219   40   25    0
## 220   49   25    0
## 221   22   26    0
```

```
## 222   23   26    1
## 223   24   26    1
## 224   25   26    0
## 225   26   26    0
## 226   27   26    1
## 227   28   26    1
## 228   29   26    0
## 229   30   26    0
## 230   31   26    1
## 231   32   26    1
## 232   33   26    1
## 233   34   26    0
## 234   35   26    0
## 235   36   26    0
## 236   37   26    0
## 237   38   26    0
## 238   39   26    0
## 239   40   26    0
## 240   49   26    0
## 241   22   27    0
## 242   23   27    1
## 243   24   27    2
## 244   25   27    2
## 245   26   27    0
## 246   27   27    0
## 247   28   27    1
## 248   29   27    0
## 249   30   27    1
## 250   31   27    0
## 251   32   27    1
## 252   33   27    0
## 253   34   27    1
## 254   35   27    1
## 255   36   27    0
## 256   37   27    0
## 257   38   27    0
## 258   39   27    0
## 259   40   27    0
## 260   49   27    0
## 261   22   28    1
## 262   23   28    0
## 263   24   28    0
## 264   25   28    0
## 265   26   28    1
## 266   27   28    0
## 267   28   28    1
## 268   29   28    0
## 269   30   28    0
## 270   31   28    0
## 271   32   28    0
## 272   33   28    1
## 273   34   28    0
## 274   35   28    0
## 275   36   28    0
```

```
## 276   37   28    0
## 277   38   28    0
## 278   39   28    0
## 279   40   28    0
## 280   49   28    0
## 281   22   29    0
## 282   23   29    0
## 283   24   29    0
## 284   25   29    2
## 285   26   29    0
## 286   27   29    1
## 287   28   29    1
## 288   29   29    0
## 289   30   29    0
## 290   31   29    0
## 291   32   29    4
## 292   33   29    1
## 293   34   29    0
## 294   35   29    0
## 295   36   29    0
## 296   37   29    0
## 297   38   29    0
## 298   39   29    0
## 299   40   29    0
## 300   49   29    0
## 301   22   30    0
## 302   23   30    0
## 303   24   30    0
## 304   25   30    0
## 305   26   30    1
## 306   27   30    0
## 307   28   30    2
## 308   29   30    1
## 309   30   30    0
## 310   31   30    0
## 311   32   30    0
## 312   33   30    1
## 313   34   30    0
## 314   35   30    0
## 315   36   30    0
## 316   37   30    0
## 317   38   30    1
## 318   39   30    0
## 319   40   30    0
## 320   49   30    0
## 321   22   31    0
## 322   23   31    0
## 323   24   31    0
## 324   25   31    0
## 325   26   31    0
## 326   27   31    0
## 327   28   31    0
## 328   29   31    0
## 329   30   31    0
```

```
## 330   31   31    2
## 331   32   31    0
## 332   33   31    0
## 333   34   31    0
## 334   35   31    1
## 335   36   31    0
## 336   37   31    0
## 337   38   31    0
## 338   39   31    0
## 339   40   31    1
## 340   49   31    0
## 341   22   32    0
## 342   23   32    0
## 343   24   32    0
## 344   25   32    0
## 345   26   32    0
## 346   27   32    0
## 347   28   32    0
## 348   29   32    0
## 349   30   32    0
## 350   31   32    0
## 351   32   32    0
## 352   33   32    1
## 353   34   32    0
## 354   35   32    0
## 355   36   32    0
## 356   37   32    0
## 357   38   32    0
## 358   39   32    0
## 359   40   32    0
## 360   49   32    0
## 361   22   33    0
## 362   23   33    0
## 363   24   33    0
## 364   25   33    0
## 365   26   33    0
## 366   27   33    0
## 367   28   33    0
## 368   29   33    0
## 369   30   33    0
## 370   31   33    0
## 371   32   33    0
## 372   33   33    0
## 373   34   33    0
## 374   35   33    0
## 375   36   33    0
## 376   37   33    0
## 377   38   33    1
## 378   39   33    0
## 379   40   33    0
## 380   49   33    0
## 381   22   34    0
## 382   23   34    0
## 383   24   34    0
```

```
## 384   25   34    0
## 385   26   34    0
## 386   27   34    0
## 387   28   34    0
## 388   29   34    0
## 389   30   34    0
## 390   31   34    1
## 391   32   34    0
## 392   33   34    0
## 393   34   34    0
## 394   35   34    1
## 395   36   34    0
## 396   37   34    1
## 397   38   34    0
## 398   39   34    0
## 399   40   34    0
## 400   49   34    0
## 401   22   35    0
## 402   23   35    0
## 403   24   35    1
## 404   25   35    0
## 405   26   35    0
## 406   27   35    0
## 407   28   35    0
## 408   29   35    0
## 409   30   35    0
## 410   31   35    0
## 411   32   35    0
## 412   33   35    1
## 413   34   35    0
## 414   35   35    0
## 415   36   35    0
## 416   37   35    0
## 417   38   35    0
## 418   39   35    0
## 419   40   35    0
## 420   49   35    0
## 421   22   36    0
## 422   23   36    0
## 423   24   36    0
## 424   25   36    0
## 425   26   36    0
## 426   27   36    0
## 427   28   36    0
## 428   29   36    0
## 429   30   36    0
## 430   31   36    0
## 431   32   36    0
## 432   33   36    0
## 433   34   36    0
## 434   35   36    0
## 435   36   36    1
## 436   37   36    0
## 437   38   36    0
```

```
## 438   39   36    0
## 439   40   36    0
## 440   49   36    0
## 441   22   37    0
## 442   23   37    0
## 443   24   37    0
## 444   25   37    0
## 445   26   37    0
## 446   27   37    1
## 447   28   37    0
## 448   29   37    0
## 449   30   37    0
## 450   31   37    1
## 451   32   37    1
## 452   33   37    0
## 453   34   37    0
## 454   35   37    0
## 455   36   37    1
## 456   37   37    0
## 457   38   37    0
## 458   39   37    0
## 459   40   37    0
## 460   49   37    0
## 461   22   38    0
## 462   23   38    0
## 463   24   38    0
## 464   25   38    0
## 465   26   38    0
## 466   27   38    0
## 467   28   38    0
## 468   29   38    0
## 469   30   38    0
## 470   31   38    0
## 471   32   38    0
## 472   33   38    0
## 473   34   38    0
## 474   35   38    1
## 475   36   38    0
## 476   37   38    0
## 477   38   38    0
## 478   39   38    0
## 479   40   38    0
## 480   49   38    0
## 481   22   39    0
## 482   23   39    0
## 483   24   39    0
## 484   25   39    0
## 485   26   39    0
## 486   27   39    0
## 487   28   39    0
## 488   29   39    0
## 489   30   39    0
## 490   31   39    0
## 491   32   39    1
```

```
## 492   33   39    0
## 493   34   39    0
## 494   35   39    0
## 495   36   39    0
## 496   37   39    0
## 497   38   39    0
## 498   39   39    0
## 499   40   39    0
## 500   49   39    0
## 501   22   40    0
## 502   23   40    0
## 503   24   40    0
## 504   25   40    0
## 505   26   40    0
## 506   27   40    0
## 507   28   40    1
## 508   29   40    0
## 509   30   40    0
## 510   31   40    0
## 511   32   40    0
## 512   33   40    0
## 513   34   40    0
## 514   35   40    0
## 515   36   40    0
## 516   37   40    0
## 517   38   40    0
## 518   39   40    0
## 519   40   40    0
## 520   49   40    0
## 521   22   41    0
## 522   23   41    0
## 523   24   41    0
## 524   25   41    0
## 525   26   41    0
## 526   27   41    0
## 527   28   41    0
## 528   29   41    0
## 529   30   41    1
## 530   31   41    0
## 531   32   41    0
## 532   33   41    0
## 533   34   41    0
## 534   35   41    0
## 535   36   41    0
## 536   37   41    0
## 537   38   41    0
## 538   39   41    1
## 539   40   41    0
## 540   49   41    0
## 541   22   42    0
## 542   23   42    0
## 543   24   42    0
## 544   25   42    0
## 545   26   42    0
```

```
## 546   27   42    0
## 547   28   42    0
## 548   29   42    0
## 549   30   42    0
## 550   31   42    0
## 551   32   42    0
## 552   33   42    0
## 553   34   42    0
## 554   35   42    0
## 555   36   42    0
## 556   37   42    0
## 557   38   42    0
## 558   39   42    0
## 559   40   42    1
## 560   49   42    0
## 561   22   43    0
## 562   23   43    0
## 563   24   43    0
## 564   25   43    0
## 565   26   43    0
## 566   27   43    0
## 567   28   43    0
## 568   29   43    0
## 569   30   43    0
## 570   31   43    0
## 571   32   43    0
## 572   33   43    1
## 573   34   43    0
## 574   35   43    0
## 575   36   43    0
## 576   37   43    0
## 577   38   43    0
## 578   39   43    0
## 579   40   43    0
## 580   49   43    0
## 581   22   44    0
## 582   23   44    0
## 583   24   44    0
## 584   25   44    0
## 585   26   44    0
## 586   27   44    1
## 587   28   44    0
## 588   29   44    0
## 589   30   44    0
## 590   31   44    0
## 591   32   44    0
## 592   33   44    0
## 593   34   44    0
## 594   35   44    0
## 595   36   44    0
## 596   37   44    0
## 597   38   44    0
## 598   39   44    0
## 599   40   44    0
```

```
## 600   49   44    0
## 601   22   46    0
## 602   23   46    0
## 603   24   46    0
## 604   25   46    0
## 605   26   46    1
## 606   27   46    0
## 607   28   46    0
## 608   29   46    0
## 609   30   46    0
## 610   31   46    0
## 611   32   46    0
## 612   33   46    0
## 613   34   46    0
## 614   35   46    0
## 615   36   46    0
## 616   37   46    0
## 617   38   46    0
## 618   39   46    0
## 619   40   46    0
## 620   49   46    0
## 621   22   48    0
## 622   23   48    0
## 623   24   48    0
## 624   25   48    0
## 625   26   48    0
## 626   27   48    0
## 627   28   48    0
## 628   29   48    0
## 629   30   48    0
## 630   31   48    0
## 631   32   48    0
## 632   33   48    0
## 633   34   48    0
## 634   35   48    0
## 635   36   48    0
## 636   37   48    0
## 637   38   48    0
## 638   39   48    0
## 639   40   48    0
## 640   49   48    1
## 641   22   50    0
## 642   23   50    0
## 643   24   50    0
## 644   25   50    0
## 645   26   50    0
## 646   27   50    0
## 647   28   50    0
## 648   29   50    0
## 649   30   50    0
## 650   31   50    0
## 651   32   50    0
## 652   33   50    0
## 653   34   50    0
```

```
## 654   35   50    0
## 655   36   50    0
## 656   37   50    0
## 657   38   50    0
## 658   39   50    0
## 659   40   50    0
## 660   49   50    1
## 661   22   57    0
## 662   23   57    0
## 663   24   57    0
## 664   25   57    0
## 665   26   57    0
## 666   27   57    1
## 667   28   57    0
## 668   29   57    0
## 669   30   57    0
## 670   31   57    0
## 671   32   57    0
## 672   33   57    0
## 673   34   57    0
## 674   35   57    0
## 675   36   57    0
## 676   37   57    0
## 677   38   57    0
## 678   39   57    0
## 679   40   57    0
## 680   49   57    0
```

```
cb(Confusion_Matrix)

#### Actual.E_C.Energy ####
#Use the following section code to copy the confusion matrix for predicted current year Actual
.E_C.Energy.

#Make Test/Training Sets#
# Now make a training label set. Ensure the selected column number matches the dependent varia
ble
#(i.e. EUI=1, Actual.E_C.Energy = 3, Actual.Thermal.Energy=4, Actual.Total.Energy=5)
data.train.depvar = data[data_ind==1, 3]
data.test.depvar = data[data_ind==2, 3]
data.trainset = cbind(data.train,data.train.depvar)
data.testset = cbind(data.test,data.test.depvar)

# Regression tres
data.rfmodel <-ranger(data.train.depvar ~ ., data = data.trainset, importance = "impurity", wr
ite.forest = TRUE, classification = FALSE)
data.rfpred <- predict(data.rfmodel, data.test)
data.rfpred$predictions
```

```
##   [1] 18.56336 25.63482 18.09769 20.92611 18.59383 20.21091 21.32659
##   [8] 19.45403 22.40366 22.47915 21.06609 20.74318 21.20776 16.28750
##  [15] 17.31068 19.63600 20.21981 24.14316 21.25897 21.09000 18.33355
##  [22] 21.21946 19.95774 20.66166 18.80967 21.75810 21.03648 22.31655
```

```
##  [29] 17.22117 22.17959 19.71055 22.72150 19.48600 20.31041 18.88361
##  [36] 20.45032 19.88929 17.97020 18.74465 16.68248 18.69813 22.22937
##  [43] 20.04539 20.82047 17.84437 19.00690 21.15905 22.67405 20.67531
##  [50] 21.83672 17.25673 21.53698 18.63306 20.13090 20.68483 17.55668
##  [57] 18.75893 19.95695 28.70077 22.73734 18.78850 19.79342 22.06446
##  [64] 18.71643 23.45062 16.48820 19.69531 21.14583 18.40328 20.41496
##  [71] 21.98464 20.63969 21.00645 18.06242 22.39791 21.33435 16.77671
##  [78] 20.55663 20.56309 21.89272 20.28979 20.37255 22.50490 20.49238
##  [85] 16.83065 21.96748 20.66002 21.29827 17.59414 21.34980 18.44334
##  [92] 18.01741 17.94434 16.89394 20.80184 19.25551 22.01341 21.61925
##  [99] 19.08026 18.00798 20.76466 16.95696 16.00363
```

```
Confusion_Matrix <- table(mround(data.rfpred$predictions,1),mround(data.test.depvar,1))
Confusion_Matrix
```

```
##
##      8 10 11 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 31 35 42
##   16 0  0  0  0  0  0  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   17 1  0  0  1  2  0  2  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   18 0  1  1  0  4  1  1  1  0  2  0  0  1  0  0  0  0  0  0  0  0  0  0
##   19 0  1  0  1  0  0  3  2  3  1  0  0  1  0  0  0  1  0  0  0  1  0  0
##   20 0  0  0  0  1  2  1  0  3  4  1  1  0  0  0  0  1  1  0  1  0  0  1
##   21 0  0  1  0  1  3  1  0  1  2  2  3  1  2  2  0  0  1  1  0  2  0  0
##   22 0  0  0  2  0  2  2  2  0  1  2  0  2  0  0  1  0  1  0  0  0  0  0
##   23 0  1  0  0  0  1  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  1  0
##   24 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
##   26 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
##   29 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##
##      48 55 62
##   16  0  0  0
##   17  0  0  0
##   18  0  0  0
##   19  0  0  1
##   20  0  0  0
##   21  1  1  0
##   22  0  0  0
##   23  0  0  0
##   24  0  0  0
##   26  0  0  0
##   29  0  0  0
```

```
Accuracy <- sum(diag(Confusion_Matrix))/sum(Confusion_Matrix)
Accuracy
```

```
## [1] 0.0776699
```

```
Confusion_Matrix = as.data.frame(Confusion_Matrix)
Confusion_Matrix
```

```
##     Var1 Var2 Freq
## 1     16    8    0
## 2     17    8    1
## 3     18    8    0
## 4     19    8    0
## 5     20    8    0
## 6     21    8    0
## 7     22    8    0
## 8     23    8    0
## 9     24    8    0
## 10    26    8    0
## 11    29    8    0
## 12    16   10    0
## 13    17   10    0
## 14    18   10    1
## 15    19   10    1
## 16    20   10    0
## 17    21   10    0
## 18    22   10    0
## 19    23   10    1
## 20    24   10    0
## 21    26   10    0
## 22    29   10    0
## 23    16   11    0
## 24    17   11    0
## 25    18   11    1
## 26    19   11    0
## 27    20   11    0
## 28    21   11    1
## 29    22   11    0
## 30    23   11    0
## 31    24   11    0
## 32    26   11    0
## 33    29   11    0
## 34    16   13    0
## 35    17   13    1
## 36    18   13    0
## 37    19   13    1
## 38    20   13    0
## 39    21   13    0
## 40    22   13    2
## 41    23   13    0
## 42    24   13    0
## 43    26   13    0
## 44    29   13    0
## 45    16   14    0
## 46    17   14    2
## 47    18   14    4
## 48    19   14    0
## 49    20   14    1
## 50    21   14    1
## 51    22   14    0
## 52    23   14    0
## 53    24   14    0
```

```
## 54    26   14    0
## 55    29   14    0
## 56    16   15    0
## 57    17   15    0
## 58    18   15    1
## 59    19   15    0
## 60    20   15    2
## 61    21   15    3
## 62    22   15    2
## 63    23   15    1
## 64    24   15    0
## 65    26   15    0
## 66    29   15    0
## 67    16   16    1
## 68    17   16    2
## 69    18   16    1
## 70    19   16    3
## 71    20   16    1
## 72    21   16    1
## 73    22   16    2
## 74    23   16    0
## 75    24   16    0
## 76    26   16    0
## 77    29   16    0
## 78    16   17    1
## 79    17   17    0
## 80    18   17    1
## 81    19   17    2
## 82    20   17    0
## 83    21   17    0
## 84    22   17    2
## 85    23   17    0
## 86    24   17    0
## 87    26   17    0
## 88    29   17    0
## 89    16   18    1
## 90    17   18    2
## 91    18   18    0
## 92    19   18    3
## 93    20   18    3
## 94    21   18    1
## 95    22   18    0
## 96    23   18    0
## 97    24   18    0
## 98    26   18    0
## 99    29   18    1
## 100   16   19    0
## 101   17   19    0
## 102   18   19    2
## 103   19   19    1
## 104   20   19    4
## 105   21   19    2
## 106   22   19    1
## 107   23   19    0
```

```
## 108   24   19    0
## 109   26   19    0
## 110   29   19    0
## 111   16   20    0
## 112   17   20    0
## 113   18   20    0
## 114   19   20    0
## 115   20   20    1
## 116   21   20    2
## 117   22   20    2
## 118   23   20    0
## 119   24   20    0
## 120   26   20    0
## 121   29   20    0
## 122   16   21    0
## 123   17   21    0
## 124   18   21    0
## 125   19   21    0
## 126   20   21    1
## 127   21   21    3
## 128   22   21    0
## 129   23   21    0
## 130   24   21    0
## 131   26   21    0
## 132   29   21    0
## 133   16   22    0
## 134   17   22    0
## 135   18   22    1
## 136   19   22    1
## 137   20   22    0
## 138   21   22    1
## 139   22   22    2
## 140   23   22    1
## 141   24   22    0
## 142   26   22    0
## 143   29   22    0
## 144   16   23    0
## 145   17   23    0
## 146   18   23    0
## 147   19   23    0
## 148   20   23    0
## 149   21   23    2
## 150   22   23    0
## 151   23   23    0
## 152   24   23    0
## 153   26   23    1
## 154   29   23    0
## 155   16   24    0
## 156   17   24    0
## 157   18   24    0
## 158   19   24    0
## 159   20   24    0
## 160   21   24    2
## 161   22   24    0
```

```
## 162   23   24    0
## 163   24   24    0
## 164   26   24    0
## 165   29   24    0
## 166   16   25    0
## 167   17   25    0
## 168   18   25    0
## 169   19   25    0
## 170   20   25    0
## 171   21   25    0
## 172   22   25    1
## 173   23   25    0
## 174   24   25    0
## 175   26   25    0
## 176   29   25    0
## 177   16   26    0
## 178   17   26    0
## 179   18   26    0
## 180   19   26    1
## 181   20   26    1
## 182   21   26    0
## 183   22   26    0
## 184   23   26    1
## 185   24   26    0
## 186   26   26    0
## 187   29   26    0
## 188   16   27    0
## 189   17   27    0
## 190   18   27    0
## 191   19   27    0
## 192   20   27    1
## 193   21   27    1
## 194   22   27    1
## 195   23   27    0
## 196   24   27    1
## 197   26   27    0
## 198   29   27    0
## 199   16   28    0
## 200   17   28    0
## 201   18   28    0
## 202   19   28    0
## 203   20   28    0
## 204   21   28    1
## 205   22   28    0
## 206   23   28    0
## 207   24   28    0
## 208   26   28    0
## 209   29   28    0
## 210   16   29    0
## 211   17   29    0
## 212   18   29    0
## 213   19   29    0
## 214   20   29    1
## 215   21   29    0
```

```
## 216   22   29    0
## 217   23   29    0
## 218   24   29    0
## 219   26   29    0
## 220   29   29    0
## 221   16   31    0
## 222   17   31    0
## 223   18   31    0
## 224   19   31    1
## 225   20   31    0
## 226   21   31    2
## 227   22   31    0
## 228   23   31    0
## 229   24   31    0
## 230   26   31    0
## 231   29   31    0
## 232   16   35    0
## 233   17   35    0
## 234   18   35    0
## 235   19   35    0
## 236   20   35    0
## 237   21   35    0
## 238   22   35    0
## 239   23   35    1
## 240   24   35    0
## 241   26   35    0
## 242   29   35    0
## 243   16   42    0
## 244   17   42    0
## 245   18   42    0
## 246   19   42    0
## 247   20   42    1
## 248   21   42    0
## 249   22   42    0
## 250   23   42    0
## 251   24   42    0
## 252   26   42    0
## 253   29   42    0
## 254   16   48    0
## 255   17   48    0
## 256   18   48    0
## 257   19   48    0
## 258   20   48    0
## 259   21   48    1
## 260   22   48    0
## 261   23   48    0
## 262   24   48    0
## 263   26   48    0
## 264   29   48    0
## 265   16   55    0
## 266   17   55    0
## 267   18   55    0
## 268   19   55    0
## 269   20   55    0
```

```
## 270   21   55    1
## 271   22   55    0
## 272   23   55    0
## 273   24   55    0
## 274   26   55    0
## 275   29   55    0
## 276   16   62    0
## 277   17   62    0
## 278   18   62    0
## 279   19   62    1
## 280   20   62    0
## 281   21   62    0
## 282   22   62    0
## 283   23   62    0
## 284   24   62    0
## 285   26   62    0
## 286   29   62    0
```

```
cb(Confusion_Matrix)

#### Actual.Thermal.Energy ####
#Use the following section code to copy the confusion matrix for predicted current year Actual
.Thermal.Energy.

#Make Test/Training Sets#
# Now make a training label set. Ensure the selected column number matches the dependent varia
ble
#(i.e. EUI=1, Actual.E_C.Energy = 3, Actual.Thermal.Energy=4, Actual.Total.Energy=5)
data.train.depvar = data[data_ind==1, 4]
data.test.depvar = data[data_ind==2, 4]
data.trainset = cbind(data.train,data.train.depvar)
data.testset = cbind(data.test,data.test.depvar)

# Regression tres
data.rfmodel <-ranger(data.train.depvar ~ ., data = data.trainset, importance = "impurity", wr
ite.forest = TRUE, classification = FALSE)
data.rfpred <- predict(data.rfmodel, data.test)
data.rfpred$predictions
```

```
##   [1]  8.990102 12.271241 13.067960 10.655313 14.634907  9.094226 10.213442
##   [8] 12.255765  6.878928 11.047329 14.996609  8.367802 17.049066 12.303797
##  [15] 12.683639 11.692183 16.738549 16.570383  7.201869 17.562003 10.463327
##  [22] 12.968807  7.588185 10.934648 19.028219 16.857736 17.130453 18.481683
##  [29] 15.301348 11.548404  9.211961  8.148847  9.544164 10.694771  8.787745
##  [36] 10.765766 12.475145 15.021953 17.676869 10.669811 14.326434  6.570367
##  [43]  9.405918 11.653171  7.364611  8.210624 16.079889 12.569504  9.472168
##  [50] 15.414029  7.152204 10.961392  8.208523  7.703865  7.479219 12.256238
##  [57]  9.871172 11.495919  6.234185 10.207804  7.962543  6.949478  7.316741
##  [64] 14.494825 10.074725  7.789403  7.931892  9.309862  8.145148 15.558627
##  [71]  8.367111  7.336410  7.461342 10.853642  6.141479  9.912300 11.413464
##  [78] 11.536726 10.161758  9.581037 10.976113 10.197516  9.145957 10.004222
##  [85]  7.354248  7.248682  8.595116  8.248510 11.619458  7.390876 11.527185
##  [92] 12.745840  8.003115  6.987210  8.557099 10.809370 11.495071  9.964947
```

```
##  [99]  9.489138 13.126197  6.512328 11.089980 11.432132
```

```
#Function to round value to the nearest increment of x (i.e. 5)
mround <- function(x,base){
  base*round(x/base)
}

Confusion_Matrix <- table(mround(data.rfpred$predictions,1),mround(data.test.depvar,1))
Confusion_Matrix
```

```
##
##       0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 21 24 25
##   6   0 1 0 0 0 0 0 0 0 0  0  1  0  0  0  0  0  0  0  0  0  0  0
##   7   0 0 0 0 0 3 1 2 2 1  4  0  0  1  0  0  1  0  0  0  0  0  0
##   8   1 0 0 1 0 2 1 1 3 2  0  1  1  0  0  0  0  0  0  0  0  0  0
##   9   0 1 1 0 2 1 1 0 0 0  2  1  1  0  0  0  0  0  1  0  0  0  0
##   10  1 1 1 0 1 2 1 1 1 2  0  0  0  0  0  1  0  0  0  0  0  0  0
##   11  0 0 1 0 1 1 0 0 1 3  2  2  0  1  0  0  1  0  0  1  0  0  1
##   12  0 0 0 0 0 0 1 1 0 2  2  0  2  0  1  1  0  0  0  0  0  1  0
##   13  0 0 0 0 0 0 1 0 0 0  0  0  0  2  1  0  1  1  0  0  0  0  0
##   14  0 0 0 0 0 0 0 0 0 0  0  0  1  0  0  0  0  1  0  0  0  0  0
##   15  0 0 0 0 0 0 0 0 0 0  0  1  0  0  1  1  1  0  0  0  0  1  0
##   16  0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  1  1  0  0  0  0  0
##   17  0 0 0 0 0 0 0 0 0 0  0  0  0  1  1  0  1  1  0  1  0  0  0
##   18  0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  2  1  0  0
##   19  0 0 0 0 0 0 0 0 0 0  0  0  0  1  0  0  0  0  0  0  0  0  0
```

```
Accuracy <- sum(diag(Confusion_Matrix))/sum(Confusion_Matrix)
Accuracy
```

```
## [1] 0.03883495
```

```
Confusion_Matrix = as.data.frame(Confusion_Matrix)
Confusion_Matrix
```

```
##     Var1 Var2 Freq
## 1      6    0    0
## 2      7    0    0
## 3      8    0    1
## 4      9    0    0
## 5     10    0    1
## 6     11    0    0
## 7     12    0    0
## 8     13    0    0
## 9     14    0    0
## 10    15    0    0
## 11    16    0    0
## 12    17    0    0
## 13    18    0    0
## 14    19    0    0
```

```
## 15     6    1    1
## 16     7    1    0
## 17     8    1    0
## 18     9    1    1
## 19    10    1    1
## 20    11    1    0
## 21    12    1    0
## 22    13    1    0
## 23    14    1    0
## 24    15    1    0
## 25    16    1    0
## 26    17    1    0
## 27    18    1    0
## 28    19    1    0
## 29     6    2    0
## 30     7    2    0
## 31     8    2    0
## 32     9    2    1
## 33    10    2    1
## 34    11    2    1
## 35    12    2    0
## 36    13    2    0
## 37    14    2    0
## 38    15    2    0
## 39    16    2    0
## 40    17    2    0
## 41    18    2    0
## 42    19    2    0
## 43     6    3    0
## 44     7    3    0
## 45     8    3    1
## 46     9    3    0
## 47    10    3    0
## 48    11    3    0
## 49    12    3    0
## 50    13    3    0
## 51    14    3    0
## 52    15    3    0
## 53    16    3    0
## 54    17    3    0
## 55    18    3    0
## 56    19    3    0
## 57     6    4    0
## 58     7    4    0
## 59     8    4    0
## 60     9    4    2
## 61    10    4    1
## 62    11    4    1
## 63    12    4    0
## 64    13    4    0
## 65    14    4    0
## 66    15    4    0
## 67    16    4    0
## 68    17    4    0
```

```
## 69    18    4    0
## 70    19    4    0
## 71     6    5    0
## 72     7    5    3
## 73     8    5    2
## 74     9    5    1
## 75    10    5    2
## 76    11    5    1
## 77    12    5    0
## 78    13    5    0
## 79    14    5    0
## 80    15    5    0
## 81    16    5    0
## 82    17    5    0
## 83    18    5    0
## 84    19    5    0
## 85     6    6    0
## 86     7    6    1
## 87     8    6    1
## 88     9    6    1
## 89    10    6    1
## 90    11    6    0
## 91    12    6    1
## 92    13    6    1
## 93    14    6    0
## 94    15    6    0
## 95    16    6    0
## 96    17    6    0
## 97    18    6    0
## 98    19    6    0
## 99     6    7    0
## 100    7    7    2
## 101    8    7    1
## 102    9    7    0
## 103   10    7    1
## 104   11    7    0
## 105   12    7    1
## 106   13    7    0
## 107   14    7    0
## 108   15    7    0
## 109   16    7    0
## 110   17    7    0
## 111   18    7    0
## 112   19    7    0
## 113    6    8    0
## 114    7    8    2
## 115    8    8    3
## 116    9    8    0
## 117   10    8    1
## 118   11    8    1
## 119   12    8    0
## 120   13    8    0
## 121   14    8    0
## 122   15    8    0
```

```
## 123   16    8    0
## 124   17    8    0
## 125   18    8    0
## 126   19    8    0
## 127    6    9    0
## 128    7    9    1
## 129    8    9    2
## 130    9    9    0
## 131   10    9    2
## 132   11    9    3
## 133   12    9    2
## 134   13    9    0
## 135   14    9    0
## 136   15    9    0
## 137   16    9    0
## 138   17    9    0
## 139   18    9    0
## 140   19    9    0
## 141    6   10    0
## 142    7   10    4
## 143    8   10    0
## 144    9   10    2
## 145   10   10    0
## 146   11   10    2
## 147   12   10    2
## 148   13   10    0
## 149   14   10    0
## 150   15   10    0
## 151   16   10    0
## 152   17   10    0
## 153   18   10    0
## 154   19   10    0
## 155    6   11    1
## 156    7   11    0
## 157    8   11    1
## 158    9   11    1
## 159   10   11    0
## 160   11   11    2
## 161   12   11    0
## 162   13   11    0
## 163   14   11    0
## 164   15   11    1
## 165   16   11    0
## 166   17   11    0
## 167   18   11    0
## 168   19   11    0
## 169    6   12    0
## 170    7   12    0
## 171    8   12    1
## 172    9   12    1
## 173   10   12    0
## 174   11   12    0
## 175   12   12    2
## 176   13   12    0
```

```
## 177   14   12    1
## 178   15   12    0
## 179   16   12    0
## 180   17   12    0
## 181   18   12    0
## 182   19   12    0
## 183    6   13    0
## 184    7   13    1
## 185    8   13    0
## 186    9   13    0
## 187   10   13    0
## 188   11   13    1
## 189   12   13    0
## 190   13   13    2
## 191   14   13    0
## 192   15   13    0
## 193   16   13    0
## 194   17   13    1
## 195   18   13    0
## 196   19   13    1
## 197    6   14    0
## 198    7   14    0
## 199    8   14    0
## 200    9   14    0
## 201   10   14    0
## 202   11   14    0
## 203   12   14    1
## 204   13   14    1
## 205   14   14    0
## 206   15   14    1
## 207   16   14    0
## 208   17   14    1
## 209   18   14    0
## 210   19   14    0
## 211    6   15    0
## 212    7   15    0
## 213    8   15    0
## 214    9   15    0
## 215   10   15    1
## 216   11   15    0
## 217   12   15    1
## 218   13   15    0
## 219   14   15    0
## 220   15   15    1
## 221   16   15    0
## 222   17   15    0
## 223   18   15    0
## 224   19   15    0
## 225    6   16    0
## 226    7   16    1
## 227    8   16    0
## 228    9   16    0
## 229   10   16    0
## 230   11   16    1
```

```
## 231   12   16    0
## 232   13   16    1
## 233   14   16    0
## 234   15   16    1
## 235   16   16    1
## 236   17   16    1
## 237   18   16    0
## 238   19   16    0
## 239    6   17    0
## 240    7   17    0
## 241    8   17    0
## 242    9   17    0
## 243   10   17    0
## 244   11   17    0
## 245   12   17    0
## 246   13   17    1
## 247   14   17    1
## 248   15   17    0
## 249   16   17    1
## 250   17   17    1
## 251   18   17    0
## 252   19   17    0
## 253    6   18    0
## 254    7   18    0
## 255    8   18    0
## 256    9   18    1
## 257   10   18    0
## 258   11   18    0
## 259   12   18    0
## 260   13   18    0
## 261   14   18    0
## 262   15   18    0
## 263   16   18    0
## 264   17   18    0
## 265   18   18    0
## 266   19   18    0
## 267    6   19    0
## 268    7   19    0
## 269    8   19    0
## 270    9   19    0
## 271   10   19    0
## 272   11   19    1
## 273   12   19    0
## 274   13   19    0
## 275   14   19    0
## 276   15   19    0
## 277   16   19    0
## 278   17   19    1
## 279   18   19    2
## 280   19   19    0
## 281    6   21    0
## 282    7   21    0
## 283    8   21    0
## 284    9   21    0
```

```
## 285   10   21    0
## 286   11   21    0
## 287   12   21    0
## 288   13   21    0
## 289   14   21    0
## 290   15   21    0
## 291   16   21    0
## 292   17   21    0
## 293   18   21    1
## 294   19   21    0
## 295    6   24    0
## 296    7   24    0
## 297    8   24    0
## 298    9   24    0
## 299   10   24    0
## 300   11   24    0
## 301   12   24    1
## 302   13   24    0
## 303   14   24    0
## 304   15   24    1
## 305   16   24    0
## 306   17   24    0
## 307   18   24    0
## 308   19   24    0
## 309    6   25    0
## 310    7   25    0
## 311    8   25    0
## 312    9   25    0
## 313   10   25    0
## 314   11   25    1
## 315   12   25    0
## 316   13   25    0
## 317   14   25    0
## 318   15   25    0
## 319   16   25    0
## 320   17   25    0
## 321   18   25    0
## 322   19   25    0
```

```
cb(Confusion_Matrix)

#### Actual.Total.Energy ###
#Use the following section code to copy the confusion matrix for predicted current year Actual
.Total.Energy.

#Make Test/Training Sets#
# Now make a training label set. Ensure the selected column number matches the dependent varia
ble
#(i.e. EUI=1, Actual.E_C.Energy = 3, Actual.Thermal.Energy=4, Actual.Total.Energy=5)
data.train.depvar = data[data_ind==1, 5]
data.test.depvar = data[data_ind==2, 5]
data.trainset = cbind(data.train,data.train.depvar)
data.testset = cbind(data.test,data.test.depvar)
```

```
# Regression tres
data.rfmodel <-ranger(data.train.depvar ~ ., data = data.trainset, importance = "impurity", wr
ite.forest = TRUE, classification = FALSE)
data.rfpred <- predict(data.rfmodel, data.test)
data.rfpred$predictions
```

```
##   [1] 28.81267 35.99657 32.06283 31.02375 32.40637 30.97939 32.91730
##   [8] 31.75539 30.61947 33.89863 33.33241 29.78875 37.16441 28.57666
##  [15] 29.82900 30.13306 38.06091 40.19821 27.69147 38.42008 29.83427
##  [22] 35.16202 28.31397 31.44601 38.20083 38.18882 38.04354 41.69251
##  [29] 32.46989 34.31977 31.34744 29.82940 28.39337 30.39846 27.20424
##  [36] 30.70231 30.08887 31.80357 35.46599 28.75486 31.33925 31.25092
##  [43] 27.51403 31.69059 25.20078 27.53204 35.32045 35.09841 31.54707
##  [50] 35.47606 22.60215 30.80723 26.81744 27.49106 28.06430 27.64593
##  [57] 29.87871 31.76092 35.61338 33.86203 26.21546 26.40540 28.81332
##  [64] 32.08103 33.88483 25.00216 27.67866 32.32863 27.59087 38.05089
##  [71] 30.24229 28.19394 28.90496 28.60928 29.37273 31.04209 28.14215
##  [78] 30.03041 34.02926 31.99446 30.51478 29.58051 32.69438 29.95389
##  [85] 25.12859 28.45844 29.30617 29.93579 28.28508 29.90456 31.89856
##  [92] 30.09036 25.35688 22.34979 29.41134 29.05072 34.21063 29.88324
##  [99] 28.35858 30.64490 25.94097 26.81746 27.81961
```

```
Confusion_Matrix <- table(mround(data.rfpred$predictions,1),mround(data.test.depvar,1))
Confusion_Matrix
```

```
##
##      17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
##   22  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   23  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
##   25  0  0  0  0  1  0  0  0  1  0  1  0  0  1  0  0  0  0  0  0  0  0  0
##   26  0  0  0  0  0  0  0  0  1  0  0  0  1  0  1  0  0  0  0  0  0  0  0
##   27  1  0  0  0  0  1  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0
##   28  0  1  0  1  1  1  0  2  1  2  0  0  2  0  0  0  1  1  0  1  0  0  0
##   29  0  0  0  0  3  0  1  0  1  0  0  1  0  0  1  0  0  1  0  0  0  0  0
##   30  0  0  1  0  1  0  2  2  0  1  1  0  3  1  2  0  0  0  1  0  0  0  0
##   31  0  0  0  1  0  0  1  0  1  1  0  0  1  0  1  1  0  1  1  1  0  0  0
##   32  0  0  1  0  0  0  0  2  0  0  2  1  1  0  2  1  0  0  0  0  0  0  0
##   33  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
##   34  0  0  0  0  0  0  0  0  0  2  1  0  1  0  0  0  0  0  0  0  1  0  1
##   35  0  0  0  0  0  0  0  1  0  1  0  0  0  1  0  0  0  0  1  0  1  0  0
##   36  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0  0  0
##   37  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   38  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  1  0  0  1  1  0  1  0
##   40  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   42  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
##
##      40 41 42 43 47 51 52 63 80
##   22  0  0  0  0  0  0  0  0  0
##   23  0  0  0  0  0  0  0  0  0
##   25  0  0  0  0  0  0  0  0  0
##   26  0  0  0  0  0  0  0  0  0
```

```
##   27  0  0  0  0  0  0  0  0  0
##   28  0  0  0  0  1  0  0  0  0
##   29  0  0  1  0  0  0  0  1  0
##   30  0  0  0  0  0  0  1  0  0
##   31  0  0  0  0  1  0  0  0  1
##   32  0  1  0  1  0  0  0  0  0
##   33  0  0  0  0  0  1  0  0  0
##   34  0  0  0  0  0  0  0  0  0
##   35  0  0  0  0  0  0  0  0  0
##   36  0  0  0  0  0  0  0  0  0
##   37  1  0  0  0  0  0  0  0  0
##   38  0  0  0  0  0  0  0  0  0
##   40  0  1  0  0  0  0  0  0  0
##   42  0  0  0  0  0  0  0  0  0
```

```
Accuracy <- sum(diag(Confusion_Matrix))/sum(Confusion_Matrix)
Accuracy
```

```
## [1] 0.05825243
```

```
Confusion_Matrix = as.data.frame(Confusion_Matrix)
Confusion_Matrix
```

```
##     Var1 Var2 Freq
## 1     22   17    0
## 2     23   17    0
## 3     25   17    0
## 4     26   17    0
## 5     27   17    1
## 6     28   17    0
## 7     29   17    0
## 8     30   17    0
## 9     31   17    0
## 10    32   17    0
## 11    33   17    0
## 12    34   17    0
## 13    35   17    0
## 14    36   17    0
## 15    37   17    0
## 16    38   17    0
## 17    40   17    0
## 18    42   17    0
## 19    22   18    0
## 20    23   18    0
## 21    25   18    0
## 22    26   18    0
## 23    27   18    0
## 24    28   18    1
## 25    29   18    0
## 26    30   18    0
## 27    31   18    0
```

```
## 28    32   18    0
## 29    33   18    0
## 30    34   18    0
## 31    35   18    0
## 32    36   18    0
## 33    37   18    0
## 34    38   18    0
## 35    40   18    0
## 36    42   18    0
## 37    22   19    0
## 38    23   19    0
## 39    25   19    0
## 40    26   19    0
## 41    27   19    0
## 42    28   19    0
## 43    29   19    0
## 44    30   19    1
## 45    31   19    0
## 46    32   19    1
## 47    33   19    0
## 48    34   19    0
## 49    35   19    0
## 50    36   19    0
## 51    37   19    0
## 52    38   19    0
## 53    40   19    0
## 54    42   19    0
## 55    22   20    1
## 56    23   20    0
## 57    25   20    0
## 58    26   20    0
## 59    27   20    0
## 60    28   20    1
## 61    29   20    0
## 62    30   20    0
## 63    31   20    1
## 64    32   20    0
## 65    33   20    0
## 66    34   20    0
## 67    35   20    0
## 68    36   20    0
## 69    37   20    0
## 70    38   20    0
## 71    40   20    0
## 72    42   20    0
## 73    22   21    0
## 74    23   21    0
## 75    25   21    1
## 76    26   21    0
## 77    27   21    0
## 78    28   21    1
## 79    29   21    3
## 80    30   21    1
## 81    31   21    0
```

```
## 82    32   21    0
## 83    33   21    1
## 84    34   21    0
## 85    35   21    0
## 86    36   21    0
## 87    37   21    0
## 88    38   21    0
## 89    40   21    0
## 90    42   21    0
## 91    22   22    0
## 92    23   22    0
## 93    25   22    0
## 94    26   22    0
## 95    27   22    1
## 96    28   22    1
## 97    29   22    0
## 98    30   22    0
## 99    31   22    0
## 100   32   22    0
## 101   33   22    0
## 102   34   22    0
## 103   35   22    0
## 104   36   22    0
## 105   37   22    0
## 106   38   22    0
## 107   40   22    0
## 108   42   22    0
## 109   22   23    0
## 110   23   23    0
## 111   25   23    0
## 112   26   23    0
## 113   27   23    0
## 114   28   23    0
## 115   29   23    1
## 116   30   23    2
## 117   31   23    1
## 118   32   23    0
## 119   33   23    0
## 120   34   23    0
## 121   35   23    0
## 122   36   23    0
## 123   37   23    0
## 124   38   23    0
## 125   40   23    0
## 126   42   23    0
## 127   22   24    0
## 128   23   24    0
## 129   25   24    0
## 130   26   24    0
## 131   27   24    0
## 132   28   24    2
## 133   29   24    0
## 134   30   24    2
## 135   31   24    0
```

```
## 136   32   24    2
## 137   33   24    0
## 138   34   24    0
## 139   35   24    1
## 140   36   24    0
## 141   37   24    0
## 142   38   24    0
## 143   40   24    0
## 144   42   24    0
## 145   22   25    0
## 146   23   25    0
## 147   25   25    1
## 148   26   25    1
## 149   27   25    0
## 150   28   25    1
## 151   29   25    1
## 152   30   25    0
## 153   31   25    1
## 154   32   25    0
## 155   33   25    0
## 156   34   25    0
## 157   35   25    0
## 158   36   25    0
## 159   37   25    0
## 160   38   25    0
## 161   40   25    0
## 162   42   25    0
## 163   22   26    0
## 164   23   26    0
## 165   25   26    0
## 166   26   26    0
## 167   27   26    0
## 168   28   26    2
## 169   29   26    0
## 170   30   26    1
## 171   31   26    1
## 172   32   26    0
## 173   33   26    0
## 174   34   26    2
## 175   35   26    1
## 176   36   26    0
## 177   37   26    0
## 178   38   26    0
## 179   40   26    0
## 180   42   26    0
## 181   22   27    0
## 182   23   27    1
## 183   25   27    1
## 184   26   27    0
## 185   27   27    0
## 186   28   27    0
## 187   29   27    0
## 188   30   27    1
## 189   31   27    0
```

```
## 190   32   27    2
## 191   33   27    0
## 192   34   27    1
## 193   35   27    0
## 194   36   27    0
## 195   37   27    0
## 196   38   27    0
## 197   40   27    0
## 198   42   27    0
## 199   22   28    0
## 200   23   28    0
## 201   25   28    0
## 202   26   28    0
## 203   27   28    1
## 204   28   28    0
## 205   29   28    1
## 206   30   28    0
## 207   31   28    0
## 208   32   28    1
## 209   33   28    0
## 210   34   28    0
## 211   35   28    0
## 212   36   28    0
## 213   37   28    0
## 214   38   28    0
## 215   40   28    0
## 216   42   28    0
## 217   22   29    0
## 218   23   29    0
## 219   25   29    0
## 220   26   29    1
## 221   27   29    1
## 222   28   29    2
## 223   29   29    0
## 224   30   29    3
## 225   31   29    1
## 226   32   29    1
## 227   33   29    0
## 228   34   29    1
## 229   35   29    0
## 230   36   29    1
## 231   37   29    0
## 232   38   29    1
## 233   40   29    0
## 234   42   29    0
## 235   22   30    0
## 236   23   30    0
## 237   25   30    1
## 238   26   30    0
## 239   27   30    0
## 240   28   30    0
## 241   29   30    0
## 242   30   30    1
## 243   31   30    0
```

```
## 244   32   30    0
## 245   33   30    0
## 246   34   30    0
## 247   35   30    1
## 248   36   30    0
## 249   37   30    0
## 250   38   30    0
## 251   40   30    0
## 252   42   30    0
## 253   22   31    0
## 254   23   31    0
## 255   25   31    0
## 256   26   31    1
## 257   27   31    0
## 258   28   31    0
## 259   29   31    1
## 260   30   31    2
## 261   31   31    1
## 262   32   31    2
## 263   33   31    0
## 264   34   31    0
## 265   35   31    0
## 266   36   31    0
## 267   37   31    0
## 268   38   31    1
## 269   40   31    0
## 270   42   31    0
## 271   22   32    0
## 272   23   32    0
## 273   25   32    0
## 274   26   32    0
## 275   27   32    0
## 276   28   32    0
## 277   29   32    0
## 278   30   32    0
## 279   31   32    1
## 280   32   32    1
## 281   33   32    0
## 282   34   32    0
## 283   35   32    0
## 284   36   32    0
## 285   37   32    0
## 286   38   32    1
## 287   40   32    0
## 288   42   32    0
## 289   22   33    0
## 290   23   33    0
## 291   25   33    0
## 292   26   33    0
## 293   27   33    0
## 294   28   33    1
## 295   29   33    0
## 296   30   33    0
## 297   31   33    0
```

```
## 298   32   33    0
## 299   33   33    0
## 300   34   33    0
## 301   35   33    0
## 302   36   33    0
## 303   37   33    0
## 304   38   33    0
## 305   40   33    0
## 306   42   33    0
## 307   22   34    0
## 308   23   34    0
## 309   25   34    0
## 310   26   34    0
## 311   27   34    0
## 312   28   34    1
## 313   29   34    1
## 314   30   34    0
## 315   31   34    1
## 316   32   34    0
## 317   33   34    0
## 318   34   34    0
## 319   35   34    0
## 320   36   34    0
## 321   37   34    0
## 322   38   34    0
## 323   40   34    0
## 324   42   34    0
## 325   22   35    0
## 326   23   35    0
## 327   25   35    0
## 328   26   35    0
## 329   27   35    0
## 330   28   35    0
## 331   29   35    0
## 332   30   35    1
## 333   31   35    1
## 334   32   35    0
## 335   33   35    0
## 336   34   35    0
## 337   35   35    1
## 338   36   35    0
## 339   37   35    0
## 340   38   35    1
## 341   40   35    0
## 342   42   35    0
## 343   22   36    0
## 344   23   36    0
## 345   25   36    0
## 346   26   36    0
## 347   27   36    0
## 348   28   36    1
## 349   29   36    0
## 350   30   36    0
## 351   31   36    1
```

```
## 352   32   36    0
## 353   33   36    0
## 354   34   36    0
## 355   35   36    0
## 356   36   36    1
## 357   37   36    0
## 358   38   36    1
## 359   40   36    0
## 360   42   36    1
## 361   22   37    0
## 362   23   37    0
## 363   25   37    0
## 364   26   37    0
## 365   27   37    0
## 366   28   37    0
## 367   29   37    0
## 368   30   37    0
## 369   31   37    0
## 370   32   37    0
## 371   33   37    1
## 372   34   37    1
## 373   35   37    1
## 374   36   37    0
## 375   37   37    0
## 376   38   37    0
## 377   40   37    0
## 378   42   37    0
## 379   22   38    0
## 380   23   38    0
## 381   25   38    0
## 382   26   38    0
## 383   27   38    0
## 384   28   38    0
## 385   29   38    0
## 386   30   38    0
## 387   31   38    0
## 388   32   38    0
## 389   33   38    0
## 390   34   38    0
## 391   35   38    0
## 392   36   38    0
## 393   37   38    0
## 394   38   38    1
## 395   40   38    0
## 396   42   38    0
## 397   22   39    0
## 398   23   39    0
## 399   25   39    0
## 400   26   39    0
## 401   27   39    0
## 402   28   39    0
## 403   29   39    0
## 404   30   39    0
## 405   31   39    0
```

```
## 406   32   39    0
## 407   33   39    0
## 408   34   39    1
## 409   35   39    0
## 410   36   39    0
## 411   37   39    0
## 412   38   39    0
## 413   40   39    0
## 414   42   39    0
## 415   22   40    0
## 416   23   40    0
## 417   25   40    0
## 418   26   40    0
## 419   27   40    0
## 420   28   40    0
## 421   29   40    0
## 422   30   40    0
## 423   31   40    0
## 424   32   40    0
## 425   33   40    0
## 426   34   40    0
## 427   35   40    0
## 428   36   40    0
## 429   37   40    1
## 430   38   40    0
## 431   40   40    0
## 432   42   40    0
## 433   22   41    0
## 434   23   41    0
## 435   25   41    0
## 436   26   41    0
## 437   27   41    0
## 438   28   41    0
## 439   29   41    0
## 440   30   41    0
## 441   31   41    0
## 442   32   41    1
## 443   33   41    0
## 444   34   41    0
## 445   35   41    0
## 446   36   41    0
## 447   37   41    0
## 448   38   41    0
## 449   40   41    1
## 450   42   41    0
## 451   22   42    0
## 452   23   42    0
## 453   25   42    0
## 454   26   42    0
## 455   27   42    0
## 456   28   42    0
## 457   29   42    1
## 458   30   42    0
## 459   31   42    0
```

```
## 460   32   42    0
## 461   33   42    0
## 462   34   42    0
## 463   35   42    0
## 464   36   42    0
## 465   37   42    0
## 466   38   42    0
## 467   40   42    0
## 468   42   42    0
## 469   22   43    0
## 470   23   43    0
## 471   25   43    0
## 472   26   43    0
## 473   27   43    0
## 474   28   43    0
## 475   29   43    0
## 476   30   43    0
## 477   31   43    0
## 478   32   43    1
## 479   33   43    0
## 480   34   43    0
## 481   35   43    0
## 482   36   43    0
## 483   37   43    0
## 484   38   43    0
## 485   40   43    0
## 486   42   43    0
## 487   22   47    0
## 488   23   47    0
## 489   25   47    0
## 490   26   47    0
## 491   27   47    0
## 492   28   47    1
## 493   29   47    0
## 494   30   47    0
## 495   31   47    1
## 496   32   47    0
## 497   33   47    0
## 498   34   47    0
## 499   35   47    0
## 500   36   47    0
## 501   37   47    0
## 502   38   47    0
## 503   40   47    0
## 504   42   47    0
## 505   22   51    0
## 506   23   51    0
## 507   25   51    0
## 508   26   51    0
## 509   27   51    0
## 510   28   51    0
## 511   29   51    0
## 512   30   51    0
## 513   31   51    0
```

```
## 514   32   51    0
## 515   33   51    1
## 516   34   51    0
## 517   35   51    0
## 518   36   51    0
## 519   37   51    0
## 520   38   51    0
## 521   40   51    0
## 522   42   51    0
## 523   22   52    0
## 524   23   52    0
## 525   25   52    0
## 526   26   52    0
## 527   27   52    0
## 528   28   52    0
## 529   29   52    0
## 530   30   52    1
## 531   31   52    0
## 532   32   52    0
## 533   33   52    0
## 534   34   52    0
## 535   35   52    0
## 536   36   52    0
## 537   37   52    0
## 538   38   52    0
## 539   40   52    0
## 540   42   52    0
## 541   22   63    0
## 542   23   63    0
## 543   25   63    0
## 544   26   63    0
## 545   27   63    0
## 546   28   63    0
## 547   29   63    1
## 548   30   63    0
## 549   31   63    0
## 550   32   63    0
## 551   33   63    0
## 552   34   63    0
## 553   35   63    0
## 554   36   63    0
## 555   37   63    0
## 556   38   63    0
## 557   40   63    0
## 558   42   63    0
## 559   22   80    0
## 560   23   80    0
## 561   25   80    0
## 562   26   80    0
## 563   27   80    0
## 564   28   80    0
## 565   29   80    0
## 566   30   80    0
## 567   31   80    1
```

```
## 568   32   80    0
## 569   33   80    0
## 570   34   80    0
## 571   35   80    0
## 572   36   80    0
## 573   37   80    0
## 574   38   80    0
## 575   40   80    0
## 576   42   80    0
```

```
cb(Confusion_Matrix)
```

Final_Random_Forest_Script_-_Future_Years.R

carle

Wed Oct 10 22:50:37 2018

```
########################################
####        Decision Trees Script     ####
########################################

####Load Packages####
library(ranger)
```

```
## Warning: package 'ranger' was built under R version 3.4.3
```

```
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.4.3
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.4.3
```

```
####Load and PreProcess Data ####
set.seed(1234)
setwd("C:/Users/carle/Documents/Data/2010-2015 Data")
getwd()
```

```
## [1] "C:/Users/carle/Documents/Data/2010-2015 Data"
```

```
# na.strings = "NA" populates every blank with "NA", which R will ignore
# stringsAsFactors = TRUE will let you read in 0's as real zeroes
# sep = "," is used for csv files
# check.names = FALSE permits multiple words in Column names to be seperated by a space.
# ^ [R] prefers names to read as one word (syntactically valid): "a.b.c" rather than "a b c"

# Load Data - Script is prepared for the 2010|2015 Dataset
data <- read.csv("2010-2015 Cleaned Data.CSV", row.names = 1, check.names = FALSE, na.strings
= "NA", sep = ",", stringsAsFactors = TRUE, header = TRUE)
#Rename Columns to have 'syntactically valid' names
cnames <- cbind("x2010.EUI","x2015.EUI","D.EUI","x2010.Qualitative.EUI","x2015.Qualitative.EUI
",
                "D.Qualitative.EUI",    "x2010.Actual.E_C.Energy",  "x2015.Actual.E_C", "D.Act
ual.E_C", "x2010.Actual.Thermal",
```

```
                "x2015.Actual.Thermal", "D.Actual.Thermal", "x2010.Actual.Energy.Use",  "D.Act
ual.Energy.Use",
                "x2015.Actual.Energy.Use",  "Portfolio.Manager.Q",  "Portfolio.Manager",    "N
et.Rentable.Area",    "Exterior.Area",
                "Gross.Floor.Area",     "Enclosed.Parking",     "Latitude", "Longitude",    "C
onstruction.Year",    "No.of.Structures",
                "Building.Class",   "Closest.Major.City",   "Climate.Zone", "Electrically.Heat
ed",  "Cooling.Tower",    "Soft.Landscaping",
                "Occupant.Density", "Vacancy.Rate", "Weekly.Operating.Hours")

colnames(data) <- cnames
#If Struggling with having syntactically valid names, use the follow function to find your err
or
#is_valid_name<- function(x)
#{length_condition = if(getRversion() < "2.13.0") 256L else 10000L
#  is_short_enough = nchar(x) <= length_condition
#  is_valid_name = (make.names(x) == x)
#  final_condition = is_short_enough && is_valid_name
#  return(final_condition)}
#sapply(cnames, is_valid_name)

# Order the factor variable Qualitative.EUI
data[,"x2010.Qualitative.EUI"]=ordered(data[,"x2010.Qualitative.EUI"], levels = c("Poor","Fair
","Good"))
data[,"x2015.Qualitative.EUI"]=ordered(data[,"x2015.Qualitative.EUI"], levels = c("Poor","Fair
","Good"))
data[,"D.Qualitative.EUI"]=ordered(data[,"D.Qualitative.EUI"], levels = c("Poor","Fair","Good"
))
data[,"D.EUI"]=as.numeric(data[,"D.Qualitative.EUI"])
# Sanity Check
is.ordered(data[,"x2010.Qualitative.EUI"]) #<- Should Say True
```

```
## [1] TRUE
```

```
is.ordered(data[,"x2015.Qualitative.EUI"]) #<- Should Say True
```

```
## [1] TRUE
```

```
is.ordered(data[,"D.Qualitative.EUI"]) #<- Should Say True
```

```
## [1] TRUE
```

```
#str(data) #<- If is.ordered=False, then look at the structure for issues
#sapply(data, class) #<- All variables should be classified as numeric or intergers (only Qual
itative EUi should be an Ordered Factor)

#Remove rows with missing data
data<-data[complete.cases(data), ]
#check.is.na=is.na(data) to confirm that there are no missing data
#View(check.is.na)
```

```
#Functions
cb <- function(df, sep="\t", dec=",", max.size=(200*1000)){
  # Copy a data.frame to clipboard
  write.table(df, paste0("clipboard-", formatC(max.size, format="f", digits=0)), sep=sep, row.
names=FALSE, dec=dec)
}
#Function to round value to the nearest increment of x (i.e. 5)
mround <- function(x,base){
  base*round(x/base)
}


#### Define Model Varibles ####
#Select Independant variables
selected.indep.variables <- c("x2010.Qualitative.EUI","x2010.Actual.E_C.Energy",    "x2010.Act
ual.Thermal",
                              "x2010.Actual.Energy.Use","Portfolio.Manager", "Exterior.Area",
                              "Latitude", "Longitude",   "Construction.Year",    "No.of.Structu
res",
                               "Building.Class", "Climate.Zone", "Electrically.Heated",  "Cooli
ng.Tower",
                                "Soft.Landscaping", "Occupant.Density",  "Vacancy.Rate", "Weekl
y.Operating.Hours")
# Edit to include chosen dependant variable: "x2015.EUI","D.EUI","x2015.Qualitative.EUI",
#                 "D.Qualitative.EUI",   "x2015.Actual.E_C", "D.Actual.E_C", "x2015.Actual.Ther
mal", "D.Actual.Thermal",
#                  "D.Actual.Energy.Use", "x2015.Actual.Energy.Use"

#### Make Test/Training Sets####

# assign rows to either the training or test sets
data_ind <- sample(2, nrow(data), replace=TRUE, prob=c(0.67, 0.33))
# move all samples for the rows associated with ind=1 to the training set
data.train <- data[data_ind==1,selected.indep.variables]
# move all samples for the rows associated with ind=2 to the test set
data.test <- data[data_ind==2,selected.indep.variables]

####Classification Tree (2015 Q.EUI)####

# now make a training label set (i.e. "x2015.Qualitative.EUI"=5, "D.Qualitative.EUI"=6)
data.train.Labels = data[data_ind==1, 5]
data.test.Labels = data[data_ind==2, 5]
data.trainset = cbind(data.train,data.train.Labels)
data.testset = cbind(data.test,data.test.Labels)
####Classification Tree (Change importance = none or impurity to see if accuraccy improves)
data.rfmodel <-ranger(data.train.Labels ~ ., data = data.trainset, min.node.size=1, importance
 = "impurity", write.forest = TRUE, classification = TRUE)
data.rfpred <- predict(data.rfmodel, data.test)
data.rfpred$predictions
```

```
##  [1] Fair Fair Fair Fair Poor Fair Fair Fair Poor Poor Good Fair Good Fair
## [15] Fair Fair Good Fair Poor Fair Fair Fair Good Fair Fair Fair Fair Fair
```

```
## [29] Fair Good Fair Fair Fair Good Fair Fair Fair Fair Fair Fair Fair Fair
## [43] Fair Fair Fair Fair Fair Fair Fair Fair Good Fair Fair Fair Fair Fair
## [57] Fair
## Levels: Poor Fair Good
```

```
Confusion_Matrix <- table(data.rfpred$predictions,data.test.Labels)
Confusion_Matrix
```

```
##       data.test.Labels
##        Poor Fair Good
##   Poor    4    0    0
##   Fair    4   34    8
##   Good    0    2    5
```

```
Accuracy <- sum(diag(Confusion_Matrix))/sum(Confusion_Matrix)
Accuracy
```

```
## [1] 0.754386
```

```
#Export Results
Confusion_Matrix = as.data.frame(Confusion_Matrix)
Confusion_Matrix
```

```
##   Var1 data.test.Labels Freq
## 1 Poor             Poor    4
## 2 Fair             Poor    4
## 3 Good             Poor    0
## 4 Poor             Fair    0
## 5 Fair             Fair   34
## 6 Good             Fair    2
## 7 Poor             Good    0
## 8 Fair             Good    8
## 9 Good             Good    5
```

```
cb(Confusion_Matrix)

####Classification Tree (D.Q.EUI)####
selected.indep.variables <- c("x2010.Qualitative.EUI","x2010.Actual.E_C.Energy",    "x2010.Actual.Thermal",
                              "x2010.Actual.Energy.Use","Portfolio.Manager",    "Exterior.Area",
                              "Latitude", "Longitude",  "Construction.Year",    "No.of.Structures",
                              "Building.Class", "Climate.Zone", "Electrically.Heated",  "Cooling.Tower",    "Soft.Landscaping",
                              "Occupant.Density",   "Vacancy.Rate", "Weekly.Operating.Hours")
# now make a training label set (i.e. "x2015.Qualitative.EUI"=5, "D.Qualitative.EUI"=6)
data.train.Labels = data[data_ind==1, 6]
data.test.Labels = data[data_ind==2, 6]
```

```
data.trainset = cbind(data.train,data.train.Labels)
data.testset = cbind(data.test,data.test.Labels)
####Classification Tree (Change importance = none or impurity to see if accuraccy improves)
data.rfmodel <-ranger(data.train.Labels ~ ., data = data.trainset, min.node.size=1, importance
 = "impurity", write.forest = TRUE, classification = TRUE)
data.rfpred <- predict(data.rfmodel, data.test)
data.rfpred$predictions
```

```
##  [1] Fair Fair Poor Poor Poor Fair Fair Fair Poor Poor Fair Good Fair Fair
## [15] Good Good Poor Fair Poor Fair Fair Good Poor Good Good Good Fair Fair
## [29] Fair Fair Fair Good Good Fair Fair Fair Fair Fair Fair Fair Fair Good
## [43] Good Fair Fair Fair Fair Good Good Fair Fair Good Fair Fair Fair Good
## [57] Good
## Levels: Poor Fair Good
```

```
Confusion_Matrix <- table(data.rfpred$predictions,data.test.Labels)
Confusion_Matrix
```

```
##       data.test.Labels
##        Poor Fair Good
##   Poor    3    4    1
##   Fair   10   17    6
##   Good    3    8    5
```

```
Accuracy <- sum(diag(Confusion_Matrix))/sum(Confusion_Matrix)
Accuracy
```

```
## [1] 0.4385965
```

```
#Export Results
Confusion_Matrix = as.data.frame(Confusion_Matrix)
Confusion_Matrix
```

```
##   Var1 data.test.Labels Freq
## 1 Poor             Poor    3
## 2 Fair             Poor   10
## 3 Good             Poor    3
## 4 Poor             Fair    4
## 5 Fair             Fair   17
## 6 Good             Fair    8
## 7 Poor             Good    1
## 8 Fair             Good    6
## 9 Good             Good    5
```

```
cb(Confusion_Matrix)

####Preprocess for regression####
#Select Independant variables
selected.indep.variables <- c("x2010.EUI","x2010.Actual.E C.Energy",    "x2010.Actual.Thermal"
```

```
,
                              "x2010.Actual.Energy.Use","Portfolio.Manager",    "Exterior.Area
",
                              "Latitude", "Longitude",  "Construction.Year",    "No.of.Structu
res",
                              "Building.Class", "Climate.Zone", "Electrically.Heated",  "Cooli
ng.Tower",    "Soft.Landscaping",
                              "Occupant.Density",   "Vacancy.Rate", "Weekly.Operating.Hours")
####For Regression-Run EUI code first as there are non-repeated formulas located in this secti
on####

####2015 EUI####

#Use the following section code to copy the confusion matrix for predicted current year EUI.

# Make Test/Training Sets#
# Now make a training label set. Ensure the selected column number matches the dependent varia
ble
#(i.e. x2015.EUI)
data.train.depvar = data[data_ind==1, 2]
data.test.depvar = data[data_ind==2, 2]
data.trainset = cbind(data.train,data.train.depvar)
data.testset = cbind(data.test,data.test.depvar)

# Regression trees
data.rfmodel <-ranger(data.train.depvar ~ ., data = data.trainset, importance = "impurity", wr
ite.forest = TRUE, classification = FALSE)
data.rfpred <- predict(data.rfmodel, data.test)
data.rfpred$predictions
```

```
##  [1] 23.09007 24.14254 38.05852 35.00124 54.12392 21.89512 27.44368
##  [8] 25.54282 42.78186 45.18488 16.18329 23.85777 19.69916 22.08732
## [15] 22.45044 23.38906 19.48640 22.71929 45.05920 33.31861 21.63641
## [22] 21.90052 19.71250 24.19366 21.89195 22.04493 24.91638 23.51457
## [29] 26.98526 20.94915 22.77064 23.15676 28.24378 31.38078 21.24082
## [36] 21.48536 23.28172 27.81736 28.05655 24.87994 21.00599 23.84142
## [43] 22.10939 23.70894 26.51065 26.79781 25.65764 28.20009 33.69907
## [50] 28.38055 24.34144 32.27726 24.24427 28.46331 30.74119 36.30644
## [57] 21.37193
```

```
Confusion_Matrix <- table(mround(data.rfpred$predictions,1),mround(data.test.depvar,1))
Confusion_Matrix
```

```
##
##      12 14 15 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 33 34 40 41 45
##   16  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   19  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   20  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   21  0  0  0  0  0  0  0  1  3  0  0  1  0  0  0  0  0  0  0  0  0  0  0
##   22  0  0  0  0  1  2  1  1  1  1  0  1  0  0  0  0  0  0  0  0  0  0  0
##   23  0  0  0  0  0  1  0  0  1  1  2  0  1  0  0  0  0  0  0  0  0  0  0
##   24  0  1  0  1  1  0  1  0  2  0  0  0  1  0  0  0  0  0  0  1  0  0  0
```

```
##    25  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0
##    26  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
##    27  0  0  0  0  0  0  0  0  1  0  0  0  2  0  0  0  0  0  1  0  0  0  0
##    28  0  0  0  0  0  0  1  1  0  0  0  0  0  1  1  0  0  1  0  1  0  0  0
##    31  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0
##    32  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
##    33  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##    34  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
##    35  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
##    36  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
##    38  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##    43  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
##    45  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##    54  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##
##       52 53 57 59 60
##    16  0  0  0  0  0
##    19  0  0  0  0  0
##    20  0  0  0  0  0
##    21  0  0  0  0  0
##    22  0  0  0  0  0
##    23  0  0  0  0  0
##    24  0  0  0  0  0
##    25  0  0  0  0  0
##    26  0  0  0  0  0
##    27  0  0  0  0  0
##    28  0  0  0  0  0
##    31  0  0  0  0  0
##    32  0  0  0  0  0
##    33  0  1  0  0  0
##    34  0  0  0  0  0
##    35  0  0  0  0  0
##    36  0  0  0  0  0
##    38  0  0  0  0  1
##    43  0  0  0  0  0
##    45  1  0  1  0  0
##    54  0  0  0  1  0
```

```
Accuracy <- sum(diag(Confusion_Matrix))/sum(Confusion_Matrix)
Accuracy
```

```
## [1] 0.0877193
```

```
Confusion_Matrix = as.data.frame(Confusion_Matrix)
Confusion_Matrix
```

```
##     Var1 Var2 Freq
## 1     16   12    1
## 2     19   12    0
## 3     20   12    0
## 4     21   12    0
```

```
## 5     22   12    0
## 6     23   12    0
## 7     24   12    0
## 8     25   12    0
## 9     26   12    0
## 10    27   12    0
## 11    28   12    0
## 12    31   12    0
## 13    32   12    0
## 14    33   12    0
## 15    34   12    0
## 16    35   12    0
## 17    36   12    0
## 18    38   12    0
## 19    43   12    0
## 20    45   12    0
## 21    54   12    0
## 22    16   14    0
## 23    19   14    0
## 24    20   14    0
## 25    21   14    0
## 26    22   14    0
## 27    23   14    0
## 28    24   14    1
## 29    25   14    0
## 30    26   14    0
## 31    27   14    0
## 32    28   14    0
## 33    31   14    0
## 34    32   14    0
## 35    33   14    0
## 36    34   14    0
## 37    35   14    0
## 38    36   14    0
## 39    38   14    0
## 40    43   14    0
## 41    45   14    0
## 42    54   14    0
## 43    16   15    0
## 44    19   15    1
## 45    20   15    1
## 46    21   15    0
## 47    22   15    0
## 48    23   15    0
## 49    24   15    0
## 50    25   15    0
## 51    26   15    0
## 52    27   15    0
## 53    28   15    0
## 54    31   15    0
## 55    32   15    0
## 56    33   15    0
## 57    34   15    0
## 58    35   15    0
## 59    36   15    0
## 60    38   15    0
## 61    43   15    0
## 62    45   15    0
## 63    54   15    0
## 64    16   17    0
## 65    19   17    0
## 66    20   17    0
## 67    21   17    0
## 68    22   17    0
## 69    23   17    0
## 70    24   17    1
## 71    25   17    0
## 72    26   17    0
## 73    27   17    0
## 74    28   17    0
## 75    31   17    0
## 76    32   17    0
## 77    33   17    0
## 78    34   17    0
## 79    35   17    0
## 80    36   17    0
## 81    38   17    0
## 82    43   17    0
## 83    45   17    0
## 84    54   17    0
## 85    16   18    0
## 86    19   18    0
## 87    20   18    1
## 88    21   18    0
## 89    22   18    1
## 90    23   18    0
## 91    24   18    1
## 92    25   18    0
## 93    26   18    0
## 94    27   18    0
## 95    28   18    0
## 96    31   18    0
## 97    32   18    0
## 98    33   18    0
## 99    34   18    0
## 100   35   18    0
## 101   36   18    0
## 102   38   18    0
## 103   43   18    0
## 104   45   18    0
## 105   54   18    0
## 106   16   19    0
## 107   19   19    0
## 108   20   19    0
## 109   21   19    0
## 110   22   19    2
## 111   23   19    1
## 112   24   19    0
```

```
## 113   25   19    0 | ## 167   45   21    0 | ## 221   28   24    0 | ## 275   19   27    0
## 114   26   19    0 | ## 168   54   21    0 | ## 222   31   24    0 | ## 276   20   27    0
## 115   27   19    0 | ## 169   16   22    0 | ## 223   32   24    0 | ## 277   21   27    0
## 116   28   19    0 | ## 170   19   22    0 | ## 224   33   24    0 | ## 278   22   27    0
## 117   31   19    0 | ## 171   20   22    0 | ## 225   34   24    0 | ## 279   23   27    0
## 118   32   19    0 | ## 172   21   22    3 | ## 226   35   24    0 | ## 280   24   27    0
## 119   33   19    0 | ## 173   22   22    1 | ## 227   36   24    0 | ## 281   25   27    0
## 120   34   19    0 | ## 174   23   22    1 | ## 228   38   24    0 | ## 282   26   27    0
## 121   35   19    0 | ## 175   24   22    2 | ## 229   43   24    0 | ## 283   27   27    0
## 122   36   19    0 | ## 176   25   22    1 | ## 230   45   24    0 | ## 284   28   27    1
## 123   38   19    0 | ## 177   26   22    0 | ## 231   54   24    0 | ## 285   31   27    0
## 124   43   19    0 | ## 178   27   22    1 | ## 232   16   25    0 | ## 286   32   27    0
## 125   45   19    0 | ## 179   28   22    0 | ## 233   19   25    0 | ## 287   33   27    0
## 126   54   19    0 | ## 180   31   22    0 | ## 234   20   25    0 | ## 288   34   27    0
## 127   16   20    0 | ## 181   32   22    0 | ## 235   21   25    1 | ## 289   35   27    0
## 128   19   20    0 | ## 182   33   22    0 | ## 236   22   25    1 | ## 290   36   27    0
## 129   20   20    0 | ## 183   34   22    0 | ## 237   23   25    0 | ## 291   38   27    0
## 130   21   20    0 | ## 184   35   22    0 | ## 238   24   25    0 | ## 292   43   27    0
## 131   22   20    1 | ## 185   36   22    0 | ## 239   25   25    0 | ## 293   45   27    0
## 132   23   20    0 | ## 186   38   22    0 | ## 240   26   25    0 | ## 294   54   27    0
## 133   24   20    1 | ## 187   43   22    0 | ## 241   27   25    0 | ## 295   16   28    0
## 134   25   20    0 | ## 188   45   22    0 | ## 242   28   25    0 | ## 296   19   28    0
## 135   26   20    1 | ## 189   54   22    0 | ## 243   31   25    0 | ## 297   20   28    0
## 136   27   20    0 | ## 190   16   23    0 | ## 244   32   25    0 | ## 298   21   28    0
## 137   28   20    1 | ## 191   19   23    0 | ## 245   33   25    0 | ## 299   22   28    0
## 138   31   20    0 | ## 192   20   23    0 | ## 246   34   25    0 | ## 300   23   28    0
## 139   32   20    0 | ## 193   21   23    0 | ## 247   35   25    0 | ## 301   24   28    0
## 140   33   20    0 | ## 194   22   23    1 | ## 248   36   25    0 | ## 302   25   28    0
## 141   34   20    0 | ## 195   23   23    1 | ## 249   38   25    0 | ## 303   26   28    0
## 142   35   20    0 | ## 196   24   23    0 | ## 250   43   25    0 | ## 304   27   28    0
## 143   36   20    0 | ## 197   25   23    0 | ## 251   45   25    0 | ## 305   28   28    1
## 144   38   20    0 | ## 198   26   23    0 | ## 252   54   25    0 | ## 306   31   28    0
## 145   43   20    0 | ## 199   27   23    0 | ## 253   16   26    0 | ## 307   32   28    0
## 146   45   20    0 | ## 200   28   23    0 | ## 254   19   26    0 | ## 308   33   28    0
## 147   54   20    0 | ## 201   31   23    0 | ## 255   20   26    0 | ## 309   34   28    0
## 148   16   21    0 | ## 202   32   23    0 | ## 256   21   26    0 | ## 310   35   28    0
## 149   19   21    0 | ## 203   33   23    0 | ## 257   22   26    0 | ## 311   36   28    0
## 150   20   21    0 | ## 204   34   23    0 | ## 258   23   26    1 | ## 312   38   28    0
## 151   21   21    1 | ## 205   35   23    0 | ## 259   24   26    1 | ## 313   43   28    0
## 152   22   21    1 | ## 206   36   23    0 | ## 260   25   26    1 | ## 314   45   28    0
## 153   23   21    0 | ## 207   38   23    0 | ## 261   26   26    1 | ## 315   54   28    0
## 154   24   21    0 | ## 208   43   23    0 | ## 262   27   26    2 | ## 316   16   29    0
## 155   25   21    0 | ## 209   45   23    0 | ## 263   28   26    0 | ## 317   19   29    0
## 156   26   21    0 | ## 210   54   23    0 | ## 264   31   26    0 | ## 318   20   29    0
## 157   27   21    0 | ## 211   16   24    0 | ## 265   32   26    0 | ## 319   21   29    0
## 158   28   21    1 | ## 212   19   24    0 | ## 266   33   26    0 | ## 320   22   29    0
## 159   31   21    0 | ## 213   20   24    0 | ## 267   34   26    0 | ## 321   23   29    0
## 160   32   21    0 | ## 214   21   24    0 | ## 268   35   26    0 | ## 322   24   29    0
## 161   33   21    0 | ## 215   22   24    0 | ## 269   36   26    0 | ## 323   25   29    0
## 162   34   21    0 | ## 216   23   24    2 | ## 270   38   26    0 | ## 324   26   29    0
## 163   35   21    0 | ## 217   24   24    0 | ## 271   43   26    0 | ## 325   27   29    0
## 164   36   21    0 | ## 218   25   24    0 | ## 272   45   26    0 | ## 326   28   29    0
## 165   38   21    0 | ## 219   26   24    0 | ## 273   54   26    0 | ## 327   31   29    0
## 166   43   21    0 | ## 220   27   24    0 | ## 274   16   27    0 | ## 328   32   29    0
```

## 382	21	33	0	## 436	35	40	1	## 490	24	52	0	## 544	43	57	0
## 381	20	33	0	## 435	34	40	0	## 489	23	52	0	## 543	38	57	0
## 380	19	33	0	## 434	33	40	0	## 488	22	52	0	## 542	36	57	0
## 379	16	33	0	## 433	32	40	0	## 487	21	52	0	## 541	35	57	0
## 378	54	31	0	## 432	31	40	0	## 486	20	52	0	## 540	34	57	0
## 377	45	31	0	## 431	28	40	0	## 485	19	52	0	## 539	33	57	0
## 376	43	31	0	## 430	27	40	0	## 484	16	52	0	## 538	32	57	0
## 375	38	31	0	## 429	26	40	0	## 483	54	45	0	## 537	31	57	0
## 374	36	31	0	## 428	25	40	0	## 482	45	45	0	## 536	28	57	0
## 373	35	31	0	## 427	24	40	0	## 481	43	45	0	## 535	27	57	0
## 372	34	31	1	## 426	23	40	0	## 480	38	45	0	## 534	26	57	0
## 371	33	31	0	## 425	22	40	0	## 479	36	45	0	## 533	25	57	0
## 370	32	31	0	## 424	21	40	0	## 478	35	45	0	## 532	24	57	0
## 369	31	31	0	## 423	20	40	0	## 477	34	45	0	## 531	23	57	0
## 368	28	31	1	## 422	19	40	0	## 476	33	45	0	## 530	22	57	0
## 367	27	31	0	## 421	16	40	0	## 475	32	45	1	## 529	21	57	0
## 366	26	31	0	## 420	54	34	0	## 474	31	45	0	## 528	20	57	0
## 365	25	31	0	## 419	45	34	0	## 473	28	45	0	## 527	19	57	0
## 364	24	31	0	## 418	43	34	0	## 472	27	45	0	## 526	16	57	0
## 363	23	31	0	## 417	38	34	0	## 471	26	45	0	## 525	54	53	0
## 362	22	31	0	## 416	36	34	0	## 470	25	45	0	## 524	45	53	0
## 361	21	31	0	## 415	35	34	0	## 469	24	45	0	## 523	43	53	0
## 360	20	31	0	## 414	34	34	0	## 468	23	45	0	## 522	38	53	0
## 359	19	31	0	## 413	33	34	0	## 467	22	45	0	## 521	36	53	0
## 358	16	31	0	## 412	32	34	0	## 466	21	45	0	## 520	35	53	0
## 357	54	30	0	## 411	31	34	0	## 465	20	45	0	## 519	34	53	0
## 356	45	30	0	## 410	28	34	1	## 464	19	45	0	## 518	33	53	1
## 355	43	30	0	## 409	27	34	0	## 463	16	45	0	## 517	32	53	0
## 354	38	30	0	## 408	26	34	0	## 462	54	41	0	## 516	31	53	0
## 353	36	30	0	## 407	25	34	0	## 461	45	41	0	## 515	28	53	0
## 352	35	30	0	## 406	24	34	1	## 460	43	41	1	## 514	27	53	0
## 351	34	30	0	## 405	23	34	0	## 459	38	41	0	## 513	26	53	0
## 350	33	30	0	## 404	22	34	0	## 458	36	41	0	## 512	25	53	0
## 349	32	30	0	## 403	21	34	0	## 457	35	41	0	## 511	24	53	0
## 348	31	30	2	## 402	20	34	0	## 456	34	41	0	## 510	23	53	0
## 347	28	30	0	## 401	19	34	0	## 455	33	41	0	## 509	22	53	0
## 346	27	30	0	## 400	16	34	0	## 454	32	41	0	## 508	21	53	0
## 345	26	30	0	## 399	54	33	0	## 453	31	41	0	## 507	20	53	0
## 344	25	30	0	## 398	45	33	0	## 452	28	41	0	## 506	19	53	0
## 343	24	30	0	## 397	43	33	0	## 451	27	41	0	## 505	16	53	0
## 342	23	30	0	## 396	38	33	0	## 450	26	41	0	## 504	54	52	0
## 341	22	30	0	## 395	36	33	0	## 449	25	41	0	## 503	45	52	1
## 340	21	30	0	## 394	35	33	0	## 448	24	41	0	## 502	43	52	0
## 339	20	30	0	## 393	34	33	0	## 447	23	41	0	## 501	38	52	0
## 338	19	30	0	## 392	33	33	0	## 446	22	41	0	## 500	36	52	0
## 337	16	30	0	## 391	32	33	0	## 445	21	41	0	## 499	35	52	0
## 336	54	29	0	## 390	31	33	0	## 444	20	41	0	## 498	34	52	0
## 335	45	29	0	## 389	28	33	0	## 443	19	41	0	## 497	33	52	0
## 334	43	29	0	## 388	27	33	1	## 442	16	41	0	## 496	32	52	0
## 333	38	29	0	## 387	26	33	0	## 441	54	40	0	## 495	31	52	0
## 332	36	29	1	## 386	25	33	0	## 440	45	40	0	## 494	28	52	0
## 331	35	29	0	## 385	24	33	0	## 439	43	40	0	## 493	27	52	0
## 330	34	29	0	## 384	23	33	0	## 438	38	40	0	## 492	26	52	0
## 329	33	29	0	## 383	22	33	0	## 437	36	40	0	## 491	25	52	0

```
## 545   45   57    1
## 546   54   57    0
## 547   16   59    0
## 548   19   59    0
## 549   20   59    0
## 550   21   59    0
## 551   22   59    0
## 552   23   59    0
## 553   24   59    0
## 554   25   59    0
## 555   26   59    0
## 556   27   59    0
## 557   28   59    0
## 558   31   59    0
## 559   32   59    0
## 560   33   59    0
## 561   34   59    0
## 562   35   59    0
## 563   36   59    0
## 564   38   59    0
## 565   43   59    0
## 566   45   59    0
## 567   54   59    1
## 568   16   60    0
## 569   19   60    0
## 570   20   60    0
## 571   21   60    0
## 572   22   60    0
## 573   23   60    0
## 574   24   60    0
## 575   25   60    0
## 576   26   60    0
## 577   27   60    0
## 578   28   60    0
## 579   31   60    0
## 580   32   60    0
## 581   33   60    0
## 582   34   60    0
## 583   35   60    0
## 584   36   60    0
## 585   38   60    1
## 586   43   60    0
## 587   45   60    0
## 588   54   60    0
```

```
cb(Confusion_Matrix)

#### 2015 Actual.E_C.Energy ####

#Use the following section code to copy the confusion matrix for predicted future E&C.

# Make Test/Training Sets#
# Now make a training label set. Ensure the selected column number matches the dependent varia
```

```
ble
#(i.e. x2015.E&C)
data.train.depvar = data[data_ind==1, 8]
data.test.depvar = data[data_ind==2, 8]
data.trainset = cbind(data.train,data.train.depvar)
data.testset = cbind(data.test,data.test.depvar)

# Regression trees
data.rfmodel <-ranger(data.train.depvar ~ ., data = data.trainset, importance = "impurity", wr
ite.forest = TRUE, classification = FALSE)
data.rfpred <- predict(data.rfmodel, data.test)
data.rfpred$predictions
```

```
##  [1] 15.04348 18.57704 17.22411 12.98038 15.09659 12.95146 15.67255
##  [8] 12.38102 17.87415 18.28624 15.27912 23.20065 15.50870 16.61024
## [15] 18.98980 14.17778 16.19288 17.38068 25.81859 16.05837 18.40129
## [22] 19.08192 15.14845 16.86328 16.14430 18.83168 13.78454 18.38007
## [29] 22.42398 18.03286 21.30357 18.26307 23.11967 18.05892 16.90327
## [36] 14.96203 16.37629 19.53116 17.75747 15.07437 17.55432 16.34884
## [43] 16.66421 14.17254 14.00280 14.79139 13.75626 22.29346 15.53302
## [50] 18.57687 16.69769 21.24344 23.17517 16.58137 14.89137 19.00710
## [57] 18.35026
```

```
#Function to round value to the nearest increment of x (i.e. 5)
mround <- function(x,base){
  base*round(x/base)
}

Confusion_Matrix <- table(mround(data.rfpred$predictions,1),mround(data.test.depvar,1))
Confusion_Matrix
```

```
##
##      10 11 12 13 14 15 16 17 18 19 20 21 22 23 25 26 27
##   12  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
##   13  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   14  0  0  1  2  1  1  0  0  0  0  0  0  0  0  0  0  0
##   15  1  1  0  1  2  0  1  1  0  1  0  0  0  0  0  0  0
##   16  0  2  1  1  2  1  0  0  0  0  0  1  0  0  0  0  0
##   17  1  0  0  2  0  2  2  0  0  1  0  0  0  0  0  0  0
##   18  0  0  0  1  0  0  2  0  2  1  2  1  0  0  0  1  0
##   19  0  0  0  0  0  0  0  1  1  0  1  0  0  1  1  1  0
##   20  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
##   21  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  1  0
##   22  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1
##   23  0  0  0  0  0  0  0  0  0  1  1  0  0  0  1  0  0
##   26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
```

```
Accuracy <- sum(diag(Confusion_Matrix))/sum(Confusion_Matrix)
Accuracy
```

```
## [1] 0.1929825
```

```
Confusion_Matrix = as.data.frame(Confusion_Matrix)
Confusion_Matrix
```

```
##     Var1 Var2 Freq
## 1     12   10    0
## 2     13   10    0
## 3     14   10    0
## 4     15   10    1
## 5     16   10    0
## 6     17   10    1
## 7     18   10    0
## 8     19   10    0
## 9     20   10    0
## 10    21   10    0
## 11    22   10    0
## 12    23   10    0
## 13    26   10    0
## 14    12   11    0
## 15    13   11    1
## 16    14   11    0
## 17    15   11    1
## 18    16   11    2
## 19    17   11    0
## 20    18   11    0
## 21    19   11    0
## 22    20   11    0
## 23    21   11    0
## 24    22   11    0
## 25    23   11    0
## 26    26   11    0
## 27    12   12    0
## 28    13   12    1
## 29    14   12    1
## 30    15   12    0
## 31    16   12    1
## 32    17   12    0
## 33    18   12    0
## 34    19   12    0
## 35    20   12    0
## 36    21   12    0
## 37    22   12    0
## 38    23   12    0
## 39    26   12    0
## 40    12   13    1
## 41    13   13    0
## 42    14   13    2
## 43    15   13    1
## 44    16   13    1
## 45    17   13    2
## 46    18   13    1
## 47    19   13    0
```

```
## 48    20   13    0
## 49    21   13    0
## 50    22   13    0
## 51    23   13    0
## 52    26   13    0
## 53    12   14    0
## 54    13   14    0
## 55    14   14    1
## 56    15   14    2
## 57    16   14    2
## 58    17   14    0
## 59    18   14    0
## 60    19   14    0
## 61    20   14    0
## 62    21   14    0
## 63    22   14    0
## 64    23   14    0
## 65    26   14    0
## 66    12   15    0
## 67    13   15    0
## 68    14   15    1
## 69    15   15    0
## 70    16   15    1
## 71    17   15    2
## 72    18   15    0
## 73    19   15    0
## 74    20   15    0
## 75    21   15    0
## 76    22   15    0
## 77    23   15    0
## 78    26   15    0
## 79    12   16    0
## 80    13   16    0
## 81    14   16    0
## 82    15   16    1
## 83    16   16    0
## 84    17   16    2
## 85    18   16    2
## 86    19   16    0
## 87    20   16    0
## 88    21   16    0
## 89    22   16    0
## 90    23   16    0
## 91    26   16    0
## 92    12   17    0
## 93    13   17    0
## 94    14   17    0
## 95    15   17    1
## 96    16   17    0
## 97    17   17    0
## 98    18   17    0
## 99    19   17    1
## 100   20   17    0
## 101   21   17    0
```

```
## 102   22   17    0
## 103   23   17    0
## 104   26   17    0
## 105   12   18    0
## 106   13   18    0
## 107   14   18    0
## 108   15   18    0
## 109   16   18    0
## 110   17   18    0
## 111   18   18    2
## 112   19   18    1
## 113   20   18    0
## 114   21   18    0
## 115   22   18    0
## 116   23   18    0
## 117   26   18    0
## 118   12   19    0
## 119   13   19    0
## 120   14   19    0
## 121   15   19    1
## 122   16   19    0
## 123   17   19    1
## 124   18   19    1
## 125   19   19    0
## 126   20   19    0
## 127   21   19    1
## 128   22   19    0
## 129   23   19    1
## 130   26   19    0
## 131   12   20    0
## 132   13   20    0
## 133   14   20    0
## 134   15   20    0
## 135   16   20    0
## 136   17   20    0
## 137   18   20    2
## 138   19   20    1
## 139   20   20    0
## 140   21   20    0
## 141   22   20    0
## 142   23   20    1
## 143   26   20    0
## 144   12   21    0
## 145   13   21    0
## 146   14   21    0
## 147   15   21    0
## 148   16   21    1
## 149   17   21    0
## 150   18   21    1
## 151   19   21    0
## 152   20   21    1
## 153   21   21    0
## 154   22   21    0
## 155   23   21    0
```

```
## 156   26   21    0
## 157   12   22    0
## 158   13   22    0
## 159   14   22    0
## 160   15   22    0
## 161   16   22    0
## 162   17   22    0
## 163   18   22    0
## 164   19   22    0
## 165   20   22    0
## 166   21   22    0
## 167   22   22    1
## 168   23   22    0
## 169   26   22    0
## 170   12   23    0
## 171   13   23    0
## 172   14   23    0
## 173   15   23    0
## 174   16   23    0
## 175   17   23    0
## 176   18   23    0
## 177   19   23    1
## 178   20   23    0
## 179   21   23    0
## 180   22   23    0
## 181   23   23    0
## 182   26   23    0
## 183   12   25    0
## 184   13   25    0
## 185   14   25    0
## 186   15   25    0
## 187   16   25    0
## 188   17   25    0
## 189   18   25    0
## 190   19   25    1
## 191   20   25    0
## 192   21   25    0
## 193   22   25    0
## 194   23   25    1
## 195   26   25    0
## 196   12   26    0
## 197   13   26    0
## 198   14   26    0
## 199   15   26    0
## 200   16   26    0
## 201   17   26    0
## 202   18   26    1
## 203   19   26    1
## 204   20   26    0
## 205   21   26    1
## 206   22   26    0
## 207   23   26    0
## 208   26   26    1
## 209   12   27    0
```

```
## 210   13   27    0
## 211   14   27    0
## 212   15   27    0
## 213   16   27    0
## 214   17   27    0
## 215   18   27    0
## 216   19   27    0
## 217   20   27    0
## 218   21   27    0
## 219   22   27    1
## 220   23   27    0
## 221   26   27    0
```

```
cb(Confusion_Matrix)

#### 2015 Actual.Thermal.Energy ####
#Use the following section of code to copy the confusion matrix for predicted future year Actu
al.Thermal.Energy.

# Make Test/Training Sets#
# Now make a training label set. Ensure the selected column number matches the dependent varia
ble
#(i.e. x2015.E&C)
data.train.depvar = data[data_ind==1, 11]
data.test.depvar = data[data_ind==2, 11]
data.trainset = cbind(data.train,data.train.depvar)
data.testset = cbind(data.test,data.test.depvar)

# Regression trees
data.rfmodel <-ranger(data.train.depvar ~ ., data = data.trainset, importance = "impurity", wr
ite.forest = TRUE, classification = FALSE)
data.rfpred <- predict(data.rfmodel, data.test)
data.rfpred$predictions
```

```
##  [1] 10.134787  8.848038 10.948676  8.298801 12.085438  9.088728 14.315565
##  [8] 12.730279 11.197289 13.114280  5.975556 10.771033  8.112334  8.296273
## [15]  8.907811 10.191873  7.560902  8.585489 20.383175 15.646677  4.772015
## [22]  7.857861  7.269992 10.418174  9.530852  8.533058 10.716072  8.509755
## [29]  7.997556  5.164942  5.768165  9.901290 11.826854  4.108608  8.134502
## [36]  7.133013  5.356967 10.159801  9.640522 10.464811  7.855780 10.382346
## [43]  9.845458 10.157630 11.775243 13.402364 15.444953  3.097484  7.155199
## [50] 11.550791  8.353384 13.038895  5.534971  8.364769 10.269850  7.345983
## [57]  6.249424
```

```
#Function to round value to the nearest increment of x (i.e. 5)
mround <- function(x,base){
  base*round(x/base)
}

Confusion_Matrix <- table(mround(data.rfpred$predictions,1),mround(data.test.depvar,1))
Confusion_Matrix
```

```
##
##       1 2 3 4 5 6 7 8 9 10 11 12 14 15 16 18 23 24 33
##    3  1 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
##    4  1 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
##    5  0 1 1 1 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
##    6  0 0 0 1 1 1 0 1 0  0  0  0  0  0  0  0  0  0  0
##    7  0 0 0 0 0 2 1 0 1  0  0  0  0  0  0  0  0  0  0
##    8  0 0 0 0 0 1 2 3 4  0  0  0  0  0  0  0  0  0  0
##    9  0 0 0 0 0 0 1 1 0  2  1  1  0  0  0  0  0  0  0
##    10 0 0 0 0 0 0 0 1 1  1  4  5  0  0  0  0  0  0  0
##    11 0 0 0 0 0 0 0 0 0  2  1  0  1  0  0  0  0  0  0
##    12 0 0 0 0 0 0 0 0 1  0  1  1  1  0  0  0  0  0  0
##    13 0 0 0 0 0 0 0 0 0  0  0  0  0  1  1  1  1  0  0
##    14 0 0 0 0 0 0 0 0 0  1  0  0  0  0  0  0  0  0  0
##    15 0 0 0 0 0 0 0 0 0  0  0  1  0  0  0  0  0  0  0
##    16 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  1
##    20 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  1  0
```

```
Accuracy <- sum(diag(Confusion_Matrix))/sum(Confusion_Matrix)
Accuracy
```

```
## [1] 0.1052632
```

```
Confusion_Matrix = as.data.frame(Confusion_Matrix)
Confusion_Matrix
```

```
##     Var1 Var2 Freq
## 1      3    1    1
## 2      4    1    1
## 3      5    1    0
## 4      6    1    0
## 5      7    1    0
## 6      8    1    0
## 7      9    1    0
## 8     10    1    0
## 9     11    1    0
## 10    12    1    0
## 11    13    1    0
## 12    14    1    0
## 13    15    1    0
## 14    16    1    0
## 15    20    1    0
## 16     3    2    0
## 17     4    2    0
## 18     5    2    1
## 19     6    2    0
## 20     7    2    0
## 21     8    2    0
## 22     9    2    0
## 23    10    2    0
```

```
## 24    11    2    0
## 25    12    2    0
## 26    13    2    0
## 27    14    2    0
## 28    15    2    0
## 29    16    2    0
## 30    20    2    0
## 31     3    3    0
## 32     4    3    0
## 33     5    3    1
## 34     6    3    0
## 35     7    3    0
## 36     8    3    0
## 37     9    3    0
## 38    10    3    0
## 39    11    3    0
## 40    12    3    0
## 41    13    3    0
## 42    14    3    0
## 43    15    3    0
## 44    16    3    0
## 45    20    3    0
## 46     3    4    0
## 47     4    4    0
## 48     5    4    1
## 49     6    4    1
## 50     7    4    0
## 51     8    4    0
## 52     9    4    0
## 53    10    4    0
## 54    11    4    0
## 55    12    4    0
## 56    13    4    0
## 57    14    4    0
## 58    15    4    0
## 59    16    4    0
## 60    20    4    0
## 61     3    5    0
## 62     4    5    0
## 63     5    5    0
## 64     6    5    1
## 65     7    5    0
## 66     8    5    0
## 67     9    5    0
## 68    10    5    0
## 69    11    5    0
## 70    12    5    0
## 71    13    5    0
## 72    14    5    0
## 73    15    5    0
## 74    16    5    0
## 75    20    5    0
## 76     3    6    0
## 77     4    6    0
```

```
## 78     5    6    0
## 79     6    6    1
## 80     7    6    2
## 81     8    6    1
## 82     9    6    0
## 83    10    6    0
## 84    11    6    0
## 85    12    6    0
## 86    13    6    0
## 87    14    6    0
## 88    15    6    0
## 89    16    6    0
## 90    20    6    0
## 91     3    7    0
## 92     4    7    0
## 93     5    7    0
## 94     6    7    0
## 95     7    7    1
## 96     8    7    2
## 97     9    7    1
## 98    10    7    0
## 99    11    7    0
## 100   12    7    0
## 101   13    7    0
## 102   14    7    0
## 103   15    7    0
## 104   16    7    0
## 105   20    7    0
## 106    3    8    0
## 107    4    8    0
## 108    5    8    0
## 109    6    8    1
## 110    7    8    0
## 111    8    8    3
## 112    9    8    1
## 113   10    8    1
## 114   11    8    0
## 115   12    8    0
## 116   13    8    0
## 117   14    8    0
## 118   15    8    0
## 119   16    8    0
## 120   20    8    0
## 121    3    9    0
## 122    4    9    0
## 123    5    9    0
## 124    6    9    0
## 125    7    9    1
## 126    8    9    4
## 127    9    9    0
## 128   10    9    1
## 129   11    9    0
## 130   12    9    1
## 131   13    9    0
```

```
## 132   14    9    0
## 133   15    9    0
## 134   16    9    0
## 135   20    9    0
## 136    3   10    0
## 137    4   10    0
## 138    5   10    0
## 139    6   10    0
## 140    7   10    0
## 141    8   10    0
## 142    9   10    2
## 143   10   10    1
## 144   11   10    2
## 145   12   10    0
## 146   13   10    0
## 147   14   10    1
## 148   15   10    0
## 149   16   10    0
## 150   20   10    0
## 151    3   11    0
## 152    4   11    0
## 153    5   11    0
## 154    6   11    0
## 155    7   11    0
## 156    8   11    0
## 157    9   11    1
## 158   10   11    4
## 159   11   11    1
## 160   12   11    1
## 161   13   11    0
## 162   14   11    0
## 163   15   11    0
## 164   16   11    0
## 165   20   11    0
## 166    3   12    0
## 167    4   12    0
## 168    5   12    0
## 169    6   12    0
## 170    7   12    0
## 171    8   12    0
## 172    9   12    1
## 173   10   12    5
## 174   11   12    0
## 175   12   12    1
## 176   13   12    0
## 177   14   12    0
## 178   15   12    1
## 179   16   12    0
## 180   20   12    0
## 181    3   14    0
## 182    4   14    0
## 183    5   14    0
## 184    6   14    0
## 185    7   14    0
```

```
## 186    8   14    0
## 187    9   14    0
## 188   10   14    0
## 189   11   14    1
## 190   12   14    1
## 191   13   14    0
## 192   14   14    0
## 193   15   14    0
## 194   16   14    0
## 195   20   14    0
## 196    3   15    0
## 197    4   15    0
## 198    5   15    0
## 199    6   15    0
## 200    7   15    0
## 201    8   15    0
## 202    9   15    0
## 203   10   15    0
## 204   11   15    0
## 205   12   15    0
## 206   13   15    1
## 207   14   15    0
## 208   15   15    0
## 209   16   15    0
## 210   20   15    0
## 211    3   16    0
## 212    4   16    0
## 213    5   16    0
## 214    6   16    0
## 215    7   16    0
## 216    8   16    0
## 217    9   16    0
## 218   10   16    0
## 219   11   16    0
## 220   12   16    0
## 221   13   16    1
## 222   14   16    0
## 223   15   16    0
## 224   16   16    0
## 225   20   16    0
## 226    3   18    0
## 227    4   18    0
## 228    5   18    0
## 229    6   18    0
## 230    7   18    0
## 231    8   18    0
## 232    9   18    0
## 233   10   18    0
## 234   11   18    0
## 235   12   18    0
## 236   13   18    1
## 237   14   18    0
## 238   15   18    0
## 239   16   18    0
```

```
## 240   20   18    0
## 241    3   23    0
## 242    4   23    0
## 243    5   23    0
## 244    6   23    0
## 245    7   23    0
## 246    8   23    0
## 247    9   23    0
## 248   10   23    0
## 249   11   23    0
## 250   12   23    0
## 251   13   23    1
## 252   14   23    0
## 253   15   23    0
## 254   16   23    0
## 255   20   23    0
## 256    3   24    0
## 257    4   24    0
## 258    5   24    0
## 259    6   24    0
## 260    7   24    0
## 261    8   24    0
## 262    9   24    0
## 263   10   24    0
## 264   11   24    0
## 265   12   24    0
## 266   13   24    0
## 267   14   24    0
## 268   15   24    0
## 269   16   24    0
## 270   20   24    1
## 271    3   33    0
## 272    4   33    0
## 273    5   33    0
## 274    6   33    0
## 275    7   33    0
## 276    8   33    0
## 277    9   33    0
## 278   10   33    0
## 279   11   33    0
## 280   12   33    0
## 281   13   33    0
## 282   14   33    0
## 283   15   33    0
## 284   16   33    1
## 285   20   33    0
```

```
cb(Confusion_Matrix)
#### 2015 Actual.total.Energy ####
#Use the following section code to copy the confusion matrix for predicted future year Actual.
Total.Energy.

# Make Test/Training Sets#
```

```
# Make a training label set. Ensure the selected column number matches the dependent variable
#(i.e. x2015.E&C)
data.train.depvar = data[data_ind==1, 15]
data.test.depvar = data[data_ind==2, 15]
data.trainset = cbind(data.train,data.train.depvar)
data.testset = cbind(data.test,data.test.depvar)

# Regression trees
data.rfmodel <-ranger(data.train.depvar ~ ., data = data.trainset, importance = "impurity", wr
ite.forest = TRUE, classification = FALSE)
data.rfpred <- predict(data.rfmodel, data.test)
data.rfpred$predictions
```

```
##  [1] -4.8013224 -4.5080893 -6.5672743 -1.9902708 -1.5315319 -1.7346892
##  [7] -5.7786431 -4.2645934 -5.5634145 -8.4469516  0.8169294 -7.0967624
## [13]  0.9451351 -1.1912858 -3.6633609 -5.8791368 -1.1413188 -1.6874630
## [19] -7.8785794 -6.5770145 -2.7020560 -1.4360484 -0.6538297 -2.1510101
## [25] -2.7614703 -3.9719211 -3.8106603 -4.2780113 -4.4692059 -1.6699081
## [31] -3.1555555 -1.4504164 -7.8504086 -1.1026131 -1.3619219 -1.5574590
## [37] -2.7328065 -2.9641985 -3.7457566 -3.9801141 -2.6904136 -4.3249187
## [43] -2.3005068 -3.1490215 -2.3478710 -4.1409999 -4.2552717 -3.9356262
## [49] -4.1059218 -2.9660692  3.8457482 -9.5818954 -1.3632080 -1.8773941
## [55] -3.2692931 -4.9123491 -1.9165695
```

```
Confusion_Matrix <- table(mround(data.rfpred$predictions,1),mround(data.test.depvar,1))
Confusion_Matrix
```

```
##
##       -13 -11 -10 -9 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 9 16
##   -10   0   0   0  0  0  0  0  0  0  0  0 0 1 0 0 0 0 0  0
##   -8    0   1   0  0  0  1  1  0  0  0  0 0 0 0 0 0 0 0  0
##   -7    1   0   0  0  0  0  0  1  0  0  0 0 0 0 0 0 0 0  1
##   -6    0   0   0  1  1  0  0  0  1  0  0 0 0 0 0 0 0 0  0
##   -5    0   0   0  0  0  0  0  0  0  1  1 0 0 0 0 0 1 0  0
##   -4    0   0   1  2  0  3  2  1  1  1  2 0 0 0 0 0 0 0  0
##   -3    0   0   0  0  0  4  0  1  2  0  0 1 1 0 0 0 0 0  0
##   -2    0   0   0  0  0  0  0  1  1  2  3 2 0 2 0 0 0 0  0
##   -1    0   0   0  0  0  1  0  0  1  2  0 1 0 0 1 1 0 1  0
##   1     0   0   0  0  0  0  0  0  0  0  1 1 0 0 0 0 0 0  0
##   4     0   0   0  0  0  0  0  0  0  1  0 0 0 0 0 0 0 0  0
```

```
Accuracy <- sum(diag(Confusion_Matrix))/sum(Confusion_Matrix)
Accuracy
```

```
## [1] 0.122807
```

```
Confusion_Matrix = as.data.frame(Confusion_Matrix)
Confusion_Matrix
```

```
##     Var1 Var2 Freq
## 1    -10  -13    0
## 2     -8  -13    0
## 3     -7  -13    1
## 4     -6  -13    0
## 5     -5  -13    0
## 6     -4  -13    0
## 7     -3  -13    0
## 8     -2  -13    0
## 9     -1  -13    0
## 10     1  -13    0
## 11     4  -13    0
## 12   -10  -11    0
## 13    -8  -11    1
## 14    -7  -11    0
## 15    -6  -11    0
## 16    -5  -11    0
## 17    -4  -11    0
## 18    -3  -11    0
## 19    -2  -11    0
## 20    -1  -11    0
## 21     1  -11    0
## 22     4  -11    0
## 23   -10  -10    0
## 24    -8  -10    0
## 25    -7  -10    0
## 26    -6  -10    0
## 27    -5  -10    0
## 28    -4  -10    1
## 29    -3  -10    0
## 30    -2  -10    0
## 31    -1  -10    0
## 32     1  -10    0
## 33     4  -10    0
## 34   -10   -9    0
## 35    -8   -9    0
## 36    -7   -9    0
## 37    -6   -9    1
## 38    -5   -9    0
## 39    -4   -9    2
## 40    -3   -9    0
## 41    -2   -9    0
## 42    -1   -9    0
## 43     1   -9    0
## 44     4   -9    0
## 45   -10   -7    0
## 46    -8   -7    0
## 47    -7   -7    0
## 48    -6   -7    1
## 49    -5   -7    0
## 50    -4   -7    0
## 51    -3   -7    0
## 52    -2   -7    0
## 53    -1   -7    0
## 54     1   -7    0
## 55     4   -7    0
## 56   -10   -6    0
## 57    -8   -6    1
## 58    -7   -6    0
## 59    -6   -6    0
## 60    -5   -6    0
## 61    -4   -6    3
## 62    -3   -6    4
## 63    -2   -6    0
## 64    -1   -6    1
## 65     1   -6    0
## 66     4   -6    0
## 67   -10   -5    0
## 68    -8   -5    1
## 69    -7   -5    0
## 70    -6   -5    0
## 71    -5   -5    0
## 72    -4   -5    2
## 73    -3   -5    0
## 74    -2   -5    0
## 75    -1   -5    0
## 76     1   -5    0
## 77     4   -5    0
## 78   -10   -4    0
## 79    -8   -4    0
## 80    -7   -4    1
## 81    -6   -4    0
## 82    -5   -4    0
## 83    -4   -4    1
## 84    -3   -4    1
## 85    -2   -4    1
## 86    -1   -4    0
## 87     1   -4    0
## 88     4   -4    0
## 89   -10   -3    0
## 90    -8   -3    0
## 91    -7   -3    0
## 92    -6   -3    1
## 93    -5   -3    0
## 94    -4   -3    1
## 95    -3   -3    2
## 96    -2   -3    1
## 97    -1   -3    1
## 98     1   -3    0
## 99     4   -3    0
## 100  -10   -2    0
## 101   -8   -2    0
## 102   -7   -2    0
## 103   -6   -2    0
## 104   -5   -2    1
## 105   -4   -2    1
## 106   -3   -2    0
## 107   -2   -2    2
## 108   -1   -2
## 109    1
## 110    4
## 111  -1
## 112   -
## 113   -
## 114   -
## 115   -
## 116   -
## 117   -
## 118   -
## 119   -1   -1
## 120    1
## 121    4
## 122  -1
## 123   -
## 124   -
## 125   -
## 126   -
## 127   -
## 128   -
## 129   -
## 130   -1
## 131         0    1
## 132         0    0
## 133  -1
## 134   -
## 135   -
## 136   -
## 137   -
## 138   -
## 139   -
## 140   -
## 141   -1
## 142         1    0
## 143         1    0
## 144  -1
## 145   -
## 146   -
## 147   -
## 148   -
## 149   -
## 150   -
## 151   -
## 152   -1
## 153         2    0
## 154         2    0
## 155  -1
## 156   -
## 157   -
## 158   -
## 159   -
## 160   -
## 161   -
```

```
## 162   -2    3    0
## 163   -1    3    1
## 164    1    3    0
## 165    4    3    0
## 166  -10    4    0
## 167   -8    4    0
## 168   -7    4    0
## 169   -6    4    0
## 170   -5    4    0
## 171   -4    4    0
## 172   -3    4    0
## 173   -2    4    0
## 174   -1    4    1
## 175    1    4    0
## 176    4    4    0
## 177  -10    5    0
## 178   -8    5    0
## 179   -7    5    0
## 180   -6    5    0
## 181   -5    5    1
## 182   -4    5    0
## 183   -3    5    0
## 184   -2    5    0
## 185   -1    5    0
## 186    1    5    0
## 187    4    5    0
## 188  -10    9    0
## 189   -8    9    0
## 190   -7    9    0
## 191   -6    9    0
## 192   -5    9    0
## 193   -4    9    0
## 194   -3    9    0
## 195   -2    9    0
## 196   -1    9    1
## 197    1    9    0
## 198    4    9    0
## 199  -10   16    0
## 200   -8   16    0
## 201   -7   16    1
## 202   -6   16    0
## 203   -5   16    0
## 204   -4   16    0
## 205   -3   16    0
## 206   -2   16    0
## 207   -1   16    0
## 208    1   16    0
## 209    4   16    0
```

```
cb(Confusion_Matrix)
```

ANN

```
#### Load Packages ####
library(neuralnet)
library("beepr")
library(ggplot2)
library(lattice)
library(caret)
library(boot)
library(plyr)
library(withr)
set.seed(1234)
#### Load Dataset ####
setwd("C:/Users/carleen.lawson/Documents/Data/2010-2015 Data/Cross Validation Sets")
data <- read.csv(file.choose(), row.names = 1, check.names = FALSE, na.strings = "NA", sep = ",", stringsAsFactors = TRUE, header = TRUE)
#### Normalize Dataset #####
cnames <- cbind("Delta.EUI","Portfolio.Manager","Exterior.Area","Lattitude","Longitude","Construction.Year","Num.of.Structures",
                "Class","Climate.Zone","Electrically.Heated","Cooling.Tower","Soft.Landscaping","X2010.EUI",
                "Occupant.Density","Vacancy.Rate","Weekly.Operating.Hours")
normalize <- function(x){(x-min(x))/(max(x)-min(x))}
normalize.delta <- function(x){(x-(-50))/(50-(-50))}
rnames <- row.names(data)
norm.indepvar<- as.data.frame(lapply(data[,-1], normalize))
norm.depvar <- as.data.frame(normalize.delta(data[,1]))
norm.data <-cbind(norm.depvar, norm.indepvar)
Delta.EUI <- norm.data$norm.depvar
row.names(norm.data) <- rnames
colnames(norm.data) <- cnames

#### Subset Data ####
#print.rows <- function(x){cat(((toString(shQuote((row.names.data.frame(x)), type = "cmd")))), "\n")}

r.test1 <-c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524")
r.test2 <- c("215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302",
"138" )
r.test3 <- c("221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293")
r.test4 <- c("272", "249", "202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175",
"582")
r.test5 <- c("224", "565", "250", "559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543",
"148" )
r.test6 <- c("637", "212", "210", "518", "645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523",
"97" )
r.test7 <- c("126", "508", "475", "513", "965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195",
"297" )
r.test8 <- c("509", "650", "86", "251", "663", "287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194" )
r.test9 <- c("96", "636", "122", "101", "83", "143", "491", "107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127" )
r.test10 <- c("560", "93", "357", "211", "109", "273", "488", "640", "85", "81", "643", "110", "112", "128", "562", "652", "579", "84", "526")

r.train1 <-c("215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302",
"138", "221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272",
"249", "202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565",
"250", "559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210",
"518", "645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513",
"965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663",
"287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491",
"107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640", "85",
"81", "643", "110", "112", "128", "562", "652", "579", "84", "526")
r.train2 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272",
"249", "202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565",
"250", "559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210",
"518", "645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513",
"965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663",
"287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491",
"107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640", "85",
"81", "643", "110", "112", "128", "562", "652", "579", "84", "526" )
r.train3 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"272", "249", "202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224",
"565", "250", "559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212",
"210", "518", "645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475",
"513", "965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251",
"663", "287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143",
"491", "107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640",
"85", "81", "643", "110", "112", "128", "562", "652", "579", "84", "526")
r.train4 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "224", "565",
"250", "559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210",
"518", "645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513",
"965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663",
"287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491",
"107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640", "85",
"81", "643", "110", "112", "128", "562", "652", "579", "84", "526")
r.train5 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272", "249",
"202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "637", "212", "210",
"518", "645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513",
"965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663",
"287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491",
"107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640", "85",
"81", "643", "110", "112", "128", "562", "652", "579", "84", "526" )
r.train6 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272", "249",
"202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565", "250",
"559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "126", "508", "475", "513",
"965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663",
"287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491",
"107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640", "85",
"81", "643", "110", "112", "128", "562", "652", "579", "84", "526" )
r.train7 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272", "249",
"202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565", "250",
"559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210", "518",
```

```
"645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "509", "650", "86", "251", "663",
"287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491",
"107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640", "85",
"81", "643", "110", "112", "128", "562", "652", "579", "84", "526" )
r.train8 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272", "249",
"202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565", "250",
"559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210", "518",
"645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513", "965",
"214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "96", "636", "122", "101", "83", "143",
"491", "107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640",
"85", "81", "643", "110", "112", "128", "562", "652", "579", "84", "526")
r.train9 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272", "249",
"202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565", "250",
"559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210", "518",
"645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513", "965",
"214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663", "287",
"190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "560", "93", "357", "211", "109", "273", "488", "640",
"85", "81", "643", "110", "112", "128", "562", "652", "579", "84", "526")
r.train10 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272", "249",
"202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565", "250",
"559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210", "518",
"645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513", "965",
"214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663", "287",
"190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491", "107",
"116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127")

data.test.1 <- norm.data[r.test1,]
data.test.2 <- norm.data[r.test2,]
data.test.3 <- norm.data[r.test3,]
data.test.4 <- norm.data[r.test4,]
data.test.5 <- norm.data[r.test5,]
data.test.6 <- norm.data[r.test6,]
data.test.7 <- norm.data[r.test7,]
data.test.8 <- norm.data[r.test8,]
data.test.9 <- norm.data[r.test9,]
data.test.10 <- norm.data[r.test10,]

data.train.1 <- norm.data[r.train1,]
data.train.2 <- norm.data[r.train2,]
data.train.3 <- norm.data[r.train3,]
data.train.4 <- norm.data[r.train4,]
data.train.5 <- norm.data[r.train5,]
data.train.6 <- norm.data[r.train6,]
data.train.7 <- norm.data[r.train7,]
data.train.8 <- norm.data[r.train8,]
data.train.9 <- norm.data[r.train9,]
data.train.10 <- norm.data[r.train10,]

#### Functions ####
#descales the dependent variable, using a max of (50) and a min of (-50).
denormalize.delta <- function(x){(x*100)-50}

nn.accuracy <- function(nn, test.x, test.y, rep, ind.variables) {
  scaled.pr.nn <- (compute(nn,test.x[,ind.variables], ncol(nn$result.matrix)))$net.result
  Predicted.Delta.EUI <- round(denormalize.delta(scaled.pr.nn),0)
  Actual.Delta.EUI <- round(denormalize.delta(test.y),0)
  Residuals <- Actual.Delta.EUI-Predicted.Delta.EUI
  Compare <- as.data.frame(cbind(Actual.Delta.EUI,Predicted.Delta.EUI,Residuals))
  colnames(Compare)<-c('Actual.Delta.EUI','Predicted.Delta.EUI','Residuals')
  Compare$Actual.Class = ifelse((Compare$Actual.Delta.EUI <= 0 ),"Poor",
                                ifelse((Compare$Actual.Delta.EUI > 0&Compare$Actual.Delta.EUI <10),"Fair",
                                       ifelse((Compare$Actual.Delta.EUI >=10),"Good",0)))
  Compare$Predicted.Class = ifelse((Compare$Predicted.Delta.EUI <= 0 ),"Poor",
                                   ifelse((Compare$Predicted.Delta.EUI > 0 & Compare$Predicted.Delta.EUI <10),"Fair",
                                          ifelse((Compare$Predicted.Delta.EUI >=10),"Good",0)))
  Compare$Correct = ifelse((Compare$Actual.Delta.EUI <= 0 & Compare$Predicted.Delta.EUI <= 0 ),1,
                           ifelse((Compare$Actual.Delta.EUI > 0&Compare$Actual.Delta.EUI <10 & Compare$Predicted.Delta.EUI > 0 &
Compare$Predicted.Delta.EUI <10),1,
                                  ifelse((Compare$Actual.Delta.EUI >10 & Compare$Predicted.Delta.EUI >10),1,0)))
  Class.Accuracy<- round((sum(Compare$Correct))/(nrow(Compare))*100,2)
  Results=list("Class.Accuracy" = Class.Accuracy, "Compare.Results" = Compare)
  return(Results)
}

nn.benchmark.1layers<- function(formula, train, test.x, test.y, min.nodes.1layer, max.nodes.1layer, threshold, reps,ind.variables) {
  nrows <- ((max.nodes.1layer-min.nodes.1layer)+1)

  benchmark.result <- matrix(ncol=4, nrow=nrows)
  result.row <- 1

  nn.results <- matrix(nrow=1, ncol=reps)


  for (i in min.nodes.1layer:max.nodes.1layer) {
    print(paste("running the network with", i, "node(s) on the hidden layer"))
    try({
      nn.tested <- neuralnet(formula=formula, train, hidden=c(i), linear.output=TRUE, threshold=threshold, rep=reps, stepmax = 10000)
      for (k in 1:reps){
        nn.results[1,k] <- (nn.accuracy(nn.tested, test.x, test.y, ncol(nn.tested$result.matrix), ind.variables))$Class.Accuracy
      }

      benchmark.result[result.row,]<- c(i, mean(nn.results[1,]), max(nn.results[1,]), min(nn.results[1,]))
    })
    result.row <- result.row+1
  }

  benchmark.result <- as.data.frame(benchmark.result)
  colnames(benchmark.result) <- c("Layer 1", "Average", "Max", "Min")
  Results = list("benchmark.result" = benchmark.result, "best.results" = benchmark.result[which.max(benchmark.result$Average),])
  return(Results)
```

```
}
nn.benchmark.2layers<- function(formula, train, test.x, test.y, min.nodes.1layer, max.nodes.1layer, min.nodes.2layer, max.nodes.2layer, threshold,
reps, ind.variables) {
  nrows <- ((max.nodes.1layer-min.nodes.1layer)+1)*((max.nodes.2layer-min.nodes.2layer)+1)

  benchmark.result <- matrix(ncol=5, nrow=nrows)
  result.row <- 1

  nn.results <- matrix(nrow=1, ncol=reps)

  for (i in min.nodes.1layer:max.nodes.1layer) {
    for (j in min.nodes.2layer:max.nodes.2layer) {
      print(paste("running the network with", i, "node(s) on the 1st hidden layer and", j, "node(s) on the 2nd"))
      try({
        nn.tested <- neuralnet(formula=formula, train, hidden=c(i,j), linear.output=TRUE, threshold=threshold, rep=reps, stepmax = 10000)
        for (k in 1:reps){
          nn.results[1,k] <- (nn.accuracy(nn.tested, test.x, test.y, ncol(nn.tested$result.matrix), ind.variables))$Class.Accuracy
        }

        benchmark.result[result.row,]<- c(i, j, mean(nn.results[1,]), max(nn.results[1,]), min(nn.results[1,]))
      })
      result.row <- result.row+1
    }
  }

  benchmark.result <- as.data.frame(benchmark.result)
  colnames(benchmark.result) <- c("Layer 1", "Layer 2", "Average", "Max", "Min")
  Results = list("benchmark.result" = benchmark.result, "best.results" = benchmark.result[which.max(benchmark.result$Average),])
  return(Results)
}
nn.benchmark.3layers<- function(formula, train, test.x, test.y, min.nodes.1layer, max.nodes.1layer, min.nodes.2layer, max.nodes.2layer,
min.nodes.3layer, max.nodes.3layer, threshold, reps, ind.variables) {
  nrows <- ((max.nodes.1layer-min.nodes.1layer)+1)*((max.nodes.2layer-min.nodes.2layer)+1)*((max.nodes.3layer-min.nodes.3layer)+1)

  benchmark.result <- matrix(ncol=6, nrow=nrows)
  result.row <- 1

  nn.results <- matrix(nrow=1, ncol=reps)

  for (i in min.nodes.1layer:max.nodes.1layer) {
    for (j in min.nodes.2layer:max.nodes.2layer) {
      for (k in min.nodes.3layer:max.nodes.3layer) {
        print(paste("Running the network with", i, "nodes on the 1st layer and", j, "nodes on the 2nd and", k, "nodes on the 3rd"))
        try({
          nn.tested <- neuralnet(formula=formula, train, hidden=c(i,j,k), linear.output=TRUE, threshold=threshold, rep=reps, stepmax = 10000)
          for (r in 1:reps){
            nn.results[1,r] <- (nn.accuracy(nn.tested, test.x, test.y, ncol(nn.tested$result.matrix), ind.variables))$Class.Accuracy
          }

          benchmark.result[result.row,]<- c(i, j,k, mean(nn.results[1,]), max(nn.results[1,]), min(nn.results[1,]))
        })
        result.row <- result.row+1
      }
    }
  }

  benchmark.result <- as.data.frame(benchmark.result)
  colnames(benchmark.result) <- c("Layer 1", "Layer 2","Layer 3", "Average", "Max", "Min")
  Results = list("benchmark.result" = benchmark.result, "best.results" = benchmark.result[which.max(benchmark.result$Average),])
  return(Results)
}

nn.best.1layer<- function(formula, train, threshold, reps, Best.result.1lay) {

  i=Best.result.1lay[,1]
  nn.tested <- neuralnet(formula=formula, train, hidden=c(i), linear.output=TRUE, threshold=threshold, rep=reps, stepmax = 10000)
  return("nn.model"=nn.tested)
}

nn.best.2layers<- function(formula, train, threshold, reps, Best.result.2lay) {

  i=Best.result.2lay[,1]
  j=Best.result.2lay[,2]
  nn.tested <- neuralnet(formula=formula, train, hidden=c(i,j), linear.output=TRUE, threshold=threshold, rep=reps, stepmax = 10000)
  return("nn.model"=nn.tested)
}

nn.best.3layers<- function(formula, train, threshold, reps, Best.result.3lay) {

  i=Best.result.3lay[,1]
  j=Best.result.3lay[,2]
  k=Best.result.3lay[,3]
  nn.tested <- neuralnet(formula=formula, train, hidden=c(i,j,k), linear.output=TRUE, threshold=threshold, rep=reps, stepmax = 10000)
  return("nn.model"=nn.tested)
}


#### i1 Define Model Varibles- All Variables ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area", "Lattitude", "Longitude", "Construction.Year",
                   "Num.of.Structures", "Class", "Climate.Zone", "Electrically.Heated",
                   "Cooling.Tower", "Soft.Landscaping", "X2010.EUI", "Occupant.Density",
                   "Vacancy.Rate", "Weekly.Operating.Hours")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15

#### i1 Training Model - All Variables####
# 1 Hidden Layer #
i1_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
```

```
i1_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i1_Summary.1lay <- cbind(Layer1=i1_results.1lay.1[,1],F1.Avg=i1_results.1lay.1[,4], F2.Avg=i1_results.1lay.2[,4], F1.Avg=i1_results.1lay.3[,4],
F4.Avg=i1_results.1lay.4[,4],
                         F5.Avg=i1_results.1lay.5[,4], F6.Avg=i1_results.1lay.6[,4], F7.Avg=i1_results.1lay.7[,4], F8.Avg=i1_results.1lay.8[,4],
                         F9.Avg=i1_results.1lay.9[,4], F10.Avg=i1_results.1lay.10[,4])
i1_Summary.1lay <- as.data.frame(i1_Summary.1lay)
i1_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i1_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i1_Summary.1lay, file = "Tuning SUmmary, ANN, h=1, 2010-2015, Actual EUI Change, All Variables")
i1_Best.result.1lay = i1_Summary.1lay[which.max(i1_Summary.1lay$CV.Average),]
write.csv(i1_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, Actual EUI Change, All Variables")

# 2 Hidden Layers #

i1_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i1_Summary.2lay <- cbind(i1_results.2lay.1[,1:2],F1.Avg=i1_results.2lay.1[,4], F2.Avg=ri1_esults.2lay.2[,4], F2.Avg=i1_results.2lay.3[,4],
F4.Avg=i1_results.2lay.4[,4],
                         F5.Avg=i1_results.2lay.5[,4], F6.Avg=i1_results.2lay.6[,4], F7.Avg=i1_results.2lay.7[,4], F8.Avg=i1_results.2lay.8[,4],
                         F9.Avg=i1_results.2lay.9[,4], F10.Avg=i1_results.2lay.10[,4])
i1_Summary.2lay <- as.data.frame(i1_Summary.2lay)
i1_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i1_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi1_(i1_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, Actual EUI Change, All Variables")
i1_Best.result.2lay = i1_Summary.2lay[which.max(i1_Summary.2lay$CV.Average),]
write.csv(i1_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, Actual EUI Change, All Variables")

# 3 Hidden Layers #

i1_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i1_Summary.3lay <- cbind(i1_results.3lay.1[,1:3],F1.Avg=i1_results.3lay.1[,4], F2.Avg=i1_results.3lay.2[,4], F3.Avg=i1_results.3lay.3[,4],
F4.Avg=i1_results.3lay.4[,4],
                         F5.Avg=i1_results.3lay.5[,4], F6.Avg=i1_results.3lay.6[,4], F7.Avg=i1_results.3lay.7[,4], F8.Avg=i1_results.3lay.8[,4],
                         F9.Avg=i1_results.3lay.9[,4], F10.Avg=i1_results.3lay.10[,4])

i1_Summary.3lay <- as.data.frame(i1_Summary.3lay)
i1_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i1_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i1_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, Actual EUI Change, All Variables")
i1_Best.result.3lay = i1_Summary.3lay[which.max(i1_Summary.3lay$CV.Average),]
write.csv(i1_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, Actual EUI Change, All Variables")

#### i2 Define Model Varibles ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area","Lattitude", "Longitude", "Construction.Year",
                   "Num.of.Structures","Class","Climate.Zone", "Electrically.Heated",
                   "X2010.EUI", "Occupant.Density",
                   "Vacancy.Rate", "Weekly.Operating.Hours")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15
#### i2 Training Model - No CT, SL####
# 1 Hidden Layer #
```

```
i2_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i2_Summary.1lay <- cbind(Layer1=i2_results.1lay.1[,1],F1.Avg=i2_results.1lay.1[,4], F2.Avg=i2_results.1lay.2[,4], F1.Avg=i2_results.1lay.3[,4],
F4.Avg=i2_results.1lay.4[,4],
                         F5.Avg=i2_results.1lay.5[,4], F6.Avg=i2_results.1lay.6[,4], F7.Avg=i2_results.1lay.7[,4], F8.Avg=i2_results.1lay.8[,4],
                         F9.Avg=i2_results.1lay.9[,4], F10.Avg=i2_results.1lay.10[,4])
i2_Summary.1lay <- as.data.frame(i2_Summary.1lay)
i2_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i2_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i2_Summary.1lay, file = "Tuning SUmmary, ANN, h=1, 2010-2015, Actual EUI Change, No CT, SL")
i2_Best.result.1lay = i2_Summary.1lay[which.max(i2_Summary.1lay$CV.Average),]
write.csv(i2_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, Actual EUI Change, No CT, SL")

# 2 Hidden Layers #

i2_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i2_Summary.2lay <- cbind(i2_results.2lay.1[,1:2],F1.Avg=i2_results.2lay.1[,4], F2.Avg=ri2_esults.2lay.2[,4], F2.Avg=i2_results.2lay.3[,4],
F4.Avg=i2_results.2lay.4[,4],
                         F5.Avg=i2_results.2lay.5[,4], F6.Avg=i2_results.2lay.6[,4], F7.Avg=i2_results.2lay.7[,4], F8.Avg=i2_results.2lay.8[,4],
                         F9.Avg=i2_results.2lay.9[,4], F10.Avg=i2_results.2lay.10[,4])
i2_Summary.2lay <- as.data.frame(i2_Summary.2lay)
i2_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i2_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi2_(i2_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, Actual EUI Change, No CT, SL or Lat")
i2_Best.result.2lay = i2_Summary.2lay[which.max(i2_Summary.2lay$CV.Average),]
write.csv(i2_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, Actual EUI Change, No CT, SL or Lat")

# 3 Hidden Layers #

i2_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i2_Summary.3lay <- cbind(i2_results.3lay.1[,1:3],F1.Avg=i2_results.3lay.1[,4], F2.Avg=i2_results.3lay.2[,4], F3.Avg=i2_results.3lay.3[,4],
F4.Avg=i2_results.3lay.4[,4],
                         F5.Avg=i2_results.3lay.5[,4], F6.Avg=i2_results.3lay.6[,4], F7.Avg=i2_results.3lay.7[,4], F8.Avg=i2_results.3lay.8[,4],
                         F9.Avg=i2_results.3lay.9[,4], F10.Avg=i2_results.3lay.10[,4])

i2_Summary.3lay <- as.data.frame(i2_Summary.3lay)
i2_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i2_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i2_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, Actual EUI Change, No CT, SL or Lat")
i2_Best.result.3lay = i2_Summary.3lay[which.max(i2_Summary.3lay$CV.Average),]
write.csv(i2_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, Actual EUI Change, No CT, SL or Lat")


#### i3 Define Model Varibles ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area", "Longitude", "Construction.Year",
                   "Num.of.Structures","Class","Climate.Zone", "Electrically.Heated",
                   "X2010.EUI", "Occupant.Density",
```

```
                  "Vacancy.Rate", "Weekly.Operating.Hours")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15
#### i3 Training Model - No CT, SL or Lat####
# 1 Hidden Layer #
i3_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i3_Summary.1lay <- cbind(Layer1=i3_results.1lay.1[,1],F1.Avg=i3_results.1lay.1[,4], F2.Avg=i3_results.1lay.2[,4], F1.Avg=i3_results.1lay.3[,4],
F4.Avg=i3_results.1lay.4[,4],
                         F5.Avg=i3_results.1lay.5[,4], F6.Avg=i3_results.1lay.6[,4], F7.Avg=i3_results.1lay.7[,4], F8.Avg=i3_results.1lay.8[,4],
                         F9.Avg=i3_results.1lay.9[,4], F10.Avg=i3_results.1lay.10[,4])
i3_Summary.1lay <- as.data.frame(i3_Summary.1lay)
i3_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i3_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i3_Summary.1lay, file = "Tuning SUmmary, ANN, h=1, 2010-2015, Actual EUI Change, No CT, SL or Lat")
i3_Best.result.1lay = i3_Summary.1lay[which.max(i3_Summary.1lay$CV.Average),]
write.csv(i3_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, Actual EUI Change, No CT, SL or Lat")

# 2 Hidden Layers #

i3_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i3_Summary.2lay <- cbind(i3_results.2lay.1[,1:2],F1.Avg=i3_results.2lay.1[,4], F2.Avg=ri3_esults.2lay.2[,4], F2.Avg=i3_results.2lay.3[,4],
F4.Avg=i3_results.2lay.4[,4],
                         F5.Avg=i3_results.2lay.5[,4], F6.Avg=i3_results.2lay.6[,4], F7.Avg=i3_results.2lay.7[,4], F8.Avg=i3_results.2lay.8[,4],
                         F9.Avg=i3_results.2lay.9[,4], F10.Avg=i3_results.2lay.10[,4])
i3_Summary.2lay <- as.data.frame(i3_Summary.2lay)
i3_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i3_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi3_(i3_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, Actual EUI Change, No CT, SL or Lat")
i3_Best.result.2lay = i3_Summary.2lay[which.max(i3_Summary.2lay$CV.Average),]
write.csv(i3_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, Actual EUI Change, No CT, SL or Lat")

# 3 Hidden Layers #

i3_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i3_Summary.3lay <- cbind(i3_results.3lay.1[,1:3],F1.Avg=i3_results.3lay.1[,4], F2.Avg=i3_results.3lay.2[,4], F3.Avg=i3_results.3lay.3[,4],
F4.Avg=i3_results.3lay.4[,4],
                         F5.Avg=i3_results.3lay.5[,4], F6.Avg=i3_results.3lay.6[,4], F7.Avg=i3_results.3lay.7[,4], F8.Avg=i3_results.3lay.8[,4],
                         F9.Avg=i3_results.3lay.9[,4], F10.Avg=i3_results.3lay.10[,4])

i3_Summary.3lay <- as.data.frame(i3_Summary.3lay)
i3_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i3_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i3_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, Actual EUI Change, No CT, SL or Lat")
i3_Best.result.3lay = i3_Summary.3lay[which.max(i3_Summary.3lay$CV.Average),]
write.csv(i3_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, Actual EUI Change, No CT, SL or Lat")
```

```
#### i4 Define Model Varibles ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area", "Longitude", "Construction.Year",
                   "Class","Climate.Zone", "Electrically.Heated",
                   "X2010.EUI", "Occupant.Density",
                   "Vacancy.Rate", "Weekly.Operating.Hours")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15
#### i4 Training Model - No CT, SL, Lat, Str####
# 1 Hidden Layer #
i4_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i4_Summary.1lay <- cbind(Layer1=i4_results.1lay.1[,1],F1.Avg=i4_results.1lay.1[,4], F2.Avg=i4_results.1lay.2[,4], F1.Avg=i4_results.1lay.3[,4],
F4.Avg=i4_results.1lay.4[,4],
                         F5.Avg=i4_results.1lay.5[,4], F6.Avg=i4_results.1lay.6[,4], F7.Avg=i4_results.1lay.7[,4], F8.Avg=i4_results.1lay.8[,4],
                         F9.Avg=i4_results.1lay.9[,4], F10.Avg=i4_results.1lay.10[,4])
i4_Summary.1lay <- as.data.frame(i4_Summary.1lay)
i4_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i4_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i4_Summary.1lay, file = "Tuning Summary, ANN, h=1, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str")
i4_Best.result.1lay = i4_Summary.1lay[which.max(i4_Summary.1lay$CV.Average),]
write.csv(i4_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str")

# 2 Hidden Layers #

i4_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i4_Summary.2lay <- cbind(i4_results.2lay.1[,1:2],F1.Avg=i4_results.2lay.1[,4], F2.Avg=ri4_esults.2lay.2[,4], F2.Avg=i4_results.2lay.3[,4],
F4.Avg=i4_results.2lay.4[,4],
                         F5.Avg=i4_results.2lay.5[,4], F6.Avg=i4_results.2lay.6[,4], F7.Avg=i4_results.2lay.7[,4], F8.Avg=i4_results.2lay.8[,4],
                         F9.Avg=i4_results.2lay.9[,4], F10.Avg=i4_results.2lay.10[,4])
i4_Summary.2lay <- as.data.frame(i4_Summary.2lay)
i4_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i4_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi4_(i4_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str")
i4_Best.result.2lay = i4_Summary.2lay[which.max(i4_Summary.2lay$CV.Average),]
write.csv(i4_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str")

# 3 Hidden Layers #

i4_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i4_Summary.3lay <- cbind(i4_results.3lay.1[,1:3],F1.Avg=i4_results.3lay.1[,4], F2.Avg=i4_results.3lay.2[,4], F3.Avg=i4_results.3lay.3[,4],
F4.Avg=i4_results.3lay.4[,4],
                         F5.Avg=i4_results.3lay.5[,4], F6.Avg=i4_results.3lay.6[,4], F7.Avg=i4_results.3lay.7[,4], F8.Avg=i4_results.3lay.8[,4],
```

```
                        F9.Avg=i4_results.3lay.9[,4], F10.Avg=i4_results.3lay.10[,4])

i4_Summary.3lay <- as.data.frame(i4_Summary.3lay)
i4_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i4_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i4_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str")
i4_Best.result.3lay = i4_Summary.3lay[which.max(i4_Summary.3lay$CV.Average),]
write.csv(i4_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str")


#### i5 Define Model Varibles ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area", "Longitude", "Construction.Year",
                   "Class","Climate.Zone",
                   "X2010.EUI", "Occupant.Density",
                   "Vacancy.Rate", "Weekly.Operating.Hours")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15
#### i5 Training Model - No CT, SL, Lat, Str, EH####
# 1 Hidden Layer #
i5_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i5_Summary.1lay <- cbind(Layer1=i5_results.1lay.1[,1],F1.Avg=i5_results.1lay.1[,4], F2.Avg=i5_results.1lay.2[,4], F1.Avg=i5_results.1lay.3[,4],
F4.Avg=i5_results.1lay.4[,4],
                         F5.Avg=i5_results.1lay.5[,4], F6.Avg=i5_results.1lay.6[,4], F7.Avg=i5_results.1lay.7[,4], F8.Avg=i5_results.1lay.8[,4],
                         F9.Avg=i5_results.1lay.9[,4], F10.Avg=i5_results.1lay.10[,4])
i5_Summary.1lay <- as.data.frame(i5_Summary.1lay)
i5_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i5_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i5_Summary.1lay, file = "Tuning Summary, ANN, h=1, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str, EH")
i5_Best.result.1lay = i5_Summary.1lay[which.max(i5_Summary.1lay$CV.Average),]
write.csv(i5_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str, EH")

# 2 Hidden Layers #

i5_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i5_Summary.2lay <- cbind(i5_results.2lay.1[,1:2],F1.Avg=i5_results.2lay.1[,4], F2.Avg=ri5_esults.2lay.2[,4], F2.Avg=i5_results.2lay.3[,4],
F4.Avg=i5_results.2lay.4[,4],
                         F5.Avg=i5_results.2lay.5[,4], F6.Avg=i5_results.2lay.6[,4], F7.Avg=i5_results.2lay.7[,4], F8.Avg=i5_results.2lay.8[,4],
                         F9.Avg=i5_results.2lay.9[,4], F10.Avg=i5_results.2lay.10[,4])
i5_Summary.2lay <- as.data.frame(i5_Summary.2lay)
i5_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i5_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi5_(i5_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str, EH")
i5_Best.result.2lay = i5_Summary.2lay[which.max(i5_Summary.2lay$CV.Average),]
write.csv(i5_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str, EH")

# 3 Hidden Layers #

i5_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
```

```
i5_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i5_Summary.3lay <- cbind(i5_results.3lay.1[,1:3],F1.Avg=i5_results.3lay.1[,4], F2.Avg=i5_results.3lay.2[,4], F3.Avg=i5_results.3lay.3[,4],
F4.Avg=i5_results.3lay.4[,4],
                         F5.Avg=i5_results.3lay.5[,4], F6.Avg=i5_results.3lay.6[,4], F7.Avg=i5_results.3lay.7[,4], F8.Avg=i5_results.3lay.8[,4],
                         F9.Avg=i5_results.3lay.9[,4], F10.Avg=i5_results.3lay.10[,4])

i5_Summary.3lay <- as.data.frame(i5_Summary.3lay)
i5_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i5_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i5_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str, EH")
i5_Best.result.3lay = i5_Summary.3lay[which.max(i5_Summary.3lay$CV.Average),]
write.csv(i5_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str, EH")


#### i6 Define Model Varibles ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area", "Longitude", "Construction.Year",
                   "Class","Climate.Zone",
                   "X2010.EUI",
                   "Vacancy.Rate", "Weekly.Operating.Hours")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15
#### i6 Training Model - No CT, SL, Lat, Str, EH, OcDe####
# 1 Hidden Layer #
i6_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i6_Summary.1lay <- cbind(Layer1=i6_results.1lay.1[,1],F1.Avg=i6_results.1lay.1[,4], F2.Avg=i6_results.1lay.2[,4], F1.Avg=i6_results.1lay.3[,4],
F4.Avg=i6_results.1lay.4[,4],
                         F5.Avg=i6_results.1lay.5[,4], F6.Avg=i6_results.1lay.6[,4], F7.Avg=i6_results.1lay.7[,4], F8.Avg=i6_results.1lay.8[,4],
                         F9.Avg=i6_results.1lay.9[,4], F10.Avg=i6_results.1lay.10[,4])
i6_Summary.1lay <- as.data.frame(i6_Summary.1lay)
i6_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i6_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i6_Summary.1lay, file = "Tuning Summary, ANN, h=1, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str, EH, OcDen ")
i6_Best.result.1lay = i6_Summary.1lay[which.max(i6_Summary.1lay$CV.Average),]
write.csv(i6_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str, EH, OcDe")

# 2 Hidden Layers #

i6_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i6_Summary.2lay <- cbind(i6_results.2lay.1[,1:2],F1.Avg=i6_results.2lay.1[,4], F2.Avg=ri6_esults.2lay.2[,4], F2.Avg=i6_results.2lay.3[,4],
F4.Avg=i6_results.2lay.4[,4],
                         F5.Avg=i6_results.2lay.5[,4], F6.Avg=i6_results.2lay.6[,4], F7.Avg=i6_results.2lay.7[,4], F8.Avg=i6_results.2lay.8[,4],
                         F9.Avg=i6_results.2lay.9[,4], F10.Avg=i6_results.2lay.10[,4])
i6_Summary.2lay <- as.data.frame(i6_Summary.2lay)
i6_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i6_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi6_(i6_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str, EH, OcDe")
i6_Best.result.2lay = i6_Summary.2lay[which.max(i6_Summary.2lay$CV.Average),]
write.csv(i6_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str, EH, OcDe")

# 3 Hidden Layers #

i6_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
```

```
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i6_Summary.3lay <- cbind(i6_results.3lay.1[,1:3],F1.Avg=i6_results.3lay.1[,4], F2.Avg=i6_results.3lay.2[,4], F3.Avg=i6_results.3lay.3[,4],
F4.Avg=i6_results.3lay.4[,4],
                         F5.Avg=i6_results.3lay.5[,4], F6.Avg=i6_results.3lay.6[,4], F7.Avg=i6_results.3lay.7[,4], F8.Avg=i6_results.3lay.8[,4],
                         F9.Avg=i6_results.3lay.9[,4], F10.Avg=i6_results.3lay.10[,4])

i6_Summary.3lay <- as.data.frame(i6_Summary.3lay)
i6_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i6_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i6_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str, EH, OcDe")
i6_Best.result.3lay = i6_Summary.3lay[which.max(i6_Summary.3lay$CV.Average),]
write.csv(i6_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, Actual EUI Change, No CT, SL, Lat, Str, EH, OcDe")


#### i7 Define Model Varibles ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area", "Longitude", "X2010.EUI")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15
#### i7 Training Model - only 4 variables####
# 1 Hidden Layer #
i7_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i7_Summary.1lay <- cbind(Layer1=i7_results.1lay.1[,1],F1.Avg=i7_results.1lay.1[,4], F2.Avg=i7_results.1lay.2[,4], F1.Avg=i7_results.1lay.3[,4],
F4.Avg=i7_results.1lay.4[,4],
                         F5.Avg=i7_results.1lay.5[,4], F6.Avg=i7_results.1lay.6[,4], F7.Avg=i7_results.1lay.7[,4], F8.Avg=i7_results.1lay.8[,4],
                         F9.Avg=i7_results.1lay.9[,4], F10.Avg=i7_results.1lay.10[,4])
i7_Summary.1lay <- as.data.frame(i7_Summary.1lay)
i7_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i7_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i7_Summary.1lay, file = "Tuning Summary, ANN, h=1, 2010-2015, Actual EUI Change, only 4 variablen ")
i7_Best.result.1lay = i7_Summary.1lay[which.max(i7_Summary.1lay$CV.Average),]
write.csv(i7_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, Actual EUI Change, only 4 variable")

# 2 Hidden Layers #

i7_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i7_Summary.2lay <- cbind(i7_results.2lay.1[,1:2],F1.Avg=i7_results.2lay.1[,4], F2.Avg=ri7_esults.2lay.2[,4], F2.Avg=i7_results.2lay.3[,4],
F4.Avg=i7_results.2lay.4[,4],
                         F5.Avg=i7_results.2lay.5[,4], F6.Avg=i7_results.2lay.6[,4], F7.Avg=i7_results.2lay.7[,4], F8.Avg=i7_results.2lay.8[,4],
                         F9.Avg=i7_results.2lay.9[,4], F10.Avg=i7_results.2lay.10[,4])
i7_Summary.2lay <- as.data.frame(i7_Summary.2lay)
i7_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i7_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi7_(i7_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, Actual EUI Change, only 4 variable")
i7_Best.result.2lay = i7_Summary.2lay[which.max(i7_Summary.2lay$CV.Average),]
write.csv(i7_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, Actual EUI Change, only 4 variable")

# 3 Hidden Layers #

i7_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
```

```
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i7_Summary.3lay <- cbind(i7_results.3lay.1[,1:3],F1.Avg=i7_results.3lay.1[,4], F2.Avg=i7_results.3lay.2[,4], F3.Avg=i7_results.3lay.3[,4],
F4.Avg=i7_results.3lay.4[,4],
                         F5.Avg=i7_results.3lay.5[,4], F6.Avg=i7_results.3lay.6[,4], F7.Avg=i7_results.3lay.7[,4], F8.Avg=i7_results.3lay.8[,4],
                         F9.Avg=i7_results.3lay.9[,4], F10.Avg=i7_results.3lay.10[,4])

i7_Summary.3lay <- as.data.frame(i7_Summary.3lay)
i7_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i7_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i7_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, Actual EUI Change, only 4 variable")
i7_Best.result.3lay = i7_Summary.3lay[which.max(i7_Summary.3lay$CV.Average),]
write.csv(i7_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, Actual EUI Change, only 4 variable")
```

```
#### Load Packages ####
library(neuralnet)
library("beepr")
library(ggplot2)
#library(lattice)
library(caret)
#library(boot)
library(plyr)
#library(withr)
set.seed(1234)
#### Load Dataset ####
#setwd("C:/Users/carleen.lawson/Documents/Data/2010-2015 Data/Cross Validation Sets")
data <- read.csv(file.choose(), row.names = 1, check.names = FALSE, na.strings = "NA", sep = ",", stringsAsFactors = TRUE, header = TRUE)
#### Normalize Dataset #####
cnames <- cbind("Delta.EUI","Portfolio.Manager","Exterior.Area","Lattitude","Longitude","Construction.Year","Num.of.Structures",
                "Class","Climate.Zone","Electrically.Heated","Cooling.Tower","Soft.Landscaping","X2010.EUI",
                "Occupant.Density","Vacancy.Rate","Weekly.Operating.Hours")
normalize <- function(x){(x-min(x))/(max(x)-min(x))}
normalize.delta <- function(x){(x-(-100))/(100-(-100))}
rnames <- row.names(data)
norm.indepvar<- as.data.frame(lapply(data[,-1], normalize))
norm.depvar <- as.data.frame(normalize.delta(data[,1]))
norm.data <-cbind(norm.depvar, norm.indepvar)
Delta.EUI <- norm.data$norm.depvar
row.names(norm.data) <- rnames
colnames(norm.data) <- cnames

#### Subset Data ####
#print.rows <- function(x){cat(((toString(shQuote((row.names.data.frame(x)), type = "cmd")))), "\n")}

r.test1 <-c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524")
r.test2 <- c("215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302",
"138" )
r.test3 <- c("221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293")
r.test4 <- c("272", "249", "202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175",
"582")
r.test5 <- c("224", "565", "250", "559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543",
"148" )
r.test6 <- c("637", "212", "210", "518", "645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523",
"97" )
r.test7 <- c("126", "508", "475", "513", "965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195",
"297" )
r.test8 <- c("509", "650", "86", "251", "663", "287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194" )
r.test9 <- c("96", "636", "122", "101", "83", "143", "491", "107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127" )
r.test10 <- c("560", "93", "357", "211", "109", "273", "488", "640", "85", "81", "643", "110", "112", "128", "562", "652", "579", "84", "526")

r.train1 <-c("215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302",
"138", "221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272",
"249", "202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565",
"250", "559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210",
"518", "645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513",
"965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663",
"287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491",
"107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640", "85",
"81", "643", "110", "112", "128", "562", "652", "579", "84", "526")
r.train2 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272",
"249", "202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565",
"250", "559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210",
"518", "645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513",
"965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663",
"287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491",
"107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640", "85",
"81", "643", "110", "112", "128", "562", "652", "579", "84", "526" )
r.train3 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"272", "249", "202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224",
"565", "250", "559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212",
"210", "518", "645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475",
"513", "965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251",
"663", "287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143",
"491", "107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640",
"85", "81", "643", "110", "112", "128", "562", "652", "579", "84", "526")
r.train4 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "224", "565",
"250", "559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210",
"518", "645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513",
"965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663",
"287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491",
"107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640", "85",
"81", "643", "110", "112", "128", "562", "652", "579", "84", "526")
r.train5 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272", "249",
"202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "637", "212", "210",
"518", "645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513",
"965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663",
"287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491",
"107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640", "85",
"81", "643", "110", "112", "128", "562", "652", "579", "84", "526" )
r.train6 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272", "249",
"202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565", "250",
"559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "126", "508", "475", "513",
"965", "214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663",
"287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491",
"107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640", "85",
"81", "643", "110", "112", "128", "562", "652", "579", "84", "526" )
r.train7 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272", "249",
"202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565", "250",
"559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210", "518",
```

```
"645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "509", "650", "86", "251", "663",
"287", "190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491",
"107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640", "85",
"81", "643", "110", "112", "128", "562", "652", "579", "84", "526" )
r.train8 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272", "249",
"202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565", "250",
"559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210", "518",
"645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513", "965",
"214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "96", "636", "122", "101", "83", "143",
"491", "107", "116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127", "560", "93", "357", "211", "109", "273", "488", "640",
"85", "81", "643", "110", "112", "128", "562", "652", "579", "84", "526")
r.train9 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272", "249",
"202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565", "250",
"559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210", "518",
"645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513", "965",
"214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663", "287",
"190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "560", "93", "357", "211", "109", "273", "488", "640",
"85", "81", "643", "110", "112", "128", "562", "652", "579", "84", "526")
r.train10 <- c("673", "479", "699", "686", "647", "144", "653", "470", "150", "203", "468", "503", "471", "655", "149", "94", "198", "103", "675",
"524", "215", "766", "649", "547", "682", "641", "355", "125", "695", "188", "651", "246", "294", "100", "121", "485", "146", "639", "302", "138",
"221", "507", "153", "92", "222", "98", "356", "124", "516", "408", "12", "477", "87", "472", "659", "668", "82", "74", "296", "293", "272", "249",
"202", "490", "247", "1286", "270", "102", "861", "535", "667", "223", "689", "216", "129", "514", "644", "300", "175", "582", "224", "565", "250",
"559", "90", "131", "177", "147", "662", "664", "220", "487", "495", "635", "137", "311", "280", "119", "543", "148", "637", "212", "210", "518",
"645", "648", "506", "269", "406", "80", "1282", "201", "289", "115", "155", "638", "196", "306", "523", "97", "126", "508", "475", "513", "965",
"214", "225", "253", "180", "654", "213", "181", "111", "254", "132", "415", "78", "481", "195", "297", "509", "650", "86", "251", "663", "287",
"190", "517", "130", "88", "271", "186", "151", "163", "252", "304", "291", "295", "194", "96", "636", "122", "101", "83", "143", "491", "107",
"116", "569", "486", "77", "671", "512", "298", "440", "248", "658", "127")

data.test.1 <- norm.data[r.test1,]
data.test.2 <- norm.data[r.test2,]
data.test.3 <- norm.data[r.test3,]
data.test.4 <- norm.data[r.test4,]
data.test.5 <- norm.data[r.test5,]
data.test.6 <- norm.data[r.test6,]
data.test.7 <- norm.data[r.test7,]
data.test.8 <- norm.data[r.test8,]
data.test.9 <- norm.data[r.test9,]
data.test.10 <- norm.data[r.test10,]

data.train.1 <- norm.data[r.train1,]
data.train.2 <- norm.data[r.train2,]
data.train.3 <- norm.data[r.train3,]
data.train.4 <- norm.data[r.train4,]
data.train.5 <- norm.data[r.train5,]
data.train.6 <- norm.data[r.train6,]
data.train.7 <- norm.data[r.train7,]
data.train.8 <- norm.data[r.train8,]
data.train.9 <- norm.data[r.train9,]
data.train.10 <- norm.data[r.train10,]



#### Functions ####
#descales the dependent variable, using a max of (100) and a min of (-100).
denormalize.delta <- function(x){(x*200)-100}

nn.accuracy <- function(nn, test.x, test.y, rep, ind.variables) {
  scaled.pr.nn <- (compute(nn,test.x[,ind.variables], ncol(nn$result.matrix)))$net.result
  Predicted.Delta.EUI <- round(denormalize.delta(scaled.pr.nn),0)
  Actual.Delta.EUI <- round(denormalize.delta(test.y),0)
  Residuals <- Actual.Delta.EUI-Predicted.Delta.EUI
  Compare <- as.data.frame(cbind(Actual.Delta.EUI,Predicted.Delta.EUI,Residuals))
  colnames(Compare)<-c('Actual.Delta.EUI','Predicted.Delta.EUI','Residuals')
  Compare$Actual.Class = ifelse((Compare$Actual.Delta.EUI <= 0 ),"Poor",
                                ifelse((Compare$Actual.Delta.EUI > 0&Compare$Actual.Delta.EUI <25),"Fair",
                                       ifelse((Compare$Actual.Delta.EUI >=25),"Good",0)))
  Compare$Predicted.Class = ifelse((Compare$Predicted.Delta.EUI <= 0 ),"Poor",
                                   ifelse((Compare$Predicted.Delta.EUI > 0 & Compare$Predicted.Delta.EUI <25),"Fair",
                                          ifelse((Compare$Predicted.Delta.EUI >=25),"Good",0)))
  Compare$Correct = ifelse((Compare$Actual.Delta.EUI <= 0 & Compare$Predicted.Delta.EUI <= 0 ),1,
                           ifelse((Compare$Actual.Delta.EUI > 0&Compare$Actual.Delta.EUI <25 & Compare$Predicted.Delta.EUI > 0 &
Compare$Predicted.Delta.EUI <25),1,
                                  ifelse((Compare$Actual.Delta.EUI >25 & Compare$Predicted.Delta.EUI >25),1,0)))
  Class.Accuracy<- round((sum(Compare$Correct))/(nrow(Compare))*100,2)
  Results=list("Class.Accuracy" = Class.Accuracy, "Compare.Results" = Compare)
  return(Results)
}

nn.benchmark.1layers<- function(formula, train, test.x, test.y, min.nodes.1layer, max.nodes.1layer, threshold, reps,ind.variables) {
  nrows <- ((max.nodes.1layer-min.nodes.1layer)+1)

  benchmark.result <- matrix(ncol=4, nrow=nrows)
  result.row <- 1

  nn.results <- matrix(nrow=1, ncol=reps)


  for (i in min.nodes.1layer:max.nodes.1layer) {
    print(paste("running the network with", i, "node(s) on the hidden layer"))
    try({
      nn.tested <- neuralnet(formula=formula, train, hidden=c(i), linear.output=TRUE, threshold=threshold, rep=reps, stepmax = 10000)
      for (k in 1:reps){
        nn.results[1,k] <- (nn.accuracy(nn.tested, test.x, test.y, ncol(nn.tested$result.matrix), ind.variables))$Class.Accuracy
      }

      benchmark.result[result.row,]<- c(i, mean(nn.results[1,]), max(nn.results[1,]), min(nn.results[1,]))
    })
    result.row <- result.row+1
  }

  benchmark.result <- as.data.frame(benchmark.result)
  colnames(benchmark.result) <- c("Layer 1", "Average", "Max", "Min")
```

```
  Results = list("benchmark.result" = benchmark.result, "best.results" = benchmark.result[which.max(benchmark.result$Average),])
  return(Results)
}
nn.benchmark.2layers<- function(formula, train, test.x, test.y, min.nodes.1layer, max.nodes.1layer, min.nodes.2layer, max.nodes.2layer, threshold,
reps, ind.variables) {
  nrows <- ((max.nodes.1layer-min.nodes.1layer)+1)*((max.nodes.2layer-min.nodes.2layer)+1)

  benchmark.result <- matrix(ncol=5, nrow=nrows)
  result.row <- 1

  nn.results <- matrix(nrow=1, ncol=reps)

  for (i in min.nodes.1layer:max.nodes.1layer) {
    for (j in min.nodes.2layer:max.nodes.2layer) {
      print(paste("running the network with", i, "node(s) on the 1st hidden layer and", j, "node(s) on the 2nd"))
      try({
        nn.tested <- neuralnet(formula=formula, train, hidden=c(i,j), linear.output=TRUE, threshold=threshold, rep=reps, stepmax = 10000)
        for (k in 1:reps){
          nn.results[1,k] <- (nn.accuracy(nn.tested, test.x, test.y, ncol(nn.tested$result.matrix), ind.variables))$Class.Accuracy
        }

        benchmark.result[result.row,]<- c(i, j, mean(nn.results[1,]), max(nn.results[1,]), min(nn.results[1,]))
      })
      result.row <- result.row+1
    }
  }

  benchmark.result <- as.data.frame(benchmark.result)
  colnames(benchmark.result) <- c("Layer 1", "Layer 2", "Average", "Max", "Min")
  Results = list("benchmark.result" = benchmark.result, "best.results" = benchmark.result[which.max(benchmark.result$Average),])
  return(Results)
}
nn.benchmark.3layers<- function(formula, train, test.x, test.y, min.nodes.1layer, max.nodes.1layer, min.nodes.2layer, max.nodes.2layer,
min.nodes.3layer, max.nodes.3layer, threshold, reps, ind.variables) {
  nrows <- ((max.nodes.1layer-min.nodes.1layer)+1)*((max.nodes.2layer-min.nodes.2layer)+1)*((max.nodes.3layer-min.nodes.3layer)+1)

  benchmark.result <- matrix(ncol=6, nrow=nrows)
  result.row <- 1

  nn.results <- matrix(nrow=1, ncol=reps)

  for (i in min.nodes.1layer:max.nodes.1layer) {
    for (j in min.nodes.2layer:max.nodes.2layer) {
      for (k in min.nodes.3layer:max.nodes.3layer) {
        print(paste("Running the network with", i, "nodes on the 1st layer and", j, "nodes on the 2nd and", k, "nodes on the 3rd"))
        try({
          nn.tested <- neuralnet(formula=formula, train, hidden=c(i,j,k), linear.output=TRUE, threshold=threshold, rep=reps, stepmax = 10000)
          for (r in 1:reps){
            nn.results[1,r] <- (nn.accuracy(nn.tested, test.x, test.y, ncol(nn.tested$result.matrix), ind.variables))$Class.Accuracy
          }

          benchmark.result[result.row,]<- c(i, j,k, mean(nn.results[1,]), max(nn.results[1,]), min(nn.results[1,]))
        })
        result.row <- result.row+1
      }
    }
  }

  benchmark.result <- as.data.frame(benchmark.result)
  colnames(benchmark.result) <- c("Layer 1", "Layer 2","Layer 3", "Average", "Max", "Min")
  Results = list("benchmark.result" = benchmark.result, "best.results" = benchmark.result[which.max(benchmark.result$Average),])
  return(Results)
}

nn.best.1layer<- function(formula, train, threshold, reps, Best.result.1lay) {

  i=Best.result.1lay[,1]
  nn.tested <- neuralnet(formula=formula, train, hidden=c(i), linear.output=TRUE, threshold=threshold, rep=reps, stepmax = 10000)
  return("nn.model"=nn.tested)
}

nn.best.2layers<- function(formula, train, threshold, reps, Best.result.2lay) {

  i=Best.result.2lay[,1]
  j=Best.result.2lay[,2]
  nn.tested <- neuralnet(formula=formula, train, hidden=c(i,j), linear.output=TRUE, threshold=threshold, rep=reps, stepmax = 10000)
  return("nn.model"=nn.tested)
}

nn.best.3layers<- function(formula, train, threshold, reps, Best.result.3lay) {

  i=Best.result.3lay[,1]
  j=Best.result.3lay[,2]
  k=Best.result.3lay[,3]
  nn.tested <- neuralnet(formula=formula, train, hidden=c(i,j,k), linear.output=TRUE, threshold=threshold, rep=reps, stepmax = 10000)
  return("nn.model"=nn.tested)
}


#### i1 Define Model Varibles- All Variables ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area", "Lattitude", "Longitude", "Construction.Year",
                   "Num.of.Structures", "Class", "Climate.Zone", "Electrically.Heated",
                   "Cooling.Tower", "Soft.Landscaping", "X2010.EUI", "Occupant.Density",
                   "Vacancy.Rate", "Weekly.Operating.Hours")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15

#### i1 Training Model - All Variables####
# 1 Hidden Layer #
i1_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
```

```
i1_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i1_Summary.1lay <- cbind(Layer1=i1_results.1lay.1[,1],F1.Avg=i1_results.1lay.1[,4], F2.Avg=i1_results.1lay.2[,4], F1.Avg=i1_results.1lay.3[,4],
F4.Avg=i1_results.1lay.4[,4],
                      F5.Avg=i1_results.1lay.5[,4], F6.Avg=i1_results.1lay.6[,4], F7.Avg=i1_results.1lay.7[,4], F8.Avg=i1_results.1lay.8[,4],
                      F9.Avg=i1_results.1lay.9[,4], F10.Avg=i1_results.1lay.10[,4])
i1_Summary.1lay <- as.data.frame(i1_Summary.1lay)
i1_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i1_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i1_Summary.1lay, file = "Tuning SUmmary, ANN, h=1, 2010-2015, % Change, All Variables")
i1_Best.result.1lay = i1_Summary.1lay[which.max(i1_Summary.1lay$CV.Average),]
write.csv(i1_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, % Change, All Variables")

# 2 Hidden Layers #

i1_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i1_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i1_Summary.2lay <- cbind(i1_results.2lay.1[,1:2],F1.Avg=i1_results.2lay.1[,4], F2.Avg=ri1_esults.2lay.2[,4], F2.Avg=i1_results.2lay.3[,4],
F4.Avg=i1_results.2lay.4[,4],
                      F5.Avg=i1_results.2lay.5[,4], F6.Avg=i1_results.2lay.6[,4], F7.Avg=i1_results.2lay.7[,4], F8.Avg=i1_results.2lay.8[,4],
                      F9.Avg=i1_results.2lay.9[,4], F10.Avg=i1_results.2lay.10[,4])
i1_Summary.2lay <- as.data.frame(i1_Summary.2lay)
i1_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i1_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi1_(i1_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, % Change, All Variables")
i1_Best.result.2lay = i1_Summary.2lay[which.max(i1_Summary.2lay$CV.Average),]
write.csv(i1_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, % Change, All Variables")

# 3 Hidden Layers #

i1_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i1_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i1_Summary.3lay <- cbind(i1_results.3lay.1[,1:3],F1.Avg=i1_results.3lay.1[,4], F2.Avg=i1_results.3lay.2[,4], F3.Avg=i1_results.3lay.3[,4],
F4.Avg=i1_results.3lay.4[,4],
                      F5.Avg=i1_results.3lay.5[,4], F6.Avg=i1_results.3lay.6[,4], F7.Avg=i1_results.3lay.7[,4], F8.Avg=i1_results.3lay.8[,4],
                      F9.Avg=i1_results.3lay.9[,4], F10.Avg=i1_results.3lay.10[,4])

i1_Summary.3lay <- as.data.frame(i1_Summary.3lay)
i1_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i1_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i1_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, % Change, All Variables")
i1_Best.result.3lay = i1_Summary.3lay[which.max(i1_Summary.3lay$CV.Average),]
write.csv(i1_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, % Change, All Variables")

#### i2 Define Model Varibles ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area","Lattitude", "Longitude", "Construction.Year",
                   "Num.of.Structures","Class","Climate.Zone", "Electrically.Heated",
                   "X2010.EUI", "Occupant.Density",
                   "Vacancy.Rate", "Weekly.Operating.Hours")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15
```

```
#### i2 Training Model - No CT, SL####
# 1 Hidden Layer #
i2_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i2_Summary.1lay <- cbind(Layer1=i2_results.1lay.1[,1],F1.Avg=i2_results.1lay.1[,4], F2.Avg=i2_results.1lay.2[,4], F1.Avg=i2_results.1lay.3[,4],
F4.Avg=i2_results.1lay.4[,4],
                         F5.Avg=i2_results.1lay.5[,4], F6.Avg=i2_results.1lay.6[,4], F7.Avg=i2_results.1lay.7[,4], F8.Avg=i2_results.1lay.8[,4],
                         F9.Avg=i2_results.1lay.9[,4], F10.Avg=i2_results.1lay.10[,4])
i2_Summary.1lay <- as.data.frame(i2_Summary.1lay)
i2_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i2_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i2_Summary.1lay, file = "Tuning SUmmary, ANN, h=1, 2010-2015, % Change, No CT, SL")
i2_Best.result.1lay = i2_Summary.1lay[which.max(i2_Summary.1lay$CV.Average),]
write.csv(i2_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, % Change, No CT, SL")

# 2 Hidden Layers #

i2_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i2_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i2_Summary.2lay <- cbind(i2_results.2lay.1[,1:2],F1.Avg=i2_results.2lay.1[,4], F2.Avg=ri2_esults.2lay.2[,4], F2.Avg=i2_results.2lay.3[,4],
F4.Avg=i2_results.2lay.4[,4],
                         F5.Avg=i2_results.2lay.5[,4], F6.Avg=i2_results.2lay.6[,4], F7.Avg=i2_results.2lay.7[,4], F8.Avg=i2_results.2lay.8[,4],
                         F9.Avg=i2_results.2lay.9[,4], F10.Avg=i2_results.2lay.10[,4])
i2_Summary.2lay <- as.data.frame(i2_Summary.2lay)
i2_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i2_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi2_(i2_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, % Change, No CT, SL or Lat")
i2_Best.result.2lay = i2_Summary.2lay[which.max(i2_Summary.2lay$CV.Average),]
write.csv(i2_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, % Change, No CT, SL or Lat")

# 3 Hidden Layers #

i2_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i2_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i2_Summary.3lay <- cbind(i2_results.3lay.1[,1:3],F1.Avg=i2_results.3lay.1[,4], F2.Avg=i2_results.3lay.2[,4], F3.Avg=i2_results.3lay.3[,4],
F4.Avg=i2_results.3lay.4[,4],
                         F5.Avg=i2_results.3lay.5[,4], F6.Avg=i2_results.3lay.6[,4], F7.Avg=i2_results.3lay.7[,4], F8.Avg=i2_results.3lay.8[,4],
                         F9.Avg=i2_results.3lay.9[,4], F10.Avg=i2_results.3lay.10[,4])

i2_Summary.3lay <- as.data.frame(i2_Summary.3lay)
i2_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i2_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i2_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, % Change, No CT, SL or Lat")
i2_Best.result.3lay = i2_Summary.3lay[which.max(i2_Summary.3lay$CV.Average),]
write.csv(i2_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, % Change, No CT, SL or Lat")


#### i3 Define Model Varibles ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area", "Longitude", "Construction.Year",
```

```
                  "Num.of.Structures","Class","Climate.Zone", "Electrically.Heated",
                  "X2010.EUI", "Occupant.Density",
                  "Vacancy.Rate", "Weekly.Operating.Hours")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15
#### i3 Training Model - No CT, SL or Lat####
# 1 Hidden Layer #
i3_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i3_Summary.1lay <- cbind(Layer1=i3_results.1lay.1[,1],F1.Avg=i3_results.1lay.1[,4], F2.Avg=i3_results.1lay.2[,4], F1.Avg=i3_results.1lay.3[,4],
F4.Avg=i3_results.1lay.4[,4],
                         F5.Avg=i3_results.1lay.5[,4], F6.Avg=i3_results.1lay.6[,4], F7.Avg=i3_results.1lay.7[,4], F8.Avg=i3_results.1lay.8[,4],
                         F9.Avg=i3_results.1lay.9[,4], F10.Avg=i3_results.1lay.10[,4])
i3_Summary.1lay <- as.data.frame(i3_Summary.1lay)
i3_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i3_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i3_Summary.1lay, file = "Tuning SUmmary, ANN, h=1, 2010-2015, % Change, No CT, SL or Lat")
i3_Best.result.1lay = i3_Summary.1lay[which.max(i3_Summary.1lay$CV.Average),]
write.csv(i3_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, % Change, No CT, SL or Lat")

# 2 Hidden Layers #

i3_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i3_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i3_Summary.2lay <- cbind(i3_results.2lay.1[,1:2],F1.Avg=i3_results.2lay.1[,4], F2.Avg=ri3_esults.2lay.2[,4], F2.Avg=i3_results.2lay.3[,4],
F4.Avg=i3_results.2lay.4[,4],
                         F5.Avg=i3_results.2lay.5[,4], F6.Avg=i3_results.2lay.6[,4], F7.Avg=i3_results.2lay.7[,4], F8.Avg=i3_results.2lay.8[,4],
                         F9.Avg=i3_results.2lay.9[,4], F10.Avg=i3_results.2lay.10[,4])
i3_Summary.2lay <- as.data.frame(i3_Summary.2lay)
i3_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i3_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi3_(i3_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, % Change, No CT, SL or Lat")
i3_Best.result.2lay = i3_Summary.2lay[which.max(i3_Summary.2lay$CV.Average),]
write.csv(i3_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, % Change, No CT, SL or Lat")

# 3 Hidden Layers #

i3_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i3_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i3_Summary.3lay <- cbind(i3_results.3lay.1[,1:3],F1.Avg=i3_results.3lay.1[,4], F2.Avg=i3_results.3lay.2[,4], F3.Avg=i3_results.3lay.3[,4],
F4.Avg=i3_results.3lay.4[,4],
                         F5.Avg=i3_results.3lay.5[,4], F6.Avg=i3_results.3lay.6[,4], F7.Avg=i3_results.3lay.7[,4], F8.Avg=i3_results.3lay.8[,4],
                         F9.Avg=i3_results.3lay.9[,4], F10.Avg=i3_results.3lay.10[,4])

i3_Summary.3lay <- as.data.frame(i3_Summary.3lay)
i3_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i3_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i3_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, % Change, No CT, SL or Lat")
```

```
i3_Best.result.3lay = i3_Summary.3lay[which.max(i3_Summary.3lay$CV.Average),]
write.csv(i3_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, % Change, No CT, SL or Lat")


#### i4 Define Model Varibles ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area", "Longitude", "Construction.Year",
                   "Class","Climate.Zone", "Electrically.Heated",
                   "X2010.EUI", "Occupant.Density",
                   "Vacancy.Rate", "Weekly.Operating.Hours")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15
#### i4 Training Model - No CT, SL, Lat, Str####
# 1 Hidden Layer #
i4_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i4_Summary.1lay <- cbind(Layer1=i4_results.1lay.1[,1],F1.Avg=i4_results.1lay.1[,4], F2.Avg=i4_results.1lay.2[,4], F1.Avg=i4_results.1lay.3[,4],
F4.Avg=i4_results.1lay.4[,4],
                         F5.Avg=i4_results.1lay.5[,4], F6.Avg=i4_results.1lay.6[,4], F7.Avg=i4_results.1lay.7[,4], F8.Avg=i4_results.1lay.8[,4],
                         F9.Avg=i4_results.1lay.9[,4], F10.Avg=i4_results.1lay.10[,4])
i4_Summary.1lay <- as.data.frame(i4_Summary.1lay)
i4_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i4_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i4_Summary.1lay, file = "Tuning Summary, ANN, h=1, 2010-2015, % Change, No CT, SL, Lat, Str")
i4_Best.result.1lay = i4_Summary.1lay[which.max(i4_Summary.1lay$CV.Average),]
write.csv(i4_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, % Change, No CT, SL, Lat, Str")

# 2 Hidden Layers #

i4_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i4_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i4_Summary.2lay <- cbind(i4_results.2lay.1[,1:2],F1.Avg=i4_results.2lay.1[,4], F2.Avg=ri4_esults.2lay.2[,4], F2.Avg=i4_results.2lay.3[,4],
F4.Avg=i4_results.2lay.4[,4],
                         F5.Avg=i4_results.2lay.5[,4], F6.Avg=i4_results.2lay.6[,4], F7.Avg=i4_results.2lay.7[,4], F8.Avg=i4_results.2lay.8[,4],
                         F9.Avg=i4_results.2lay.9[,4], F10.Avg=i4_results.2lay.10[,4])
i4_Summary.2lay <- as.data.frame(i4_Summary.2lay)
i4_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i4_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi4_(i4_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, % Change, No CT, SL, Lat, Str")
i4_Best.result.2lay = i4_Summary.2lay[which.max(i4_Summary.2lay$CV.Average),]
write.csv(i4_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, % Change, No CT, SL, Lat, Str")

# 3 Hidden Layers #

i4_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i4_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i4_Summary.3lay <- cbind(i4_results.3lay.1[,1:3],F1.Avg=i4_results.3lay.1[,4], F2.Avg=i4_results.3lay.2[,4], F3.Avg=i4_results.3lay.3[,4],
```

```
F4.Avg=i4_results.3lay.4[,4],
                        F5.Avg=i4_results.3lay.5[,4], F6.Avg=i4_results.3lay.6[,4], F7.Avg=i4_results.3lay.7[,4], F8.Avg=i4_results.3lay.8[,4],
                        F9.Avg=i4_results.3lay.9[,4], F10.Avg=i4_results.3lay.10[,4])

i4_Summary.3lay <- as.data.frame(i4_Summary.3lay)
i4_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i4_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i4_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, % Change, No CT, SL, Lat, Str")
i4_Best.result.3lay = i4_Summary.3lay[which.max(i4_Summary.3lay$CV.Average),]
write.csv(i4_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, % Change, No CT, SL, Lat, Str")


#### i5 Define Model Varibles ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area", "Longitude", "Construction.Year",
                   "Class","Climate.Zone",
                   "X2010.EUI", "Occupant.Density",
                   "Vacancy.Rate", "Weekly.Operating.Hours")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15
#### i5 Training Model - No CT, SL, Lat, Str, EH####
# 1 Hidden Layer #
i5_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i5_Summary.1lay <- cbind(Layer1=i5_results.1lay.1[,1],F1.Avg=i5_results.1lay.1[,4], F2.Avg=i5_results.1lay.2[,4], F1.Avg=i5_results.1lay.3[,4],
F4.Avg=i5_results.1lay.4[,4],
                        F5.Avg=i5_results.1lay.5[,4], F6.Avg=i5_results.1lay.6[,4], F7.Avg=i5_results.1lay.7[,4], F8.Avg=i5_results.1lay.8[,4],
                        F9.Avg=i5_results.1lay.9[,4], F10.Avg=i5_results.1lay.10[,4])
i5_Summary.1lay <- as.data.frame(i5_Summary.1lay)
i5_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i5_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i5_Summary.1lay, file = "Tuning Summary, ANN, h=1, 2010-2015, % Change, No CT, SL, Lat, Str, EH")
i5_Best.result.1lay = i5_Summary.1lay[which.max(i5_Summary.1lay$CV.Average),]
write.csv(i5_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, % Change, No CT, SL, Lat, Str, EH")

# 2 Hidden Layers #

i5_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i5_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i5_Summary.2lay <- cbind(i5_results.2lay.1[,1:2],F1.Avg=i5_results.2lay.1[,4], F2.Avg=ri5_esults.2lay.2[,4], F2.Avg=i5_results.2lay.3[,4],
F4.Avg=i5_results.2lay.4[,4],
                        F5.Avg=i5_results.2lay.5[,4], F6.Avg=i5_results.2lay.6[,4], F7.Avg=i5_results.2lay.7[,4], F8.Avg=i5_results.2lay.8[,4],
                        F9.Avg=i5_results.2lay.9[,4], F10.Avg=i5_results.2lay.10[,4])
i5_Summary.2lay <- as.data.frame(i5_Summary.2lay)
i5_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i5_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi5_(i5_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, % Change, No CT, SL, Lat, Str, EH")
i5_Best.result.2lay = i5_Summary.2lay[which.max(i5_Summary.2lay$CV.Average),]
write.csv(i5_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, % Change, No CT, SL, Lat, Str, EH")

# 3 Hidden Layers #

i5_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
```

```
i5_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i5_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i5_Summary.3lay <- cbind(i5_results.3lay.1[,1:3],F1.Avg=i5_results.3lay.1[,4], F2.Avg=i5_results.3lay.2[,4], F3.Avg=i5_results.3lay.3[,4],
F4.Avg=i5_results.3lay.4[,4],
                         F5.Avg=i5_results.3lay.5[,4], F6.Avg=i5_results.3lay.6[,4], F7.Avg=i5_results.3lay.7[,4], F8.Avg=i5_results.3lay.8[,4],
                         F9.Avg=i5_results.3lay.9[,4], F10.Avg=i5_results.3lay.10[,4])

i5_Summary.3lay <- as.data.frame(i5_Summary.3lay)
i5_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i5_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i5_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, % Change, No CT, SL, Lat, Str, EH")
i5_Best.result.3lay = i5_Summary.3lay[which.max(i5_Summary.3lay$CV.Average),]
write.csv(i5_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, % Change, No CT, SL, Lat, Str, EH")


#### i6 Define Model Varibles ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area", "Longitude", "Construction.Year",
                   "Class","Climate.Zone",
                   "X2010.EUI",
                   "Vacancy.Rate", "Weekly.Operating.Hours")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15
#### i6 Training Model - No CT, SL, Lat, Str, EH, OcDe####
# 1 Hidden Layer #
i6_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i6_Summary.1lay <- cbind(Layer1=i6_results.1lay.1[,1],F1.Avg=i6_results.1lay.1[,4], F2.Avg=i6_results.1lay.2[,4], F1.Avg=i6_results.1lay.3[,4],
F4.Avg=i6_results.1lay.4[,4],
                         F5.Avg=i6_results.1lay.5[,4], F6.Avg=i6_results.1lay.6[,4], F7.Avg=i6_results.1lay.7[,4], F8.Avg=i6_results.1lay.8[,4],
                         F9.Avg=i6_results.1lay.9[,4], F10.Avg=i6_results.1lay.10[,4])
i6_Summary.1lay <- as.data.frame(i6_Summary.1lay)
i6_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i6_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i6_Summary.1lay, file = "Tuning Summary, ANN, h=1, 2010-2015, % Change, No CT, SL, Lat, Str, EH, OcDen ")
i6_Best.result.1lay = i6_Summary.1lay[which.max(i6_Summary.1lay$CV.Average),]
write.csv(i6_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, % Change, No CT, SL, Lat, Str, EH, OcDe")

# 2 Hidden Layers #

i6_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i6_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i6_Summary.2lay <- cbind(i6_results.2lay.1[,1:2],F1.Avg=i6_results.2lay.1[,4], F2.Avg=ri6_esults.2lay.2[,4], F2.Avg=i6_results.2lay.3[,4],
F4.Avg=i6_results.2lay.4[,4],
                         F5.Avg=i6_results.2lay.5[,4], F6.Avg=i6_results.2lay.6[,4], F7.Avg=i6_results.2lay.7[,4], F8.Avg=i6_results.2lay.8[,4],
                         F9.Avg=i6_results.2lay.9[,4], F10.Avg=i6_results.2lay.10[,4])
i6_Summary.2lay <- as.data.frame(i6_Summary.2lay)
i6_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i6_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi6_(i6_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, % Change, No CT, SL, Lat, Str, EH, OcDe")
i6_Best.result.2lay = i6_Summary.2lay[which.max(i6_Summary.2lay$CV.Average),]
write.csv(i6_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, % Change, No CT, SL, Lat, Str, EH, OcDe")

# 3 Hidden Layers #

i6_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
```

```
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i6_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i6_Summary.3lay <- cbind(i6_results.3lay.1[,1:3],F1.Avg=i6_results.3lay.1[,4], F2.Avg=i6_results.3lay.2[,4], F3.Avg=i6_results.3lay.3[,4],
F4.Avg=i6_results.3lay.4[,4],
                         F5.Avg=i6_results.3lay.5[,4], F6.Avg=i6_results.3lay.6[,4], F7.Avg=i6_results.3lay.7[,4], F8.Avg=i6_results.3lay.8[,4],
                         F9.Avg=i6_results.3lay.9[,4], F10.Avg=i6_results.3lay.10[,4])

i6_Summary.3lay <- as.data.frame(i6_Summary.3lay)
i6_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i6_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i6_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, % Change, No CT, SL, Lat, Str, EH, OcDe")
i6_Best.result.3lay = i6_Summary.3lay[which.max(i6_Summary.3lay$CV.Average),]
write.csv(i6_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, % Change, No CT, SL, Lat, Str, EH, OcDe")


#### i7 Define Model Varibles ####

ind.variables <- c("Portfolio.Manager", "Exterior.Area", "Longitude", "X2010.EUI")
formula <- as.formula(paste("Delta.EUI ~", paste(ind.variables[!ind.variables %in% "Delta.EUI"], collapse = " + ")))
reps <- 5
threshold <- 0.01
max.nodes <- 15
#### i7 Training Model - only 4 variables####
# 1 Hidden Layer #
i7_results.1lay.1 <-(nn.benchmark.1layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.2 <-(nn.benchmark.1layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.3 <-(nn.benchmark.1layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.4 <-(nn.benchmark.1layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.5 <-(nn.benchmark.1layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.6 <-(nn.benchmark.1layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.7 <-(nn.benchmark.1layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.8 <-(nn.benchmark.1layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.9 <-(nn.benchmark.1layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.1lay.10 <-(nn.benchmark.1layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result

#Summary.1lay <- NULL
i7_Summary.1lay <- cbind(Layer1=i7_results.1lay.1[,1],F1.Avg=i7_results.1lay.1[,4], F2.Avg=i7_results.1lay.2[,4], F1.Avg=i7_results.1lay.3[,4],
F4.Avg=i7_results.1lay.4[,4],
                         F5.Avg=i7_results.1lay.5[,4], F6.Avg=i7_results.1lay.6[,4], F7.Avg=i7_results.1lay.7[,4], F8.Avg=i7_results.1lay.8[,4],
                         F9.Avg=i7_results.1lay.9[,4], F10.Avg=i7_results.1lay.10[,4])
i7_Summary.1lay <- as.data.frame(i7_Summary.1lay)
i7_Summary.1lay$CV.Average <- round((c(Means=rowMeans(i7_Summary.1lay[,2:11], na.rm=TRUE))),2)
write.csv(i7_Summary.1lay, file = "Tuning Summary, ANN, h=1, 2010-2015, % Change, only 4 variablen ")
i7_Best.result.1lay = i7_Summary.1lay[which.max(i7_Summary.1lay$CV.Average),]
write.csv(i7_Best.result.1lay, file = "Best Results, ANN, h=1, 2010-2015, % Change, only 4 variable")

# 2 Hidden Layers #

i7_results.2lay.1 <-(nn.benchmark.2layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.2 <-(nn.benchmark.2layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.3 <-(nn.benchmark.2layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.4 <-(nn.benchmark.2layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.5 <-(nn.benchmark.2layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.6 <-(nn.benchmark.2layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.7 <-(nn.benchmark.2layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.8 <-(nn.benchmark.2layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.9 <-(nn.benchmark.2layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, threshold, reps,
ind.variables))$benchmark.result
i7_results.2lay.10 <-(nn.benchmark.2layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, threshold,
reps, ind.variables))$benchmark.result

i7_Summary.2lay <- cbind(i7_results.2lay.1[,1:2],F1.Avg=i7_results.2lay.1[,4], F2.Avg=ri7_esults.2lay.2[,4], F2.Avg=i7_results.2lay.3[,4],
F4.Avg=i7_results.2lay.4[,4],
                         F5.Avg=i7_results.2lay.5[,4], F6.Avg=i7_results.2lay.6[,4], F7.Avg=i7_results.2lay.7[,4], F8.Avg=i7_results.2lay.8[,4],
                         F9.Avg=i7_results.2lay.9[,4], F10.Avg=i7_results.2lay.10[,4])
i7_Summary.2lay <- as.data.frame(i7_Summary.2lay)
i7_Summary.2lay$CV.Average <- round(c(Means=rowMeans(i7_Summary.2lay[,3:12], na.rm=TRUE)),2)
write.csvi7_(i7_Summary.2lay, file ="Tuning SUmmary, ANN, h=2, 2010-2015, % Change, only 4 variable")
i7_Best.result.2lay = i7_Summary.2lay[which.max(i7_Summary.2lay$CV.Average),]
write.csv(i7_Best.result.2lay, file ="Best Results, ANN, h=2, 2010-2015, % Change, only 4 variable")

# 3 Hidden Layers #

i7_results.3lay.1 <-(nn.benchmark.3layers(formula, data.train.1, data.test.1[,-1], data.test.1[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.2 <-(nn.benchmark.3layers(formula, data.train.2, data.test.2[,-1], data.test.2[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.3 <-(nn.benchmark.3layers(formula, data.train.3, data.test.3[,-1], data.test.3[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
```

```
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.4 <-(nn.benchmark.3layers(formula, data.train.4, data.test.4[,-1], data.test.4[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.5 <-(nn.benchmark.3layers(formula, data.train.5, data.test.5[,-1], data.test.5[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.6 <-(nn.benchmark.3layers(formula, data.train.6, data.test.6[,-1], data.test.6[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.7 <-(nn.benchmark.3layers(formula, data.train.7, data.test.7[,-1], data.test.7[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.8 <-(nn.benchmark.3layers(formula, data.train.8, data.test.8[,-1], data.test.8[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.9 <-(nn.benchmark.3layers(formula, data.train.9, data.test.9[,-1], data.test.9[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
i7_results.3lay.10 <-(nn.benchmark.3layers(formula, data.train.10, data.test.10[,-1], data.test.10[,1], 1, max.nodes, 1, max.nodes, 1, max.nodes,
threshold, reps, ind.variables))$benchmark.result
beep(sound = 2, expr = NULL)

i7_Summary.3lay <- cbind(i7_results.3lay.1[,1:3],F1.Avg=i7_results.3lay.1[,4], F2.Avg=i7_results.3lay.2[,4], F3.Avg=i7_results.3lay.3[,4],
F4.Avg=i7_results.3lay.4[,4],
                         F5.Avg=i7_results.3lay.5[,4], F6.Avg=i7_results.3lay.6[,4], F7.Avg=i7_results.3lay.7[,4], F8.Avg=i7_results.3lay.8[,4],
                         F9.Avg=i7_results.3lay.9[,4], F10.Avg=i7_results.3lay.10[,4])

i7_Summary.3lay <- as.data.frame(i7_Summary.3lay)
i7_Summary.3lay$CV.Average <- round(c(Means=rowMeans(i7_Summary.3lay[,4:13], na.rm=TRUE)),2)
write.csv(i7_Summary.3lay, file ="Tuning SUmmary, ANN, h=3, 2010-2015, % Change, only 4 variable")
i7_Best.result.3lay = i7_Summary.3lay[which.max(i7_Summary.3lay$CV.Average),]
write.csv(i7_Best.result.3lay, file ="Best Results, ANN, h=3, 2010-2015, % Change, only 4 variable")
```