MODEL SELECTION IN STOCK CORRELATION NETWORKS

by

Narges Alipourjeddi

Bachelor of Science, Alzahra University, 2008

A thesis presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the program of

Applied Mathematics

Toronto, Ontario, Canada, 2018

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

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.

Model Selection In Stock Correlation Networks

Master of Science, 2018

Narges Alipourjeddi

Applied Mathematics

Ryerson University

Abstract

In this research, we construct market networks to study correlation between the price returns for all Dow Jones, NASDAQ-100 and S&P 100 indices that were traded over a period of time. We consider market networks, which have stocks as nodes and edges corresponding to correlated stocks. Specifically, a winner-take-all approach is used to determine if two nodes are adjacent. We identify that all networks based on the connecting stocks of highly correlated price returns display a scale-free degree distribution.

Additionally, we use features for representing different aspects of the network. The feature includes small connected sub-graphs with three and four vertices. We use an algorithm to count frequently the number of the graphlets for our mathematical models and our constructed networks. Each network is assigned an 8-dimensional vector.

We present a model selection algorithm based on supervised learning. Our algorithm classifies our market networks with the best fitting mathematical model.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Dr. Anthony Bonato for the continuous support of my MSc study and related research, and for his patience, motivation, and immense knowledge. His guidance helped me throughout the research and writing of this thesis.

Besides my supervisor, I would like to thank the faculty and staff in the Department of Mathematics at Ryerson University for their support. In particular, I would like to thank Dr. Dejan Delic and Dr. Alexey Rubstov for being part of my thesis committee. I would also like to thank Dr. Kathleen Wilkie for chairing my defense.

Finally, I must express my very profound gratitude to my parents and my spouse, Hadi, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Contents

List of Figures

CHAPTER 1

Introduction

1.1. Motivation

The study of networks appears in diverse disciplines through the analysis of complex relational data. The earliest known paper in this field is the famous Seven Bridges of Königsberg written by Leonhard Euler in 1736 [**23**]. Euler's mathematical description of vertices and edges was the foundation of graph theory, a branch of mathematics that studies the properties of pairwise relations between objects. The field of graph theory continued to develop and found applications in varied areas [**13, 49**].

In recent years, there has been a growing interest in financial networks. These networks not only help visualize the relationship between different financial entities (such as stocks, companies and hedge funds) they may also be used to predict future market conditions. We can consider different types of financial networks. Networks based on company ownership have been studied in the US stock market [**18**]. This study showed how company ownership was a power law distribution with small number of people controlling the mass of companies.

Another network studied involves board membership on companies listed on the New York Stock Exchange (NYSE) [**9**]. If a person sits on

the board of two companies, then there is a connection between the companies. Here also board membership satisfies a power law distribution with a small number of board members sitting in most boardrooms. This leads to a concentration of decision making in the hands of a few people.

Another type of financial network is based on stock price correlations [**19**]. It is used for observing, analysing and predicting the stock market dynamics. For studying the correlations of stock prices, the usual approach involves a procedure of finding correlation between each pair of time series of stock prices, and a subsequent procedure of constructing a network that connects the individual stocks based on the levels of correlation. The resulting networks are usually large and their analysis is complex. For reducing the complexity, we may apply a criterion and filtering process to connect the stocks based on their correlation [**32**].

In this thesis, we consider the full network of correlations based on the return prices of stocks and constructed cross correlation networks using a winner-take-all method for establishing edges of the network. In the winner-take-all method, we make a binary decision on connecting two stock prices according to the value of their cross correlation being larger than a threshold value. In our networks, the nodes are stocks and we study correlations between the return prices of all Dow Jones index, S&P 100 index and NASDAQ 100 index that were traded over the period of October 2016 to September 2017.

After constructing our synthetic networks, we use machine learning to determine which models on mathematics are close to the real-world, financial networks. For mathematical models, we consider Erdős-Rényi (ER) model and the Preferential Attachment model. We use a feature known as a graphlet (or sub-graph isomorphism type) for implementing a model selection method based on supervised learning. Linear Support Vector Machine (SVM) algorithm is one way to define a good classifier for the amount of separation between the two classes. The SVM algorithm predicts which model fits best for our networks with varied thresholds. We found that for large value of threshold, the stock correlation networks are scale-free and the degree distributions follow a power-law. However, for small value of the threshold, the networks tend to be fully connected and do not exhibit power-law distributions.

1.2. Graph theory

1.2.1. Graphs. A *graph* G is a non-empty set of points, called *vertices* that are connected by lines, called *edges*. Each edge is associated with a set consisting of one or two vertices called its *endpoints*. An edge with just one endpoint is called a *loop*. Two vertices that are connected by an edge are called *adjacent* and we can say that the vertices are *neighbours*. A vertex on which no edges are incident is called *isolated*. Let $G = (V, E)$ be a graph. We write $V = V(G)$ for the vertices of G and $E = E(G)$ for the finite set of edges. A *simple graph* is a graph that does

not have any loops edges, and we denote set of edges by

$$E(G) = \{(u, v) | u, v \in V, u \neq v\}.$$

The figure below is a geometric representation of the simple graph G with $V(G) = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $E(G) = \{v_1v_2, v_1v_3, v_2v_3, v_2v_4, v_5v_6\}$.

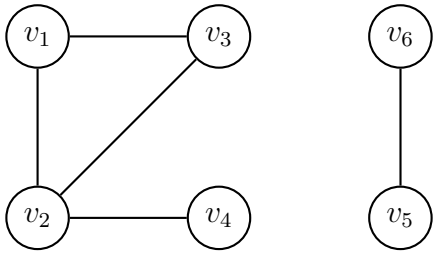


FIGURE 1.1. An example of graph.

Generally, we consider simple graph in this research and when we say "graph" we assume that the graph is simple unless otherwise specified. Vertices are also called *nodes*.

For a graph G, the number of vertices is *order* of graph and we denote by $|V(G)|$ and the *size* of graph is the number of edges and we indicate by $|E(G)|$. For instance, in Figure 1, the order of graph is $|V(G)| = 6$, and the size of graph is $|E(G)| = 5$.

An *undirected graph* is a graph in which edges have no orientation, the edge (v_1, v_2) is identical to the edge (v_2, v_1). A *directed graph* or *digraph* is a graph in which edges have orientations. An arrow (v_1, v_2) is

considered to be directed from v_1 to v_2; v_2 is called the *head* and v_1 is called the *tail* of the arrow.

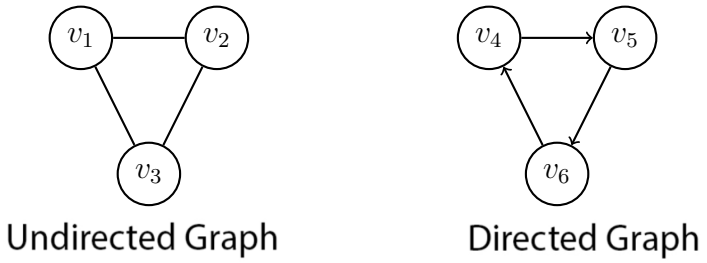


FIGURE 1.2. Examples of undirected and directed graphs.

Suppose that G and H be graphs and f is a bijective function, $f : V(G) \longrightarrow V(H)$, we say that G and H are *isomorphic* if for all $u, v \in V(G)$, uv is an edge in G if and only if $f(u)f(v)$ is an edge in H. If G and H are isomorphic, then we write $G \cong H$. For example, the following graphs are isomorphic.

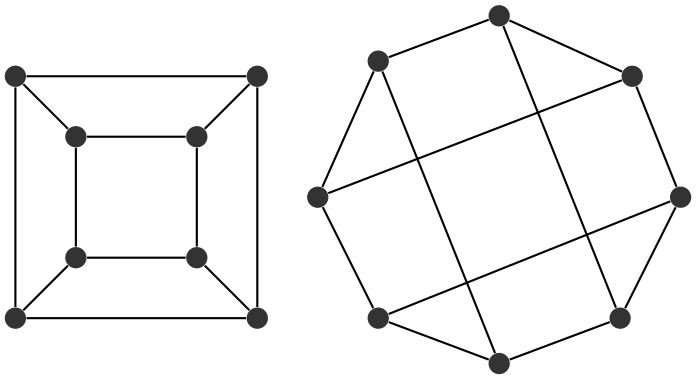


FIGURE 1.3. Isomorphic graphs.

Two graphs that are not isomorphic are said to be *non-isomorphic*. Figure 4 is a example of non-isomorphic graphs.

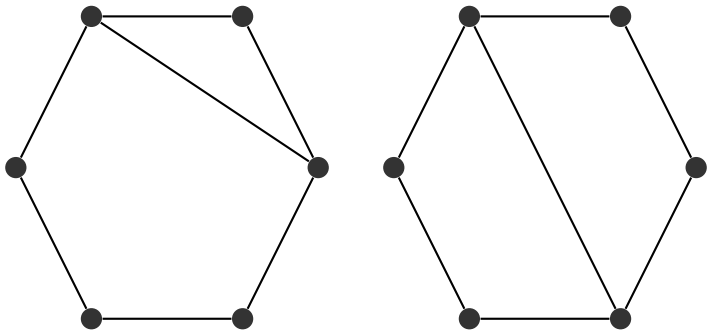


FIGURE 1.4. Non-isomorphic graphs.

A *sub-graph* S of a graph G is a graph such that $V(S) \subseteq V(G)$, and $E(S) \subseteq E(G)$. A sub-graph $S \subseteq G$ is a *spanning sub-graph* of G if $V(S) = V(G)$. An *induced sub-graph* S of a graph G has vertices S and $E(S) = E(G) \cap E(V(S))$. In an induced sub-graph $S \subseteq G$, the set $E(S)$ of edges consists of all edges belong to G such that the both endpoints of that edges are in S. We denote the sub-graph induced by S in G by $\langle S \rangle_G$.

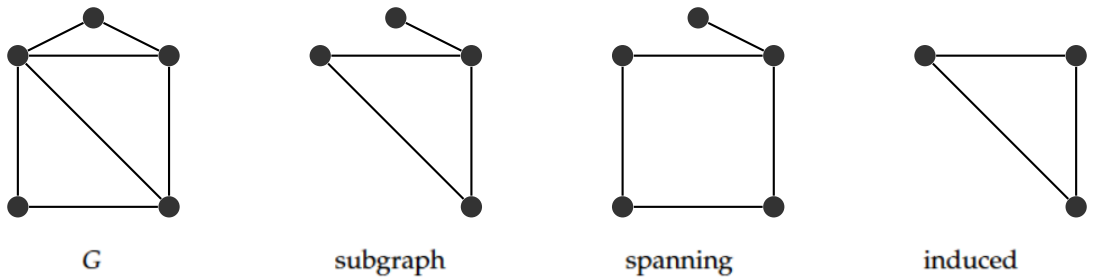


FIGURE 1.5. An example of sub-graph, a spanning sub-graph, and an induced sub-graph.

A graph G is called *connected* if there is a path between every pair of vertices. If the graph has an isolated vertex (a vertex with no incident edges), then graph is not connected. When the graph is not connected, it is called *disconnected* graph.

In Figure 6, the graph has two connected components. A *connected component* is a maximal (with respect to inclusion) connected induced subgraph.

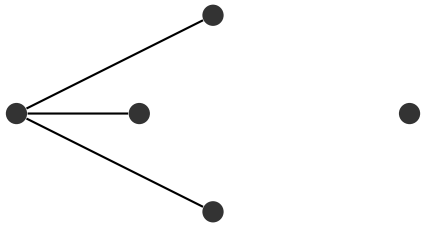


FIGURE 1.6. A disconnected graph.

Let $G = (V, E)$ be the graph and $v \in V$ the *degree* of a vertex v, written $\deg(v)$ is the number of edges that incident with v. The following theorem, due to Euler (1736) [**23**], tells that if several people shake hands, then the number of hands shaken is even. We may write the sum of the degrees in two equivalent forms. Let $d_1, d_2, ...$ be the degrees of vertices in G and n_i be the number of vertices with degree i. Then $d_1+d_2+d_3+... = n_1 + 2 \cdot n_2 + 3 \cdot n_3 + ...$; the Handshaking theorem tells us that each is equal to twice the number of edges. In particular, both sums are even.

THEOREM 1. *(Handshaking theorem) For each graph G*

$$\sum_{v \in V(G)} \deg(\mathrm{v}) = 2|E(G)|. \tag{1.1}$$

Moreover, the number of vertices of odd degree is even.

PROOF. When we sum the degrees, each edge is counted twice. □

1.2.2. Important classes of graph. A graph $G = (V, E)$ is called *regular* graph if are vertices have the same degree. A regular graph with vertices of degree k is called a *k-regular graph*. See Figure 7.

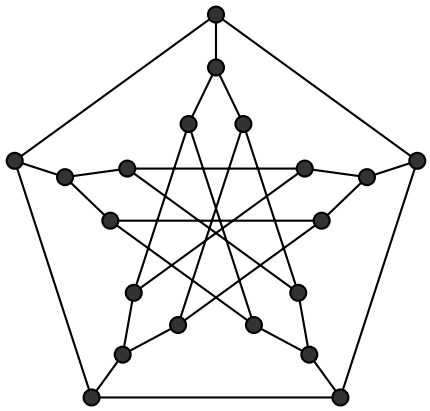


FIGURE 1.7. An example of 3-regular graph.

When any two vertices are adjacent and every pair of distinct vertices is connected by a unique edge, the graph is called *complete graph* and the complete graph with n nodes is denoted by K_n. Complete graphs are sometimes called *clique* graphs. The number of edges in complete graph is $\binom{n}{2} = \frac{n(n-1)}{2}$. Note that all complete graphs with n nodes are regular graph of degree $n - 1$.

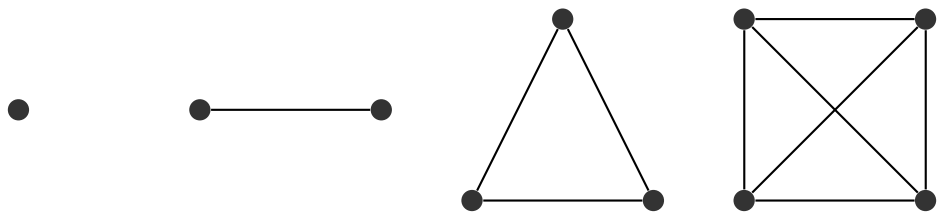


FIGURE 1.8. K_n for $n = 1, 2, 3, 4$.

The *path* P_n is the graph on n vertices that can be listed in the order $v_1, v_2, v_3, ..., v_n$ and with edges $v_1v_2, v_2v_3, v_3v_4, ..., v_{(n-1)}v_n$. The graph P_n has n vertices and $n-1$ edges.

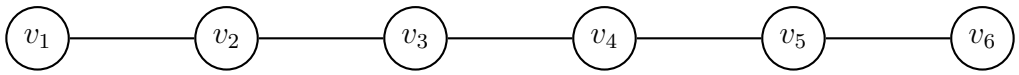


FIGURE 1.9. The graph P_6.

The *cycle* C_n is a graph with n vertices and n edges obtained from P_n by adding an edge between the two ends; it is the graph of a polygon with n sides. In a cycle graph, every vertex has degree 2.

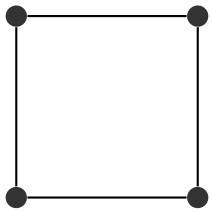


FIGURE 1.10. The graph C_4.

A *tree* is a connected graph by a unique path with no cycles. If G is a tree, then G is connected and the size of the graph is $|V(G)| - 1$. A *forest* is a graph with each connected component a tree.

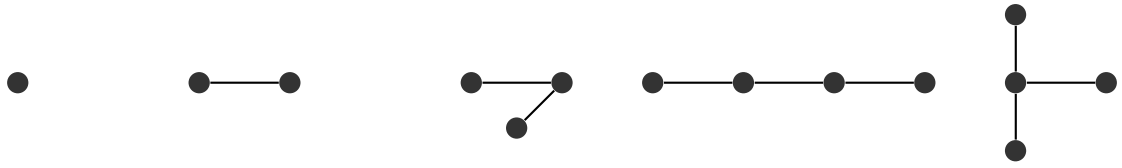


FIGURE 1.11. Trees with n=1,2,3,4.

1.3. Probability theory and complex networks

1.3.1. Degree distribution. The information from the degrees of nodes gives important clues into the structure of a network. The key concept for us is the distribution of degrees. This concept can be mathematically presented in terms of probability density function. The probability of a node having a degree k is $p(k)$ and if we plot $p(k)$ against k, then we obtain a distribution function.

In the simplest types of networks, we can find that most nodes in the network had similar degrees. For example, the random graph, in which each of n nodes is connected (or not) with independent probability p, has a binomial distribution of degrees k [**11**]:

$$p(k) = \binom{n-1}{k} p^k p^{n-1-k}. \tag{1.2}$$

However, in many real-world networks most nodes have a relatively small degree, but a few nodes will have very large degree. In a graph G of order n, let N_k be the number of nodes of degree k. The degree distribution of G follows a *power law degree distribution* if N_k is proportional to k^{-b}, for some range of k and for a fixed exponent $b > 2$ [**14**]. In particular, we have that

$$N_k \sim k^{-b} n.$$

1.3.2. Complex networks. A complex network is a graph that arises in real world networks such as computer networks, social, biological and also financial networks. There are general properties in complex networks. In this part, we describe briefly some features that appear to be common to complex networks.

(1) Large scale relative to order and size.
(2) Evolving over time.
(3) Have a power law degree distribution.
(4) Have the small world property, introduced by Duncan J. Watts and Steven Strogatz in 1998 [**46**]. The small world property in a graph of order n demands a low diameter of $O(\log n)$ and a higher clustering coefficient than a binomial random graph with same expected degree.

1.3.3. Models of networks. Models for networks uncover their hidden reality and can explain the generative mechanisms underlying them. There are existing models for complex networks but our focus is on two models:

i) Erdős-Rényi model,
ii) Preferential attachment model.

Erdős-Rényi

In 1959, Erdős and Rényi published an article in which they introduced the concept of a random graph [**22**]. First, take some positive integer n. Fix a positive integer n and real number $p \in (0, 1)$. We consider a set of n nodes. For each pair of nodes, we add an edge, independently and with probability p. The Erdős-Rényi model is denoted $G(n, p)$.

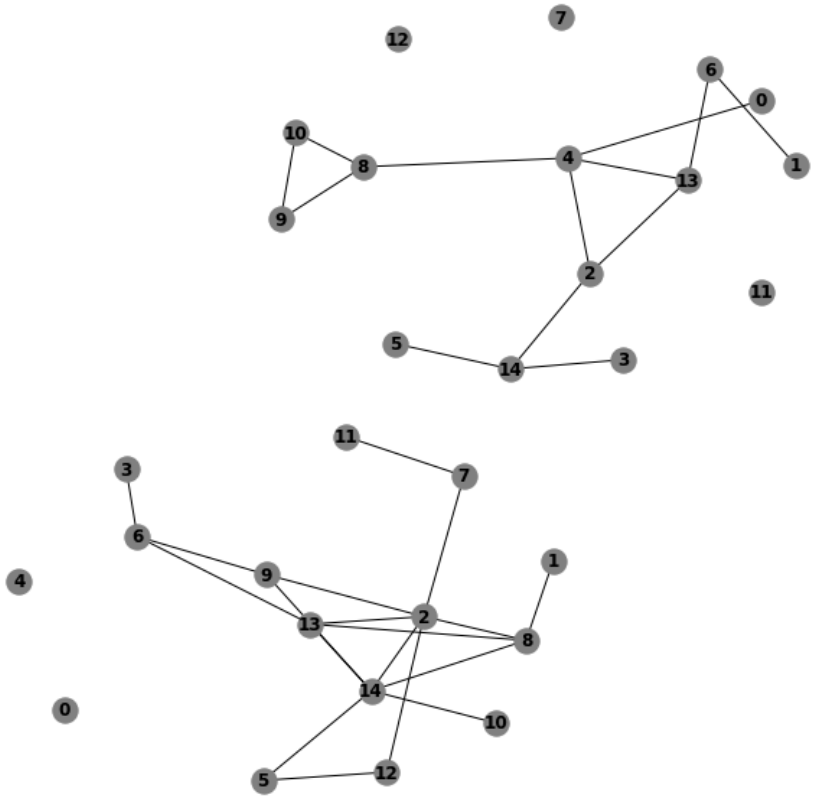


FIGURE 1.12. An example describing the Erdős-Rényi model on a graph with 15 nodes, with $p = 0.1$ on the left and $p = 0.2$ on the right.

Preferential attachment model

In 1999 Barabási and Albert proposed the application of preferential attachment to the growth of the World Wide Web [**5**]. First at time $t = 0$ we have $n = n_0$ initial vertices then we add a vertex on every step with

m edges such that $m < n_0$, that are preferentially attached to existing nodes with high degree. More precisely, we have $n_0 + t$ nodes and mt edges after t steps. The Barabási-Albert model denoted $BA(n, m)$.

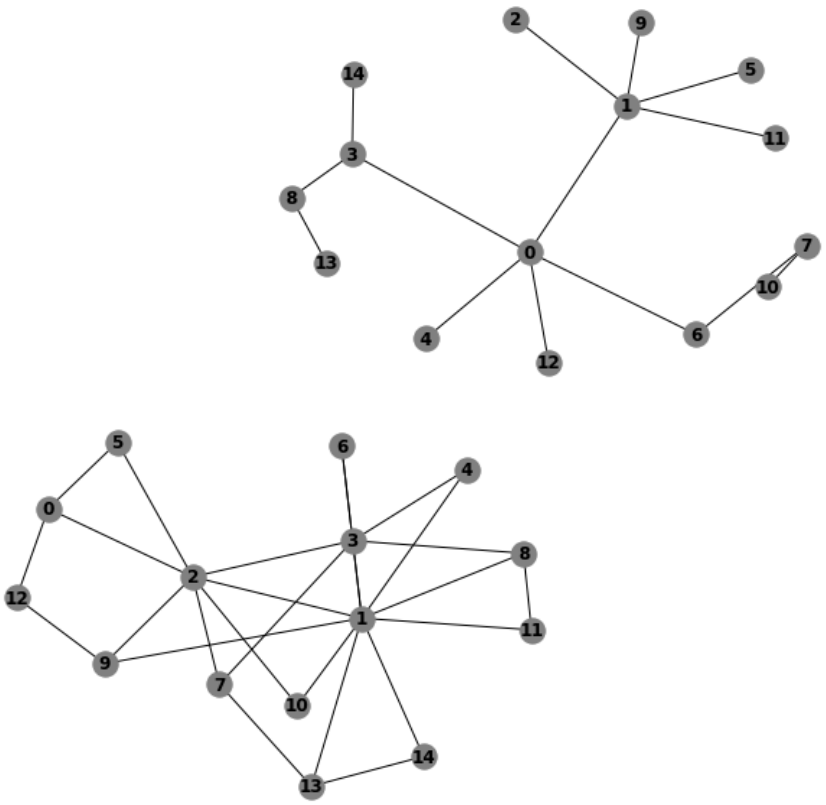


FIGURE 1.13. An example describing the preferential attachment process on a graph with 15 nodes, with $m = 1$ on the left and $m = 2$ on the right.

1.4. Summary of Thesis

The main goal of this thesis is to analyse stock correlation networks and survey best fit mathematical models for the networks. The thesis is composed of five chapters. Chapter 1 was introductory and reviewed

basic terminology of graph theory and networks. Chapter 2 considers machine learning that is a useful tool for model selection. Machine learning consists of algorithms that learn patterns in existing data and then predicts similar patterns in new data. We will explain machine learning methods and algorithms in this chapter. Chapter 3 consists of two parts and explains the mathematical models: Part 1 illustrates the Erdős-Rényi model and implements the model with Python. Part 2 analyses the Barabási-Albert model and we provide simulations of that model.

Chapter 4 is divided into three parts and provides an outline of selecting the best fit model for our networks. Part 1 describes that how to construct stock correlation networks and how extracted network information from Yahoo Finance. Part 2 focuses on a feature known as graphlet and the distribution of graphlet counts. We consider graphlets with three and four vertices. There are two possible non-isomorphic connected graphs on 3 vertices and six possible non-isomorphic connected graphs on 4 vertices. By counting that how many times every graph appears as an induced sub-graph in a network, we have a vector in 8-dimensional space. The final part recommends the SVM algorithm for finding the best model. The SVM algorithm chooses the hyperplane so that the distance from it to the nearest data point on each side is maximized. In the final chapter, we provide our conclusions and a set of open problems.

CHAPTER 2

Machine learning

2.1. Introduction

In 1959, Arthur Samuel, who worked in the field of computer gaming and artificial intelligence at IBM, was the first to use the term *machine learning* [**42**]. Machine learning is a field of study that gives computer systems the ability to learn with data, without being explicitly programmed [**30, 43**]. In 1990, machine learning became its own field and in 1997, Tom Mitchell gave a well-posed, more formal definition of the algorithms studied in the machine learning field: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." See [**33**].

Over the past two decades, machine learning has become one of the fundamental areas of computer science. The goal of machine learning is to understand the structure of data and fit that data into models that can be understood and utilized by users. With the increasing amounts of data becoming available, data analysis is a necessity for technological progress. Finding patterns in data up to now, was only possible by humans. If the data is massive, then the time taken to compute is increased,

and this is where machine learning is most powerful. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Fraud detection, on-line recommendation systems such as those from Amazon and Netflix, and even self-driving cars are examples of machine learning applications. Each of these applications perform complex mathematical calculations to perform their tasks.

In this chapter, we present the requisite background on the common machine learning methods, including supervised, semi-supervised and unsupervised learning, and common algorithmic approaches in supervised learning, including support vector machines. Also, we will explore which programming languages are typically used in machine learning.

2.2. Machine learning methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed. The common machine learning methods are:

(1) Reinforcement learning.

(2) Supervised learning.

(3) Semi-supervised learning.

(4) Unsupervised learning.

2.2.1. Reinforcement learning. In reinforcement learning, the learning system repeatedly observes the environment, performs an action and receives a reward. The goal is to choose the actions that maximize the future rewards. For example, consider teaching a dog a new trick; you cannot tell it what to do, but you can either reward or punish it if it does the right or wrong thing. It has to figure out what it did that made it get the reward or punishment, which is known as the *credit assignment problem*. We can use a similar method to train computers to do many tasks, such as playing backgammon or chess, scheduling jobs, and controlling robot limbs [**29, 34, 47**].

2.2.2. Supervised Learning. In the majority of supervised learning applications, the computer is provided with example inputs that are labelled with their desired outputs. The set of labelled examples used for learning is called *training data*. The training set consists of (feature, label) pairs, denoted by $\{(x_1, y_1), ..., (x_n, y_n)\}$. The goal of effective machine learning algorithms is to be able to recognize y for any new example with feature x. A supervised learning task is called *regression* when $y \in \mathbb{R}$ and called *classification* when y takes a set of discrete values.

There is a famous Iris flower data set that is an example of supervised classification machine learning problem [**3**]. We want to identify what type of flower we have based on different measurements, like the length and width of the petal. The data set includes three different types of flowers that all are species of Iris: Setosa, Versicolor and Virginia. Given 50 examples of each type, then we have a total of 150 examples. There are four features that describe each example: length and width of the sepal and petal.

Our goal is to find some function that maps data item in X to a label in Y summarized in a function $f : X \longrightarrow Y$. The predictor $f(x)$ use the training examples. For each training example, we have an input value x-train, for which a corresponding output, y, is known in advance. In practice, if the sepal length is x_1, sepal width is x_2, petal length is x_3 and petal width is x_4, then for each example the value of $f(x\text{-train})$ is given with the label y which is one of the species of Iris flowers. With enough training examples, we identify the function $f(x)$ that best maps the input to the desired output.

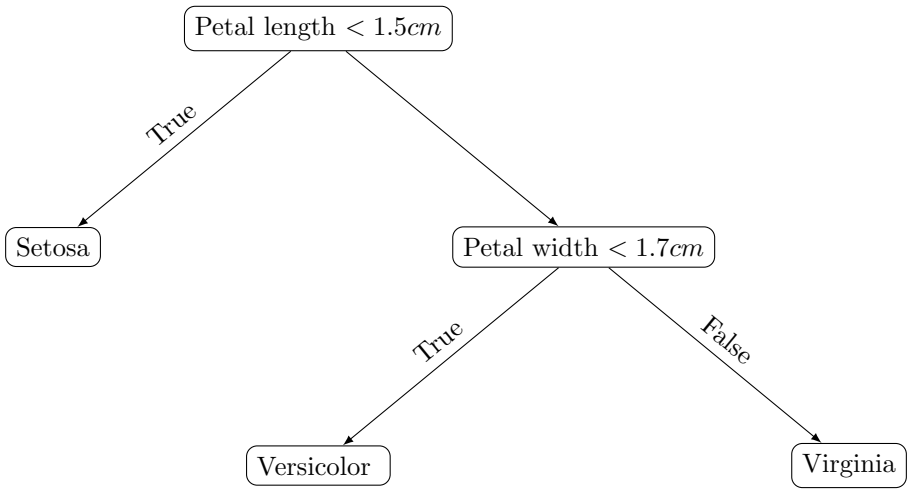


FIGURE 2.1. Decision tree for a sample of Iris data set.

Sample of Iris data set					
Data set order	Sepal length (x_1)	Sepal width (x_2)	Petal length (x_3)	Petal width (x_4)	Species (y)
1	5.1	3.5	1.4	0.2	Setosa
3	5	3.3	1.4	0.2	Setosa
4	7	3.2	4.7	1.4	Versicolor
5	5.7	2.8	4.1	1.3	Versicolor
6	6.3	3.3	6	2.5	Virginia
7	5.9	3	5.1	1.8	Virginia

To explain how this works, we include the above table and figure. The table describes measurements of the six training examples randomly from the Iris flower data set. Figure 2.1 exhibits the simple algorithm of

how the function $f(x)$ can predict the label for our new data from our training data.

There are many supervised learning algorithms that scientists have developed. These algorithms can be used to estimate the function f from the training data. We will examine the support vector machine algorithm next.

Support vector machine

Support vector machines (SVMs, also *support vector networks)* are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. For classifying data in support vector machine, a data point is viewed as a p-dimensional vector and we want to know whether we can separate such points with a $(p-1)$-dimensional hyperplane. This is called a *linear classifier*. We have a training dataset of n points of the form $(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_n}, y_n)$. Each $\mathbf{x_i}$ be a p-dimensional real vector and y_i be 1 or -1 that indicate the class of point $\mathbf{x_i}$. We explain how the algorithm works in the below sample binary classification problem with two informative features.

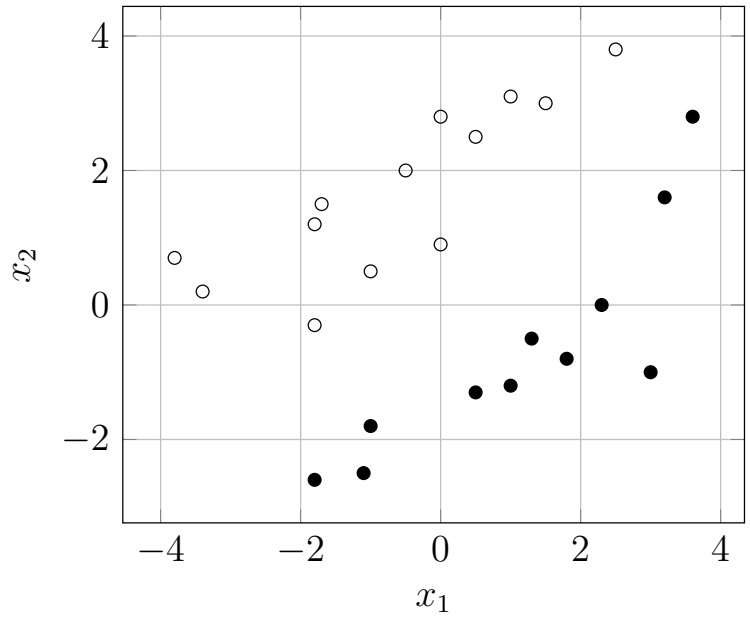


FIGURE 2.2. Binary classification problem with two features.

In Figure 2.2, we have a binary problem that has two different classes y, and where each data is represented by two informative features, x_1 and x_2. We will explain how we can predict the class of a new data. The linear classifier defined by computing a linear function of x_1 and x_2. It means that we input $x \longrightarrow \boxed{f} \longrightarrow y$ and for output we have two class values (1 or -1). For example, the equation can be $f(x, w, y) = \text{sign}(wx + b)$ such that the w is vector of weights and b is a bias term that gets added in. For this example, we define the equation $x_1 - x_2 = 0$. This corresponds to $w = (1, -1)$ and $b = 0$.

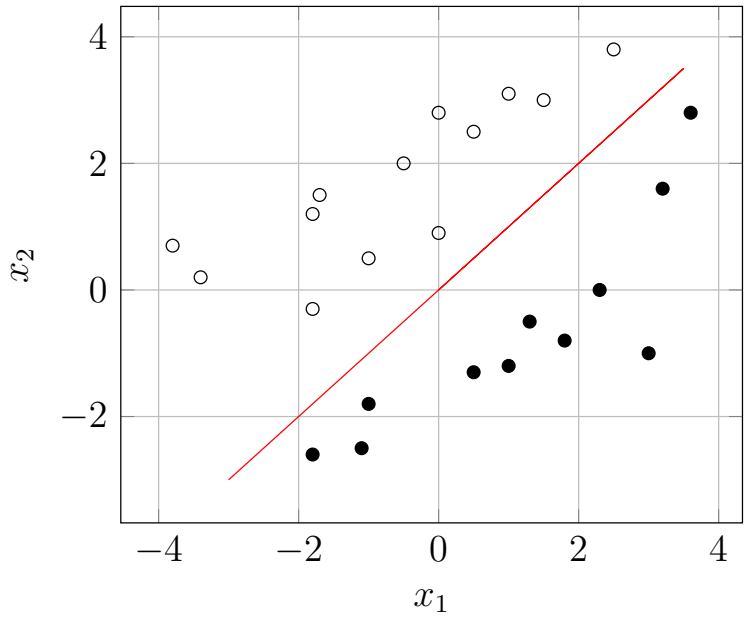


FIGURE 2.3. Decision line $x_1 - x_2 = 0$ as a classifier.

This line can be used as a decision rule for the classifier. Suppose that we want to classify the point $(-1.8, -2.5)$. We substitute these values into the function, and then apply the weights and the bias term that describes the decision boundary.

$$\begin{aligned} f &= ((-1.8, -2.5), w, b) \\ &= \text{sign}(1 \cdot (-1.8) + (-1) \cdot (-2.5) + 0) \\ &= \text{sign}(-1.8 + 2.5) \\ &= \text{sign}(+0.7) \\ &= 1. \end{aligned} \tag{2.1}$$

Therefore, in this case, because 0.7 is greater than zero, the output from the decision function would be 1 and the point classify the class one. If we consider a different point, such as $(-3.4, 0.2)$ and then substitute these values into the function, then we obtain the following:

$$\begin{aligned} f &= ((-3.4, 0.2), w, b) \\ &= \text{sign}(1 \cdot (-3.4) + (-1) \cdot (0.2) + 0) \\ &= \text{sign}(-3.4 - 0.2) \\ &= \text{sign}(-3.6) \\ &= -1. \end{aligned} \tag{2.2}$$

The classifier predicts a class -1 for the point. We observe that by applying a simple linear formula, we have been able to produce a class value for any point in this two dimensional feature space. That is one of the rules that we use for converting a data with its features to an output prediction.

One way to define a good classifier is to reward a classifier for the amount of separation that can provide between the two classes. We need to define the concept of *classifier margin*. For our given classifier, the margin is the width that the decision boundary can be increased before hitting a data point.

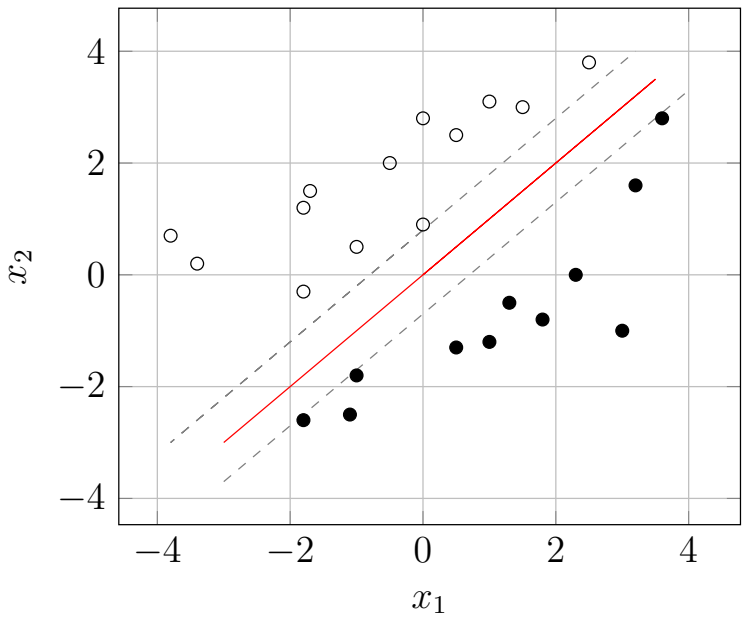


FIGURE 2.4. Margin of the classifier.

For every classifier that we have considered so far, we can do the same calculation or simulation to find the margin. Among all possible classifiers that separate these two classes, we can define the best classifier that has the maximum amount of margin. This maximum margin classifier is called the *linear support vector machine (SVM)*.

SVMs can be used to solve various real world problems such as the following.

(1) SVMs are helpful in text and hypertext categorization as their application can significantly reduce the need for labelled training

instances in both the standard inductive and transductive settings [**1**].

(2) Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true of image segmentation systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik [**6**].

(3) Hand-written characters can be recognized using SVM [**21**].

In Chapter 4 of the thesis, we will use the SVM algorithm to predict a best fitting mathematical model for the stock correlation networks with varied thresholds.

2.2.3. Semi-supervised learning. Semi-supervised learning (SSL) methods use small amounts of labelled data along with large amounts of unlabelled data to train the prediction system. Because unlabelled data is available everywhere and labelling the data is expensive and more rare, the semi-supervised learning method has gained widespread usage in recent years. We are given a set of labelled examples $x_1, \ldots, x_l \in X$ with corresponding labels $y_1, \ldots, y_l \in Y$. Additionally, we are given u unlabelled examples $x_{l+1}, \ldots, x_{l+u} \in X$, where usually l is taken as much smaller than u. Semi-supervised learning attempts to make use of

this combined information to predict labels for all data points. There are methods for SSL algorithm such as generative models, low-density separation and graph-based method.

Recently, graph-based semi-supervised learning methods have received a lot of attention [**4, 10, 28**]. These methods start with a graph where the nodes are the labelled and unlabelled data points, and weighted edges reflect the similarity of nodes. Graph-based methods based on label propagation work by using class labels associated with each labelled node, and each node propagates its label to its neighbours and the process is repeated until we have the desired coverage. The recognition of handwritten digits is one of the examples of this algorithm [**50**].

2.2.4. Unsupervised Learning. This learning system observes an unlabelled set of items, represented by their features $\{x_1, x_2, \ldots, x_n\}$. The goal is to organize the items. Typical unsupervised learning tasks include finding relationships within data. There are no training examples used in this process. Instead, the system is given data and tasked with finding patterns and correlations. Unsupervised learning is commonly used for transactional data. For an example, we may have a large dataset of customers and their purchases, but we can not determine what similar attributes can be drawn from customer profiles and their types of purchases. With this data, unsupervised learning algorithms may be used to determine that women of a certain age range who buy soaps

are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases [**41**].

2.3. Programming languages used in machine learning

There are various languages that can be used for machine learning processes, such as Python, Java, R and C++. After considering each of these and the libraries available for them, we choose Python for our thesis. Python's popularity may be due to the increased development of deep learning frameworks available for this language recently, including TensorFlow, PyTorch, and Keras. As a language that has readable syntax and the ability to be used as a scripting language, Python proves to be powerful and straightforward both for preprocessing data and working with data directly. The Scikit-learn machine learning library is built on top of several existing Python packages that Python developers may already be familiar with, namely NumPy, SciPy, and Matplotlib [**45**].

CHAPTER 3

Network models

3.1. Introduction

The study of networks has emerged in diverse disciplines as a means of analyzing relational data. Many of real-world networks are large, and some with upwards of trillions of nodes and edges [**44**]. To explore them, it is helpful to use mathematical models to identify the structural patterns and correlations within a network. Models for networks give insight into the underlying generative properties of networks and can serve as a predictive tool in their evolution. Network models help as a foundation for understanding interactions within networks. Various random graph generation models produce network structures that may be used in comparison to real-world networks.

In this chapter, we will explain two models of networks including the Erdős-Rényi model and the Preferential Attachment model, and explain the properties and characteristics of the networks. We will use Python 3 and the NetworkX library to simulate these models. Also, we will simulate the models with the Gephi application for visualizing networks.

3.2. Erdős-Rényi model

Network science aims to build models that reproduce the properties of real-world networks. Random network models incorporate randomness in their design to better simulate real-world networks. The Erdős-Rényi model is used for generating random graphs. We use $G(n, p)$ to denote the undirected random graph with n nodes, where each node pair is connected with probability p. A closely related model is $G(n, m)$. Of all possible graphs with n nodes and exactly m edges, one is uniformly randomly selected. The two models have very similar properties. We will be mostly focusing on the Erdős-Rényi model. To construct a random network we follow these steps:

(1) Start with n isolated nodes.

(2) Select a node pair and generate a random number between 0 and 1. If the number exceeds p, then connect the selected node pair with an edge; otherwise, leave them disconnected.

(3) Repeat step (2) for each of the $\binom{n}{2}$ node pairs.

The network obtained after this procedure is called *a binomial random graph*. Paul Erdős and Alfréd Rényi, have played an important role in understanding the properties of these networks. In their honour, a random network is called the *Erdős-Rényi network*.

3.2.1. Properties of Erdős-Rényi model. Let $e_{ij} \in \{0, 1\}$ be a Bernoulli random variable indicating the presence of the edge $\{i, j\}$. For

the Erdős-Rényi model, the random variables e_{ij} are independent and we have that:

$$e_{ij} = \begin{cases} 1 & \text{with probability } p, \\ 0 & \text{with probability } 1-p. \end{cases}$$

The expected number of edges in $G(n,p)$ is

$$\begin{aligned} m &= \mathbf{E}(\sum_{i \neq j} e_{ij}) = \binom{n}{2} p \\ &= \frac{n(n-1)}{2} p. \end{aligned} \tag{3.1}$$

In summary, because the expected value is determined by n and p, the network becomes denser by increasing p. It means that the average number of edges increases linearly with n.

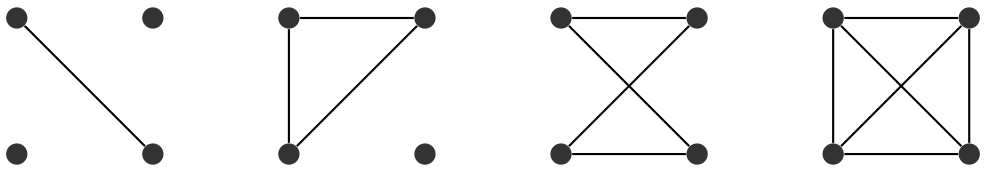


FIGURE 3.1. This Erdős-Rényi model is generated with $n = 4$ nodes with $p = 0.2$, $p = 0.5$, $p = 0.7$, $p = 1$ from left to right, respectively.

According to the definition of the degree of nodes, if the k_i denotes the degree of node i in a graph G, then we obtain the average degree of a random graph in the Erdős-Rényi model by using the Handshaking theorem:

$$
\begin{aligned}
AD(G) &= \frac{1}{n}\sum_i k_i = \frac{2m}{n} \\
&= \frac{2}{n}\frac{n(n-1)}{2}p \\
&\sim pn.
\end{aligned}
\tag{3.2}
$$

In the Erdős-Rényi random graph, the probability that node i has exactly k edges is the product of three terms [**12, 36**].

(1) The probability that k of its edges are present, or p^k.

(2) The probability that the remaining $(n-1-k)$ edges are missing, or $(1-p)^{n-1-k}$.

(3) The number of ways we can select k edges from $n-1$ potential node can have, or $\binom{n-1}{k}$.

Consequently, the degree distribution of a random network follows a binomial distribution:

$$
p_k = \binom{n-1}{k} p^k (1-p)^{n-1-k}. \tag{3.3}
$$

The shape of this distribution depends on the system size n and the probability p.

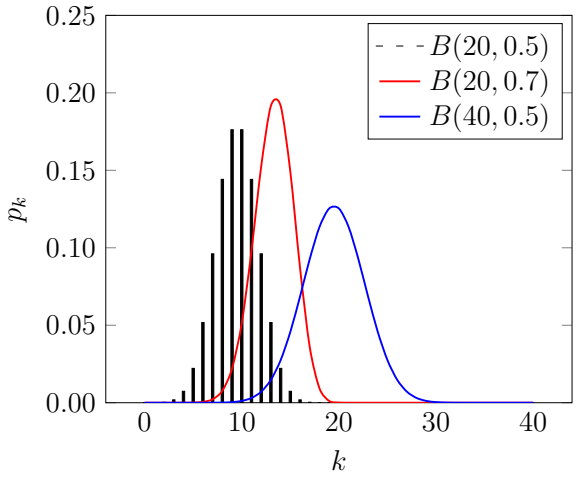


FIGURE 3.2. Simulating the binomial distribution $B(20, 0.5)$ with $n = 20$ and $p = 0.5$, $B(20, 0.7)$ with $n = 20$ and $p = 0.7$, and $B(40, 0.5)$ with $n = 40$ and $p = 0.5$.

3.3. Preferential Attachment model

In real networks, new nodes tend to link to the more connected nodes. For example, on Twitter, we would expect that popular users gather more new followers than less popular ones. At each time-step, we add a new node with m edges that connect to nodes already existing in the network. Preferential attachment is thought to be one mechanism behind the generation of complex networks. A model that incorporates preferential attachment is named after its inventors Albert-László Barabási and Réka Albert. The *Preferential Attachment model* or *Barabási-Albert model*, which can generate scale-free networks is also known as the *BA model* or the *scale-free model* [**5**]. A *scale-free network* is a network whose degree distribution of the network follows a power law. Thus, the proportion of nodes of degree k is:

$$p_k \sim k^{-b}. \tag{3.4}$$

If a graph satisfies (3.4), then we say that it has a *power law degree distribution* with *exponent* $b > 2$. If we take logarithm of equation (3.4), then we obtain

$$\log p_k \sim -b \log k.$$

The $\log p_k$ term is linearly dependent on $\log k$ and the slope of the line is the degree exponent $-b$.

3.3.1. Properties of Preferential Attachment model. Preferential attachment is a probabilistic mechanism. A new node is free to connect to any node in the network, whether it is high or low degree node. The network begins with m_0 nodes. New nodes are added to the network one at a time. Each new node is connected to $m \leq m_0$ existing nodes with a probability that is proportional to the number of edges that the existing nodes already have [**2**]. The probability p_i that a link of the new node connects to node i depends on the degree k_i is given by:

$$p_i = \frac{k_i}{\sum_j k_j}. \tag{3.5}$$

For example, the equation (3.5) implies that if a new node has a choice between a degree-two and a degree-four node, then it is twice as likely that it connects to the degree-four node.

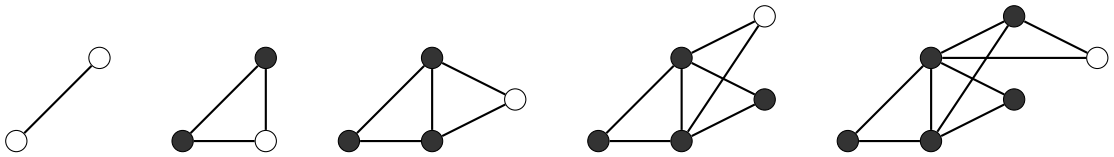


FIGURE 3.3. A sequence of four steps of the Preferential Attachment model from left to right. Empty circles are the newly added node to the network with $m = 2$.

After t time steps, the Barabási-Albert model generates a network with $n = t + m_0$ nodes and $e_0 + mt$ edges, where e_0 is the number of edges of the initial graph. As Figure 3 indicates, due to preferential attachment, new nodes are more likely to connect to the more connected nodes than to the low degree nodes. Hence, the high degree nodes have more edges. The networks contain few nodes with unusually high degree as compared to the other nodes, and these nodes are called *hubs* of the network.

According to the definition of the degree of nodes, if the k_i be a degree of node i, then we obtain the average degree of a Preferential Attachment model by using the Handshaking theorem:

$$
\begin{aligned}
AD(G) &= \frac{1}{n}\sum_{i=1}^{n} k_i \\
&= \frac{2|E|}{n} \\
&= \frac{2(e_0 + mt)}{t + m_0}.
\end{aligned} \tag{3.6}
$$

3.4. Simulations

3.4.1. Simulating the Erdős-Rényi model with Python. To generate graphs via the Erdős-Rényi model, we use the NetworkX library [**25**] for simulating graphs using Python. The $G(n, p)$ graph algorithm chooses each of the $\binom{n}{2}$ undirected possible edges with probability p. This algorithm runs in $O(n + m)$ time, where m is the expected number of edges, which equals $\binom{n}{2}p$ [**8**].

The graph below is a visualization of a simulation of the Erdős-Rényi model with $n = 100$ and $p = 0.08$. For the visualization, we use Gephi which is an open-source network analysis and visualization software package [**7**].

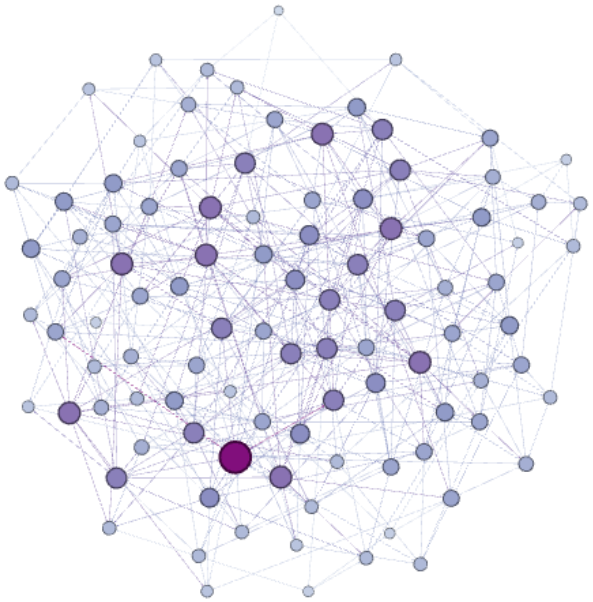


FIGURE 3.4. A simulation of the Erdős-Rényi model with $n = 100$ and $p = 0.08$. The darker nodes have higher degree.

We obtain the expected number of edges and average degree as follows:

$$m = \mathbf{E}(\sum_{ij} e_{ij}) = \binom{n}{2} p = \frac{n(n-1)}{2} p = \frac{100 \cdot 99}{2} 0.08 \sim 400. \tag{3.7}$$

$$AD(G) = \frac{1}{n} \sum k_i \sim pn \sim 0.08 \cdot 100 \sim 8. \tag{3.8}$$

The degree distribution for the graph simulated from $G(100, 0.08)$ looks like a binomial distribution according to Figure 3.5.

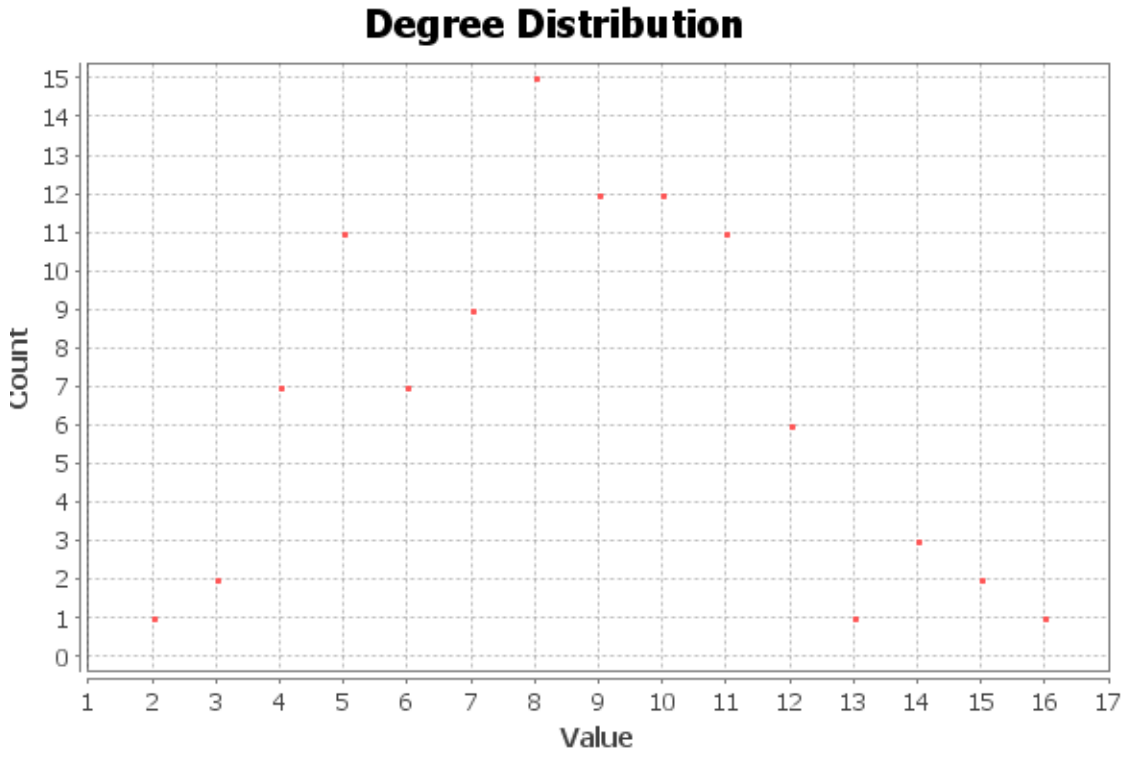


FIGURE 3.5. The degree distribution of the above simulation of $G(100, 0.08)$.

3.4.2. Simulating Preferential Attachment model with Python. We use the NetworkX package for simulating the Preferential Attachment model in Python. A graph of n nodes is grown by attaching new nodes each with m edges that are preferentially attached to existing nodes with high degree [**5**]. The initialization is a graph with m nodes and no edges. The $BA(n, m)$ graph is an algorithm for generating random scale-free networks using a preferential attachment mechanism.

The graph below is the visualization of the Preferential Attachment model with $n = 100$ and $m = 2$. We use Gephi for the visualization.

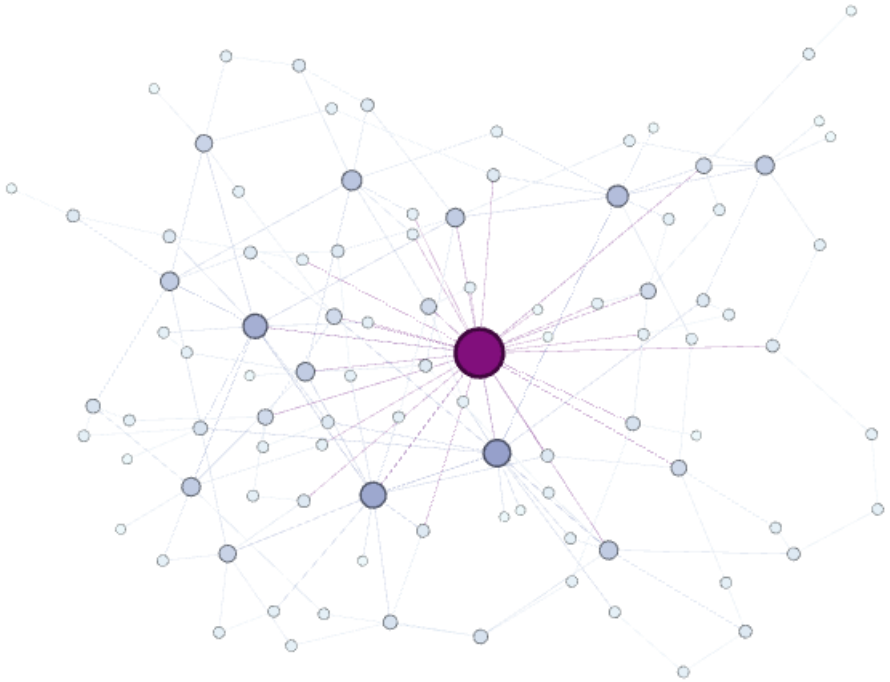


FIGURE 3.6. A simulation of the Preferential Attachment model with $n = 100$ and $m = 2$. The darker nodes have higher degree.

We obtain the average degree as follows:

$$\begin{aligned} AD(G) &= \frac{1}{n}\sum_i k_i = \frac{2|E|}{n} \\ &= \frac{2(e_0 + mt)}{t + m_0} \\ &= \frac{2(0 + 2 \cdot 98)}{100} \sim 3.9. \end{aligned} \tag{3.9}$$

The degree distribution for our simulation of $BA(100, 2)$ looks like a power law distribution according to Figure 3.7.

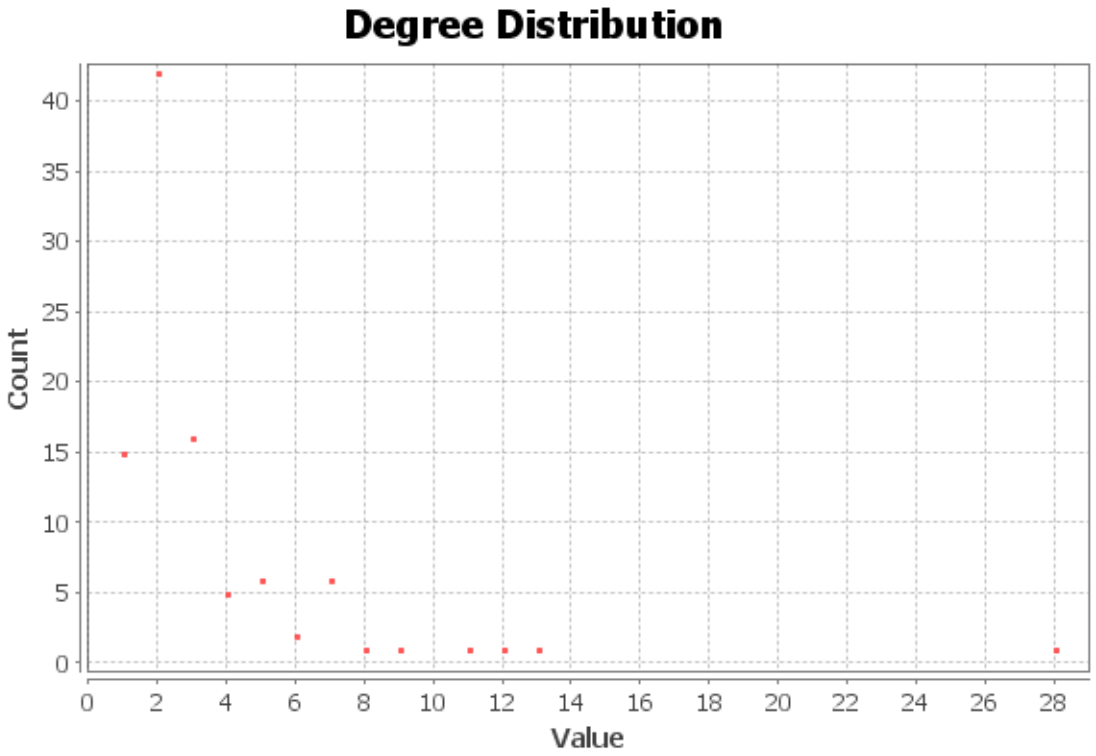


FIGURE 3.7. The degree distribution of the above simulation of $BA(100, 2)$.

CHAPTER 4

Model selection in stock correlation network

4.1. Introduction

In the last decade, financial networks have garnered a lot of interest from researchers as a way of visualizing the relationships between financial entities. This makes them useful in assessing current market dynamics and in predicting future market conditions. The stock market is a kind of multi-factor financial network. Fluctuations in stock prices are not independent but are highly inter-coupled with strong correlations with the business sectors and industries to which the stocks belong. Recently, analyses based on network models have been proposed for studying the correlations of stock prices [**15, 16, 17, 35, 37, 48**].

For constructing a stock correlation network, nodes represent the different stocks and edges are formed between the stocks based on the level of correlation between each pair of time series of stock prices. The resulting networks are usually large and their analysis is complex. A filtering process helps us to reduce the complexity and have a small size network for analysis.

In this chapter, we consider full networks of positive correlation of the return stock prices of all Dow Jones index, S&P 100 index and NASDAQ-100 index that were traded over a period of October 2016 to September 2017. We calculate the cross correlation values for each pair of stocks in each index, and we use a winner-take-all approach in establishing edges of the networks. After constructing the networks, we use machine learning to determine which models on mathematics are close to our networks. We use a feature known as graphlets (or sub-graph isomorphism types) for implementing a model selection method based on supervised learning. The SVM algorithm predicts which model fits best for our networks with varied thresholds.

4.2. Data

First of all, we consider a network of Dow Jones stock prices of 65 nodes. Then, we constructed a network for the S&P 100 index with 102 nodes and the NASDAQ-100 index with 107 nodes. Each node corresponds to one of the stocks traded between October 2016 to September 2017. We will evaluate the cross correlation of the time series of their monthly price returns.

Let $p_i(t)$ be the adjusted close price which is the closing price of the stock after accounting for any dividends and stock splits, and let i be the stock on month t. Then the price return of stock i on month t, denoted by $r_i(t)$, is defined as

$$r_i(t) = \frac{p_i(t)}{p_i(t-1)} - 1.$$

Suppose $x_i(t)$ and $x_j(t)$ are the monthly price returns of stock i and stock j, respectively, over the period $t = 0$ to $N-1$. We now compare the two time series with no relative time shift. The cross correlation between x_i and x_j with no time shift is given by [**20**]:

$$C_{ij} = \frac{\sum_t [(x_i(t) - \bar{x_i})((x_j(t) - \bar{x_j}))]}{\sqrt{\sum_t (x_i(t) - \bar{x_i})^2}\sqrt{\sum_t (x_j(t) - \bar{x_j})^2}},$$

where $\bar{x_i}$ and $\bar{x_j}$ are the means of the time series and the summations are taken over $t = 0$ to $N - 1$.

In the winner-take-all method, we need to apply a criterion to add edges between the stocks based on their correlation. In defining our criterion for connecting a pair of nodes, we need a threshold value for the cross correlation. Since cross correlation is a measure of similarity and its value is between 0 and 1, we choose a positive fractional value as the threshold. Suppose the threshold is ρ. Then the connection criterion for stock i and stock j is

$$C_{ij} > \rho.$$

For our networks, we consider two thresholds for finding the best fit model for networks. We suppose that $\rho = 0.7$ for a large value of

threshold and $\rho = 0.3$ for a small value of the threshold. We use Gephi for finding the average degree and to visualize the networks.

4.2.1. Dow Jones network. We begin with the Dow Jones Index. Dow Jones index keeps track of the performance of 30 large the United States Industrial companies, 15 prominent utility companies and 20 companies of the transportation sector. We examine values $\rho = 0.7$ and $\rho = 0.3$ to construct stock networks that reflect connections of correlated stock price time series. We calculate the number of edges and average degrees for both networks. Figures 1 and 2 visualize the networks with $\rho = 0.7$ and $\rho = 0.3$, respectively. Figure 4.3 shows various key parameters of our networks.

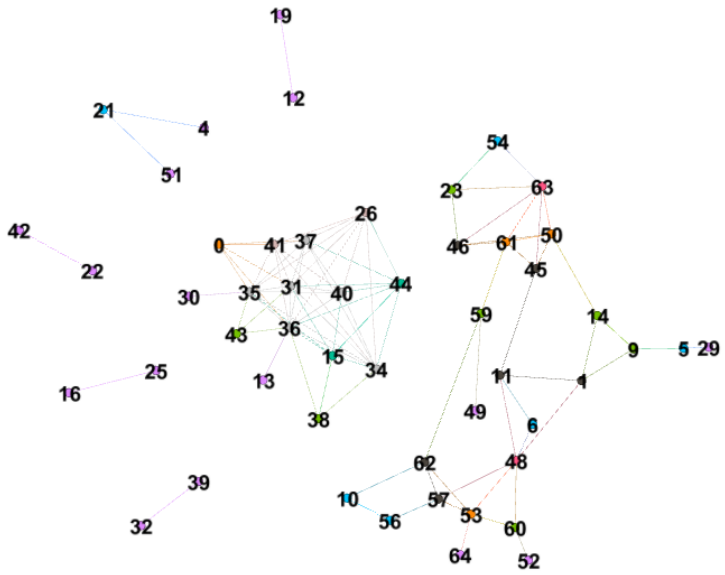


FIGURE 4.1. The Dow Jones stock network with $\rho = 0.7$.

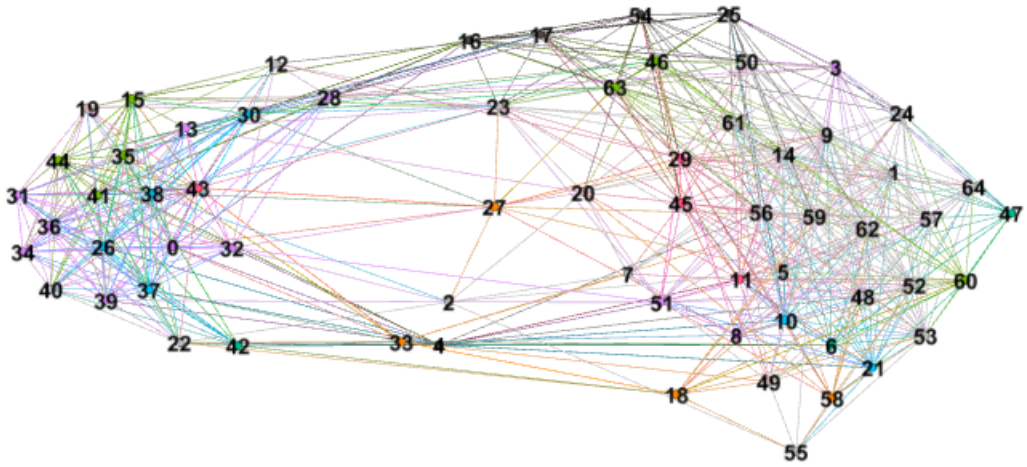


FIGURE 4.2. The Dow Jones stock network with $\rho = 0.3$.

Dow Jones networks		
Parameters	$\rho = 0.7$	$\rho = 0.3$
Number of nodes, N	65	65
Number of edges, E	99	626
Average degree, AD	3.05	19.26

FIGURE 4.3. Network parameters from Dow Jones stock networks.

We found that the degree distributions display scale-free characteristics when ρ is large. Figure 4 illustrates the power-law degree distribution for $\rho = 0.7$.

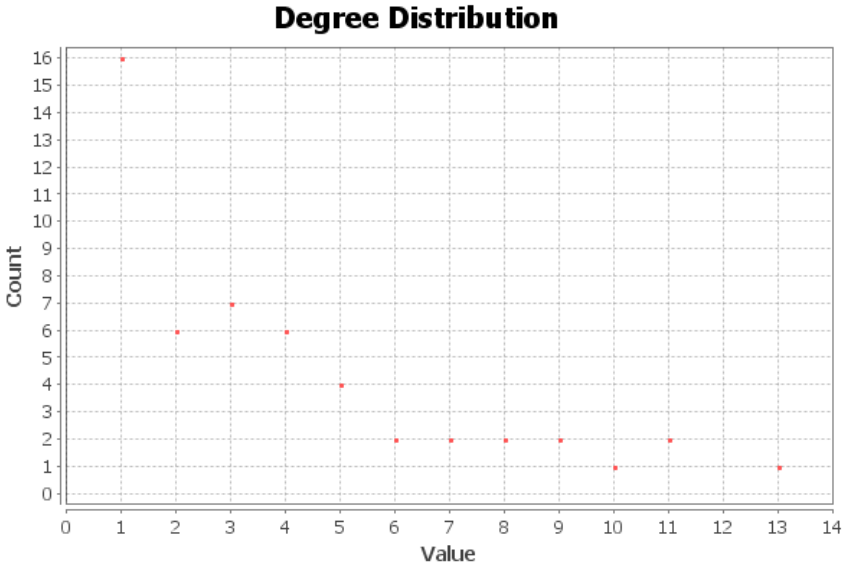


FIGURE 4.4. The degree distribution for a Dow Jones network with $\rho = 0.7$.

With the smaller value of ρ, the degree distribution does not show clear scale-free characteristics and the network tends to be randomly connected. Figure 5 shows the degree distribution for $\rho = 0.3$.

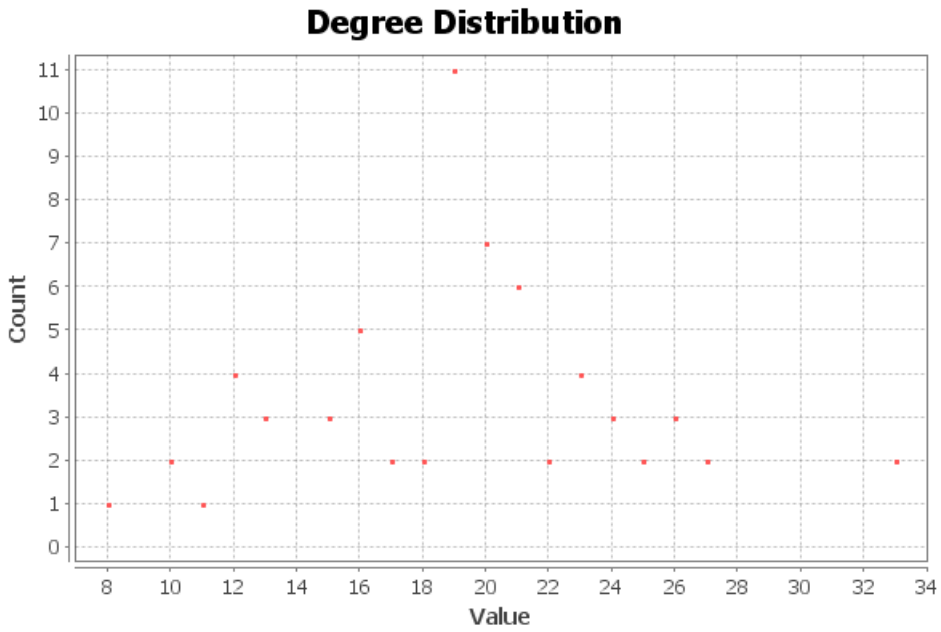


FIGURE 4.5. The degree distribution for a Dow Jones network with $\rho = 0.3$.

4.2.2. S&P 100 network. The S&P 100 Index is a stock market index of United States stocks maintained by Standard and Poor's. The S&P 100 includes 102 leading U.S. stocks. For constructing the stock networks, we examine values $\rho = 0.7$ and $\rho = 0.3$. We also calculate the number of edges and average degrees for both networks. Figures 6 and 7 visualize the networks with $\rho = 0.7$ and $\rho = 0.3$, respectively. Figure 8 shows parameters of our networks.

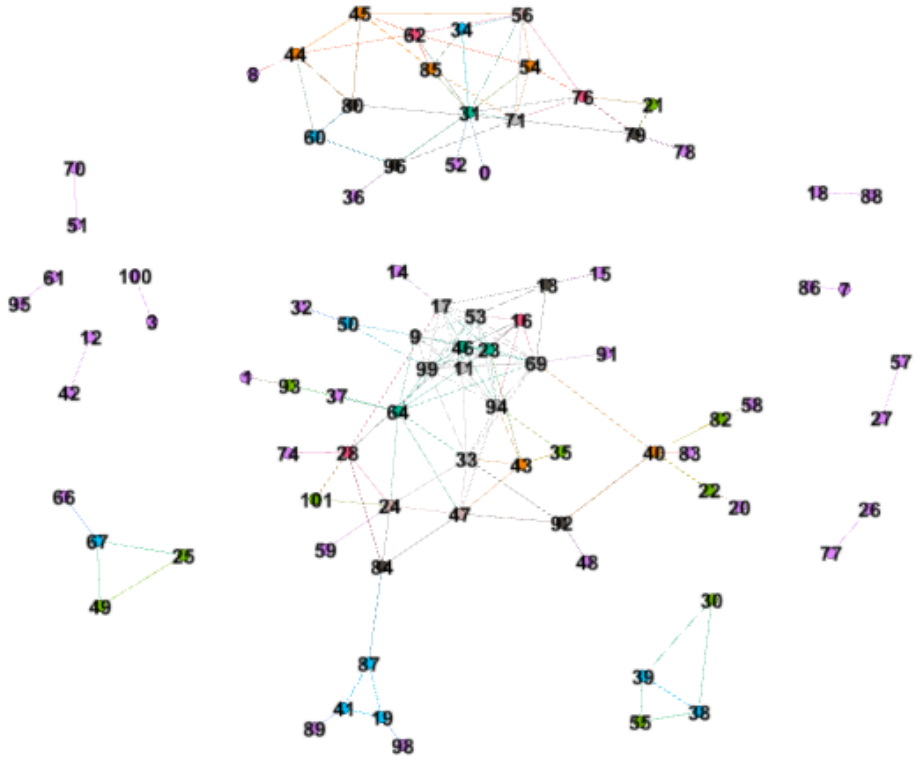


FIGURE 4.6. The S&P 100 stock network with $\rho = 0.7$.

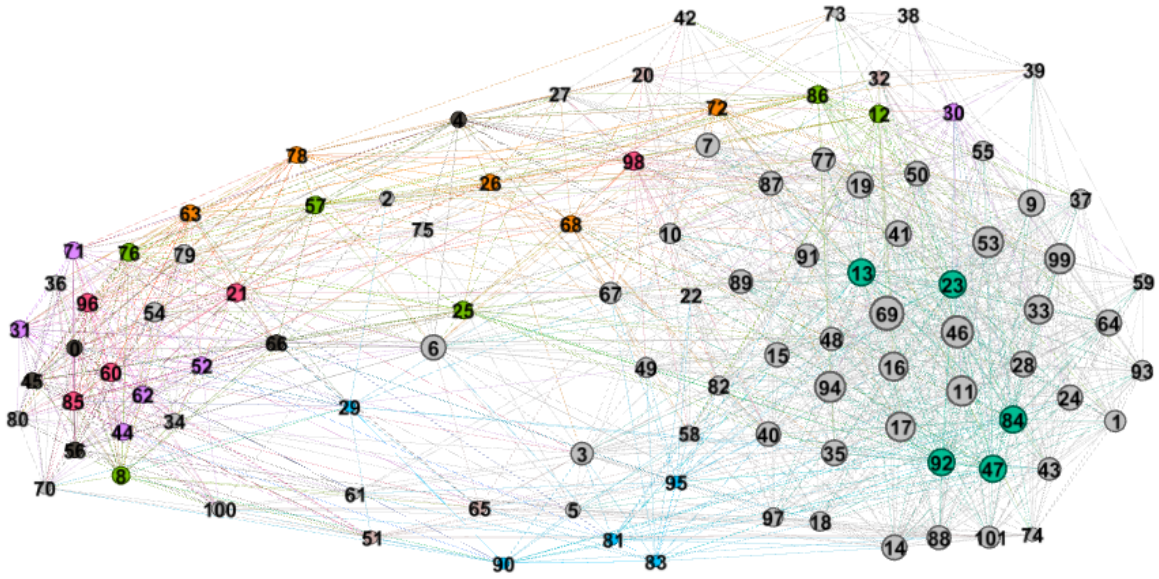


FIGURE 4.7. The S&P 100 stock network with $\rho = 0.3$.

S&P 100 networks		
Parameters	$\rho = 0.7$	$\rho = 0.3$
Number of nodes, N	102	102
Number of edges, E	158	1474
Average degree, AD	3.1	28.9

FIGURE 4.8. Network parameters from S&P 100 stock networks.

According to the Figures 9 and 10, the degree distribution for the networks with $\rho = 0.7$ exhibits that the network is scale-free and has the power law distribution. However, with a small value of $\rho = 0.3$, the degree distribution does not show clear scale-free characteristics and the network tends to be connected.

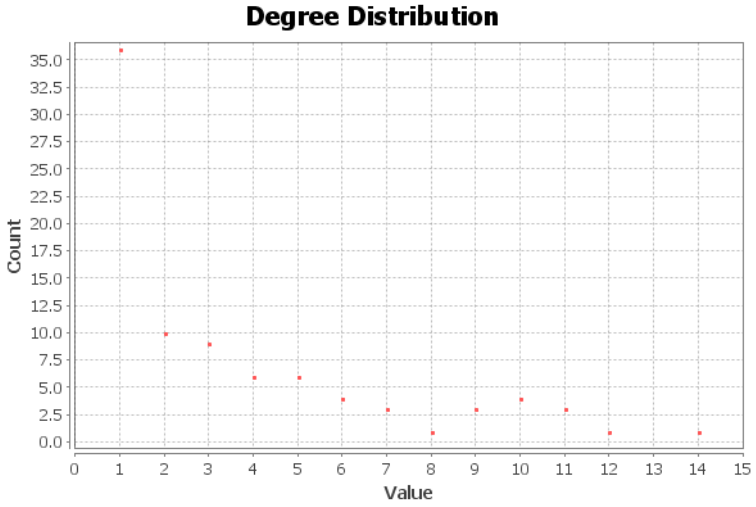


FIGURE 4.9. The degree distribution for a S&P 100 network with $\rho = 0.7$.

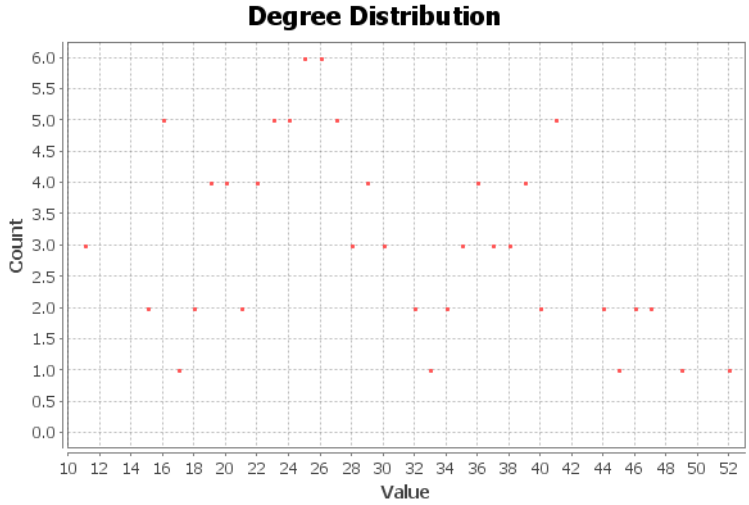


FIGURE 4.10. The degree distribution for a S&P 100 network with $\rho = 0.3$.

4.2.3. NASDAQ-100 index. The NASDAQ-100 is a stock market index made up of 107 equity securities issued by the largest non-financial companies listed on the NASDAQ. We calculate the number of connections and average degrees for both networks with $\rho = 0.7$ and $\rho = 0.3$ in Figure 13, and also we use Gephi to visualize the networks in Figures 11 and 12.

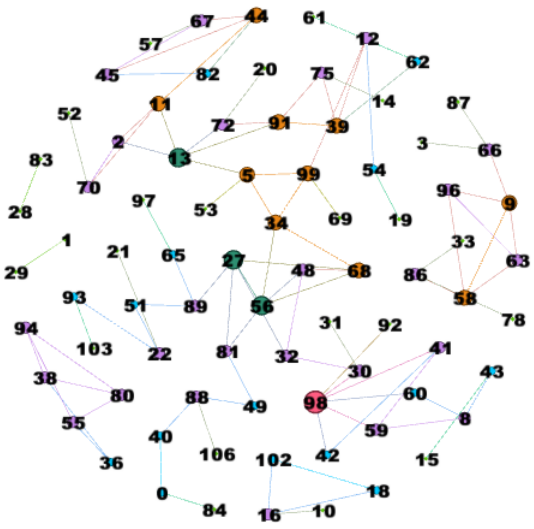


FIGURE 4.11. The NASDAQ-100 stock network with $\rho = 0.7$.

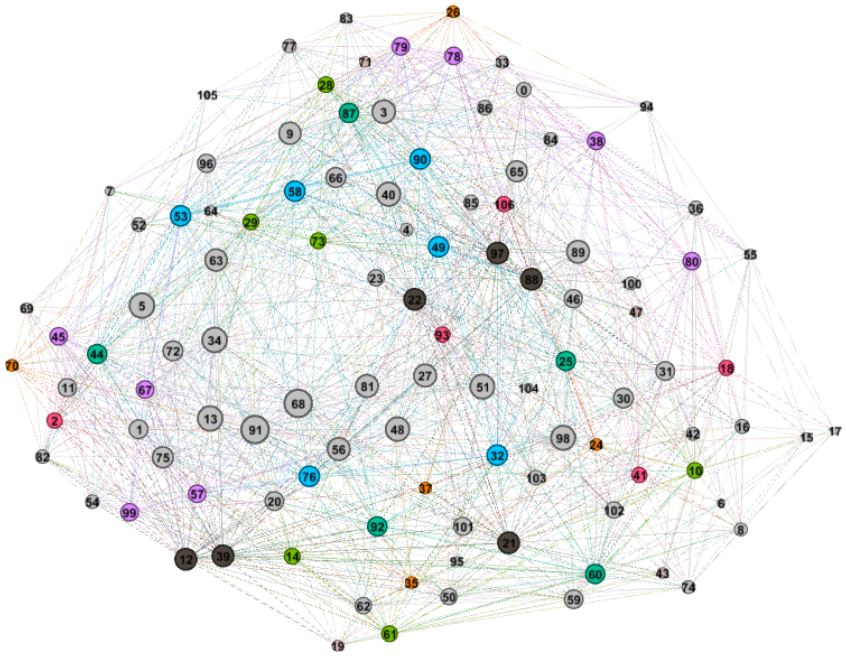


FIGURE 4.12. The NASDAQ-100 stock network with $\rho = 0.3$.

NASDAQ-100 networks		
Parameters	$\rho = 0.7$	$\rho = 0.3$
Number of nodes, N	107	107
Number of edges, E	98	1483
Average degree, AD	1.8	27.72

FIGURE 4.13. Network parameters From NASDAQ-100 stock networks.

The degree distribution for the NASDAQ-100 stock correlation networks with thresholds $\rho = 0.7$ and $\rho = 0.3$ are in Figures 14 and 15, respectively. With small threshold $\rho = 0.3$, the network distribution looks like the binomial distribution.

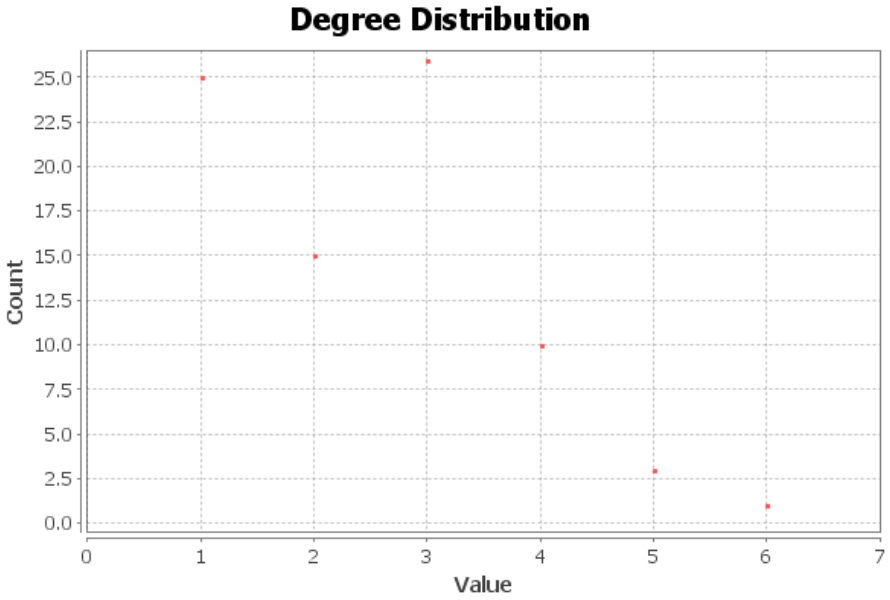


FIGURE 4.14. The degree distribution for a NASDAQ-100 network with $\rho = 0.7$.

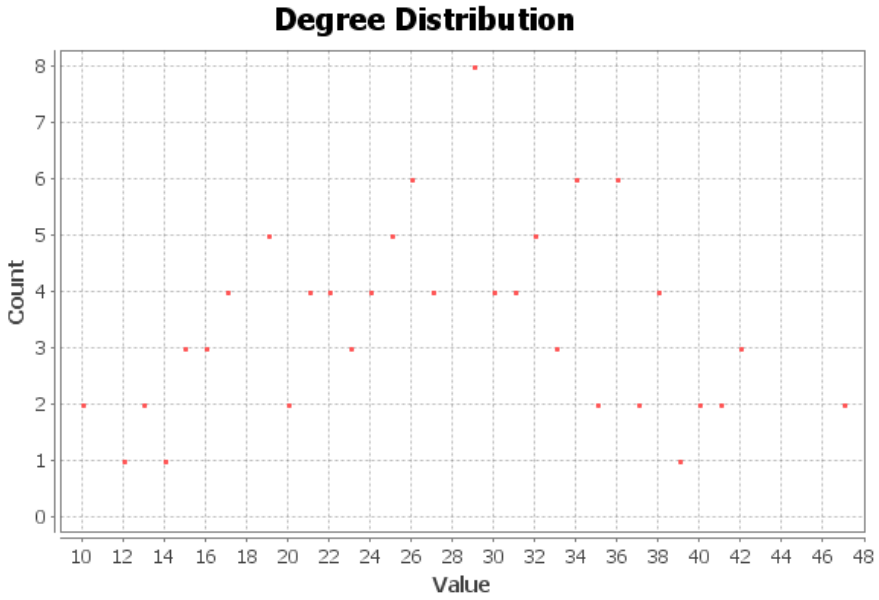


FIGURE 4.15. The degree distribution for a NASDAQ-100 network with $\rho = 0.3$.

To perform our experiments on networks of different orders, we built six stock correlation networks. Recall that $\mathrm{AD}(G)$ is the average degree of a network. For the Erdős-Rényi model, we have that $\mathrm{AD}(G) \sim np$. Also, according to the NetworkX package, because the initial graph has m nodes and no edges, the $\mathrm{AD}(G)$ a graph generated by the PA model is $\mathrm{AD}(G) = 2m(n-m)/n$. By finding $\mathrm{AD}(G)$, we can calculate the probability (p) of $G(n,p)$ model and also solve the quadratic equation for gaining the number of edges (m) in each step for Preferential Attachment model, $BA(n,m)$. For example, suppose that G is Dow Jones with $\rho = 0.7$, then we find that $\mathrm{AD}(G) = 3.05$:

$$p \sim \mathrm{AD}(G)/n = 3.05/65 \sim 0.05.$$

The probability of the Erdős-Rényi model is 0.05 and we have that $G(65, 0.05)$. To find the value of m for Preferential Attachment model, we need to solve the quadratic equation:

$$m(65 - m) = \mathrm{AD}(G) \cdot n/2 = \frac{3.05 \cdot 65}{2} \sim 99.$$

Hence, the number of edges (the parameter m) in each step for Preferential Attachment model is 1 and we have $BA(65, 1)$. We do the same process for all six networks and find the mathematical models parameters for each one. Figure 4.16 shows results of all networks.

Networks	Average degree	$G(n, p)$	$BA(n, m)$
Dow Jones with $\rho = 0.7$	3.05	$G(65, 0.05)$	$BA(65, 1)$
Dow Jones with $\rho = 0.3$	19.26	$G(65, 0.3)$	$BA(65, 12)$
S&P 100 with $\rho = 0.7$	3.1	$G(102, 0.03)$	$BA(102, 1)$
S&P 100 with $\rho = 0.3$	28.9	$G(102, 0.28)$	$BA(102, 17)$
NASDAQ-100 with $\rho = 0.7$	1.8	$G(107, 0.02)$	$BA(107, 1)$
NASDAQ-100 with $\rho = 0.3$	27.72	$G(107, 0.26)$	$BA(107, 15)$

FIGURE 4.16. Stock correlation networks with different AD(G).

4.3. Graphlet

Decomposition of networks is a widely used an approach in network analysis to factorize the complex structure of real-world networks into small sub-graph patterns of order k nodes. These patterns are called *graphlets* [**40**]. Graphlets are small connected non-isomorphic induced subgraphs of a large network [**39, 40**]. The number of appearances of graphlets in the network provides a description of the networks structural

properties. Such analysis is usually limited to the 30 graphlets between two and five nodes [**39**]. On a local level, counting how many graphlets in the network gives a significant information about the local network structure in a variety of domains [**24, 26**]. In this thesis, we work with undirected connected sub-graphs with 3 and 4 nodes as our graphlets. This is a set of eight graphs shown in Figure 17.

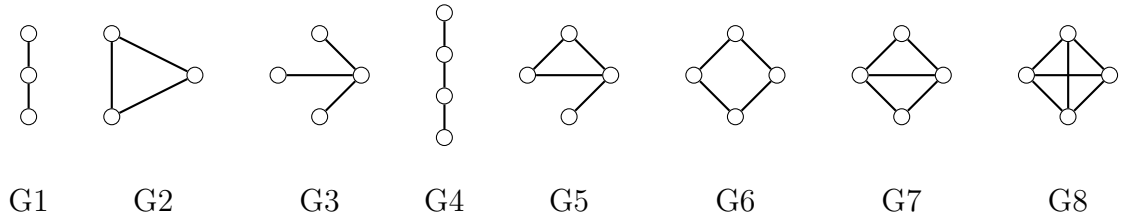


FIGURE 4.17. Connected sub-graphs of order 3 (two non-isomorphic graphs) and order 4 (six non-isomorphic graphs).

By counting that how many times every graph appears as an induced sub-graph in a network, we have a vector on 8-dimensional space as $(G1, G2, G3, G4, G5, G6, G7, G8)$. We use a fast and efficient algorithm for counting graphlets with 3 and 4 nodes [**31**]. The performance of the algorithm gives us the number of each graphlet in each network. The tables 18, 19, 20, 21, 22 and 23 illustrate the number of graphlets for graphs. Figure 24 shows us the number of graphlets for our constructed financial networks.

$G(65, 0.05)$								
Graphlets	2-star	Triangle	3-star	4-path	4-tailed triangle	4-cycle	4-chordal cycle	4-clique
Description	G1	G2	G3	G4	G5	G6	G7	G8
Data 1	231	3	153	554	17	7	0	0
Data 2	293	1	254	798	8	9	0	0
Data 3	344	9	277	967	74	12	2	0
Data 4	249	6	179	617	37	7	2	0
Data 5	464	11	590	1516	120	31	1	0
Data 6	208	5	133	436	65	5	1	0
$BA(65, 1)$								
Data7	229	0	713	416	0	0	0	0
Data8	268	0	1069	506	0	0	0	0
Data9	407	0	2691	679	0	0	0	0
Data10	323	0	1816	516	0	0	0	0
Data11	272	0	881	365	0	0	0	0
Data12	244	0	698	427	0	0	0	0

FIGURE 4.18. Number of graphlets for $G(65, 0.05)$ and $BA(65, 1)$ networks with implementing each network model six times.

$G(65, 0.3)$								
Graphlets	2-star	Triangle	3-star	4-path	4-tailed triangle	4-cycle	4-chordal cycle	4-clique
Description	G1	G2	G3	G4	G5	G6	G7	G8
Data 1	8459	1144	26269	78011	32129	8631	6596	403
Data 2	7402	1001	20935	67652	26508	6150	5176	316
Data 3	8194	1160	24568	75357	31503	7867	6553	470
Data 4	8126	1162	23908	74896	31638	7518	6524	464
Data 5	7882	1007	24726	71339	28022	7666	5562	318
Data 6	8039	1140	24084	73364	30589	7644	6572	480
$BA(65, 12)$								
Data7	8609	1854	35235	59260	47128	6287	15749	2466
Data8	8589	1671	30834	66036	42876	7668	13035	1803
Data9	8698	1804	34904	61594	46640	6925	15164	2106
Data10	8703	1703	32782	64787	44255	7723	13816	1924
Data11	8704	1826	35188	61479	46651	6683	15404	2357
Data12	8766	1887	37282	59273	48053	6606	16388	2526

FIGURE 4.19. Number of graphlets for $G(65, 0.3)$ and $BA(65, 12)$ networks with implementing each network model six times.

$G(102, 0.03)$								
Graphlets	2-star	Triangle	3-star	4-path	4-tailed triangle	4-cycle	4-chordal cycle	4-clique
Description	G1	G2	G3	G4	G5	G6	G7	G8
Data 1	493	2	454	1399	14	8	0	0
Data 2	470	7	442	1367	53	7	1	0
Data 3	504	1	501	1473	7	8	0	0
Data 4	394	8	458	1009	61	6	1	0
Data 5	531	6	484	1589	54	8	0	0
Data 6	341	2	275	840	12	4	0	0
$BA(102, 1)$								
Data7	573	1	3078	1028	0	0	0	0
Data8	353	0	791	633	0	0	0	0
Data9	572	0	3524	1243	0	0	0	0
Data10	367	0	1085	887	0	0	0	0
Data11	350	0	817	653	0	0	0	0
Data12	393	0	1313	797	0	0	0	0

FIGURE 4.20. Number of graphlets for $G(102, 0.03)$ and $BA(102, 1)$ networks with implementing each network model six times.

$G(102, 0.28)$								
Graphlets	2-star	Triangle	3-star	4-path	4-tailed triangle	4-cycle	4-chordal cycle	4-clique
Description	G1	G2	G3	G4	G5	G6	G7	G8
Data 1	30692	4244	147137	443014	183485	44969	37824	2667
Data 2	30379	4026	148616	437373	176509	44588	35302	2235
Data 3	29260	3763	141731	419983	163123	41149	31708	2081
Data 4	29764	3867	141284	433599	166990	42254	32227	2042
Data 5	28254	3548	135698	403637	152819	38493	29120	1753
Data 6	28329	3634	134549	405107	155490	38458	29813	1893
$BA(102, 17)$								
Data 7	29537	5493	169850	350376	220804	35734	65485	9171
Data 8	29647	5673	175337	344861	229352	34509	68340	9525
Data 9	30276	5963	198541	329994	242134	34255	76276	11459
Data 10	30097	5706	186242	342314	234033	35986	70474	9841
Data 11	30254	5636	191474	343405	231715	36697	70579	9549
Data 12	30192	5783	193241	337368	236092	35227	72457	10526

FIGURE 4.21. Number of graphlets for $G(102, 0.28)$ and $BA(102, 17)$ networks with implementing each network model six times.

$G(107, 0.02)$								
Graphlets	2-star	Triangle	3-star	4-path	4-tailed triangle	4-cycle	4-chordal cycle	4-clique
Description	G1	G2	G3	G4	G5	G6	G7	G8
Data 1	231	0	133	447	0	1	0	0
Data 2	214	2	172	437	7	5	0	0
Data 3	257	3	170	532	16	7	1	0
Data 4	184	0	96	306	0	0	0	0
Data 5	219	1	130	397	6	0	0	0
Data 6	146	0	70	214	0	2	0	0
$BA(107, 1)$								
Data 7	446	0	1727	1039	0	0	0	0
Data 8	465	0	2013	744	0	0	0	0
Data 9	453	0	1493	886	0	0	0	0
Data 10	428	0	1487	753	0	0	0	0
Data 11	444	0	2184	731	0	0	0	0
Data 12	432	0	1555	886	0	0	0	0

FIGURE 4.22. Number of graphlets for $G(107, 0.02)$ and $BA(107, 1)$ networks with implementing each network model six times.

$G(107, 0.0.26)$								
Graphlets	2-star	Triangle	3-star	4-path	4-tailed triangle	4-cycle	4-chordal cycle	4-clique
Description	G1	G2	G3	G4	G5	G6	G7	G8
Data 1	30495	3633	151125	453893	161781	40405	29008	1717
Data 2	30476	3587	152525	452347	161086	40692	28532	1701
Data 3	27872	3108	135646	407767	135409	33750	22715	1282
Data 4	29062	3299	144815	427537	146425	36845	25202	1427
Data 5	28200	3063	138203	414850	135647	35006	22060	1108
Data 6	29203	3469	142474	429951	152390	36974	26868	1666
$BA(107, 15)$								
Data 7	27752	4459	169964	331513	189512	28933	49126	6417
Data 8	28068	4509	174899	334345	193606	30020	50344	6166
Data 9	28210	4571	185548	327297	197510	28780	52556	6621
Data 10	27972	4467	173826	334500	191344	29300	49671	6318
Data 11	28219	4707	187988	322242	203423	27663	54778	6877
Data 12	28164	4474	183348	329688	195043	29316	50847	6147

FIGURE 4.23. Number of graphlets for $G(107, 0.26)$ and $BA(107, 15)$ networks with implementing each network model six times.

Graphlets	2-star	Triangle	3-star	4-path	4-tailed triangle	4-cycle	4-chordal cycle	4-clique
Description	G1	G2	G3	G4	G5	G6	G7	G8
Dow Jones $\rho = 0.7$	188	116	100	159	30	3	154	120
Dow Jones $\rho = 0.3$	4545	2575	5342	27216	25803	1220	11994	6560
S&P 100 $\rho = 0.7$	435	153	322	1012	783	21	248	136
S&P 100 $\rho = 0.3$	18667	9015	40856	190256	169258	8851	73830	35847
NASDAQ-100 $\rho = 0.7$	138	20	142	183	54	5	8	1
NASDAQ-100 $\rho = 0.3$	21996	7140	59033	268323	188857	11565	60806	17825

FIGURE 4.24. Number of graphlets for our stock correlation networks with varied threshold.

4.4. Model selection and discussion

We use the support vector machine algorithm (SVM) based classification tool from Scikit-learn [**38**] that models best fit out data. Our model selection method follows three steps. First, we generate the training data, consisting of the number of graphlets in Erdős-Rényi (ER) model and the Preferential Attachment model. Next, we use the training data to build a classifier with SVM algorithm. By labelling the -1 for Erdős-Rényi model and label 1 for the Preferential Attachment model, we build the classifier between the two classes. Finally, we predict the fit model for our stock correlation networks as testing data. For example, we consider the number of graphlets for the Erdős-Rényi model with average

degree 3.05 as an 8 dimension vector. Hence, we represent our graphs by eight features in a vector representation. We implement the algorithm six times, so we have six vectors for $G(65, 0.05)$ as testing data.

Another testing set comes from the the number of graphlets for the Preferential Attachment model. So, we have six vectors for $BA(65, 1)$ as well. We label -1 for the vectors for Erdős-Rényi model and label 1 for the Preferential Attachment vectors. We then predict the fit model for our Dow Jones networks with threshold $\rho = 0.7$.

The results of our experiments show that the networks with threshold $\rho = 0.7$ for Dow Jones, S&P 100 and NASDAQ-100 fit with the Erdős-Rényi (ER) model. Otherwise, the Dow Jones, S&P 100 and NASDAQ-100 networks with $\rho = 0.3$ are in the Preferential Attachment class. This result is not our expected result according to our average degree as showing in Figures 4, 5, 9, 10, 14 and 15. The hypothesis was that the result should, in fact, be exactly the opposite, based on the degree distributions. In particular, when $\rho = 0.3$, we would expect the Erdős-Rényi model and when $\rho = 0.7$, the network fit with the Preferential Attachment model.

We can conclude that considering graphlets may be not sufficient to separate the models for financial networks. Another feature that we can consider in future work to separate the models is degree distribution percentiles [**27**]. In this method, we measure the spread of the degree

Networks	Predicted label
Dow Jones with $\rho = 0.7$	-1
Dow Jones with $\rho = 0.3$	1
S&P 100 with $\rho = 0.7$	-1
S&P 100 with $\rho = 0.3$	1
NASDAQ-100 with $\rho = 0.7$	-1
NASDAQ-100 with $\rho = 0.3$	1

FIGURE 4.25. The SVM classifier results for stock correlation networks.

distribution. We consider the percentiles of the distribution formed by breaking it evenly into eight different pieces. This gives us seven features, called *deg1, deg2, deg3, deg4, deg5, deg6* and *deg7*.

Another observation is that the variation of stock prices is strongly influenced by a relatively small number of stocks. Because power law distributions have been found in the stock correlation networks with a large threshold ($\rho = 0.7$), this means that the stocks corresponding to nodes of high degrees are leaders of the entire market. Our analysis shows that in Dow Jones network with $\rho = 0.7$, the high degree is 13 and it belongs to a node number 36 and the node 36 corresponding to the Duke energy company. The Duke energy company stock is a leader stock for Dow Jones correlation network. For our S&P 100 correlation network with $\rho = 0.7$, the node number 69 has high degree and it is 14. The node number 69 represent the Morgan Stanley stock. And our last network, NASDAQ-100 correlation network with $\rho = 0.7$, suggest that the stock number 98 is a leader stock; note that is the media company Viacom Incorporation. As a result, by identifying leaders, investors can predict

the trend of stocks and market movements and also they can use the leader stocks to compose a new index that can naturally and adequately reflect the market variation.

CHAPTER 5

Conclusion and Open Problems

In our final chapter, we summarize our results and present some open problems collected from the entire thesis. The main goal of the thesis was to analyse stock correlation networks and determine the best fitting mathematical models for the networks.

5.1. Summary of results

We introduced the terminology of graph theory and networks in Chapter 1. In Chapter 2, we explained machine learning that is a subset of artificial intelligence. We described how machine learning explores the study and construction of algorithms that can learn from and make predictions based on real-world data. We also explained how the support vector machine algorithm worked in the supervised learning system. In Chapter 3, we recalled two mathematical models for networks, focusing on the Erdős-Rényi (ER) model and the Preferential Attachment (BA) model. Both models were implemented and simulated in Python.

In Chapter 4, we explained how to construct the stock correlation networks for the Dow Jones index, the S&P 100 index and the NASDAQ 100 index that were traded over a period of October 2016 to September 2017. We visualized the degree distribution of networks for small and

large thresholds. In addition, we represented our networks by graphlets counts for the connected sub-graphs of size 3 and size 4 in a vector representation. With the SVM algorithm, we predicted which model fits best for our networks. The result is shown in Figure 5.1.

Networks	Degree distribution	SVM Predicted label
Dow Jones with $\rho = 0.7$	BA	ER
Dow Jones with $\rho = 0.3$	ER	BA
S&P 100 with $\rho = 0.7$	BA	ER
S&P 100 with $\rho = 0.3$	ER	BA
NASDAQ-100 with $\rho = 0.7$	BA	ER
NASDAQ-100 with $\rho = 0.3$	ER	BA

FIGURE 5.1. Outcomes for stock correlation networks.

This result is not our expected result. We expected the SVM predictions would match the observed degree distributions. We can conclude that either graphlets do not separate the models for financial networks, or the small size of the networks involved influences the output. In future work, we will determine which feature is the most appropriate for model selection in financial networks.

5.2. Open problems

In what follows, we discuss problems left open by this thesis.

(1) Another feature that should be examined for financial networks is the degree distribution percentile. This is a logical feature to use that would measure the "shape" of the degree distribution. What is the result if we use these features?

(2) Our method compared two mathematical models. We will consider additional models for comparison in future work.

(3) In this thesis, we consider the positive correlation and thresholds for networks, and it would be interesting also to analyse them from the perspective of negative correlation.

(4) We used the threshold $\rho = 0.3$ as a small threshold and $\rho = 0.7$ for a large threshold. What about the other numbers between 0 and 1? Which model fits best for other thresholds?

(5) We can examine another period of time for our networks. Would the results differ if we take different intervals of time?

Appendices

APPENDIX A

List of Stocks

We present the list of stocks we used from the indices Dow Jones, S&P 100 and NASDAQ-100.

Dow Jones Stocks		
Id	Label	Industry
0	AAPL	Apple Inc.
1	AXP	American Express Company
2	BA	The Boeing Company
3	CAT	Caterpillar Inc.
4	CSCO	Cisco Systems, Inc.
5	CVX	Chevron Corporation
6	DWDP	DowDuPont Inc.
7	DIS	The Walt Disney Company
8	GE	General Electric Company
9	GS	The Goldman Sachs Group, Inc.
10	HD	The Home Depot, Inc.
11	IBM	International Business Machines
12	INTC	Intel Corporation
13	JNJ	Johnson & Johnson
14	JPM	JPMorgan Chase Co.
15	KO	The Coca-Cola Company
16	MCD	McDonald's Corporation
17	MMM	3M Company
18	MRK	Merck & Co., Inc.
19	MSFT	Microsoft Corporation
20	NKE	NIKE, Inc.
21	PFE	Pfizer Inc.
22	PG	The Procter & Gamble Co.
23	TRV	The Travelers Companies, Inc.
24	UNH	UnitedHealth Group
25	UTX	United Technologies Corp.

Dow Jones Stocks		
Id	Label	Industry
26	V	Visa Inc.
27	VZ	Verizon Communications Inc.
28	WMT	Wal-Mart Stores, Inc.
29	XOM	Exxon Mobil Corporation
30	AES	The AES Corporation
31	AEP	American Electric Power
32	AWK	American Water Works Co.
33	CNP	CenterPoint Energy, Inc.
34	ED	Consolidated Edison Inc.
35	D	Dominion Energy Inc.
36	DUC	Duke Energy
37	EIX	Edison International
38	EXC	Exelon
39	FE	FirstEnergy Corp
40	NEE	NextEra Energy
41	NI	NiSource Inc.
42	PCG	PG&E
43	PEG	Public Service Enterprise Group
44	SO	Southern Co
45	ALK	Alaska Air Group, Inc.
46	AAL	American Airlines Group Inc.
47	CAR	Avis Budget Group, Inc.
48	CHRW	C.H. Robinson Worldwide, Inc.
49	CSX	CSX Corp.
50	DAL	Delta Air Lines
51	EXPD	Expeditors International
52	FDX	FedEx Corporation
53	JBHT	JB Hunt Inc.
54	JBLU	JetBlue Airways Corp.
55	KSU	Kansas City Southern
56	KEX	Kirby Corp.
57	LSTR	Landstar System, Inc.
58	MATX	Matson, Inc.
59	NSC	Norfolk Southern Corp.
60	R	Ryder System, Inc.
61	LUV	Southwest Airlines, Inc.
62	UNP	Union Pacific Corp.
63	UAL	United Continental Holdings
64	UPS	United Parcel Service, Inc.

S&P 100 Stocks		
Id	Label	Industry
0	AAPL	Apple Inc.
1	ABBV	AbbVie Inc.
2	ABT	Abbott Laboratories
3	ACN	Accenture plc
4	AGN	Allergan plc
5	AIG	American International Group
6	ALL	Allstate Corp.
7	AMGN	Amgen Inc.
8	AMZN	Amazon.com
9	AXP	American Express Inc.
10	BA	Boeing Co.
11	BAC	Bank of America Corp.
12	BIIB	Biogen Idec
13	BK	The Bank of New York Mellon
14	BLK	BlackRock Inc.
15	BMY	Bristol-Myers Squibb
16	BRK-B	Berkshire Hathaway
17	C	Citigroup Inc
18	CAT	Caterpillar Inc.
19	CELG	Celgene Corp.
20	CHTR	Charter Communications
21	CL	Colgate-Palmolive Co.
22	CMCSA	Comcast Corporation
23	COF	Capital One Financial Corp.
24	COP	ConocoPhillips
25	COST	Costco
26	CSCO	Cisco Systems
27	CVS	CVS Health
28	CVX	Chevron
29	DHR	Danaher
30	DIS	The Walt Disney Company
31	DUK	Duke Energy
32	DWDP	DowDuPont
33	EMR	Emerson Electric Co.
34	EXC	Exelon
35	F	Ford Motor
36	FB	Facebook
37	FDX	FedEx
38	FOX	21st Century Fox
39	FOXA	21st Century Fox
40	GD	General Dynamics
41	GE	General Electric Co.
42	GILD	Gilead Sciences

S&P 100 Stocks		
Id	Label	Industry
43	GM	General Motors
44	GOOG	Alphabet Inc.
45	GOOGL	Alphabet Inc.
46	GS	Goldman Sachs
47	HAL	Halliburton
48	HD	Home Depot
49	HON	Honeywell
50	IBM	International Business Machines
51	INTC	Intel Corporation
52	JNJ	Johnson & Johnson Inc.
53	JPM	JP Morgan Chase & Co
54	KHC	Kraft Heinz
55	KMI	Kinder Morgan Inc/DE
56	KO	The Coca-Cola Company
57	LLY	Eli Lilly and Company
58	LMT	Lockheed-Martin
59	LOW	Lowe's
60	RMA	MasterCard Inc.
61	MCD	McDonald's Corp.
62	MDLZ	Mondelz International
63	MDT	Medtronic Inc.
64	MET	Metlife Inc.
65	MMM	3M Company
66	MO	Altria Group
67	MON	Monsanto
68	MRK	Merck & Co.
69	MS	Morgan Stanley
70	MSFT	Microsoft
71	NEE	NextEra Energy
72	NKE	Nike
73	ORCL	Oracle Corporation
74	OXY	Occidental Petroleum Corp.
75	PCLN	Priceline Group Inc.
76	PEP	Pepsico Inc.
77	PFE	Pfizer Inc.
78	PG	Procter & Gamble Co
79	PM	Phillip Morris International
80	PYPL	PayPal Holdings
81	QCOM	Qualcomm Inc.
82	RTN	Raytheon Company
83	SBUX	Starbucks Corporation
84	SLB	Schlumberger
85	SO	Southern Company

S&P 100 Stocks		
Id	Label	Industry
86	SPG	Simon Property Group, Inc.
87	T	AT& T Inc.
88	TGT	Target Corp.
89	TWX	Time Warner Inc.
90	TXN	Texas Instruments
91	UNH	UnitedHealth Group Inc.
92	UNP	Union Pacific Corp.
93	UPS	United Parcel Service Inc.
94	USB	US Bancorp
95	UTX	United Technologies Corp.
96	V	Visa Inc.
97	VZ	Verizon Communications Inc.
98	WBA	Walgreens Boots Alliance
99	WFC	Wells Fargo
100	WMT	Wal-Mart
101	XOM	Exxon Mobil Corp.

NASDAQ-100 Stocks		
Id	Label	Industry
0	AAL	American Airlines Group Inc.
1	AAPL	Apple Inc.
2	ADBE	Adobe Systems Inc.
3	ADI	Analog Devices Inc.
4	ADP	Automatic Data Processing Inc.
5	ADSK	Autodesk Inc.
6	AKAM	Akamai Technologies Inc.
7	ALGN	Align Technology Inc.
8	ALXN	Alexion Pharmaceuticals Inc.
9	AMAT	Applied Materials Inc.
10	AMGN	Amgen Inc.
11	AMZN	Amazon.com Inc.
12	ATVI	Activision Blizzard Inc.
13	AVGO	Broadcom Inc.
14	BIDU	Baidu Inc.
15	BIIB	Biogen Inc.
16	BMRN	Biomarin Pharmaceutical Inc.
17	CA	CA Inc.
18	CELG	Celgene Corp.
19	CERN	Cerner Corp.
20	CHKP	Check Point Technologies

NASDAQ-100 Stocks		
Id	Label	Industry
21	CHTR	Charter Communications Inc.
22	CMCSA	Comcast Corp.
23	COST	Costco Wholesale Corp.
24	CSCO	Cisco Systems Inc.
25	CSX	CSX Corp.
26	CTAS	Cintas Corp.
27	CTRP	Ctrip.Com International Ltd
28	CTSH	Cognizant Technology Solutions
29	CTXS	Ctrip.Com International Ltd
30	DISCA	Discovery Inc.
31	DISCK	Discovery Inc.
32	DISH	DISH Network Corp.
33	DLTR	Dollar Tree Inc.
34	EA	Electronic Arts
35	EBAY	eBay Inc.
36	ESRX	Express Scripts Holding Co.
37	EXPE	Expedia Group Inc.
38	FAST	Fastenal Co
39	FB	Facebook
40	FISV	Fiserv Inc.
41	FOX	Twenty-First Century Fox Inc.
42	FOXA	Twenty-First Century Fox Inc.
43	GILD	Gilead Sciences Inc.
44	GOOG	Alphabet Class C
45	GOOGL	Alphabet Class A
46	HAS	Hasbro Inc.
47	HOLX	Hologic Inc.
48	HSIC	Henry Schein Inc.
49	IDXX	IDEXX Laboratories Inc.
50	ILMN	Illumina Inc.
51	INCY	Incyte Corp
52	INTC	Intel Corp
53	INTU	Intuit Inc.
54	ISRG	Intuitive Surgical Inc.
55	JBHT	J.B. Hunt Inc.
56	JD	JD.com Inc.
57	KHC	Kraft Heinz Co.
58	KLAC	KLA-Tencor Corp.
59	LBTYA	Liberty Global PLC
60	LBTYK	Liberty Global PLC
61	LILA	Liberty Latin America Ltd.
62	LILAK	Liberty Latin America Ltd.
63	LRCX	Lam Research Corp.

NASDAQ-100 Stocks		
Id	Label	Industry
64	MAR	Marriott International Inc.
65	MAT	Mattel Inc.
66	MCHP	Microchip Technology Inc.
67	MDLZ	Mondelez International Inc.
68	MELI	MercadoLibre Inc.
69	MNST	Monster Beverage Corp.
70	MSFT	Microsoft Corp.
71	MU	Micron Technology Inc.
72	MXIM	Maxim Integrated Products Inc.
73	MYL	Mylan NV
74	NCLH	Norwegian Cruise Line Holdings
75	NFLX	Netflix Inc.
76	NTES	NetEase Inc.
77	NVDA	NVIDIA Corp.
78	ORLY	O'Reilly Automotive Inc.
79	PAYX	Paychex Inc.
80	PCAR	PACCAR Inc.
81	PCLNX	PIMCO Commodities
82	PYPL	PayPal Holdings Inc.
83	QCOM	Qualcomm Inc.
84	QVCA	Liberty Interactive Corp.
85	REGN	Regeneron Pharmaceuticals Inc.
86	ROST	Ross Stores Inc.
87	SBUX	Starbucks Corp.
88	SHPG	Shire PLC
89	SIRI	Sirius XM Holdings Inc.
90	STX	Seagate Technology PLC
91	SWKS	Skyworks Solutions Inc.
92	SYMC	Symantec Corp.
93	TMUS	T-Mobile US Inc.
94	TSCO	Tractor Supply Co.
95	TSLA	Tesla Inc.
96	TXN	Texas Instruments Inc.
97	ULTA	Ulta Beauty Inc.
98	VIAB	Viacom Inc.
99	VOD	Vodafone Group PLC
100	VRSK	Verisk Analytics Inc.
101	VRTX	Vertex Pharmaceuticals Inc.
102	WBA	Walgreens Boots Alliance Inc.
103	WDC	Western Digital Corp.
104	WYNN	Wynn Resorts Ltd
105	XLNX	Xilinx Inc.
106	XRAY	Dentsply Sirona Inc.

APPENDIX B

Codes

B.0.1. Graphlet count code. We customized the graphlet count code that we mentioned in Chapter 4. The original source code did not count the number of four vertices graphlets, so we add the code below for counting.

```
print("number of total four path: %d" % len(self.total_four_path))
print("number of totla three star: %d" % len(self.total_three_star))
print("number of totla four cycle: %d" % len(self.total_four_cycle))
print("number of totla four tailed_triangle: %d" % len(self.total_four_tailed_triangle))
print("number of totla four chordal_cycle: %d" % len(self.total_four_chordal_cycle))
print("number of totla four clique: %d" % len(self.total_four_clique))
```

Also, we generate another source code for the Random Graph model and the Preferential Attachment model for getting different results in each implementation.

$G = nx.barabasi\_albert\_graph(n, m, seed = None)$

$G = nx.fast\_gnp\_random\_graph(n, p, seed = None)$

B.0.2. SVM code. We present the code discussed in Chapter 4. The SVM algorithm predicted the best fit model based on the graphlet count for our networks.

```
In [1]: from sklearn.svm import SVC
        X = [[231,3,153,554,17,7,0,0],
             [293,1,254,798,8,9,0,0],
             [344,9,277,967,74,12,2,0],
             [249,6,179,617,37,7,2,0],
             [464,11,590,1516,120,31,1,0],
             [208,5,133,436,65,5,1,0],
             [229,0,713,416,0,0,0,0],
             [268,0,1069,506,0,0,0,0],
             [407,0,2691,679,0,0,0,0],
             [323,0,1816,516,0,0,0,0],
             [272,0,881,365,0,0,0,0],
             [244,0,698,427,0,0,0,0]]
        y = [-1,-1,-1,-1,-1,-1,1,1,1,1,1,1]
        svclassifier = SVC(kernel='linear')
        svclassifier.fit(X, y)
        svclassifier.predict
        [[188,116,100,159,30,3,154,120]])
```

```
Out[1]: array([-1])

In [2]: from sklearn.svm import SVC
        X=[
        [8459,1144,26269,78011,32129,8631,6596,403],
        [7402,1001,20935,67652,26508,6150,5176,316],
        [8194,1160,24568,75357,31503,7867,6553,470],
        [8126,1162,23908,74896,31638,7518,6524,464],
        [7882,1007,24726,71339,28022,7666,5562,318],
        [8039,1140,24084,73364,30589,7644,6572,480],
        [8609,1854,35235,59260,47128,6287,15749,2466],
        [8589,1671,30834,66036,42876,7668,13035,1803],
        [8698,1804,34904,61594,46640,6925,15164,2106],
        [8703,1703,32782,64787,44255,7723,13816,1924],
        [8704,1826,35188,61479,46651,6683,15404,2357],
        [8766,1887,37282,59273,48053,6606,16388,2526]]
        y = [-1,-1,-1,-1,-1,-1,1,1,1,1,1,1]
        svclassifier = SVC(kernel='linear')
        svclassifier.fit(X, y)
        svclassifier.predict ([
        [4545,2575,5342,27216,25803,1220,11994,6560]])

Out[2]: array([1])
```

```
In [3]: from sklearn.svm import SVC
        X = [[493,2,454,1399,14,8,0,0],
             [470,7,442,1367,53,7,1,0],
             [504,1,501,1473,7,8,0,0],
             [394,8,458,1009,61,6,1,0],
             [531,6,484,1589,54,8,0,0],
             [341,2,275,840,12,4,0,0],
             [573,1,3078,1028,0,0,0,0],
             [353,0,791,633,0,0,0,0],
             [572,0,3524,1243,0,0,0,0],
             [367,0,1085,887,0,0,0,0],
             [350,0,817,653,0,0,0,0],
             [393,0,1313,797,0,0,0,0]]
        y = [-1,-1,-1,-1,-1,-1,1,1,1,1,1,1]
        svclassifier = SVC(kernel='linear')
        svclassifier.fit(X, y)
        svclassifier.predict
        ([[435,153,322,1012,783,21,248,136]])

Out[3]: array([-1])

In [4]: from sklearn.svm import SVC
        X = [
        [30692,4244,147137,443014,183485,44969,37824,2667],
```

```
        [30379,4026,148616,437373,176509,44588,35302,2235],
        [29260,3763,141731,419983,163123,41149,31708,2081],
        [29764,3867,141284,433599,166990,42254,32227,2042],
        [28254,3548,135698,403637,152819,38493,29120,1753],
        [28329,3634,134549,405107,155490,38458,29813,1893],
        [29537,5493,169850,350376,220804,35734,65485,9171],
        [29647,5673,175337,344861,229352,34509,68340,9525],
        [30276,5963,198541,329994,242134,34255,76276,11459],
        [30097,5706,186242,342314,234033,35986,70474,9841],
        [30254,5636,191474,343405,231715,36697,70579,9549],
        [30192,5783,193241,337368,236092,35227,72457,10526]]
        y = [-1,-1,-1,-1,-1,-1,1,1,1,1,1,1]
        svclassifier = SVC(kernel='linear')
        svclassifier.fit(X, y)
        svclassifier.predict ([
        [18667,9015,40856,190256,169258,8851,73830,35847]])
```

Out[4]: array([1])

```
In [5]: from sklearn.svm import SVC
        X = [[231,0,133,447,0,1,0,0],
            [214,2,172,437,7,5,0,0],
            [257,3,170,532,16,7,1,0],
            [184,0,96,306,0,0,0,0],
```

```
        [219,1,130,397,6,0,0,0],
        [146,0,70,214,0,2,0,0],
        [446,0,1727,1039,0,0,0,0],
        [465,0,2013,744,0,0,0,0],
        [453,0,1493,886,0,0,0,0],
        [428,0,1487,753,0,0,0,0],
        [444,0,2184,731,0,0,0,0],
        [432,0,1555,886,0,0,0,0]]
    y = [-1,-1,-1,-1,-1,-1,1,1,1,1,1,1]
    svclassifier = SVC(kernel='linear')
    svclassifier.fit(X, y)
    svclassifier.predict
    ([[138,20,142,183,54,5,8,1]])
```

Out[5]: array([-1])

```
In [6]: from sklearn.svm import SVC
        X =[
        [30495,3633,151125,453893,161781,40405,29008,1717],
        [30476,3587,152525,452347,161086,40692,28532,1701],
        [27872,3108,135646,407767,135409,33750,22715,1282],
        [29062,3299,144815,427537,146425,36845,25202,1427],
        [28200,3063,138203,414850,135647,35006,22060,1108],
        [29203,3469,142474,429951,152390,36974,26868,1666],
```

```
        [27752,4459,169964,331513,189512,28933,49126,6417],
        [28068,4509,174899,334345,193606,30020,50344,6166],
        [28210,4571,185548,327297,197510,28780,52556,6621],
        [27972,4467,173826,334500,191344,29300,49671,6318],
        [28219,4707,187988,322242,203423,27663,54778,6877],
        [28164,4474,183348,329688,195043,29316,50847,6147]]
        y = [-1,-1,-1,-1,-1,-1,1,1,1,1,1,1]
        svclassifier = SVC(kernel='linear')
        svclassifier.fit(X, y)
        svclassifier.predict ([
        [21996,7140,59033,268323,188857,11565,60806,17825]])
```

Out[6]: array([1])

APPENDIX C

Correlation Networks

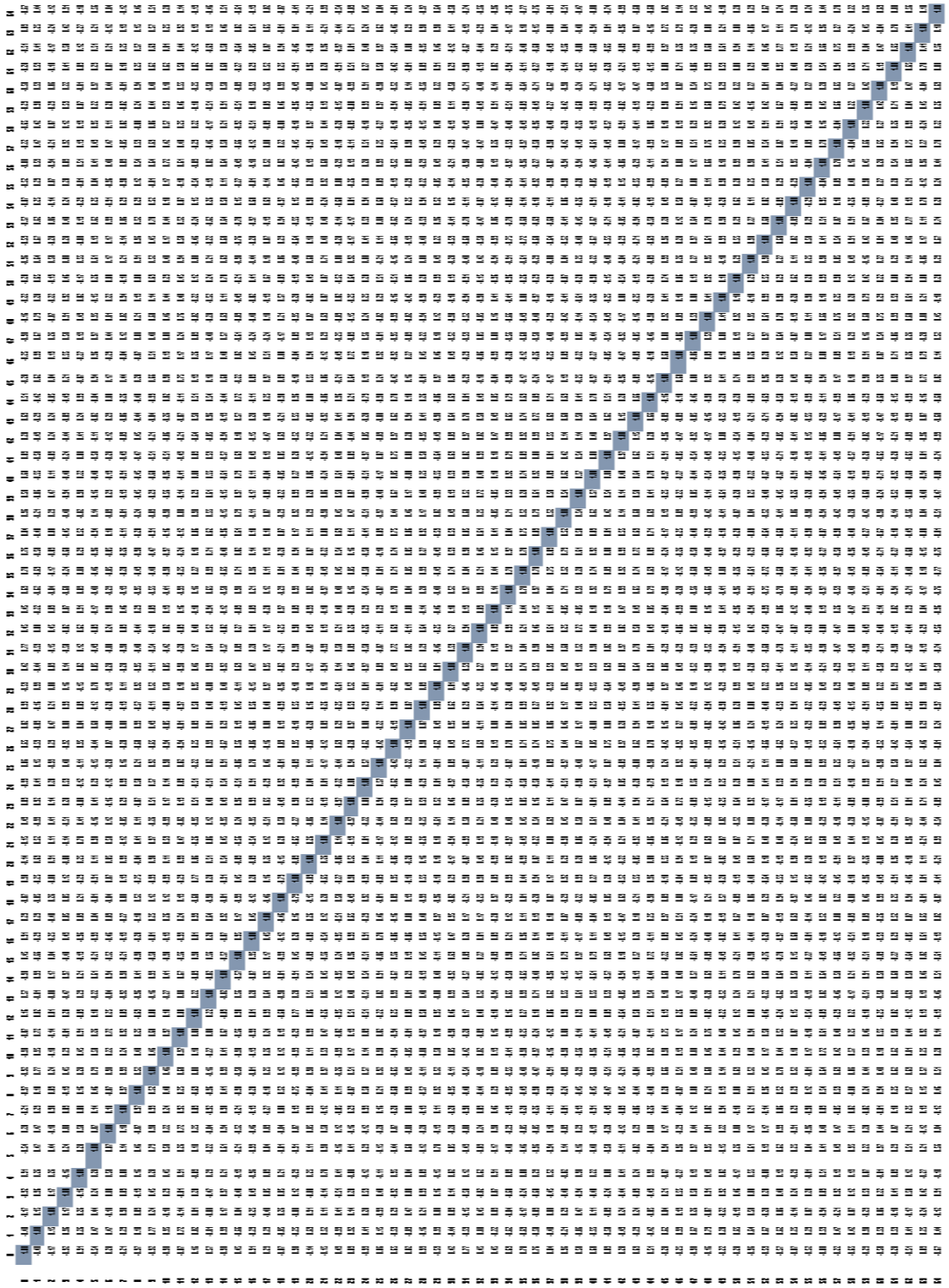
In Chapter 4, we present the networks with $\rho = 0.3$ and $\rho = 0.7$ for our stocks. To create the networks, first, we compute the cross correlation for all the stocks and create a cross correlation matrix. We built cross correlation matrices for three indices.

(1) Dow Jones stock correlation matrix.

(2) S&P 100 stock correlation matrix.

(3) NASDAQ-100 stock correlation matrix.

Dow Jones stock correlation matrix.



S&P 100 stock correlation matrix part one.

S&P 100 stock correlation matrix part two.

NASDAQ-100 stock correlation matrix part one.

NASDAQ-100 stock correlation matrix part two.

Bibliography

[1] J. Aitchison, The Bibliographic Classification of H. E. Bliss as a source of thesaurus terms and structure, *Journal of Documentation* **42** (1986) 160–181.

[2] R. Albert, A. L. Barabási, Statistical mechanics of complex networks, *Reviews of Modern Physics* **74** (2002) 47–97.

[3] E. Anderson, The species problem in Iris, *Annals of the Missouri Botanical Garden* **23** (1936) 45–509.

[4] X. Argyriou, M. Herbster, M. Pontil, Combining Graph Laplacians for Semi-Supervised Learning, *Neural Information Processing Systems*, 2005.

[5] A. L. Barabási, R. Albert, Emergence of scaling in random networks, *Science* **286** (1999) 509–512.

[6] L. Barghout, Spatial-taxon information granules as used in iterative fuzzy-decision-making for image segmentation, *Granular Computing and Decision-Making*, Springer 2015 pp. 285–318.

[7] M. Bastian, S. Heymann, M. Jacomy, Gephi : An Open Source Software for Exploring and Manipulating Networks, AAAI Publications, 2009.

[8] V. Batagelj, U. Brandes, Efficient generation of large random networks, *Phys. Rev. E.* **71** (2005) 036–113.

[9] S. Battiston, M. Catanzaro, *Statistical properties of corporate board and director networks*, Springer 2004 pp. 345–352.

[10] A. Blum, T. Mitchell, Combining Labelled and Unlabelled Data with Co-Training, *Computational Learning Theory* 1998 pp. 92–100.

[11] B. Bollobás, *Probabilistic Combinatorics and Its Applications*, American Mathematical Society, 1991.

[12] B. Bollobás, *Random Graphs*, Cambridge University Press, 2001.

[13] A. Bonato, *A Course on the Web Graph*, American Mathematical Society Graduate Studies Series in Mathematics, Providence, Rhode Island, 2008.

[14] A. Bonato, A. Tian, Complex Networks and Social Networks, *Social Networks*, Springer 2011 pp. 280–291.

[15] G. Bonnanno, G. Caldarelli, F. Lillo, R. N. Mantegna, Topology of correlation-based minimal spanning trees in real and model markets, *Phys. Rev. E.* **68** (2003) 046–103.

[16] G. Bonanno, G. Caldarelli, F. Lillo, S. Micciche, N. Vandewalle, R. N. Mantegna, Networks of equities in financial markets, *Euro. Phys.J. B.* **38** (2004) 363–371.

[17] G. Bonanno, F. Lillo, R. N. Mantegna, High-frequency cross correlation in a set of stocks, *Quantit. Finance* **1** (2001) 96–104.

[18] A. Chapelle, Separation of ownership and control where do we stand, *Corp. Ownersh. Control* **15** (2005) 91–101.

[19] K. Tse. Chi, Liu. Jing, C.M. Francis, A. Lau, Network perspective of the stock market, *Journal of Empirical Finance* **17** (2004) 659–667.

[20] J. Cohen, P. Cohen, S. G. West, L. S. Aiken, *Applied Multiple Regression/Correlation Analysis for the Behavioural Sciences, Third Edition*, 2003.

[21] D. DeCoste, Training Invariant Support Vector Machines, *Machine Learning* **46** (2002) 161–191.

[22] P. Erdős, A. Rényi, On Random Graph, *Publications Mathematica* **6** (1959) 290–297.

[23] L. Euler, Solutio problematis ad geometriam situs pertinentis, *Comment. Acad. Sci. U. Petrop* **8**, 1736.

[24] O. Frank, Triad count statistics, *Annals of Disc. Math.* (1988) 141–149.

[25] A. Hagberg, A. Schult, J. Swart, Exploring network structure, dynamics, and function using NetworkX, In: *Proceedings of the 7th Python in Science Conference*, Editors: G. Varoquaux, T. Vaught, J. Millman, 2008.

[26] P. W. Holland, S. Leinhardt, Local structure in social networks, *Sociological Methodology* **7** (1976) 1–45.

[27] J. Janssen, M. Hurshman, N. Kalyaniwalla, Model selection for social networks using graphlets, *Internet Math* **8** (2012) 338–363.

[28] T. Joachims, Transductive Learning via Spectral Graph Partitioning, *Machine Learning* 2003 pp. 290–297.

[29] P. Kaelbling, L. Littman, W. Moore, Reinforcement Learning: A Survey, *Journal of Artificial Intelligence Research* **4** (1996) 237–285.

[30] C. Koza, J. Bennett, F. Andre, D. Keane, A. Martin, Automated design of both the topology and sizing of analogue electrical circuits using genetic programming, *Artificial Intelligence in Design*, Springer 1996 pp. 151–170.

[31] X. Liu, Graphlet counting, GitHub repository, https://github.com/liuxt/graphlet counting, 2017.

[32] E. Mandere, Financial Networks and their applications to the stock Market, 2009.

[33] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[34] V. Mnih, Human-level control through deep reinforcement learning, *Nature* **518** (2015) 529–533.

[35] R. N. Mantegna, Hierarchical structure in financial markets, *Euro. Phys. J. B.* **11** (1999) 193–197.

[36] M. Newman, S. H. Strogatz, D. J. Watts, (2001), Random graphs with arbitrary degree distributions and their applications, *Physical Review E.* **64** (2009) 026–118.

[37] J. P. Onnela, A. Chakraborti, K. Kaski, Dynamics of market correlations: taxonomy and portfolio analysis, *Phys. Rev. E.* **68** (2003) 056–110.

[38] F. Pedregosa, Scikit-learn: Machine Learning in Python,*Journal of Machine Learning Research* **12** (2011) 2825–2830.

[39] N. Prźulj, Biological Network Comparison Using Graphlet Degree Distribution, *Bioinformatics* 2007.

[40] N. Prźulj, DG. Corneil, I. Jurisica, Modeling Interactome, Scale-Free or Geometric, *Bioinformatics* 2004.

[41] S. Rajagopal, Customer data clustering using data mining technique, *International Journal of Database Management Systems* **3**, 2011.

[42] A. Samuel ,*Computer Games I*, Springer 1998 pp. 335–365.

[43] A. Samuel, Some studies in machine learning using the game of checkers, *IBM Journal of Research and Development* **3**, 1959.

[44] B. Schwartz, Google search knows about over 130 trillion pages, https://searchengineland.com/googles-search-indexes-hits-130-trillion-pages-documents, 2016.

[45] Scikit-learn, Machine Learning in Python, Pedregosa et al., *Journal of Machine Learning Research* **12** (2011) 2825–2830.

[46] S. H. Strogatz, D. J. Watts, Collective dynamics of small-world networks, *Nature* **393** (1998) 440–442.

[47] S. Sutton, G. Barto, *Reinforcement Learning*, MIT Press, Cambridge 1998 Chapter 11.

[48] N. Vandewalle, F. Brisbois, X. Tordoir, Self-organized critical topology of stock markets, *Quantit. Finan.* **1** (2001) 372–375.

[49] D. B. West, *Introduction to Graph Theory, Second Edition*, Prentice Hall, 2001.

[50] X. Zhu, J. Kandola, Z. Ghahramani, J. Lafferty, Non-parametric Transforms of Graph Kernels for Semi-Supervised Learning, *Neural Information Processing Systems*, 2005.