QOS-BASED DATA ANALYTIC SERVICE SELECTION:

A COMPARATIVE STUDY OF DIFFERENT LEARNING MODELS

By

Bayan Alghofaily

Bachelor of Science in Information Systems

Prince Sultan University

Saudi Arabia, 2013

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the program of

Computer Science

Toronto, Ontario, Canada, 2015

©Bayan Alghofaily 2015

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

QOS-BASED DATA ANALYTIC SERVICE SELECTION: A COMPARATIVE STUDY OF DIFFERENT LEARNING MODELS

Bayan Alghofaily

Master of Science, Computer Science, 2015

Ryerson University

ABSTRACT

QoS-based web service selection has been studied in the service computing community for some time; however, data characteristics are not considered. In this work, we have studied the use of different machine learning algorithms as meta-learners in predicting the performance of data analytic services for the given dataset. We used a meta-learning algorithm to incorporate meta-features in the selection process and we used clustering services as an example of data analytic services. We have also investigated the impact of the number of data features on the performance of the meta-learners. We found that, out of the 5 classification models, SVM showed the best results in predicting the recommended service for the given dataset with an accuracy of 78%. When it comes to regression models, MLP was the best regressor. We recommend considering only simple meta-features that can be collected for most datasets, as those proved to be sufficient to achieve good prediction accuracy.

ACKNOWLEDGEMENTS

This thesis could not have been written without the support of my supervisor Dr. Cherie Ding, who encouraged and challenged me throughout my academic program. She guided me throughout my master and thesis and never accepted less than my best efforts. Thank you Dr. Ding.

I would like to acknowledge and extend my warm appreciation to the Department of Computer Science at Ryerson University, my professors and classmates for their cooperation and assistance. I would also like to thank my defense examiners Dr. Woungang, Dr. Sadeghian, and Dr. Harley.

Special thanks to my husband, words alone cannot express what I owe him for his encouragement, patience and care to complete this period. Without him, this period would have been much harder and more difficult.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AHP:	Analytic Hierarchical Process
CP:	Constraints Programming
DAaaS:	Data Analytic as a Service
EM:	Expectation Maximization
IBL:	Instance Based Learning
KNN:	k Nearest Neighbor Algorithm
MCDM:	Multi Criteria Decision Making
MIP:	Mixed-integer programming
MLP:	Multi Layer Perceptron
QoS:	Quality of Service
QP:	Quadratic Programming
RAE:	Relative Absolute Error
REST:	Representational State Transfer
SMO:	Sequential Minimal Optimization
SRC:	Spearman Rank Correlation
SVM:	Support Vector Machine

CHAPTER 1

INTRODUCTION

1.1 Background and the Problem Statement

1.1.1 Background

The rapid increase in use of social media, online e-commerce web sites, sensors, Internet of Things and big-scale scientific projects has drastically increased the amount of digital raw data we produce. From here, a concept of big data has been introduced which refers to the enormous, varied and often real-time streaming of digital data that cannot be processed by traditional data management tools or techniques. For years, IT departments have collected and stored vast amounts of data. Now they are expected not only to store the data but also to provide an infrastructure to analyse these data.

Data mining offers various techniques for finding, describing and analysing patterns in large amounts of data to understand the data better and make predictions[1]. Weka [2], R [3] and RapidMiner [4] are examples of common data mining applications that provide different algorithms and techniques such as classification and clustering to analyse data. However, big data analytics require intensive data mining techniques that many small businesses cannot afford in cost, time or even in equipment to run such techniques on their machines. For this reason, a client may prefer to use data analytics as services over the Internet to examine his or her data.

Cloud computing can serve as a suitable solution for both computational and data storage that big data applications require. Combining big data analytics and knowledge discovery techniques with scalable computing systems is the next step towards cloud-based big data analytics. Service-oriented computing is a potential solution for big data analytics in the cloud, which means we can provide data analytics as software services through a cloud platform [5].

1.1.2 Problem Statement

Different types of services are available on the Internet such as commerce and communication services. Another type of service that is currently flourishing is data analytic services. As more and more data becomes available on the Internet, there is a need for services to process these raw data with the purpose of drawing useful information and creating knowledge. Data Analytic as a Service (DAaaS) [6] is a new topic in this area that uses a cloud-based model to provide an extensible platform for analytical capabilities. There are many common data analytics services offered in the cloud such as Microsoft Azure [7] is a cloud computing platform for integrated services and Amazon Machine Learning [8] services that allow you to create predictive applications and machine-learning models easily. Other data analytics software services include Spark Machine Learning Library [9], IBM Watson Analytics [10], etc.

Many of these data analytic services offer similar functionalities and the challenge here is how to select the best service for the given data. The traditional approach of service selection depends on only a general understanding of the functionality of the service to make the selection [11]. A better way to select services is to consider both functional and non-functional requirements of the service. Functional requirements determine the overall behaviour of the services and establish how relevant a service is to the user’s query. Non-functional requirements refer to the quality attributes of a service or its Quality of Services (QoS). It supports prioritisation between several services. Web service selection is the process that distinguishes functionally similar services according to their quality features to recommend the best web service for a client's requirement. For example, a user initiates a request through a service selection system to search for an online communication service where “the cost of the service” should be less than $2, “the availability of the service” should be greater than 80% and “the security of the service” should be higher than 90%. Then the selection system finds all the functionally matching services (i.e. communication services) and ranks them based on their values with these three QoS attributes. The highest ranked is the one recommended to the user, or sometimes a ranked list is returned and the user makes a decision accordingly.

QoS-based web service selection and recommendation has been studied in the service computing community for some time; however, data characteristics are not considered. In other words, the selection process only focuses on the QoS values for the service without taking into consideration the dataset characteristics (e.g. size, dimensions, pattern, missing values, etc.) when making a service selection. Data characterisation is important for the selection process of data analytic services because it affects the QoS values. For instance, a service will take less time on a smaller dataset, and a service implementing one particular algorithm will achieve a more accurate result on a certain dataset. Most of the current service selection approaches take the average QoS values from past invocations to rank services. However, some QoS values are closely related to dataset features and an average value cannot represent QoS for a particular dataset. We address these problems in this thesis work.

1.2 Objectives

This thesis aims to study the service selection problem based on data-dependent QoS properties, where the selection process is to find the best service for a given dataset.

We have the following objectives in this work:

1. To provide a comparative study on the accuracy of different machine-learning models when applying them to select the best data analytic service for a given dataset.
2. Since a Support Vector Machine (SVM) approach usually performs well for classification problems, we particularly want to evaluate the performance of various SVM models for service selection.
3. There are multiple features that can be used to describe a dataset. In many traditional meta-learning or algorithm selection papers a good number of features are used. However, for real datasets, especially big data, we might not be able to get all these measurements. In this work, we investigate whether the number of features has a significant impact on the result.

1.3 Methodology

In this comparative study, we use five common machine-learning models: Support Vector Machine (SVM), K-nearest Neighbours (KNN), Multilayer Perceptrons (MLP), Naïve Bayes and the C4.5 algorithm. For the SVM, we specifically use the Sequential Minimal Optimisation algorithm for training a support vector. For the C4.5 algorithm, the

J48 is used to generate a decision tree. We choose SVM, KNN and MLP because they are frequently used for algorithm selection in similar contexts. Naïve Bayes and J48 are simple, straightforward classifiers and we wanted to study whether simple classifiers can work as well as more complex algorithms in this problem.

In our framework, we choose clustering services as an example of data analytic services; other analytic services can also be used in the same experiment. The goal is to select the best service for a given dataset. To achieve our goal, we first create a database that collects the meta-features of the datasets and QoS information for all previous service requests. To identify the best service, we first use normalisation functions to normalise QoS values and give a weight to each QoS attribute. After that, the weighted sum is used in a utility function to get the overall score of the service. Two types of selection are considered here: using classification models and using regression models. In the classification model, the service with the highest overall score is picked as the class value for the given dataset. Then all the datasets with their meta-features and class values are fed to the model to predict the best service. In the regression model, each chosen machine-learning algorithm builds a regression function that estimates the QoS values for each service for the given dataset and then uses the predicted QoS values in the utility function to predict the best services. This way, a service requester has more knowledge about the service that works best with his or her dataset without having to try all the services.

1.4 Contributions

Our contributions in this work are as listed below:

- We provided a comparative study for a QoS-based data analytic service selection by five different meta-learners when applying them to recommend the best clustering service and predict QoS attributes for the given dataset.
- We studied the impact of the number of meta-features in the performance of the meta-learner predictions.

1.5 Thesis Outline

The rest of the thesis is organised as follows. Chapter 2 reviews and analyses current QoS-driven web service selection methods. We also discuss some meta-learning approaches for algorithm selection and describe the different algorithms used in the experiments. In Chapter 3 there is a detailed description of the different meta-learners and prediction models used in the selection process. Both QoS properties and meta-feature attributes are discussed in this chapter.

Chapter 4 presents our experiment design, tools and system configuration with our results and observations. It also describes the metrics used for evaluating the algorithms. Finally, Chapter 5 concludes the thesis work along with some suggestions for future work.

CHAPTER 2

LITERATURE REVIEW

This chapter begins by reviewing the classification and regression algorithms that have been used in our later experiment. Two types of related work are discussed: i) QoS-Based Web Service Selection Methods and ii) Algorithm Selection Using Meta-learning Approaches.

2.1 Classification and Regression Algorithms

Two types of data mining algorithms are discussed here: i) classification model where a new observation is assigned to a class based on the training model. ii) regression model which predicts the target value based on a regression function. Some machine-learning algorithms can be used for both classification and regression models such as SVM and Artificial Neural Network.

Sequential Minimal Optimization (SMO) implements John Platt's Sequential Minimal Optimisation algorithm for training a support vector classifier [12]. Like other SVM-training algorithms, it solves Quadratic Programming (QP) optimisation problems by breaking down the QP problem into smaller problems. SMO utilises the smallest possible QP problem, which makes it faster and it has better scaling than any other SVM algorithm. SMO consists of two parts: i) an analytical solution to a QP problem of the two Lagrange multipliers and ii) a set of heuristics for efficiently choosing which multipliers to optimise. An improved version of the SMO algorithm is used for training

SVM regression. The main modification is to add caching kernel output with other heuristics [13].

K-Nearest neighbours (KNN) algorithm is considered one of the easiest machine-learning algorithms since there are no assumptions made on the underlying data distribution. It is a non-parametric algorithm for solving classification and regression problems. KNN is a lazy learning algorithm because it does not utilise the training data for generalisation. Therefore, the training phase is quite minimal or non-existent [14].

Multilayer perceptrons (MLP) are feed-forward neural networks that consist of multilayer nodes with at least one hidden layer. Each neuron has a non-linear activation function that defines its output given a set of inputs. MLP uses a back-propagation learning technique to train the network to find the weight that maps an input to an output. Depending on the activation function, a neural network can solve either classification or regression problems [15].

The Naïve Bayes classifier depends on the Bayes rule of conditional probability. It is a simple classifier that can only work with nominal classes. The Naïve Bayes classifier is easy to implement and, in some cases, outperforms many other complex algorithms. It is a powerful probabilistic representation that is robust to noise and can handle null values. However, Naïve Bayes works with the assumption that each attribute is working independently from other attributes, which can sometimes negatively impact the accuracy of the model. In addition, practical dependencies exist among variables in the Naïve Bayes classifier [16].

The J48 decision tree classifier is a predictive machine-learning model that assigns a target value for a new sample based on different attribute values of the dataset. This classifier works well in most data classification problems since it structures a tree that makes the model more understandable. Furthermore, the J48 decision tree does not make any prior assumptions about the data, and is able to process both categorical and numerical data. However, the output attribute has to be categorical and trees that are formed from numeric datasets can be somewhat complex. The decision tree construction process is unstable because small changes in the data can lead to quite different decision trees [1].

2.2 Review of QoS-based Web Service Selection Methods

As mentioned previously, the web service selection process checks available web services to find those most relevant to the user's query. However, service selection based solely on functional requirements is insufficient and usually leads to problems. For instance, if a user's functional query is "weather forecasting service", there may be 20 such services found with exactly the same functionality, and they are ordered randomly or based on alphabetical order of their providers' names. So a service with very low availability and a very long response time could be in a top position, which is not desirable. This is why non-functional requirements should also be included in the selection so that a more reasonable ranking can be created.

QoS-based web service selection is usually considered as an optimisation problem and different approaches have been proposed to find the best solution. One way to represent QoS attributes is through a QoS matrix where each service is assigned to a vector. In [17], domain specific criteria such as business-related criteria are considered in the QoS model for web services. A web service is selected based on the user's constraints, and arranged in a matrix where each row shows a service and each column represents a constraint. After normalising QoS attributes and QoS groups, services are ranked. A service broker is another way to provide quality information by managing the supply of information from the service provider and the QoS metrics. Al-Masri and Mahmoud used a third party to collect web services' QoS metrics to ensure quality-driven discovery and ranking of web services [18]. They used the weighted sum of seven attributes to measure the relevance of a web service to a user's query. A similar approach is used in [19] where QoS requirements are compared with QoS advertisements in the repository and services with matching QoS are discovered. Then, these discovered services are ranked based on QoS vector models and the consumer's given weight to each attribute. Even though using metrics to represent QoS values is very useful for choosing the most suitable service when there is more than one similar service matching the request, this method of selection does not focus on defining the relationship between consumers' preferences and the utility the consumer derives from the service.

One way to look at service selection is through utility functions. A utility is a dimensionless quantity that is used to measure the usefulness of a system in terms of a set of attributes. In [20], utility functions with cost constraints are used to perform service selection by a QoS broker. A client sends a request to the broker that indicates the service type, the cost constraint and the utility function. Average response time and throughput are used here as QoS metrics for the utility functions. The idea is to select the service provider that maximises a global utility for a service requester under a cost constraint. Hang and Singh [21] used a probabilistic trust model to describe the service provider and create quality distribution based on the utility functions. They considered the reputation of the service provider by the quality distribution and the utility function that reflects the consumer's preferences. The expected utility of providers with respect to the sum of qualities is calculated here. Based on the expected utility and using utility policies, consumers decide which provider promises the utility they expect or which service maximises their utility. An advantage of using utility functions in the selection process is that they can be adjusted according to the situation. For instance, in composite services the utility can be adjusted based on the selected subservices. In other situations, new attributes can be added into the utility function and the weights adjusted according to their degree of importance. The main issue in this approach is how to create the utility function and handle the complexity of it as more parameters are added.

Sometimes, QoS conditions are used as constraints programming (CP) to find similar values. In [22], description logics reasoning is introduced into the QoS aspects to ensure semantic matchmaking. Next, the Constraint Programming layer is used where QoS conditions are translated into constraints based on intrinsic semantics. Finally, the optimal web service is selected by combining all parameters and taking their relative weights. In [23], semantic consistency and conformance checking are used here such that demands and offers can be modelled using Quality Requirement Language. QoS offers are assessed using weights for every QoS metric and using the metric's utility function to rank each offer. A service discovery approach that uses CP alone is not accurate enough because CP usually uses syntactic methods to match QoS parameters and lacks semantic matchmaking metrics. So by adding semantic information, the proposed method can achieve a more accurate result.

In [24], the semantic QoS description is transformed into a Mixed-Integer Programming (MIP) problem, and then an MIP engine is exploited for matchmaking. Two different constraints programming-based service matchmaking algorithms were developed: a unary-constraint where each demand's constraint has only one QoS metric and a more generic algorithm where n-ary constraints are accepted. Several versions of these two matchmaking algorithms were compared based on the average execution time of the matchmaking algorithm. According to their experiment [24], MIP outperforms CP in matchmaking algorithms, especially with linear constraints. However, non-linear problems are harder to solve by MIP and should be solved using CP.

Multi-Criteria Decision Making (MCDM) is a common method for web service selection. In [25], an outranking algorithm is used to compare services based on their QoS priorities. These priorities are integrated over QoS into current service selection models by first identifying priorities over QoS considerations, then adding priorities to the Unified Modelling Language QoS framework meta-model, and finally converting priorities into weights to be used in a service selection method. This approach depends on priorities instead of utility functions for attributes, which is why it is important to establish priority ordering in constraint form. The other technique of MCDM that has been used is the Analytic Hierarchical Process (AHP) process that has three essential steps: decomposing the problem, comparing decisions and synthesising priorities. The selection model in [26] is based on AHP. First, the decision-maker is responsible for creating a hierarchy for the problem that ranks the offers based on their QoS values. Then, pair-wise comparison is done to prioritise QoS attributes by weighting them to find the final scores that are the weighted sum of all the QoS values. Finally, relative weights of the QoS attributes are calculated by aggregating the relative local ranks. Garg et al. introduces a tool (SMICloud) for selecting cloud services, their main example is infrastructure as a service [27]. It uses previous data of service performance and user experiences to predict attribute values. Then, a relative index is computed for comparing different cloud services and these services are ranked based on Service Measurement Index attributes by the AHP process. The main issue in the AHP technique is that it does not focus on user-defined QoS constraints over the QoS attributes.

Skyline, which is a d-dimensional space where points are not dominated by any other point, is also used for the process of QoS-based web service selection. For example, in [28], the authors used skyline services to address the challenge of freeing users from assigning weights to present their preferences over the QoS attributes. Given a set of service providers and a specified probability threshold, the p-dominant skyline service is computed to find the optimised solutions by measuring their dominance relationships. The average QoS value for each provider is used to calculate the aggregate skyline. Only providers that survive the pruning heuristics steps are computed for dominant probability and are compared with other providers. Likewise, in [29] the concept of top-k retrieval of web services was introduced by defining three ranking steps: i) aggregating the degree of match in all parameters in the matched descriptions, ii) having multiple matching criteria through combining the match results from the individual similarity measures and iii) ranking the results. Taking into consideration these three ranking criteria, *TKDD*, *TKDG*, and *TKM* ranking algorithms are used for selecting the top-k matches for a service request.

As mentioned earlier in this section, to the best of our knowledge, most of the existing service selection methods such as the ones in [20] and [28] only consider the average QoS values from past invocations to rank services. However, some QoS values depend on dataset features; for example, the response time for a large dataset will be higher than the time for a smaller dataset, and taking the average will sometimes be misleading. In this thesis, we show an approach to help in the selection process for data analytics services, taking into consideration that the quality of services can vary for different types of data.

2.3 Review on Algorithm Selection using Meta-learning

Some research efforts in the area of meta-learning have focused on the use of meta-learning algorithms to predict the performance of algorithms for a new problem based on the relationship between data characteristics and algorithm performance. The most common domain using a meta-learning approach for algorithm selection is recommending classification algorithms [30]. Other application areas include regression, time-series forecasting, sorting, constraint satisfaction problems and optimisation.

The framework in [31] performs selection of classification algorithms using an IBL (Instance-Based Learning) approach to identify similar datasets as the performance of candidate algorithms is expected to be similar for similar datasets. The similarity between the datasets is measured by data characteristics and a list of measures using a KNN algorithm. Performance of algorithms is measured through an "Adjusted Ratio of Ratios (ARR)" that considers accuracy and time. Their result suggests that ARR with KNN leads to significantly better rankings than the baseline ranking method; however, the KNN algorithm was sensitive to irrelevant attributes that were eliminated in the experiment.

It is very common in a meta-learning context to use a machine-learning SVM to predict algorithm performance based on features of the learning problem. In fact, the SVM is proved to have higher accuracy than neural networks and KNN [32]. In [33], they want to predict the performance of MLP networks using meta-regression. They generate a set of meta-examples by evaluating the performance of the MLP in each regression problem and calculate the value of 10 meta-features for each meta-example. The generated meta-examples are input to the SVM algorithm to produce a regression model that is responsible for predicting the algorithm's performance for new problems based on its meta-features.

The framework in [32] consists of a Feature Extractor (FE) to generate the meta-features, a database that stores examples of datasets used in the training phase and a meta-learner for ranking the candidate algorithms. This is done with the help of a number of regressors, each one being responsible for predicting the ranking of a specific algorithm for the input dataset. In their case study, they used an SVM regression algorithm as the meta-learner to rank seven clustering algorithms for 32 cancer gene expression microarray datasets with the help of eight meta-features. Each of the algorithms is run 30 times with the non-normalised version of the dataset to produce the respective partitions. A “corrected Rand index (cR)” is generated to measure the agreement or disagreement between the partitions, which then indicates the position of the clustering algorithm in the ranking.

The above framework was extended in [34]. It used two different sets of meta-features and ranked the algorithms using an MLP network and Support Vector Regression as the meta-learners. The ranking was based on the performance of the algorithms in terms of their global error rate. The research here was based on artificial datasets generated using [35]. The authors worked with eight different combinations (of number of clusters and dimensions) and created 10 different instances of each combination, thus generating 80 datasets using a Gaussian cluster generator. They also generated another 80 datasets with higher dimensions for the ellipsoid cluster generator.

In [36], a meta-learning approach was used to recommend clustering algorithms for seven UCI datasets. Each clustering algorithm was run 30 times and the best FBCubed metric was selected to perform the ranking. The KNN algorithm, MLP, neural networks, decision tree and Naïve Bayes were used as the meta-learners. A Spearman Ranking Correlation coefficient (SRC) was used to evaluate the prediction of the ranking for the clustering algorithms. In their experiment, the KNN algorithm performed best for their dataset.

Similarly, in [37], developers defined 32 meta-examples (each one corresponding to a dataset) described by a set of meta-features for a gene expression dataset, and a vector with the ranking of the clustering algorithms for that dataset. To measure the similarity between meta-features, they used Euclidean distance, Pearson correlation and cosine. For a new dataset, a meta-regressor used the meta-examples to predict the algorithms' ranks for the input dataset. As in [36], the similarity between the predicted and the default rankings for the input dataset was done by the SRC coefficient.

Our approach differs from the traditional meta-learning approach in the way that our meta-learners are used to predict the best clustering service for the given dataset. The meta-learners are used for both classifying services and predicting QoS values. In addition, in order to evaluate the performance of our selected services, we consider a multi-criteria evaluation measure unlike the evaluation method used in [32, 36] that only uses a single performance measure.

CHAPTER 3

METHODOLOGY

3.1 Introduction

Quality-based web service selection is the process that incorporates QoS attributes to distinguish functionally similar services according to their quality features to recommend desirable web services to a client's request. Unlike traditional selection processes, QoS-based selection ensures priorities between the services, which yields more related and desirable services. QoS is usually divided into two types: positive QoS parameters that need to be maximised such as security and reliability, and negative parameters that need to be minimised such as cost and response time. In our work we consider both positive and negative parameters.

As discussed in Chapter 2, several methods of QoS-based service selection are available in the literature. QoS requirements can be represented as vector-based metrics, and the services are then ranked based on the distance between the required QoS vector and the actual one [17-19]. Other approaches such as utility function and constraint programming are used to find services that optimise the QoS criteria [20, 23]. Sometimes, prioritising between QoS attributes is considered based on user preferences using the MCDM (Multi Criteria Decision Making) approach [25, 26]. Skyline is another approach that has been used for web service selection based on QoS attributes [28, 29]. However, most of the selection methods proposed in the literature lack an important aspect for data-processing or data-analysing services: the data characteristics or features of the processed datasets. It is obvious that the same service might behave differently for different types of dataset, because of the limitations or requirements of the service, or the dependency between some QoS values and datasets. For example, some services require the data to be categorical, others cannot process unary class values. Furthermore, response time is generally longer on a bigger dataset.

As more and more data become available online and the speed at which they are being produced rises, users can no longer rely on using tools installed locally on their machines to process their data because of capacity, memory or even cost limitations. When it comes to processing big data, the public cloud is the obvious solution. The emergence of big data and cloud computing presents the potential to use Data Analytics as a Service (DAaaS) [6]. DAaaS is an extensible platform for using analytic tools that can process huge quantities with different forms of data using cloud delivery methods. Users can easily feed their data to the platform and all the processing is done externally to provide fast and reliable results. Various types of data analytics services are offered such as data mining, data modelling, visualisation and data cleansing.

In this thesis, we use data mining services as example data analytic services to solve the problem of service selection based on QoS attributes and dataset characteristics, i.e. the meta-features of the dataset, to provide the recommended data mining service for the given dataset.

3.2 System Architecture

The system consists of different components as shown in Figure 3-1 and described as follows:

- **Invocation Proxy:** a computer network service that connects user requests to the Service Provisioning Server so that the QoS values of the invoked services can be monitored and recorded.
- **Service Provisioning Server:** the place that runs the data mining services on the given dataset.
- **QoS Repository:** the database that keeps track of the QoS values for the applied services.
- **Meta-features Repository:** where all the dataset characteristics such as type and size of the dataset are stored. For every service request made, the system records the meta-features of the input dataset to keep a log of the meta-features of the dataset used.
- **QoS Normalisation:** the process of adjusting QoS values to a common scale.
- **Utility Function:** the process of calculating the overall score of each service for the given dataset so the service with the maximised utility values is usually ranked higher.
- **Functional Matching:** the first step of the selection process where functionally similar services are found based on the functional requirements of the user request.

- **Non-functional Matching:** the process of identifying the best service for the given dataset based on the user’s non-functional requirements which is done with the help of:
    - **Classification model:** using meta-features and the identified class values to predict the best service.
    - **Regression function:** using meta-features to estimate the QoS values for each service for the given dataset and using the predicted QoS values in the utility function to predict the best services.

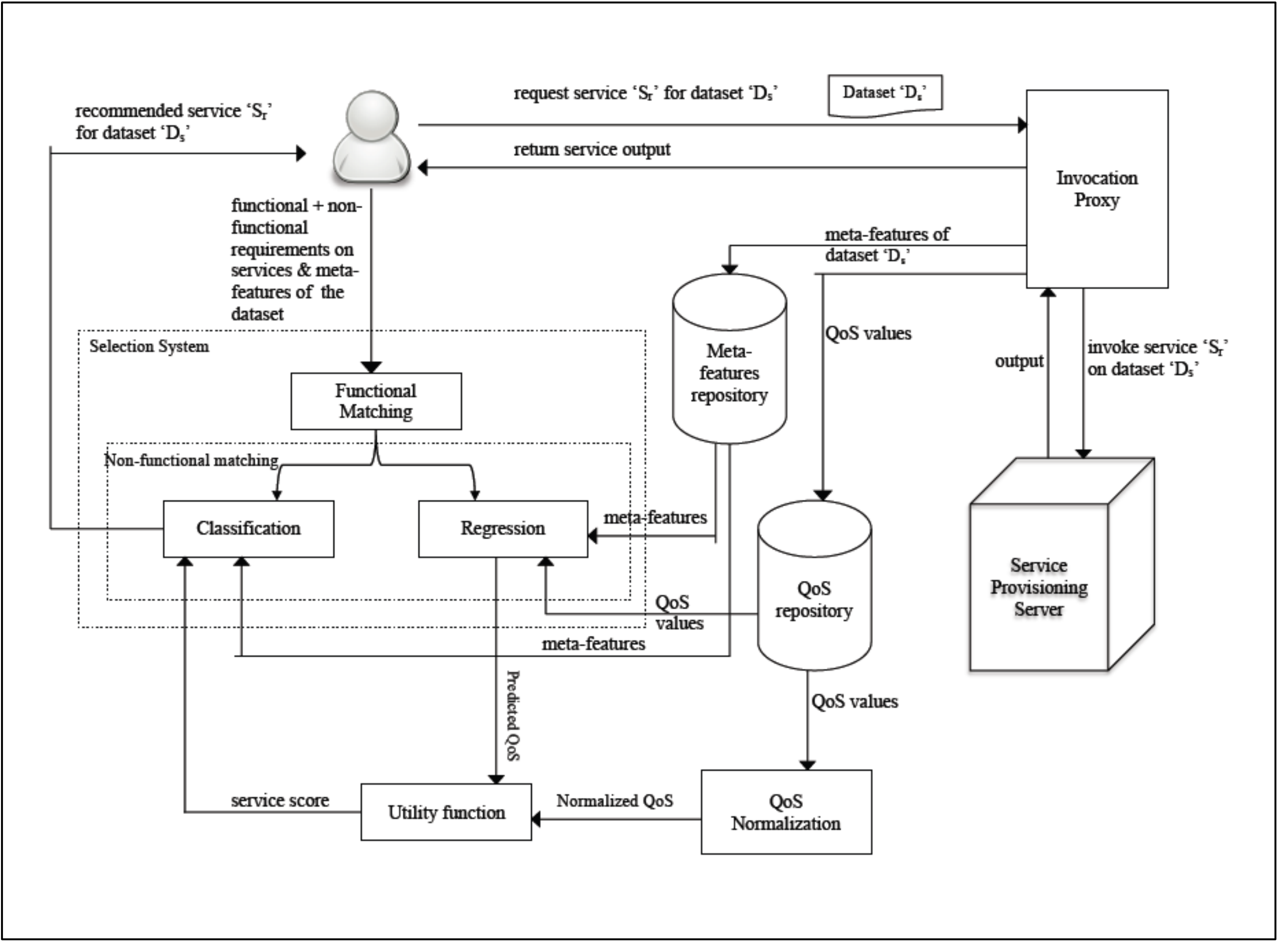

Figure 3-1: System Architecture

There are two workflows existing in our system: i) service selection workflow and ii) invocation workflow. The first workflow is the selection process that begins by first finding services that have similar functionalities to the functional requirements of the user, which is called the functional matching step. The second step is the non-functional matching that uses QoS attributes to prioritise between the services. The goal of this process is to find the best service S_r that matches user requirements and works the best for the input dataset D_s.

The second workflow starts when a user sends an invocation request for service S_r on dataset D_s. The request goes through an Invocation Proxy, and then is forwarded to the

Service Provisioning Server where the required service is applied to the input dataset. The delivered service also goes through the Invocation Proxy to allow for the actual QoS data to be monitored and recorded in the QoS Repository and meta-feature values to be stored in the Meta-features Repository. The invocation workflow is completed once the service output is returned to the user.

In order to identify the best service, we use a utility function based on the objective function taken from [24] that normalises the QoS values first, then gives a weight for each attribute and finally calculates an overall score for each service based on its QoS values. This utility function considers different QoS metric tendencies, and it also adapts to user preferences by taking different QoS weights.

The utility function for service *Sr* is given below:

$$U(S_r) = \sum_{i \in X} \frac{\mathrm{V_{ir}} - \mathrm{MIN_i}}{\mathrm{MAX_i} - \mathrm{MIN_i}} * W_i \; + \; \sum_{j \in Y} \frac{\mathrm{MAX}_j - \mathrm{V_{jr}}}{\mathrm{MAX}_j - \mathrm{MIN_j}} \; * W_j \quad (3.1)$$

where X is the set of positively monotonic QoS metrics and Y is the set of negatively monotonic QoS metrics. $\mathrm{V_{ir}}$ (or $\mathrm{V_{jr}}$) is the value of the QoS measure for service S_r. r is the range of n services where $1 \le r \le n$. W_i (or W_j) represents the corresponding weight for the i^{th} (or j^{th}) QoS attribute. $\mathrm{MAX_i}$ and $\mathrm{MIN_i}$ (or $\mathrm{MAX_j}$ or $\mathrm{MIN_j}$) represent the maximum and minimum values for these QoS metrics among all the candidate services.

In our selection model, we consider two types of approaches that are described below:

1) Classification

In this type, the meta-features and the identified class values (i.e. the best performing services according to their utility scores) for all the given datasets are fed to train the classifier which then generates the predicted service name that is the best for this new dataset. We also consider classification via regression, which is an approach for predicting the service name through regression estimation. The class values are replaced by binary digits and one regression function is built for each class value.

2) Regression

   In this scenario, a regression function is built for each service. The meta-features and the actual QoS values from all available services for the given datasets are input to the regression model. The model then predicts the QoS values from each service on the new dataset. The estimated QoS values are subsequently used in the utility function to identify the service which maximises the utility value and obtain the best service for the given dataset.

Finally, the recommended service is returned to the user along with the service output.

3.3 Learning/Selection Algorithm

In our system, we take software services implementing data clustering algorithms as an example of data analytic services. Clustering is a data mining technique used to place data elements that are similar to each other into related groups without previous knowledge of group definitions.

A learning algorithm may perform differently in different learning problems, which may result in restricting the machine-learning or the data mining techniques since the relationship between the learning problem and the use of the learning algorithm is not well established. To tackle this issue, meta-learning has been effectively used in the machine-learning community in selecting and recommending machine-learning algorithms.

We consider the data-dependent QoS-based service selection problem as similar to the meta-learning problem, and the target is to select the best data analytic service for a given dataset based on its meta-features and service QoS values which are affected by these features. Therefore in our methodology, we use a meta-learning approach to select the best clustering service for a given dataset. Meta-learning can be defined as the automatic process of relating the performance of learning algorithms to the features of the learning problems (e.g. the meta-features of the datasets) to obtain knowledge [38]. There are various approaches to meta-learning; the one used here is the classification of data clustering services as recommended and non-recommended, based on meta-features of the datasets. The meta-feature is formed by general characteristics of the data in the learning problem. A common type of meta-knowledge obtained from the data includes statistical or information-theoretic features. Statistical meta-features describe statistical properties of the data such as canonical and absolute correlation. Information-theoretic features are the ones based on the entropy measures of the dataset, for example, the mean entropy of discrete attributes, joint entropy and mutual information.

3.3.1 Meta-features

In traditional meta-learning, many meta-features are considered. We consider simple ones because they are easier to calculate and are more likely to be obtained for real data. We also want to study whether more meta-features bring higher accuracy, and thus we include five simple descriptive features and two statistical features. Simple descriptive measures include:

1. Dataset size: the number of instances in a dataset.
2. Number of dimensions: the number of attributes in a dataset.
3. Percentage of missing data: the amount of missing data in a dataset computed as a percentage value.
4. Data type: whether the data type is numeric or nominal.
5. Data distribution pattern: whether the data are randomly distributed or have a grid pattern.

Statistical meta-features give an overview of how the data is distributed and describes the statistical properties of the dataset and include:

1. Skewness: this is a measure of the asymmetry of the data around the sample mean. If skewness is negative, the data is shifted to the left of the mean. If skewness is positive, the data is shifted to the right. The skewness of normal distribution is zero. We used the formula provided in [39] to calculate the mean skewness of continuous attributes based on the equation below:

$$s = \frac{(x - \mu)^3}{\sigma^3} \qquad (3.2)$$

where *x* is the numeric attributes of the dataset, μ is the mean of *x*, σ is the standard deviation of *x*.

2. Kurtosis: this is a measure of whether the data are peaked or flat relative to a normal distribution. The kurtosis of a normal distribution is 3. Distributions that are more outlier-prone than the normal distribution have kurtosis greater than 3; distributions that are less outlier-prone have kurtosis less than 3. We used the formula provided in [39] to calculate the mean kurtosis of continuous attributes based on the equation below:

$$k = \frac{(x - \mu)^4}{\sigma^4} \qquad (3.3)$$

where *x* is the numeric attributes of the dataset, μ is the mean of *x*, σ is the standard deviation of *x*.

3.4 Support Vector Machine

SVM is one of the top performing algorithms for solving prediction problems. It was introduced by Vladimir Vapnik [40] and his team in the mid-1990s. SVM was initially developed for classification problems and has been extended for regression. Since then, SVM has proved to be a very effective technique for classification and regression problems. In addition, it has been successfully applied to different selection problems such as algorithm selection. In this work, SVM is the major meta-learning model we consider for service selection.

3.4.1 SVM Classification

SVM is a supervised machine-learning technique that finds the optimal linear hyperplane to maximise the distance to the closest training point from two classes to have better classification on test data. The decision is made based only on the support vectors that are data points located at the margin of the decision boundary. Figure 3-2 illustrates SVM for linearly separable data.

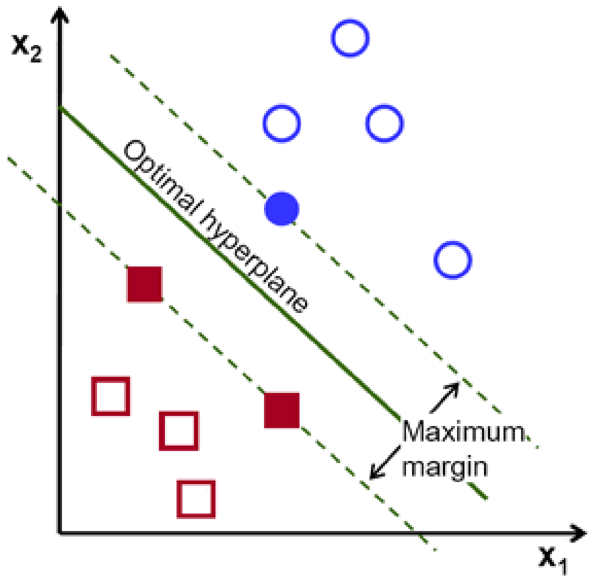

Figure 3-2: Linearly separable data [41]

Circles and squares are data points belonging to two different classes; in our case, two different clustering services (since each clustering service is considered as a class). For every data point (x,y), x is the actual service for a given dataset, y is either 1 or -1 denoting the class to which point x belongs. The classes can be fully separated by the optimal hyperplane. The decision boundary is the optimal hyperplane that leaves the maximum margin from both classes. The margin is the distance between the hyperplane and the closest data point to the hyperplane. The goal of SVM is to maximise the margin to avoid misclassification. The classification is done based on the hyperplane function: $\boldsymbol{w}^T\, \boldsymbol{v} + b = 0$, where w is a weight vector (w1,w2) and b is the bias, which will be computed by SVM in the training process. A binary SVM classifies data point $\boldsymbol{v_i}$ if it is above the decision boundary as class +1 i.e. $\boldsymbol{w}^T\, \boldsymbol{v} + b > 0$, and data point $\boldsymbol{v_i}$ as class -1 if it is below the decision boundary i.e. $\boldsymbol{w}^T\, \boldsymbol{v} + b < 0$.

In the real world, datasets may not be free of noise or not perfectly linearly separable; for example, some data points of one class appear to be on the other side of the boundary. Figure 3-3 illustrates this scenario. In this case, SVM introduces *slack variables* which measure the degree of misclassification. Since there is no hyperplane that can clearly separate the entire data, SVM uses a soft margin method to select a hyperplane that allows data points to be misclassified while maximising the margin. However, if this set of points becomes large, then this causes a computational problem and a major slowdown for using SVMs at test time.

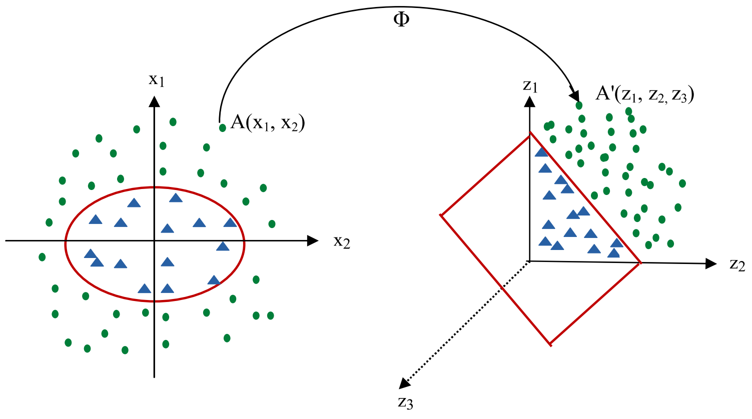

Figure 3-3: Non-linear separable data [42].

To solve the computational problem occurring in non-linearly separable data, SVM finds a classification function for non-linear SVM. This process is called the *Kernel Function Trick*. A kernel function is a similarity function between two input vectors and measures how similar they are. The output of the function is at its maximum when the inputs become equivalent. First, the input vectors are mapped into high-dimensional feature vectors where the chance of finding a separating hyperplane is better for the training points. Then, SVMs are used to find the hyperplane of maximal margin in the new feature space. For every data point, SVM maps the input space into high-dimensional space by computing the dot product of all pairs of data in the feature space $\Phi: x \rightarrow \varphi(x)$. In Figure 3-3, SVM transforms the non-linear data into a 3D feature space where the two classes are separated by a hyperplane.

SVM uses various kernel function techniques: below are common kernel functions used in non-linear SVM classification where *a* and *b* are feature vectors in the input space, the operation between them is the dot product and K determines the mapping Φ [43]:

i. Polynomial: $K(a, b) = (a . b + 1)^d$

ii. Radial basis function: $K(a,b)=exp(-\gamma \, ||a-b||^2)$

iii. Sigmoid: $K(a,b) = tanh(\mathrm{k}a \cdot b+c)$

Further explanations of these kernel functions are out of the scope of this thesis. Instead, we give the basic properties of the linear polynomial kernel that was used later in the experiment. The polynomial kernel is the product between two vectors Φ(a) and Φ(b).

The general function for the polynomial kernel is $K(a, b) = (a.b + c)^d$ where $c \geq 0$ is a free parameter trading off the influence of higher-order versus lower-order terms and d is the degree of the function. When c = 0, the kernel is called homogeneous. The homogeneous kernel $K(a, b) = (a.b)^d$ gives the same result as explicit mapping to a higher feature space and the dot product.

As mentioned previously, SVM is a binary classifier where instances are classified into two classes ± 1. However, in our experiment, our dataset has more than two classes, so we needed a classifier that is able to solve multiclass problems. SVM is extended to solve multiclass classification by combining multiple binary classifiers. The most common methods for solving multiclass classification using SVM are:

i. **One-against-all:** for an n-class problem, an n-binary SVM is built and trained to distinguish the data in an n-class from the samples in all remaining (n-1) classes. A decision function is made by each classifier to calculate the test data belonging to that class. The test data is classified to the class that obtained the highest decision function value.

ii. **One-against-one:** for each pair of classes, a binary SVM classifier is built to classify data between that pair of classes. When an instance is assigned to one of the two classes, the class gets a vote. The class with the highest number of votes is considered as the true class for that instance.

3.4.2 SVM Regression

SVM can also be applied to regression problems. Support Vector Regression (SVR) is a technique to find a function that maps from an input object to a real number based on the training data. Like SVM, SVR has the same method of margin maximisation and kernel trick for non-linear mapping. For a training data point (x,y) x represents the actual QoS values for a given dataset and y represents the predicted QoS attributes. In the case of linear function, the goal is to find a function in the form $F(x) \Rightarrow w \cdot x + b$ that has at most ε deviation from the target y_i for all training data. It means that errors are ignored as long as they are less than ε deviation. The idea is to make *w*, which is the weight vector, as small as possible to ensure a flat function. One way to do this is to minimise the norm, i.e. $\|w\|^2 = (w,w)$. We can write this problem as: minimise $\frac{1}{2}\ ||w||^2$, under the constraints $y_i - (w,x_i) - b \le \varepsilon$ and $(w, x_i) + b - y_i \le \varepsilon$. In the case of a non-linear function, data points must be mapped into a higher-dimension space similar to the process used in the non-linear SVM classifier that was explained in the previous section.

3.5 Summary

The beginning of this chapter emphasised the need to consider quality attributes in the selection process. We examined our recommended methodology towards selection of services based on the dataset given by the user. One way to ensure high quality results for data analytic services is to consider dataset characteristics or the meta-features of the processed data. We used a meta-learning algorithm to incorporate meta-features in the selection process of clustering services as an example of data analytic services.

CHAPTER 4

EXPERIMENTS

4.1 Introduction

In this chapter, we first discuss our experimental design, followed by the implementation details and then proceed to our results. We compare the prediction of QoS values as well as the best service name for each dataset in the experiment using five machine-learning models (SVM, KNN, MLP, Naïve Bayes and C4.5 decision tree algorithm). SVM was successfully used in predicting algorithm performance in a similar context [32, 33, 44]. MLP was also previously implemented as a meta-learner in a similar fashion [34]. KNN was previously used successfully on the same datasets used in this work, and this potentially allows us to compare the results of the different methodologies [45]. Naïve Bayes and the C4.5 decision tree algorithm are simple and straightforward classifiers, as we wanted to examine whether simple classifiers can work as well as more complex algorithms in this problem. We also considered the impact of the number of meta-features on prediction accuracy; the details of meta-features used were explained in the previous chapter.

4.2 Experiment Design

Our aim is to prove the importance of data characteristics, which we call here meta-features, for the selection process of clustering services. For this purpose, we incorporated five simple and two statistical meta-features for our datasets. We compared the performance when we used only five meta-features and when we used all seven of them. We check by including more meta-features to see whether we get a more accurate result. Many of the statistical or information-theoretic features are hard to measure for real datasets. If by only considering the simple features which are usually available for real datasets we can still achieve reasonable accuracy, then we can effectively apply the meta-learning approaches for data analytic service selection in real scenarios. Another major goal of our experiment is to discover which machine-learning model works the best with our dataset. This mainly refers to prediction accuracy, e.g. which algorithm is the most accurate in predicting the recommended service name or QoS values and has the lowest error rate. As mentioned previously, we used services that implement several data-clustering algorithms; however, any other data mining services can be used as well, such as data classification and data regression services.

The entirety of the experiment was conducted on an Apple MacBook Air 2013 with the following configuration: 1.3GHz dual-core Intel Core i5 (Turbo Boost up to 2.6GHz) with 3MB shared L3 cache, 4GB of 1600MHz LPDDR3 on-board memory and 128GB flash storage.

4.2.1 Dataset Collection

We started our experiment by using synthetic datasets that were used in a different project done by a member of our group, to compare the potential results of the two different methodologies used [45]. The decision to use synthetic as opposed to real datasets was mainly due to the fact that we can generate a much wider variety of datasets and meta-features to test our methodology. Only a small number of real datasets that could be used in our research is available in the UCI repository [46]. In our opinion, it is hard to study the impact of meta-features on system performance using a small number of datasets. Therefore, we used a synthetic data generator provided by the Weka toolkit, version 3.6.10 [2]. We used 560 datasets with different meta-features, i.e. dataset size, number of dimensions, type of data (nominal and numeric), data distribution pattern and percentage of missing data. The size of these datasets ranges between 500 and 5000, and the range of the numbers of dimensions is between 5 and 70. These datasets differ in the type of data they hold, including nominal and numeric types. The distribution pattern of the datasets is either “grid” or “random”.

For the statistical meta-features, we created a program using MathWorks MATLAB® 2014b [47] that runs each numeric dataset and calculates the average kurtosis and average skewness of the dataset. For the nominal datasets, since we cannot calculate the kurtosis and skewness for them, we assigned zero for these meta-features. The value of skewness for the dataset ranges between -0.146, which means that the data is spread out to the left of the mean by 0.146, and +0.463, which means that the data is spread out to the right of the mean by 0.463. The value of the kurtosis is between 0 and 5.35. The kurtosis of the normal distribution is 3, so if the kurtosis of a dataset is less than 3 it means that the distribution of the data is flat compared to the normal distribution, and if the kurtosis of a dataset is greater than 3 it means that the data has a sharper peak compared to normal distribution. The values of the statistical meta-features are then put together with the other meta-features.

Our complete dataset consists of seven attributes, comprising five numeric attributes which are Dimensions, DatasetSize, MissingData, Average_skewness and

Average_kurtosis and two nominal attributes, Type and Pattern. Table 4-1 shows a sample of our complete dataset for the classification problem.

Table 4-1: Sample for Classification Dataset

ID	Dimension	Dataset Size	Missing Data %	Skewness (Average)	Kurtosis (Average)	Type	Pattern	Service Name (Class)
440	5	692	3	0.37	1.96	Numeric	Random	Xmeans
441	20	906	1	0.00	1.90	Numeric	Random	FarthestFirst
442	18	945	3	0.00	1.91	Numeric	Random	FarthestFirst
443	6	588	3	0.21	2.01	Numeric	Random	EM
444	11	725	1	0.08	1.69	Numeric	Random	FarthestFirst
750	63	856	3	0.03	5.33	Numeric	Grid	EM
751	67	575	0	0.01	4.64	Numeric	Grid	sIB
868	6	755	3	0.00	0.00	Nominal	Grid	EM
869	9	564	0	0.00	0.00	Nominal	Grid	HierarchicalClustering _WARD

As shown in the above table, for classification algorithms the target value is the recommended clustering algorithm (Service Name). Table 4-2 shows a sample of the regression dataset where we use Accuracy and Latency as the target values, and we filter the service name for one service at a time for each regression function.

Table 4-2: Sample for Regression Dataset

ID	Service Name	Dimension	Dataset Size	Missing Data %	Type	Skewness (Average)	Kurtosis (Average)	Pattern	Accuracy	Latency (ms)
514	DBSCAN	10	2958	1	Numeric	-0.03	2.30	Random	16.67	1595.80
521	DBSCAN	6	2604	0	Numeric	0.20	1.99	Random	16.67	831.20
540	DBSCAN	33	2754	15	Numeric	-0.11	2.12	Random	16.67	3730.50
917	HierarchicalClustering _SINGLE	40	924	20	Nominal	0.00	0.00	Grid	16.88	5410.40
742	HierarchicalClustering _SINGLE	61	846	5	Numeric	0.05	4.65	Grid	16.9	3202.60
742	HierarchicalClustering _AVERAGE	61	846	5	Numeric	0.05	4.65	Grid	16.9	2132.10
809	FarthestFirst	63	2550	5	Numeric	0.03	4.61	Grid	20.16	54.60
962	KMeans	35	2904	20	Nominal	0.00	0.00	Grid	23.66	239.44
862	CLOPE	5	540	5	Nominal	0.00	0.00	Grid	24.26	388.00

4.2.2 QoS Computation

We used services created in [48] that apply various clustering algorithms available through the Weka Toolkit API. These services are built to work on the web and implemented as RESTful services. By applying these services on the datasets, we can collect QoS values of the services for each dataset. Table 4-3 shows the clustering algorithms used in these services.

Table 4-3: List of services

No.	Clustering Algorithm used in the service
1	CLOPE
2	Cobweb
3	DBSCAN
4	EM
5	FarthestFirst
6	SimpleKMeans
7	XMeans
8	sIB
9	Hierarchical Clustering with SINGLE link
10	Hierarchical Clustering with COMPLETE link
11	Hierarchical Clustering with AVERAGE link
12	Hierarchical Clustering with WARD link

In our experiment, we computed two QoS attributes for all the services for each dataset – latency and accuracy:

a. **Accuracy:** used to determine the quality of the clustering results. It is the accuracy of a clustering service on a dataset, or the percentage of correctly clustered instances.

b. **Latency:** the time it takes the service to build the clustering model for a given dataset.

After applying the datasets to each service, we did several steps to preprocess our dataset to be in the correct format for the model inputs. First, we normalized the accuracy and latency values for all datasets and for each service. Then, the normalized QoS values were combined using the utility function explained earlier to get the overall score. In the utility function, we used equal weights for both of the QoS attributes, but different QoS weights can be used based on the importance of the attribute. The overall scores are then ranked from smallest value to the largest. Each score is matched to its corresponding service name. Last, the service with the highest score is considered as the class service name. These service names are the ones that will be predicted by each classifier.

4.2.3 Experiment Settings

In all our experiments, we used the 10-fold cross validation technique to split the datasets into training and test datasets. The datasets are randomly split into 10 samples: nine as training datasets, and one as a test dataset. We assessed the accuracy of the service selection result on these test datasets.

As mentioned previously, we considered two types of selection model. The first one is the classification model where each machine-learning model should predict the best clustering algorithm for the given dataset. In this model, we applied the following classifiers from Weka: SMO, IBK, MLP, Naïve Bayes and J48. The inputs of the models are: Dimensions, DatasetSize, MissingData, Average_skewness, Average_kurtosis, Type and Pattern. The output of the classification models is the class value for each dataset which is the recommended service for the given dataset. The output of the regression models is the predicted latency and accuracy for each service. We used the multiclass classifier option since the class attribute has more than two classes.

The second one is the regression model where each machine-learning model builds a regression function for each service name that learns the relationship among the meta-features to predict the QoS values. Then these predicted QoS values are used in the utility function to find the service with the highest utility score that then becomes the recommended service for the given dataset.

4.2.4 Evaluation Metrics

To evaluate our experimental results, we used three different metrics: Spearman's Rank Correlation (SRC), Relative Absolute Error (RAE) and the prediction accuracy of the classifier. The SRC coefficient [49] can be used to assess ranking accuracy by comparing the rankings for the services based on the actual and the predicted QoS values. We also considered the average SRC coefficient for each dataset. The SRC for the ranked list of services for a dataset *D* is shown in the equation below:

$$SRC(D) = 1 - \frac{6\sum_{i=1}^{p} d_i^2}{p(p^2-1)} \qquad (4.1)$$

where d_i is the difference between the actual and predicted ranking value for the *ith* service and *p* is the number of services being ranked. The value of SRC ranges from -1 to +1 and represents how similar the two rankings are. The closer the number to 1, the more similar are the two rankings.

We also used RAE [50] to assess the accuracy of the prediction of the QoS attributes. The RAE value for QoS attribute Q is computed as given in equation 4.2

$$RAE(Q) = \frac{\sum_{i=1}^{n} |p_i - a_i|}{\sum_{i=1}^{n} |a_i - \bar{a}|} \qquad (4.2)$$

where p_i represents the predicted QoS value and a_i is the actual QoS value for the *ith* service, and $\bar{a}$ is the average for the actual QoS values.

Another evaluation metric is the prediction accuracy of the classifier. This prediction accuracy is different from the QoS accuracy defined earlier. The prediction accuracy of the classifier means the percentage of correctly classified instances to the total number of instances in the testing set. The formula below shows the equation of the classifier accuracy used in our experiments.

$$Prediction\ Accuracy = \frac{Correctly\ Classified\ Instances}{Total\ number\ of\ Instances} \times 100 \qquad (4.3)$$

4.3 Results

In this section, we compare the prediction among the five models using the complete set of seven meta-features based on the different metrics described in the previous section.

We start with the various SVM models. In this experiment, we applied all the services on the set of 560 datasets. Then we used the accuracy and the time needed to complete the service (latency) in the utility function to identify the best service for each dataset. After that, we used the SMO classifier to predict the best service name for each dataset and compared the actual with the predicted service name. To evaluate the classifier performance, we use the classifier predicted accuracy and the RAE. We measured the overall RAE for each classifier by computing the average RAE over all the datasets for each service to assess our prediction quality. Table 4-4 represents the results for each SVM model. Figure 4-1 and Figure 4-2 present the prediction accuracy and the RAE, respectively, for the different SVM models when seven meta-features are considered.

Table 4-4: Results for Different SVM Models

SVM Model	Accuracy	RAE
SVM for Classification	78.21%	0.3291
SVM Classification via Regression	71.07%	0.4376
SVM Regression	66.61%	0.4797

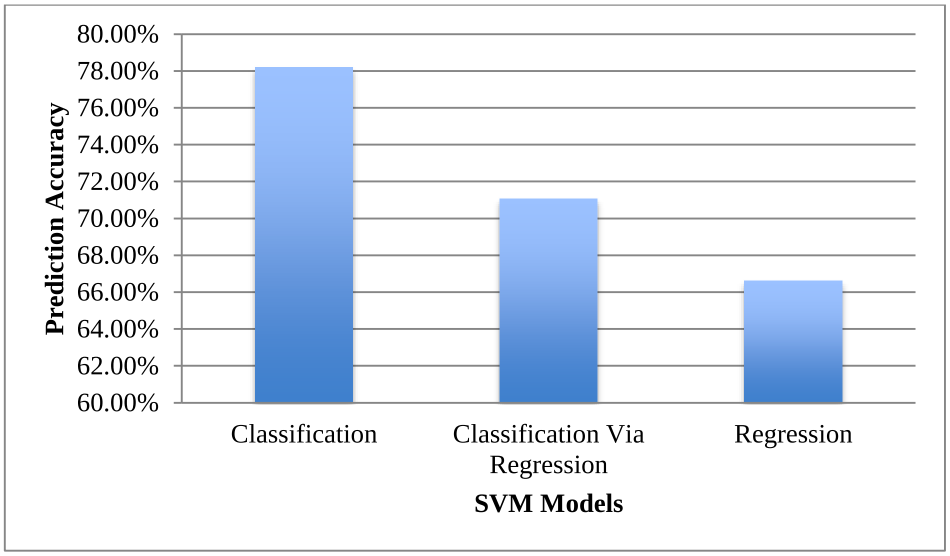

Figure 4-1: Prediction Accuracy for SVM Models

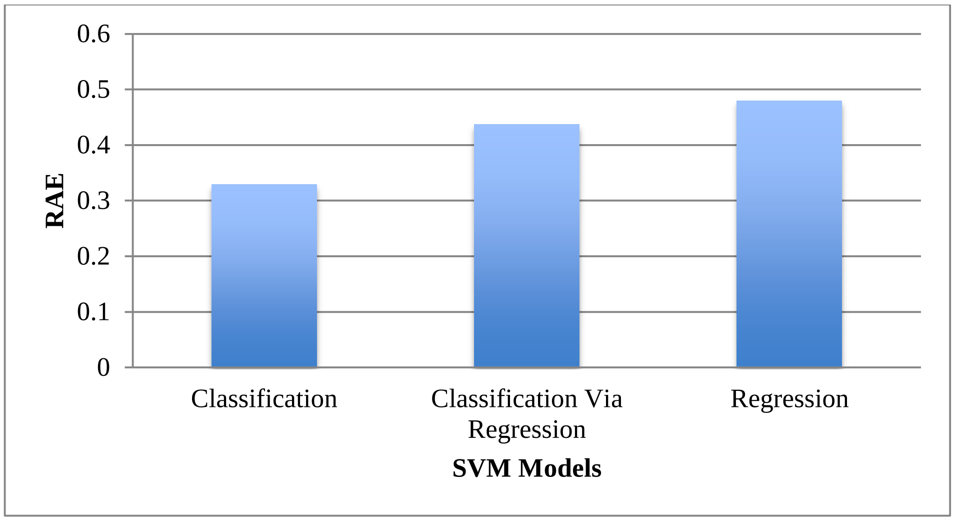

Figure 4-2: RAE for SVM Models

From this experiment, we can see that the SVM classification model is more accurate in predicting the best service than the SVM regression model. The above figure also indicates that SVM for classification has a lower error rate than SVM regression.

Our second experiment focuses on choosing the best classifier for each selection model: classification and regression. For the classification problem, the results for the five machine learning models are presented in Table 4-5.

Table 4-5: Results for Classification Models

Type	Model	Accuracy	RAE
Classification	SVM	78.21%	0.3291
	KNN	72.68%	0.4184
	MLP	75.17%	0.4217
	Bayes	68.39%	0.5029
	Decision Tree	74.28%	0.4405

Figures 4-3 and 4-4 compare the prediction accuracy and error rates, respectively for each of the classification model.

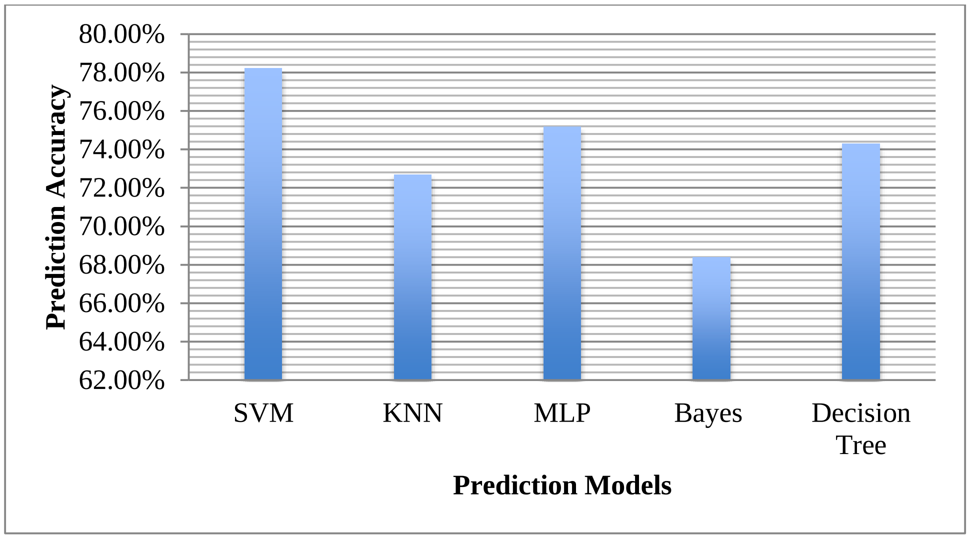

Figure 4-3: Prediction Accuracy for Classification Models

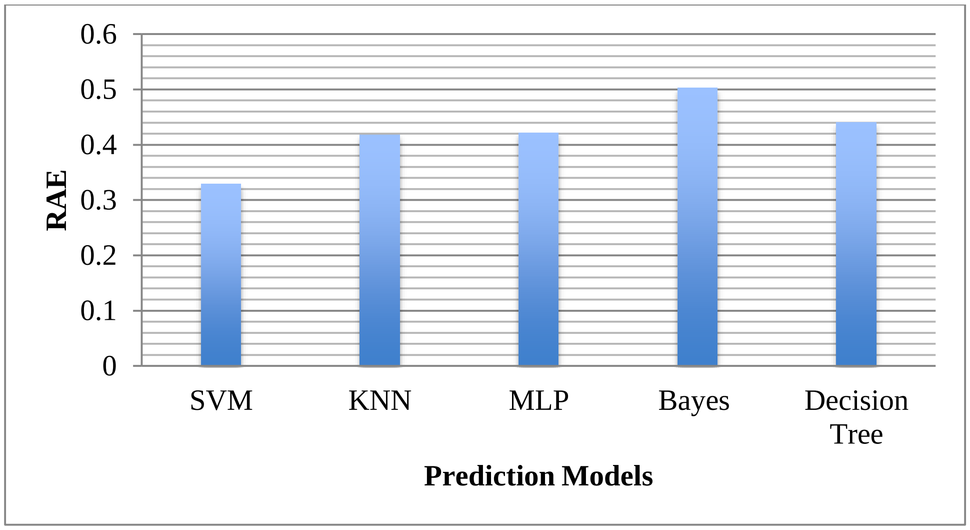

Figure 4-4: RAE for Classification Models

The results show that the SVM has the highest accuracy in assigning the best service name for each dataset with the lowest error rate.

Our next experiment is the regression problem. Since Naïve Bayes and the J48 decision tree cannot usually handle numeric classes, we only use the three machine learning models, SVM, KNN and MLP, that can be used for regression prediction. Two types of experiment were done in this part. First is classification via regression, which is for converting the target into a binary class and building a regression model for each class. Second is the regression method where a regression function is built for each service name to predict QoS attribute values.

As mentioned before, in the regression problem, we use the predicted QoS values in the utility function to predict the best service. The RAE results for predicted QoS values (latency and accuracy) are presented in Table 4-6, and Figure 4-5 shows the comparison between RAE values of predicted QoS values for the regression models. The chart shows that SVM is better on predicting accuracy, and KNN is better on predicting latency.

Table 4-6: RAE for Predicted QoS Values

Model	Accuracy	Latency
SVM	0.4623	0.4971
KNN	0.5070	0.4125
MLP	0.5034	0.4902

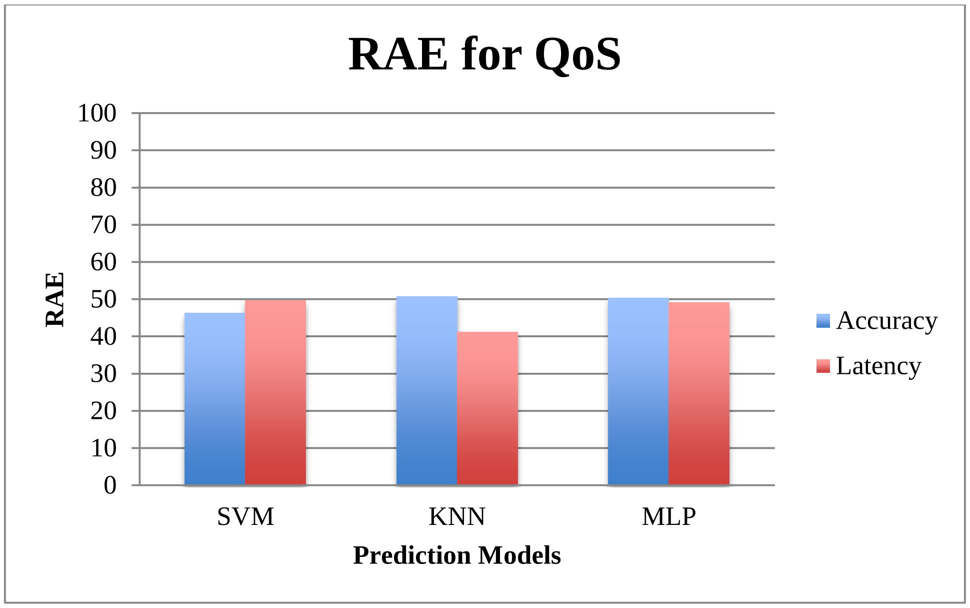

Figure 4-5: RAE for Predicted QoS for Regression Models

Table 4-7 presents the results for regression model. Despite the good results SVM showed in the classification problem, for the regression model it shows negative results compared with the other machine-learning models.

Table 4-7: Results for Regression Models

Type	Model	Accuracy	RAE
Regression	SVM	66.60%	0.4797
	KNN	71.25%	0.4597
	MLP	73.51%	0.4968

As shown in Figure 4-6 and Figure 4-7, the SVM has the lowest accuracy and high error rates in both types.

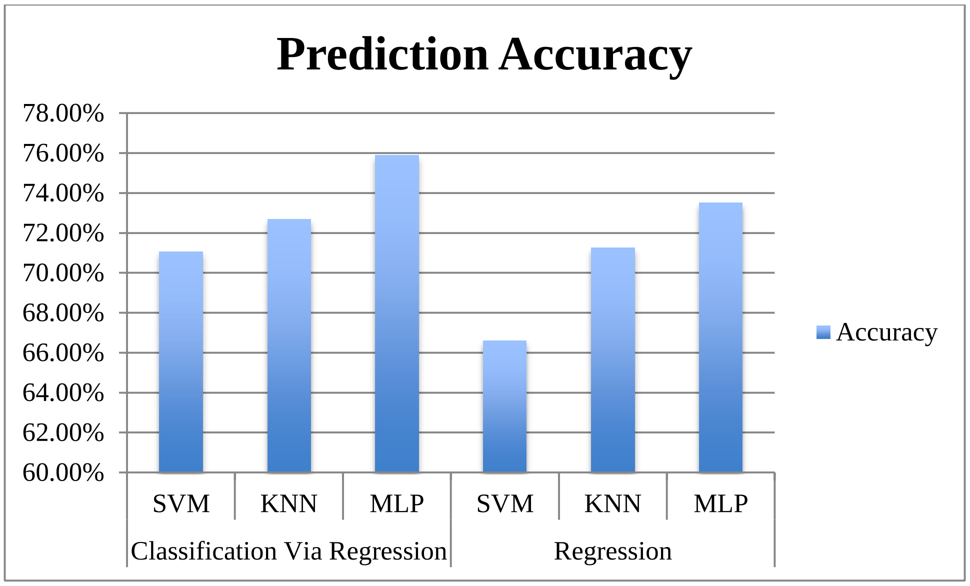

Figure 4-6: Prediction Accuracy for Regression Models

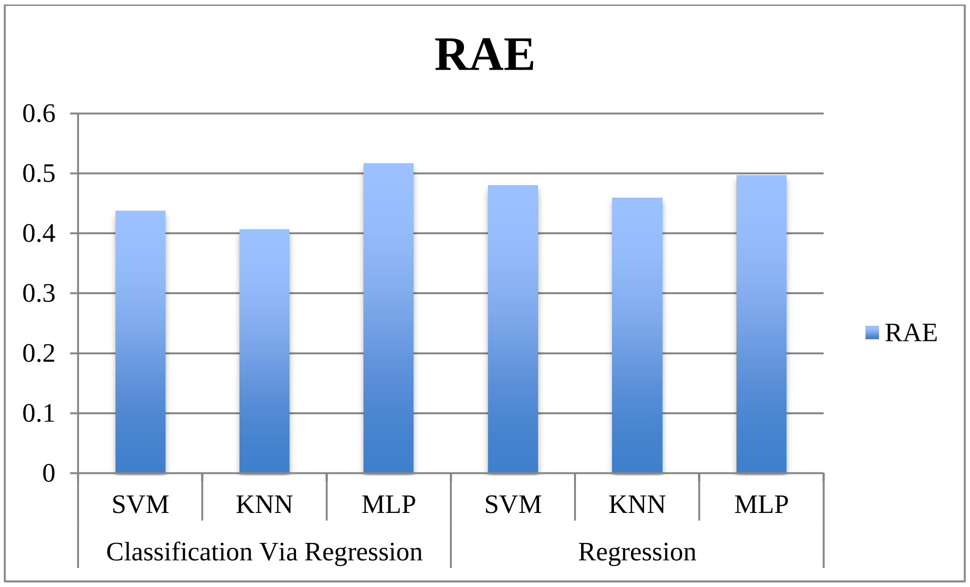

Figure 4-7: RAE for Regression Models

We also compute the overall SRC for each classifier in the regression problem by computing the average SRC for all the datasets. The overall SRC result for SVM, KNN and MLP is provided in Table 4-8. We compute the overall SRC as the average SRC over all the datasets. As seen in Figure 4-8, the three models showed similar SRC values, but considering the accuracy, RAE and SRC results, MLP presents the best classifier for regression problems. It shows the highest accuracy and SRC with an average error rate.

Table 4-8: SRC for Regression Models

Model	SRC
SVM	0.7686
KNN	0.7798
MLP	0.7823

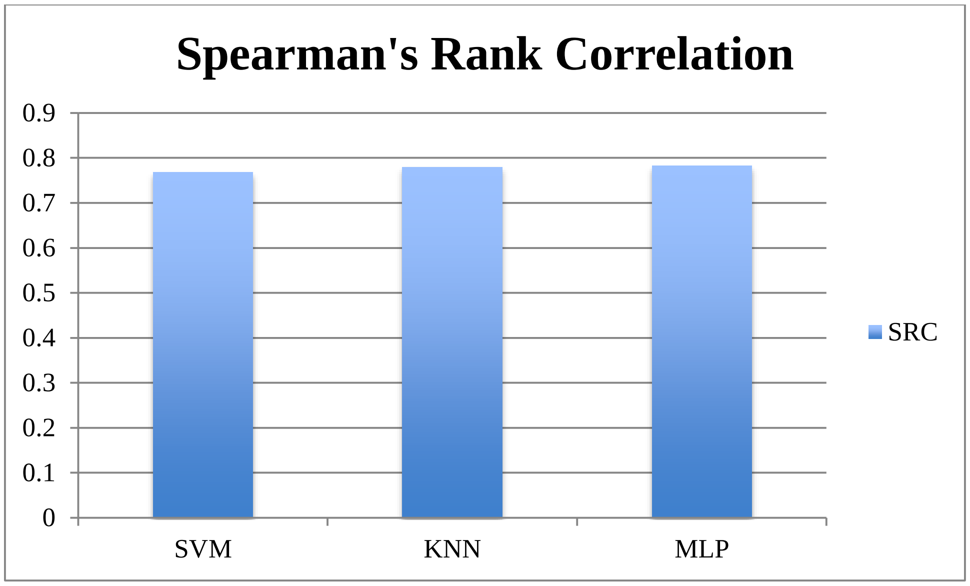

Figure 4-8: SRC for Regression Models

If we compare classification results against regression, SVM for classification presents the highest accuracy (78.21%) with the lowest RAE (0.3291). This means that in order to return the best service name for the user dataset for our system we should use the SVM classifier to get the most accurate prediction. However, if the goal is to return a ranked list of services to the user, a regression model is preferred because, unlike classification, regression models can also predict the QoS value so that ranking based on the predicted values can be generated.

4.4 Effects of the Number of Meta-features

As explained previously, the similarity between datasets is based on the set of meta-features. In this set of experiments, we study the influence of meta-features on the performance of the classifier. We do this by using only the five basic meta-features and eliminating the statistical meta-features. We worked with the same set of 560 datasets mentioned in Section 4.3. We compare five cases, each case being a model with two experiments: the first experiment considers only the five basic meta-features (size, dimension, missing data, data type and pattern), and the second experiment considers both basic and statistical meta-features (size, dimension, missing data, data type, pattern, skewness and kurtosis). We repeat this set of experiments for classification, classification via regression and regression. Figure 4-9 represents the prediction accuracy of each model using five meta-features versus seven meta-features for the classification problem.

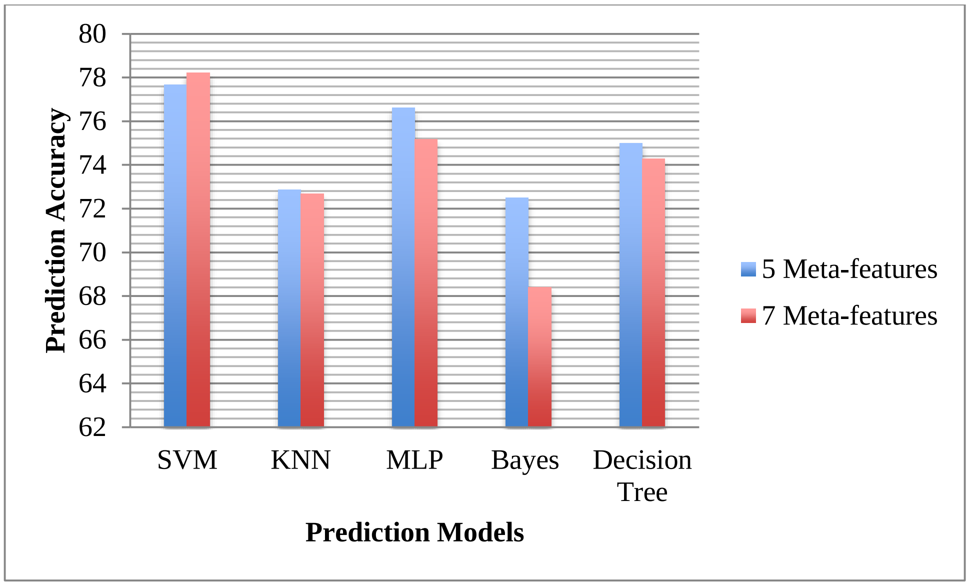

Figure 4-9: Prediction Accuracy of Five Meta-features vs. Seven Meta-features for Classification Models.

The chart indicates that, except for the SVM model, five meta-features provide a higher accuracy than seven meta-features. Even for the SVM model, the difference between the two results is quite small.

Regression and classification via regression models present a similar pattern. Figure 4-10 shows that KNN and MLP are performing better using five meta-features rather than using seven meta-features. The same pattern is shown in the average SRC values for the three cases presented in Figure 4-11.

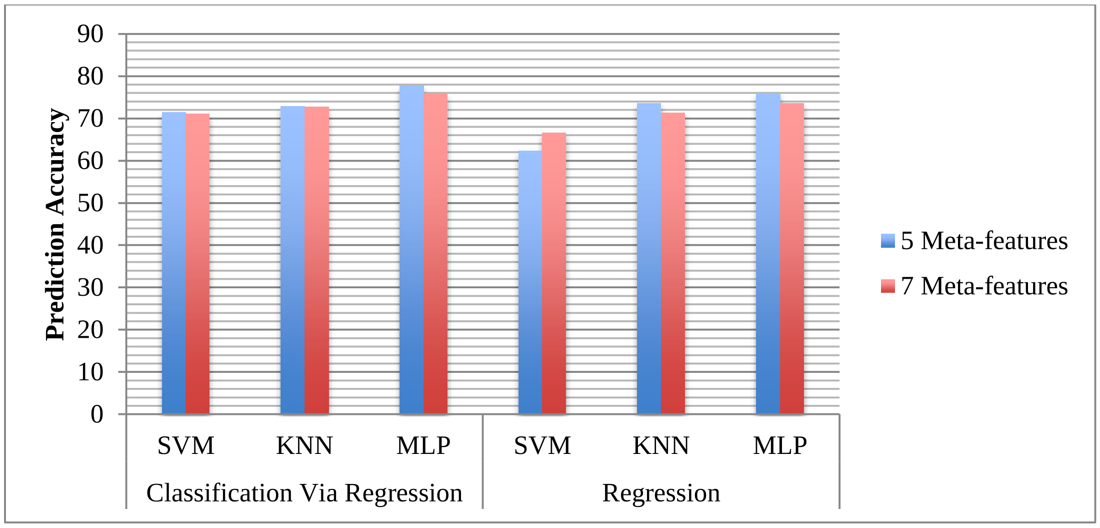

Figure 4-10: Prediction Accuracy of Five Meta-features vs. Seven Meta-features for Regression Models.

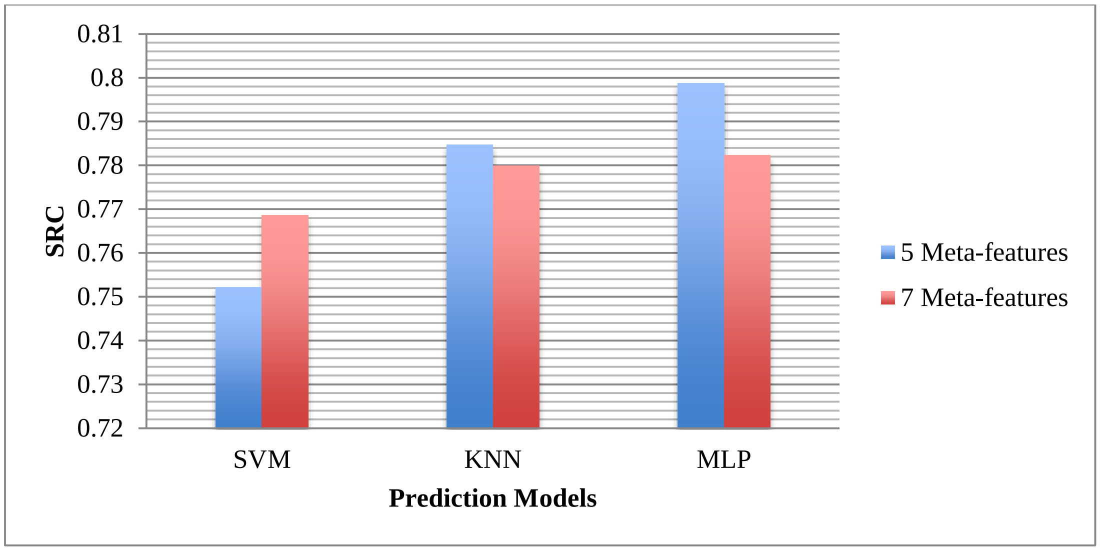

Figure 4-11: SRC of Five Meta-features vs. Seven Meta-features for Regression Models.

Hence we can say that the five basic and simple meta-features that can be collected for most datasets would be sufficient input to get an accurate prediction.

4.5 Summary

In this chapter, we first discussed our experimental design, followed by the implementation details and our results. We compared the prediction of QoS values as well as the best service name for each dataset in the experiment using classification or regression models. To evaluate our experimental results, we used three different metrics: SRC, RAE and the prediction accuracy of the classifier. We also considered the impact of the number of meta-features on the prediction accuracy.

We found that, out of the 5 classification models, SVM showed the best results in predicting the recommended service name for the user dataset. When it comes to regression models, MLP was the best classifier. It showed the highest accuracy and SRC with an average error rate. When comparing classification models against regression, SVM for classification presented the highest accuracy and the lowest RAE. In order to return the best service name for the user dataset, the SVM classifier should be used. However, if the goal is to return a ranked list of services to the user, a regression model is preferred because, unlike classification, regression models can also predict the QoS value so that a ranking based on the predicted values can be generated.

To study the influence of meta-features on the performance of the classifier, we repeated the same experiment using five versus seven meta-features. For classification models we found that, except for the SVM model, five meta-features provided a higher accuracy than seven meta-features. Regression and classification via regression models presented a similar pattern. And even for the SVM model, the difference between the results using five and seven features is small, and we could say that the accuracy is comparable. We conclude that the five basic and simple meta-features that can be collected for most datasets are sufficient to acquire an accurate prediction result.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

Data Analytics as a Service is one of the hottest topics in the area of web services. It combines the on-demand aspects of cloud computing with the analytic techniques proposed by data mining. Many analytic services are offered on the web, but the challenge is to choose the right service that suits the data on hand.

In this thesis, we addressed the problem of QoS-based web service selection for data analytic services. We used clustering services as an example of data analytic services. Meta-features can affect service performance, therefore we have considered data characterisation when making a service selection. A meta-learning approach can assist in defining the relationship of the dataset meta-features and the performance of clustering algorithms, thus we studied five machine-learning models as different meta-learners. Our meta-features include: dataset size, number of dimensions, percentage of missing data, type of data, data distribution pattern, and average skewness and kurtosis for continuous attributes.

In our selection process, we considered two different models: classification and regression. We incorporated two important QoS attributes: clustering accuracy and latency. We chose those QoS attributes because these values change for each service for a specific dataset.

Our contributions are as listed below:

- We provided a comparative study for a QoS-based data analytic service selection by five different meta-learners when applying them to recommend the best clustering service and predict QoS attributes for the given dataset.
- We studied the impact of the number of meta-features in the performance of the meta-learner predictions.

We found that SVM showed the best results in predicting the recommended service name for the user dataset. We also found that MLP is the most accurate model for estimating QoS attributes and in predicting the service name using regression functions.

We recommend considering only simple meta-features that can be collected for most datasets, as those proved to be sufficient to achieve good prediction accuracy for service selection.

5.2 Future Work

QoS-based web service selection is expanding and growing to include different theories and is being applied in different domains. There are a few aspects we would like to work on in the future. We acknowledge the importance of using real datasets in any research, therefore our next target is to apply our methodology and set of experiments on real datasets to check whether we observe similar performances in synthetic and real datasets. We have already collected 20 datasets from the UCI repository [46] and calculated their meta-features as the first step to pursuing our goal.

We will also work with more QoS attributes such as response time and throughput, and study how the five different meta-learners perform in predicting these

QoS attributes. We can incorporate user constraints in choosing different weight values for the QoS attributes based on the importance of the attributes.

We are interested in integrating a ranking mechanism based on the predicted QoS attributes in our selection model, especially using the meta-learner SVM since it has been used for ranking problems, and compare the results with different usage of SVM (classification and regression).

We can also work with a wider range of meta-features and consider some information-theoretic and model-based meta-features. In addition, we would like to study the relationship between the types of meta-features and the performance of the meta-learner model.

Finally, we can expand our evaluation metrics to include other measures such as Matthews's correlation coefficient and receiver operating characteristic areas to study deeply the differences between the five meta-learner performances.

REFERENCES

[1] Witten, I. and Hall, M., *Data Mining : Practical Machine Learning Tools and Techniques*. 3rd ed. Morgan Kaufmann Series in Data Management Systems. 2011, Amsterdam ; London: Morgan Kaufmann. xxxiii, 629 p.

[2] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I., *The Weka Data Mining Software: An Update.* SIGKDD Explorations, 2009. **11**(1): p. 10-18.

[3] Rcore-Team, *R: A Language and Environment for Statistical Computing.* The R Foundation for Statistical Computing, 2013.

[4] Hofmann, M. and Klinkenberg, R., *Rapidminer : Data Mining Use Cases and Business Analytics Applications*. Chapman & Hall/Crc Data Mining and Knowledge Discovery Series. xx, 465 pages.

[5] Talia, D., *Clouds for Scalable Big Data Analytics.* Computer, 2013. **46**(5): p. 98-101.

[6] Güemes, C., *Data Analytics as a Service: Unleashing the Power of Cloud and Big Data.* 2013.

[7] Microsoft, *Azure.* http://azure.microsoft.com/en-us/services/machine-learning/.

[8] Rastogi, R., *Machine Learning @ Amazon*, in *Proceedings of the 2nd IKDD Conference on Data Sciences*. 2015, ACM: Bangalore, India. p. 1-1.

[9] Zaharia, M., Chowdhury, M., Franklin, M., Shenker, S., and Stoica, I., *Spark: Cluster Computing with Working Sets*, in *Proceedings of the 2nd USENIX*

conference on Hot topics in cloud computing. 2010, USENIX Association: Boston, MA. p. 10-10.

[10] Ferrucci, D.A., *Ibm's Watson/Deepqa.* SIGARCH Comput. Archit. News, 2011. **39**(3): p. 1-10.

[11] Wang, Y. and Stroulia, E., *Structural and Semantic Matching for Assessing Web-Service Similarity.* First International Conference, Trento, Italy, 2003. Proceedings, 2003: p. 194-207.

[12] Platt, J., *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines.* MSR-TR-98-14, 1998.

[13] Flake, G.W. and Lawrence, S., *Efficient Svm Regression Training with Smo.* Mach. Learn., 2002. **46**(1-3): p. 271-290.

[14] Thirumuruganathan, S., *A Detailed Introduction to K-Nearest Neighbor (Knn) Algorithm.* 2010.

[15] Gutierrez, R., *L18: Multi-Layer Perceptrons.* CSCE 666 Pattern Analysis 2013.

[16] Witten, I. and Frank, E., *Data Mining : Practical Machine Learning Tools and Techniques*. 2nd ed. 2005, Amsterdam ; London: Elsevier. xxxi, 524 p.

[17] Liu, Y., Ngu, A., and Zeng, L., *Qos Computation and Policing in Dynamic Web Service Selection.* Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters 2004: p. 66-73.

[18] Al-Masri, E. and Mahmoud, Q., *Qos-Based Discovery and Ranking of Web Services.* IEEE, 2007: p. 529 - 534

[19] Yan, J. and Piao, J., *Towards Qos-Based Web Services Discovery.* ICSOC, 2008: p. 200-210.

[20] Menasc, D.A. and Dubey, V., *Utility-Based Qos Brokering in Service Oriented Architectures.* IEEE, 2007: p. 422 - 430

[21] Hang, C.W. and Singh, M., *From Quality to Utility- Adaptive Service Selection Framework.* ICSOC, 2010: p. 456-470.

[22] Ma, Q., Wang, H., Li, Y., Xie, G., and Liu, F., *A Semantic Qos-Aware Discovery Framework for Web Services.* 2008: p. 129-136.

[23] Ruz-Cortes, A., *Improving the Automatic Procurement of Web Services Using Constraint Programming.* International Journal of Cooperative Information Systems, 2005: p. 439.

[24] Kritikos, K. and Plexousakis, D., *Mixed-Integer Programming for Qos-Based Web Service Matchmaking.* IEEE TRANSACTIONS ON SERVICES COMPUTING, 2009. **2**: p. 122-139.

[25] Herssens, C., Jureta, I., and Faulkner, S., *Dealing with Quality Tradeoffs During Service Selection.* 2008: p. 77-86.

[26] Tran, V., Tsuji, H., and Masuda, R., *A New Qos Ontology and Its Qos-Based Ranking Algorithm for Web Services.* Simulation Modelling Practice and Theory, 2009. **17**(8): p. 1378-1398.

[27] Garg, S., Versteeg, S., and Buyya, R., *Smicloud: A Framework for Comparing and Ranking Cloud Services.* 2011: p. 210-218.

[28] Yu, Q. and Bouguettaya, A., *Computing Service Skyline from Uncertain Qows.* IEEE TRANSACTIONS ON SERVICES COMPUTING, 2010. **3**: p. 16-29.

[29] Skoutas, D., Sacharidi, D., Simitsis, A., Kantere, V., and Sellis, T., *Top-K Dominant Web Services under Multi-Criteria Matching.* Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology 2009: p. 898-909

[30] Lemke, C., Budka, M., and Gabrys, B., *Metalearning: A Survey of Trends and Technologies.* Artif Intell Rev, 2015. **44**(1): p. 117-130.

[31] Brazdil, P., Soares, C., and Da Costa, J., *Ranking Learning Algorithms- Using Ibl and Meta-Learning on Accuracy and Time Results.* Machine Learning, 2003. **50**: p. 271-277.

[32] Souto, M.D., Prudencio, R., Soares, R., Araujo, D., Costa, I., Ludermir, T., and Schliep, A., *Ranking and Selecting Clustering Algorithms Using a Meta-Learning Approach.* Neural Networks, 2008. IJCNN 2008. : p. 3729 - 3735

[33] Guerra, S., Prudencio, R., and Ludermir, T.B., *Predicting the Performance of Learning Algorithms Using Support Vector Machines as Meta-Regressors.* ICANN, 2008: p. 523-532.

[34] Soares, R., Ludermir, T., and Carvalho, F., *An Analysis of Meta-Learning Techniques for Ranking Clustering Algorithms Applied to Artificial Data.* ICANN, 2009: p. 131-140.

[35] Handl, J., *Cluster Generators for Large High-Dimensional Data Sets with Large Numbers of Clusters.* ICANN, 2009.

[36] Ferrari, D.G. and Castro, L.N.D., *Clustering Algorithm Recommendation: A Meta-Learning Approach.* SEMCCO, 2012: p. 143-150.

[37] Prudêncio, R., De Souto, M., and Ludermir, T., *Selecting Machine Learning Algorithms Using the Ranking Meta-Learning Approach*, in *Meta-Learning in Computational Intelligence*, N. Jankowski, W. Duch, and K. Grąbczewski, Editors. 2011, Springer Berlin Heidelberg. p. 225-243.

[38] Brazdil, P., Carrier, C.G., and Soares, C., *Metalearning: Application to Data Mining.* 2008.

[39] Martinez, W. and Martinez, A., *Computational Statistics Handbook with Matlab*. 2nd ed. Chapman & Hall: Crc Computer Science and Data Analysis Series. 2008. xxiii, 767 p.

[40] Vapnik, V., *The Nature of Statistical Learning Theory*. 2000: Springer.

[41] Boswell, D., *Introduction to Support Vector Machines.* 2002.

[42] Weston, J., *Support Vector Machine and Statistical Learning Theory.* NEC Labs America.

[43] Yu, H. and Kim, S., *Svm Tutorial: Classification, Regression and Ranking*, in *Handbook of Natural Computing*, G. Rozenberg, T. Bäck, and J. Kok, Editors. 2012, Springer Berlin Heidelberg. p. 479-506.

[44] Beverly, R., Sollins, K., and Berger, A., *Svm Learning of Ip Address Structure for Latency Prediction*, in *Proceedings of the 2006 SIGCOMM workshop on Mining network data*. 2006, ACM: Pisa, Italy. p. 299-304.

[45] Jain, N., *Data Dependent Web Service Selection.* MSc Computer Science Thesis, Ryerson University, 2015.

[46] Lichman, M., *Uci Machine Learning Repository* 2013.

[47] Inc., T.M., *Matlab and Statistics Toolbox Release 2014b.* 2014.

[48] Rahman, S., *A Test-Bed for Qos-Based Data Analytic Service Selection in the Cloud.* MSc Computer Science Thesis, Ryerson University, 2015.

[49] Sun, Q. and Pfahringer, B., *Pairwise Meta-Rules for Better Meta-Learning-Based Algorithm Ranking.* Machine Learning, 2013. **93**(1): p. 141-161.

[50] Gepsoft, *Analyzing Genexprotools Models Statistically*. http://www.gepsoft.com, 2014.