Forecasting Stock Markets Index by Using ARIMA, Artificial Neural Networks and Support Vector Machine

A dissertation submitted to The University of Manchester for
the degree of Master of Science in the Faculty of Humanities

2018

10212201

ALLIANCE MANCHESTER BUSSINESS SCHOOL

Table of Contents

Word Count: 15,567

List of Figures

List of Tables

Abstract

Forecasting future prices of financial markets has always been, and still is, one of the biggest challenges for academics and the private sector. The trend that stock markets follow is a result of factors ranging from political to economical, thus identification of a straightforward approach is extremely hard. Regardless, the most popular methods include statistical and machine learning techniques due to their ability to recognise patterns among vast amount of data which capture underlying stock price dynamics.

This study attempts to predict future movements of the Greek stock market (ASE) by utilizing the properties of autoregressive integrated moving average (ARIMA), artificial neural network (ANN) and support vector machine (SVM). Traditionally ARIMA has been in the top preferences of market researchers as one of the best for short-term predictions that largely outperformed other more complex models, however its application is limited since it cannot capture non-linear trends adequately. On the other hand, artificial neural networks which are known for their dynamic and effectiveness when dealing with large datasets perform well when facing non-linearities. Last comes the support vector machine which, similar to an ANN, can provide very accurate results capturing a big proportion of the relationships and patterns in the data. The comparison of the models applied was made in one view point: Mean Absolute Error (MAE), using historical data acquired from Financial Times database ranging from 06/14/2017 to 06/14/2018.

The ARIMA and SVM models were implemented in Python, one of the most powerful and flexible, data handling and analysis tools, while on the other hand IBM SPSS Modeler 14.2 was used for the ANN due to flexible deployment and ability to make use of all the data for maximum insight. After the deployment of the models took place, the accuracy metrics acquired for each model place ANN as the best performer with a MAE of 6.98052 while ARIMA and SVM achieved 7.567558 and 8.536055 respectively.

Declaration

No portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or another institute of learning.

Intellectual Property Statement

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Ludmil Mikhailov, for giving me the opportunity to work with him during this study and his continuous guidance, understanding and patience. The academic world will miss you after your retirement. May you have a relaxing new chapter ahead.

In addition, I would like to thank my parents, sister and friends for their sacrifices, constant support in this journey and for never letting me lose my spirit. I feel lucky that I have you in my life and with you by my side there is nothing I cannot achieve. I love you more than you know.

Chapter 1. Introduction

1.1 Background

In this era of continuous change, where market preferences shift faster than ever, and after many countries experienced (some still experiencing) the results of the financial crisis that shake the foundations of the global economy, people are more cautious than ever regarding the financial endeavours they attempt. From farmers that want to know the amount of rainfall in the next days in order to maximise their production to investors seeking to determine stock movements and gain profit out of it the art of forecasting has become more pivotal than ever. When it comes to prediction of stock markets there are many factors that influence the index that can be political, economic and social. It has to be clarified that forecasting comes with a margin error thus making the process of perfect prediction impossible. This error gets even steeper when the forecast attempted goes deep into the future since many things can change or even more variables can surface along the way. In that context, both firms and academics have increased their efforts of deploying models with adequate predictive power.

Daily data were collected for the purpose of the study and acquired from the Financial Times database ranging in a time frame between 06/14/2017 to 06/14/2018. A stock market index is usually characterized by Open Price, Close Price, Minimum Price, Maximum Price and Volume traded. Open Price represents the value of the index at the opening of the market and Volume traded the number of stocks traded. In addition, Maximum and Minimum prices refer to the maximum and minimum value of the index respectively during the day. Finally, Close Price, is the closing value of the market index at the end of the trading day and is the one used in this study for forecasting purposes. The reason behind it is that Close Price accounts for all the activities performed during the day and has an effect on the index. The model used were either statistical (ARIMA) or artificial intelligent (ANN, SVM) and the final ranking is based on the Mean Absolute Error. When it comes to the data analysis and ARIMA implementation it was developed in Python while IBM SPSS Modeller 14.2 was used for the ANN and SVM.

1.2 Overview of Past Studies

Prediction models, as proposed by literature, fall under two categories, statistical and artificial intelligence (AI). Statistical methods include, among others, exponential smoothing, autoregressive integrated moving average and generalized autoregressive conditional heteroskedasticity (GARCH) while AI techniques include artificial neural networks, support vector machines and many more. The search for an efficient predictive model is profound in the literature due to its complexity, dynamic behaviour and the need for better results. Especially when it comes to time series for financial forecasting the application of ARIMA and neural networks has become very popular the last years (Zhang, 2004; Windrow, 1994; Refenes, 1995; Kate, 2000; J. T. Yao, 1999; Abu- Mostafa, 2001).

Comparison of linear optimisation and ANNs models for time series analysis was made by Tansel et al. (1999) based on accuracy, convenience and computational time. Similarly, Lee, Sehwan, and J. Jongdae et al. (2007) performed a forecasting for the Korean stock market index using ARIMA and a Back Propagation Neural Network (BPNN). In both cases the study resulted in superiority of the ARIMA model compared to the neural network.

Adebiyi et al. (2014) studies the application of both an ARIMA model and an multi - layer perceptron neural network to forecast the movement of the New York stock market index. The result of the research suggests that in both cases the prediction rate is adequately good and can be applied in many real - world problems.

In line with the previous study, Sterba and Hilovska et al. (2010) argues that both models behave accurately enough and are highly suggested for future applications. On the other, hand Yao et. Al (1996) showed in his study that neural networks largely outperform the classical Box – Jenkins method and this is something that Hansen et al. (1999) also agrees arguing that the neural network is more capable in identifying hidden patterns underlying in the historical data. Wijaya et al. (2010) attempted a forecasting on the Indonesian exchange rate and identified that the ARIMA model cannot perform equally or better than the neural network used. Tang, Zaiyong, Chrys De Almeida, and Paul A. Fishwick et al. (1991) experiments demonstrate that for time series with long memory, both methods produced comparable results. However, for series with short memory, neural networks outperformed the Box-Jenkins model.

Several more studies contradict each other arguing about the superiority or not of a model when it comes to time series predictions. Zhibin et al. (2014) attempted to forecast river flow in semiarid mountain region by comparing different models such as artificial neural networks and support vector machines and concluded that all of them can provide satisfactory results but the latter manages to outperform the ANN.

Kyoung-Jae et al. (2003) studied the prediction of financial data by applying back propagation neural network along with SVM before deciding about the superiority of Support Vector Machines over neural networks. In line with the previous researchers, Y Ding, X Song, Y Zen et al. (2008) used various modelling techniques in an attempt to forecast the stock return of several Chinese companies. The result of their study ranked SVM as the most accurate method compared to neural networks and other statistical procedures. Chen, Sheng-Hsun Hsu, and Hwang-Pin Shen et al. (2005) in their attempt to develop an application for intrusion detection compared the relative performance of traditional neural networks with support vector machines and suggested that the second performs better as well.

Kara, Yakup, Melek Acar Boyacioglu, and Ömer Kaan Baykan et al. (2011) evaluated the prediction accuracy of neural networks and support vector machines in the Turkish stock market. Experimental results showed that average performance of ANN model (75.74%) was found significantly better than that of SVM model (71.52%). This research seeks to bridge the gap between different views and help establish a more coherent view of the best methodology to address the problem of stock market forecasting.

1.3 Objective of the Study

The objective of this thesis is to explore further the issue of stock market forecasting and supplement the existing literature by using various techniques in order to provide empirical evidence of their utility and predictive performance. It focuses on the technical analysis and tries to determine, by comparison, a model that captures relationships in historical data of stock markets and by doing so forecast future movements. An application of ARIMA, ANN and SVM in time series will be attempted on the Greek market (Athens Stock Index – ASE), one that has suffered the consequences of the global financial crisis in the maximum degree and perhaps the most volatile market in the globe.

1.4 Thesis Structure

The thesis has been organized into four chapters as follows:

Chapter 1 gives a background and introduces the problem of forecasting from different perspectives. The context of this study is established such as the area of application and it describes the methodology followed in order to compare different models and acquire an adequate predictive technique.

Chapter 2 provides a literature survey and analysis for the components and architecture of ARIMA, ANNs and SVMs and explains their application on forecasting problems.

Chapter 3 is about the implementation of all three models in Python and SPSS Modeller respectively and includes a step by step process, problems faced and how they were addressed. The results of the modelling process are also presented.

Chapter 4 includes a critical comparison and ranking of the models developed with limitations and suggestions for future improvements.

Chapter 2. Literature Review

2.1 Stock Market Index Forecasting

Stock market index is a measurement that refers to a section of a market. Usually computed as a weighted average of the prices of selected stocks, it is used to represent the market state and track its movement over time. Observation of indices trend has always been essential for investors and firms in an effort to maximise their profit and reduce risk that comes from variance and uncertainty. As it was mentioned though the index is a representation of movement ***over time,*** so it is safe to assume that time affects the index significantly, thus investors soon turned in searching an efficient way to predict future prices of the market based on data rather than estimations and many times gut. Due to many factors that affect either positively or negatively the market such as political, economic or social the process of predicting the movement of the index is extremely challenging and a field of continuous study. Various approaches have been examined and proposed, each addressing a different problem of forecasting, in an effort to identify the optimal performing method.

Analysts argue that financial markets are divided in to two broad categories of thought, technical and fundamental analysis.

2.2 Fundamental Analysis

Fundamental analysis, or intrinsic, tries to determine and utilize underlying factors that influence movement of index and result in abnormal returns (Greig, 1992). In general terms, supporters of fundamental analysis suggest that the movement of a market index can be explained with the help of the economic factors that influence the specific market studied. Dunis and Feeny et al. (1989) while studying the factors that affect exchange rates discovered that fundamental analysis is a subject falling into limitations since it is based on long term indications and trends thus a short - term prediction will not be accurate.

2.3 Technical Analysis

Technical analysis, or chartist, seeks to predict the movement of stock market by looking at historical movements and identifying patterns. In other words, technical analysis supports the theory that history repeats itself and its purpose is to establish familiarity with past patterns of price behaviour in order to recognize situations of likely recurrence (Fama, 1986).

Technical analysis, the one that this thesis adopts, has been widely used to predict the tendency of markets since the early 1980’s. Economists have been arguing for many years whether the technical methods are suitable and sufficient for forecasting markets.

Chen et al. (2008) and Moosa et al. (2000) also supported that the factors influencing the movement of the market index are embodied in and reflected by the actual behaviour of the index concluding that study of historical data can be beneficial in predicting the future.

It is a fact that technical analysis contradicts the long held Efficient Market Hypothesis (EMH) which states that index movement cannot be predicted only by its historical data because information entering the market affect prices instantaneously. Although EMH may sound logical, if it was true it would mean that machine learning techniques are incapable of dealing with forecasting something that success of empirical studies have made look as a null hypothesis.

2.4 Time Series

Time series is a sequence of observations taken sequentially in time (Box and Jenkins, 2015). Examples of time series can be found in many fields such as engineering, business, economics and appear in various forms from hourly to annually time series. The main characteristic of a time series is that observations are dependent from past ones. This feature by its own is highly interesting for time series analysis, a process that seeks to identify and understand the dependencies between observations by constructing models that are applicable in everyday life problems.

Box and Jenkins et. al (2015) identified five major areas where time series can be applied:

1. The determination of the transfer function of a system subject to inertia—the determination of a dynamic input – output model that can show the effect on the output of a system of any given series of inputs.
2. The use of indicator input variables in transfer function models to represent and assess the effects of unusual intervention events on the behaviour of a time series.
3. The examination of interrelationships among several related time series variables of interest and determination of appropriate multivariate dynamic models to represent these joint relationships among variables over time.

4. The design of simple control schemes by means of which potential deviations of the system output from a desired target may, so far as possible, be compensated by adjustment of the input series values.
5. The forecasting of future values of a time series from current and past values which, consequently, constitutes the area for which this study is concerned about.

The foundation of time series analysis is stationarity. Definition of stationarity though can be flexible depending on the problem studied. A strict definition of stationarity mentions that a series is stationary if, and only if, its joint distribution does not vary with a time shift.

Tsay and Ruey et al. (2005) provided a definition that a time series $\{r_t\}$ is said to be strictly stationary if the joint distribution of $(r_{t1,\ldots,}r_{tk)}$ is identical with the distribution of $(r_{t1+1,\ldots,}r_{tk+1)}$ for all t where t, are positive integers. This definition though is very difficult to be applied in the majority of empirical problems and for that purpose a weaker definition of stationarity is used, one that mandates that a series $\{r_t\}$ is stationary if it has a constant mean and the co - variance between two observations is independent of time shift.

2.5 Auto - Regressive Integrated Moving Average (ARIMA)

2.5.1 Overview

Many practical occurring time series (e.g. stock return) have non - stationary characteristics thus making forecasting a challenge. Box and Jenkins et al. (1970) introduced a model for describing stationary and non - stationary time series which is called an autoregressive integrated moving average process, of order (p, d, q). The Auto - Regressive Integrated Moving Average (ARIMA), or Box – Jenkins, is the most widely used statistical method for time series forecasting that states that future values have a clear and direct functional relationship with current and past values (Tseng and Fang-Mei, 2001). In many cases ARIMA is used as benchmark in order to compare the newly developed models and approximate their performance. Despite its popularity for short-term predictions the Box – Jenkins method has certain limitations. It is a univariate model and is limited by the pre-assumed linear characteristics of the model (Zhang, 2003). Although similar models to ARIMA have been developed that deal with multivariate problems as well (e.g. ARIMAX) there is still no way for it to handle non-linearities in the data by itself which is the most common occurrence in real world problems.

2.5.2 Stationarity

A strict assumption required for the development of time series models is about stationarity. Usually a stationary time series can be described by its mean, variance and autocorrelation function or alternatively by its mean, variance and spectral density function. Similarly, a process is said to be stationary if its properties stay unaffected by change in time. Such a process has the same probability distribution p(z) for all the observations z in all times t and a mean μ than can be described by the following relationship:

$$\mu = \int_{-\infty}^{\infty} zp(z)dz$$

Consequently, the variance σ_z^2 can be referred as a spread and expressed:

$$\sigma_z^2 = \int_{-\infty}^{\infty} (z-\mu)^2 p(z)\, dz$$

2.5.2.1 Stationary Models

Stationary models, a sub category of stochastic models, have attracted a lot of attention from researchers when it comes to time series forecasting. According to Box and Jenkins stationary models assume that the process remains in statistical equilibrium with probabilistic properties that do not change over time and more specifically they vary around a constant mean and variance.

Brooks et al. (2010) defined a model as a system that has to satisfy certain conditions in order to have stationary properties:

$$(1)\ E(y_t) = \mu$$

$$(2)\ E(y_t - \mu)\ (y_t - \mu) = \sigma^2 < \infty$$

$$(3)\ E(y_{t_1} - \mu)\ (y_{t_2} - \mu) = r_{t_2-t_1} \text{ , for any } t_1, t_2$$

Where y_t represents the output for a given time period t, σ^2 is the finite variance term of the disturbance and r accounts for the relationship between y in any given time with the corresponding value of y in the past.

The above equations define that a stationary process has to present a constant mean, covariance and constant auto – covariance respectively for any given point in time. In addition, empirical studies have showed that extreme values will eventually be "absorbed" by the stationary system which is not the case in a non -stationary process.

The Box – Jenkins model is comprised by two more models: AR which includes auto – regressive terms and MA which refers to a moving average model. Both of them are classified as stationary models.

The ***Auto – Regressive Model*** specifies that that the output variable y depends only on its own past values and an imperfectly predictable term which can be described as an error term. The essence of the auto – regressive model is that it is a regression against itself. Mathematically the AR model of order p can be represented as:

$$Y_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

where c is a constant,φ is a parameter, x is a predictor and ε is the error term (white noise).
The ***Moving Average Model*** in time series is a common univariate modelling technique and specifies that an output variable Y is linearly dependent only on its own current and past values

of an imperfectly predicted term (error). The Moving Average model of order q can be expressed with the following mathematic formula:

$$Y_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

where ε is the error term, μ is the mean and θ the parameter of the moving average part.

2.5.2.2 Non – Stationarity Models

For non - stationary processes a combination of the Auto – Regressive (AR) and Moving Average (MA) models have been used which is known as ARMA (p, q) model. It is defined as one that the output value is depended in a linear relationship from the past output values of the time series and the error term is also depended on the past error term values as well. A representation of the ARMA (p, q) model follows:

$$y_t = \varepsilon_t + c + \sum_{i=1}^{p} \varphi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-q}$$

Where ε represents the error term, φ, θ are the parameters of the aggregated individual models and c a constant.

As expected the new developed ARMA model will present the properties of both the Auto – Regressive and Moving Average models. Tran and Reed et al. (2004) outlined that if someone wants to discriminate the two models (AR, MA) can observe the auto correlation (ACF) and partial auto correlation (PACF) functions correlograms. The same research though highlighted that this was not possible when it came to distinguish an ARMA and an auto – regressive model because both have a function that decays rapidly. As a result, the partial auto correlation was found to be more suitable for this purpose.

The ARIMA model derived as an alternative to ARMA which handles empirical data differently and thus managing to acquire better results. In general, ARIMA is a transformation of ARMA that includes a degree of differencing to the time series (one or two degrees) which deals with non - stationarity if present. This model is mostly seen as ARIMA (p, d, q) where p is the order (number of time lags) of the autoregressive model, d is the degree of differencing (the number of times the data have had past values subtracted), and q is the order of the moving-average model.

2.5.3 ARIMA Modelling Process

ARIMA models are, in theory, the most general class of models for forecasting a time series. The ARIMA model can be viewed as a “filter” that tries to separate the signal from the noise, and the signal is then extrapolated into the future to obtain forecasts.

In time series the equation representing the ARIMA has linear properties in which the predictors consist of lags of the output variable and lags of the forecast errors:

$$\begin{aligned}\text{Predicted value of Y} = &\text{ a constant and} \\ &/\text{or a weighted sum of one or more recent values of Y a} \\ &/\text{or a weighted sum of one or more recent values of the errors.}\end{aligned}$$

The model is called purely auto – regressive if its predictors consist only of lagged values of the dependent variable, which can be fitted easily with a simple regression model. In case the predictors are lags of the error it is not possible to fit a regression model due to the fact that that the predictor of the last time frame cannot be estimated. To simplify the above, the error parameters need to be approximated on a time to time basis. From technical standpoint, the reason that the lagged error terms cannot be used as predictors is that, despite the fact that predictions of the model have a linear relationship with past data that’s not the case with the coefficients. As a result, adopting non- linear optimisation methods is mandatory when coefficients of the ARIMA model include lagged errors.

The forecasting equation is constructed as follows. First, let y denote the dth difference of Y, which means:

$$\text{If d=0: } y_t = y_t$$

If d=1: $y_t \ = \ y_t \ - \ y_{t-1}$

If d=2: $y_t \ = \ (y_t \ - \ y_{t-1}) - (y_{t-1} \ - \ y_{t-2}) \ = \ y_t \ - \ 2y_{t-1} \ + \ y_{t-2}$

In terms of y, the general forecasting equation is:

$$\hat{y}t \ = \ \mu + \varphi_1 y_y + \cdots + \varphi_p y_t - p - \theta_1 e_t - 1 - \cdots - \theta_q e_t - q$$

To identify the appropriate ARIMA model, the raw data must be visualised in order to estimate potential trend or seasonal patterns. In real world problems visualisations can be inconclusive thus more formal ways of determining the existence or not of stationarity are used. One of these methods is the Dickey – Fuller statistical test. The Dickey - Fuller test provides with a test statistic value, a p-value and several critical values corresponding to different level of confidence. If the p-value is less than 0.05 or the test statistic is less than the critical value it can be concluded that the series is stationary to a degree of confidence. Another largely applied method to check if a time series has a constant mean and variance is the auto correlation function (ACF) correlogram. According to Startz et al. (2008), if the ACF correlogram depicts a fast decay to zero then the time series can be regarded as stationary. If the data do not have stationary properties an order of differencing (d) is needed in order to remove the gross features of trend and/or seasonality, perhaps in conjunction with a variance - stabilizing transformation such as logging. The p and q parameters of the ARIMA can be estimated by the ACF and PACF (Partial auto correlation function) correlograms as well by considering the values in different lags. When this is done, the process is continued with diagnostics in order to specify if the model is adequate for the purpose in hand. Initially, the model is trained using a part of the historical data and validated by forecasting data already in hand and then comparing those with the actual values. There are several measurements of accuracy in forecasting some of the most usually used being mean squared error (MSE), mean absolute error (MAE), root mean squared error (RMSE). If the model developed is judged to be of sufficient accuracy the out of sample forecast can take place else the process continuous from re - estimating the model parameters until an acceptable result is reached.

Random-walk and random-trend models, autoregressive models, and exponential smoothing models are all special cases of ARIMA models.

2.7 Artificial Neural Networks

2.7.1 Overview

Artificial neural networks are defined as computing systems whose main theme derives from the way biological neurons, of humans or animals, process information (Mehrotra, Kishan, Chilukuri K. Mohan, and Sanjay Ranka, 1997). Initially introduced from McCulloch and Pitts et al. (1943), artificial neural networks (ANNs) sought to deal simple intelligence or pattern recognition problems that seemed to be hard to automate but appeared easy for biological organizations. At the same time Von Neumann and Turing et al. (1943) discussed some ground - breaking information about the statistical and robust behavior of systems imitating a biological brain. Although the hardware limitations of the time restricted further development on the subject Rosenblatt et al. (1950) introduced the perceptron, an algorithm that would constitute one of the most important components of neural networks.

The basic principles of ANNs were first formulated by McCulloch and Pitts et al. (1943), in terms of five assumptions, as follows:

1) The activity of a neuron in an ANN is all-or-nothing, a binary procedure.
2) A certain fixed number of synapses larger than one must be excited within a given interval of neural addition for a neuron to be excited.
3) The only significant delay within the neural system is the synaptic delay.
4) The activity of any inhibitory synapse absolutely prevents the excitation of the neuron at that time.
5) The structure of the interconnection network does not change over time.

Undoubtedly the key characteristic of neural networks is their ability to learn from experience and by doing so improving their performance (Fyfe, 2000). The previous property sets ANNs as the most appropriate structures for pattern identification in fuzzy or ill - defined information.

Briefly, the procedure that allows artificial neural networks to cope with patterns in data is an algorithm which identifies relationships between inputs and outputs.

Each connection, called synapses, can transmit a signal from one artificial neuron to another. An artificial neuron can receive a signal and after processing can transfer it to the next one.

In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by a non - linear function which aggregates the inputs. Artificial neurons and the connections between them typically have a weight that adjusts as learning proceeds. The decisive factor of whether the signal will be transferred from neuron to neuron is a function.

Artificial neural networks also referred to as "neural nets," "artificial neural systems," "parallel distributed processing systems," or "connectionist systems", at a simplified topography are comprised of an input layer, a hidden layer and an output layer and information flow from one layer to another.

2.7.2 Biological Neural networks

As reported earlier the basic concept behind artificial neural networks is that they approximate the function of the human and animal brain. The four basic components of a biological neural network are dendrites, cell body, synapses and axon (Figure 2.1).

A biological neural network is comprised of edges called “dendrites which receive signals and pass them to the cell body. Dendrites are “edges” who are responsible for receiving and passing information to the cell body after the weighting process has occurred into synapses. Axon receives the signal from the cell body and transfers it to the dendrites of other adjustment neurons. Nucleus is inside the cell body and keeps information important to the neuron. Obviously, since every neuron has multiple dendrites and is connected to many other neurons as well, multiple signals can be transferred to multiple directions simultaneously (Fyfe, 2000).

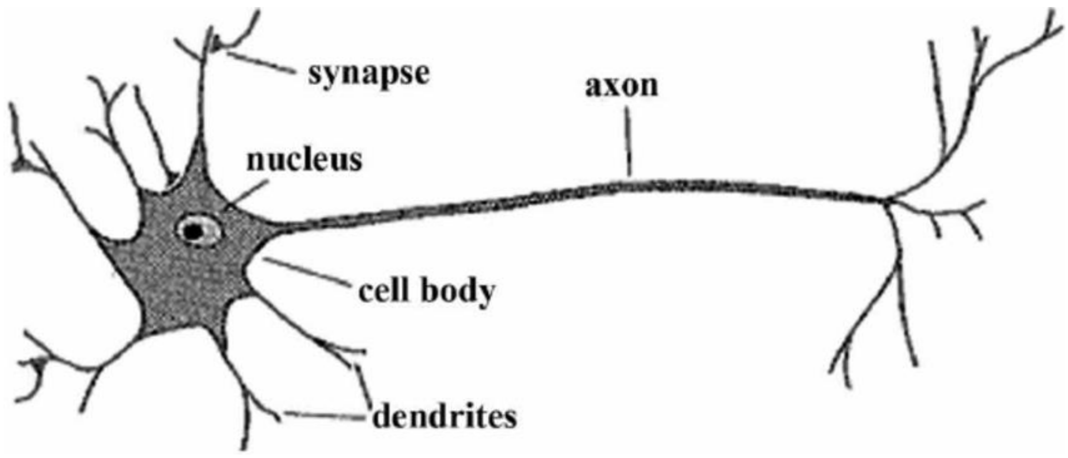


Figure 2.1 A biological neuron anatomy

Source: Borah, Tripti, Kandarpa, and Pranhari et al. (2016)

The resemblance of biological and artificial neural networks can be visualised in Figure 2.2 where dendrites receive the data mimicking dendrites. Then the data are stored into hidden nodes of the middle layer having been assigned a specific weight first. Finally, a function aggregates the weighted signals and then depending on the threshold value information can be transferred into the output nodes. From 1943, when the first ANN was introduced, it has become significantly more complicated than the one depicted below. As a result, artificial neural networks tend to resemble the biological ones even more accurately.

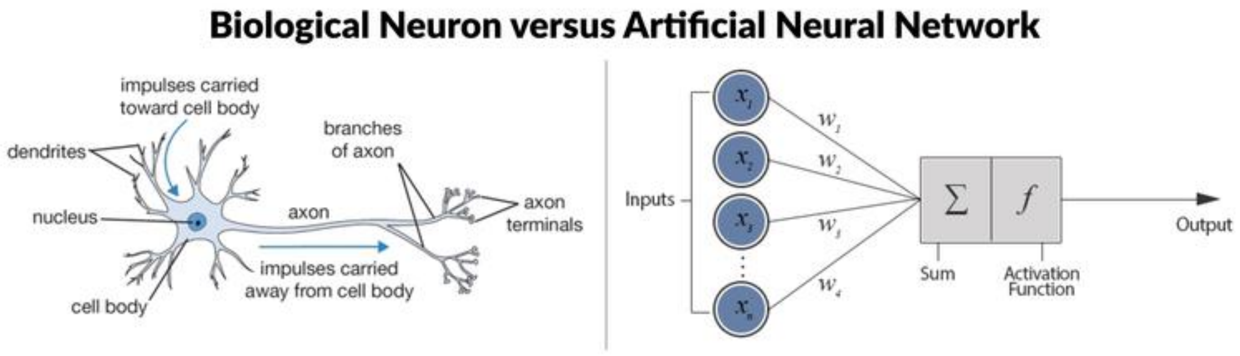


Figure 2.2 Comparison of a biological and artificial neuron

For ease the parts of a biological neuron and the corresponding artificial are summarised in Table 2.1.

Table 2.1 Terminology in Biological and Artificial Neural Networks

Biological Terminology	**Artificial Neural Network Terminology**
Neuron	Node/Unit/Cell/Neurode
Synapse	Connection/Edge/Link
Firing Frequency	Node Output

2.7.2 The Artificial Neuron Topology

As mentioned before an artificial neural network is a processing system that seeks to simulate the functions of biological brain and is comprised of multiple interconnected artificial neurons.

According to Vo Chi My et al. (2014) there are three basic components of a neuron:

1. A tuple of inputs [$x_{1, \ldots,} x_n$] arrives at the dendrites (probably from other neurons) and then is multiplied by a corresponding weight in the synapses. The value of the weight can range from negative to positive values for each of the inputs.
2. A propagation function f_{prop} combines the weights and produces an output net_i which is considered as the input for the neuron to work and is represented as:

   $$net_i \; = \; f_{prop}\left(x_1, x_2, \ldots , x_n, w_{i_{1,j}}, w_{i_{2,j}}, \ldots , w_{i_{n,j}}\right) = \sum_{j=1}^{n} w_{ij}x_j$$

   Where x is the input signal, w is the weight assigned from the synapses and net_i is the output of the propagation function.
3. The activation function f_{act} operates as filter which allows the information to go through if the input is greater than an activation value. In addition, a bias term θ_j,

influences the horizontal offset of the function. The bias may be treated as the weight from the supplementary input unit, and is utilized to increase to either increase or decrease the input of the activation function by an equivalent number of the bias. (Schalkoff, 1997).

The output of the artificial neuron can be denoted mathematically as:

$$y_j = f(net_i + \theta_j)$$

Where y_j is the output of the activation function and θ_j the bias term applied.

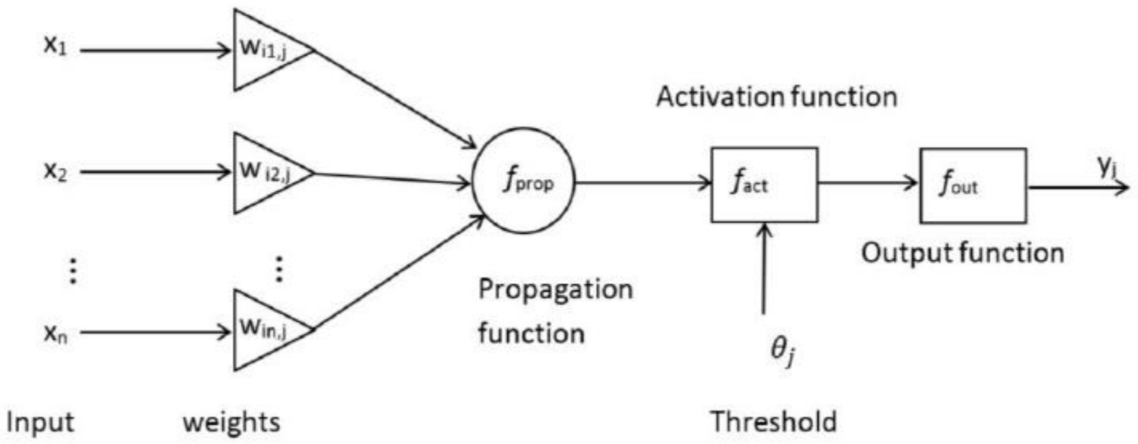


Figure 2.3 The structure of an artificial neuron

Source: Vo Chi My et al. (2014)

Let the output of the propagation function f_{prop} be denoted as:

$$z = \sum_{i=1}^{\infty} x_i w_i + \theta_i$$

The first artificial neuron, the perceptron, was formulated by Rosenblatt et al. (1958) and serves as a building block to most later models. According to him a perceptron is a machine that learns, using examples and a linear function of the inputs in order to assign samples to different classes. A bit later, Minsky and Papert et al. (1969) described the perceptron as a stochastic gradient-descent algorithm that attempts to linearly separate a set of n - dimensional training data. In its simplest form, a perceptron has a single output whose values determine to which of two classes each input pattern belongs. A single node applies a linear step function to the inputs and forces them to a zero or one output.

In comparison to the simplified approach that Rosenblatt et al. (1958) proposed for classification problems, since then there have been many different algorithmic procedures. The most modern artificial neural networks are generalization of the classical perceptron approach. An ANN sums weighted inputs, but instead of producing a binary output of 0 or 1 based on a threshold, it tries to classify the samples in to similar groups by producing a smooth graded value between 0 and 1 based on the distance between an observation and each of those groups. The network nodes are usually biased toward the extreme values of 0 or 1.

2.7.3 Activation Functions

As described in the previous sections one of the characteristics of artificial neural networks is the activation function they apply in the weighted sum of inputs in order to classify samples in to different categories based on the output value. The activation function of the perceptron as proposed by Rosenblatt et al. (1958), assumed a linear relationship but since then many more have been introduced the majority of which is non - linear since most of the practical problems in the real world include non - linear data as well. Despite their differences in how they classify objects in general terms the purpose of all activation functions is to receive the weighted inputs and determine the output based on the algorithm they follow. The most popular and widely used activation functions are described in detail in the next sections.

2.7.3.1 Heaviside Step Function

This particular function was introduced from McCulloch and Pitts et al. (1943) and was the first to be used in pattern recognition with artificial neural networks. The basic idea behind this approach is that the output of the neuron will equal to one if the weighted sum of inputs arriving to the activation node is positive or zero and zero in every other case. In mathematical terms the above is expressed as follows:

$$y = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

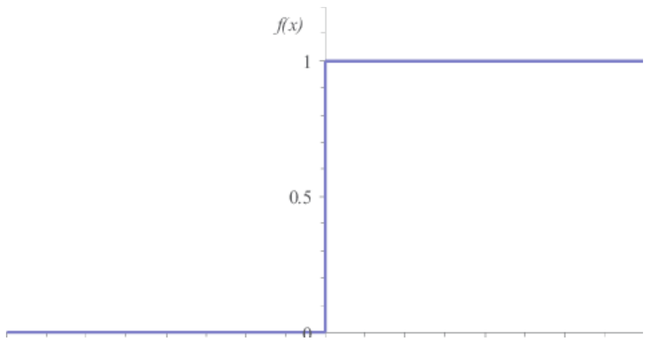


Figure 2.4 Heaviside Step function

2.7.3.2 Sigmoid/Logistic Function

The most frequently used activation function is called Sigmoid or Logistic whose main characteristic is that is continuously differentiable (Graupe, 2013). It manages to satisfy the relation depicted in Figure 2.5 with the following formula:

$$y = \frac{1}{1 + \exp(-z)} = f(z)$$

Such as that: $y = \begin{cases} 0, & z \to -\infty \\ 0.5, & z \to= 0 \\ 1, & z \to \infty \end{cases}$

Jain et al. (1996) writes that the sigmoid function is a strictly increasing function with asymptotic properties and smoothness and its merits lie on the balance between a linear and non – linear behaviour. Although the sigmoid function has an upper limit of infinity and a lower limit of minus infinity it should be noted that it will never reach these values. Consequently, setting the sigmoid function to 0.9 and 0.1 should be considered as fully activated and turned off respectively.

Roughly speaking, the sigmoid function assumes minimal structure and reflects our general state of ignorance about the underlying model.

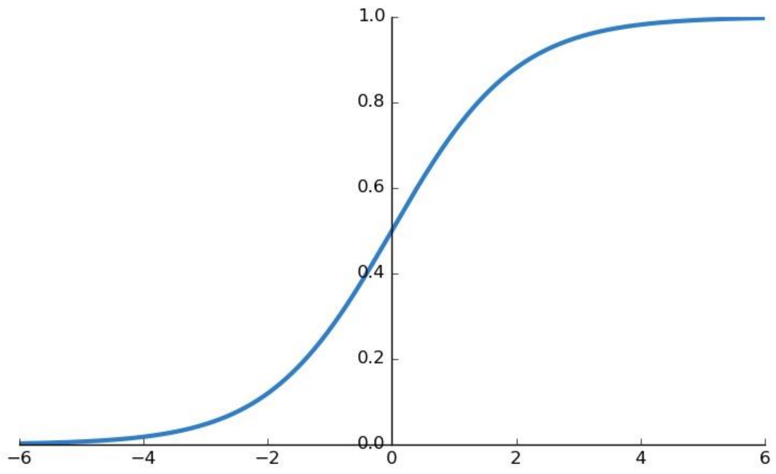


Figure 2.5 Sigmoid Function

2.7.3.3 Hyperbolic Tangent Function

Another popular activation function is the hyperbolic tangent function that is expressed as:

$$y = \frac{\sinh(z)}{cosh(z)} = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} = \tanh(z)$$

It ranges between [-1, 1] and represents the ratio between the hyperbolic sine and cosine functions. Many more functions similar functions have been developed based on the hyperbolic tangent such as the softsign proposed by Bergstra et al. (2009) which approaches its asymptotes much slower.

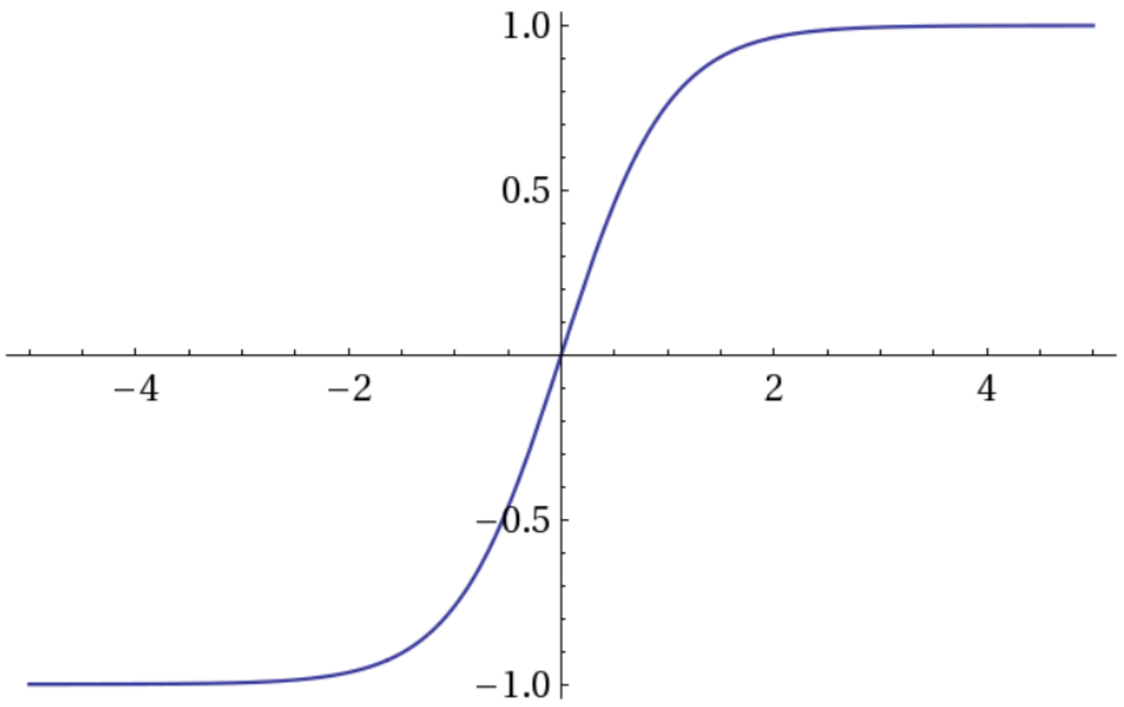


Figure 2.6 Hyperbolic tangent function

2.7.4 Architectures of Neural Networks

A group of interconnected artificial neurons is regarded as a neural network. In general, the neural networks can be divided into three parts, named layers, which are known as:

1) **Input layer**

   Information, signals, features or measurements from the external environment flows into this layer. These inputs (features) are represented by input nodes which correspond to the number of the variables (columns in data) of the prediction model. The inputs are usually normalised within the limit values produced by activation functions which helps obtaining better results after the network operations (Da Silva, 2017). The non - linear model that is produced in the time series forecasting uses as inputs the results of the learning process.

2) **Hidden/Invisible layer**

The composition of this part of the network is with one ore more hidden nodes. In this part of the structure takes place the pattern recognition and most of the analysis operations for the problem that is studied. One of the biggest problems in ANN analysis is the number of the hidden nodes and layers that should be used during the formulation process and still attracts a lot of research attention since an optimal number has not been found yet in theory for dealing with practical problems. The generalisation of Akaike's information criterion (AIC) has been proposed by Murata et al. (1994) as a static procedure to determine the right number of hidden nodes through general loss criteria. Many more researchers studied in practice the problem and applied two, three and four hidden layers in order to test their performances. The results were that two hidden layers in many cases perform better than one layer and according to Villiers and Barnard et al. (1992) three and four - layer networks of same complexity perform equally good in all aspects but the model with the four layers led to bad local minima. When it comes to the size of each layer some empirically-derived rules-of-thumb suggest that the number of neurons should be between the size of the input and the output layer.

Heaton et al. (2008) noted that there are some rules-of-thumb for determining the correct number of neurons to use in the hidden layers, such as the following:

- The number of hidden neurons should be between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
- The number of hidden neurons is correlated to the size of the training set (a small training set can not have many hidden neurons neither a large one can have very few neurons).

Ultimately, the selection of an architecture for your neural network will come down to trial and error.

3) **Output layer**

   Information from the previous layers flows in the third layer which is responsible for producing and presenting the outputs of the whole system. Every input neuron represents some independent variable that has an influence over the output of the neural network thus the input and output layer should be examined as a whole and not as individual parts.

2.7.4.1 Feedforward Neural Network

Feedforward neural networks have been largely used to handle complex problems of the real world and due to their simplicity compared to other models and success rate. In this type of ANNs, the outputs that derive from the neurons are directly feed forward to subsequent layers and there is only one direction that information travels meaning that individual units of the system do not belong in to a cycle. In a graphical representation of a feedforward neural network where the vertices are the neurons and edges are the connections the graph will be acyclic meaning that there is no path from the ending that can trace back to the start. The main characteristic of theirs is their quick responsiveness to inputs. Data access input layers and with the assistance of hidden layers end up to the output layers.

Benardos and Vosniakos et al. (2007) noted that there are four elements in the architecture that heavily affect the ANN's performance:

1. The number of layers.
2. The number of neurons in each layer.
3. The activation function of each layer.
4. The training algorithm (because this determines the final value of the weights and biases).

The number of each kind of layer and the size (number of neurons) can vary and ultimately are the most critical elements in a feedforward network architecture.

2.7.4.2 Recurrent/Feedback Neural Network

The characteristic of this class of neural networks is that they there are inherent feedback connections between the neurons of the network. More specifically recurrent neural networks (RNNs) allow node connections between the output layer, the hidden and/or input layer and they can even include connections between nodes of the same layer. They form a dynamic system with a graph that unlike feedforward neural networks represent a cyclic diagram and they can store information over time which changes until they reach an equilibrium point. Since the output of a neuron cannot be a function of itself, such an architecture requires that time be explicitly taken into account so that the output of a neuron cannot be a function of itself at the same instant of time, but it can be a function of its past value(s) (Dreyfus, 2005).

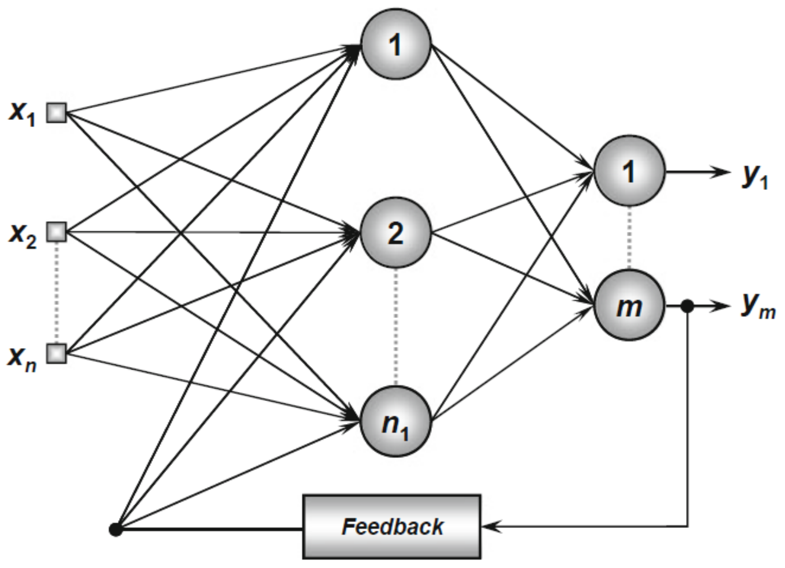


Figure 2.7 Recurrent neural network with one hidden layer

Source: Da Silva and Ivan Nunes et al. (2017)

Hochreiter and Schmidhuber et al. (1997) proposed a revolutionary RNN type network the Long – Short Term Memory (LSTM). LSTM in practice was a dynamic system that adopted the same structure units as the classical RNN but also included a long - term memory which allowed it to store a state for longer periods and empowered it with the ability to control the whole "storing – forgetting" procedure. Artificial neural networks and especially LSTMs have been the subject of study for many years due to their high performance when dealing with complex problems and are widely applied. Some of their advantages follow below:

1. Recognition of temporally extended patterns in noisy input sequences
2. Recognition of the temporal order of widely separated events in noisy input streams
3. Extraction of information conveyed by the temporal distance between events
4. Stable generation of precisely timed rhythms, smooth and non-smooth periodic trajectories
5. Robust storage of high-precision real numbers across extended time intervals

2.7.5 The Multilayer Perceptron (MLP)

Minsky and Papert et al. (1969) outlined some essential flaws of Rosenblatt’s perceptron. Indeed, studies pointed out that the single layer of the perceptron cannot solve problems such as a 2-state Exclusive-Or (XOR) and behaves poorly in classification when the inputs of the problem is very large. The frustration and disappointment of the research community of the limitations of the perceptron became soon obvious something that resulted in an exponential reduce in using of the model.

The multilayer perceptron (MLP) came as a counter measure to all the limitations faced by the perceptron. MLP is a feedforward neural network model which follows the function of the simple perceptron but it differentiates itself by including more than one layers. What also distinguishes it from the first is also the fact that each node has a non - linear activation function which enables it to deal with nonlinearities in the data and furthermore be applied in more than one dataset after training. It is famous for its generalisation and has been widely used to solve many complex problems.

In Figure 2.8 a MLP with three input nodes, four hidden nodes and one output node is presented. Weighted neurons linearly sum up and transfer to the next layer. Non – linear activation functions are implemented in the structure as well which allows recognition of nonlinearities in the data.

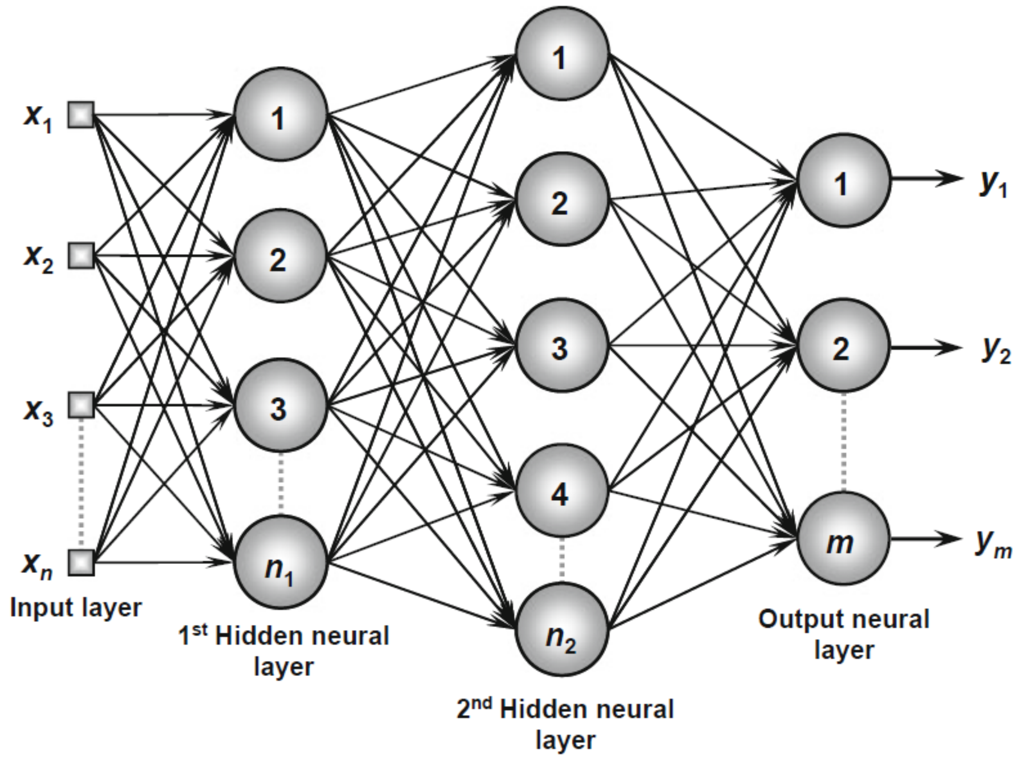


Figure 2.8 Example of an MLP with one input layer, two hidden neural layers and one output layer

Source: Da Silva and Ivan Nunes et al. (2017)

Trenn et al. (2008) identifies its advantages to be the following:

1. It simple to apply
2. Great generalisation properties
3. It can be extended and measured
4. It can be applied to numerous problems

2.7.6 The Radial Basis Function (RBF)

As was mentioned in the previous section, development of multilayer perceptron contributed in handling nonlinearities in the data that it could not be done by the classical perceptron method. The research around MLP has been focused in models that included either two or three hidden layers and usually the results had a very good accuracy. Although the multilayer perceptron could approximate any continuous variable it suffered from long training times. Broomhead and Lowe et al. (1988) formulated a new type of architecture whose output was a linear combination of radial basis functions of the inputs and the model parameters. The key element of the new system was the function calculation which was performed in the hidden layer and converted in a non – linear way the data from the input space to the hidden layer space (using a basis activation function) before the output neurons perform a weighted sum procedure. A widely used activation function has been the Gaussian whose mean and standard deviation can be determined by the data. In Figure 2.9 it is shown its graphical representation which is described by the following formula:

$$\varphi(x) = \left(\varphi_1(x), \varphi_2(x), \dots, \varphi_M(x)\right)^T = \sum_{i=1}^{N} \alpha_i \rho(||x - c_i||)$$

where i is a neuron and the centers c_i of the Gaussians will be determined by the input data x (Fyfe, 2000). Note that the term $||x - c_i||$ represents the Euclidean distance between the inputs and the ith center while the α_i parameter corresponds to the weight of the neuron i.

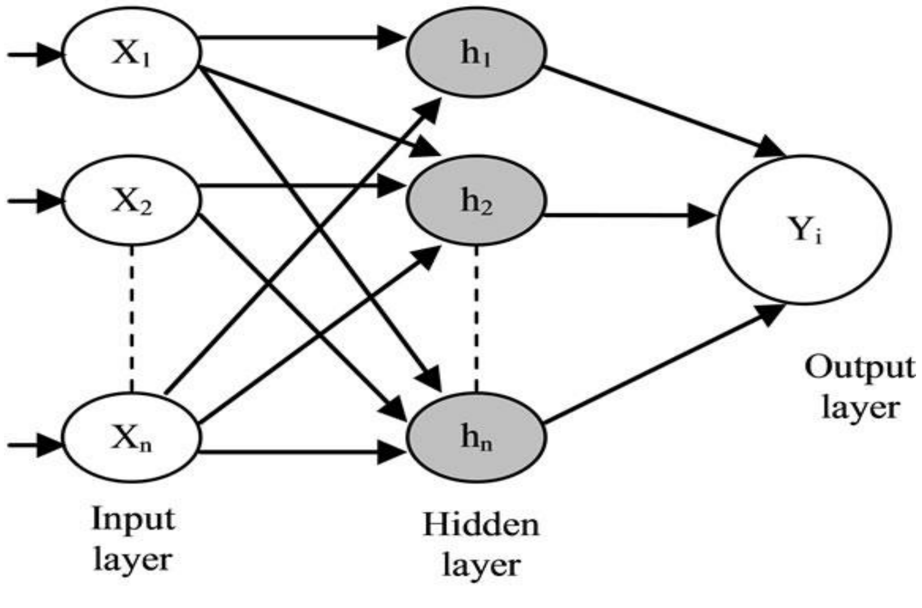


Figure 2.9 Radial Basis Function Network

Source: S. Deshmukh, J. Senthilnath, R. Dixit, S. Malik, R. Pandey, A. Vaidya, S. Omkar and S. Mudliar et al. (2012)

In a later study, Kl Du et al. (2016) summarised the basic advantages of the Radial Basis Function network as:

1. It can approximate any multivariate continuous function on a compact domain to an arbitrary accuracy, given a sufficient number of units.
2. The approximation has the best-approximation property since the unknown coefficients are linear.
3. The solution is optimal in the sense that it minimizes a functional that measures how much it oscillates.
4. Short training times for complex problems.

2.7.7 Learning Schemes

It has been noted that the most common neural networks architectures are the feedforward and recurrent. Whatever their architecture, philosophy and the rules they obey both of these classes are characterised by their ability to “learn”. Learning is the process by which a neural system acquires ability to carry out certain tasks by adjusting its internal parameters according to some learning scheme (karagiannis and Venetsopoulos, 2013). The learning process in the neural networks is what enables them to adapt in different environments which is achieved by changing the weights of the network. Many algorithms initially assign random weights and then their validity is examined. After that depending the performance of the neural network on the validity check the weights are adjusted and the process repeats itself until an acceptable accuracy is achieved. At the end of the whole procedure the neural network will have “learned” by itself without explicitly programmed and will be ready for application in the problem studied. Depending on the particular neural architecture considered, learning can be either supervised, unsupervised or reinforcement.

- **<u>Supervised Learning</u>**

  This type of learning process can be thought as “learning with a teacher”. A sample of data is provided to the network with the anticipated outputs of these samples. Information is propagated forward according to the network architecture until it reaches the output layer. If the results agree sufficiently with the expected values of the sample then the model developed can be considered for further use. On the other hand, if the results provided from the network are not acceptable then the weights adjust ("learn") and the process repeats for a number of epochs until the error rate is reasonably small. Each epoch is a pass through the sample data. Common algorithms in supervised learning include logistic regression, naive bayes, support vector machines, artificial neural networks, and random forests.

- **<u>Unsupervised Learning</u>**

  In unsupervised learning the network is fed with sample data as inputs but unlike supervised learning the target values are not provided. In this case the network is expected to self – organise and act on the information provided without guidance by identifying a function that describes the structure of the unlabelled data. Since no labels are provided, it is obvious that there is no specific way to compare model performance in most unsupervised learning methods. Some common algorithms include k-means clustering, principal component analysis, and autoencoders.

- **<u>Reinforcement Learning</u>**

  Reinforcement learning process is considered by many as a hybrid method of the supervised and unsupervised methods of training. This is because in this type of learning, information is fed in to the input layer but as it happens in unsupervised leaning the target values are not provided. Instead, the network is required to produce outputs based only to the samples provided which at the end are evaluated and only then the system is told if the result is correct enough or not. In the latter case, the weights are adjusted and the process repeats.

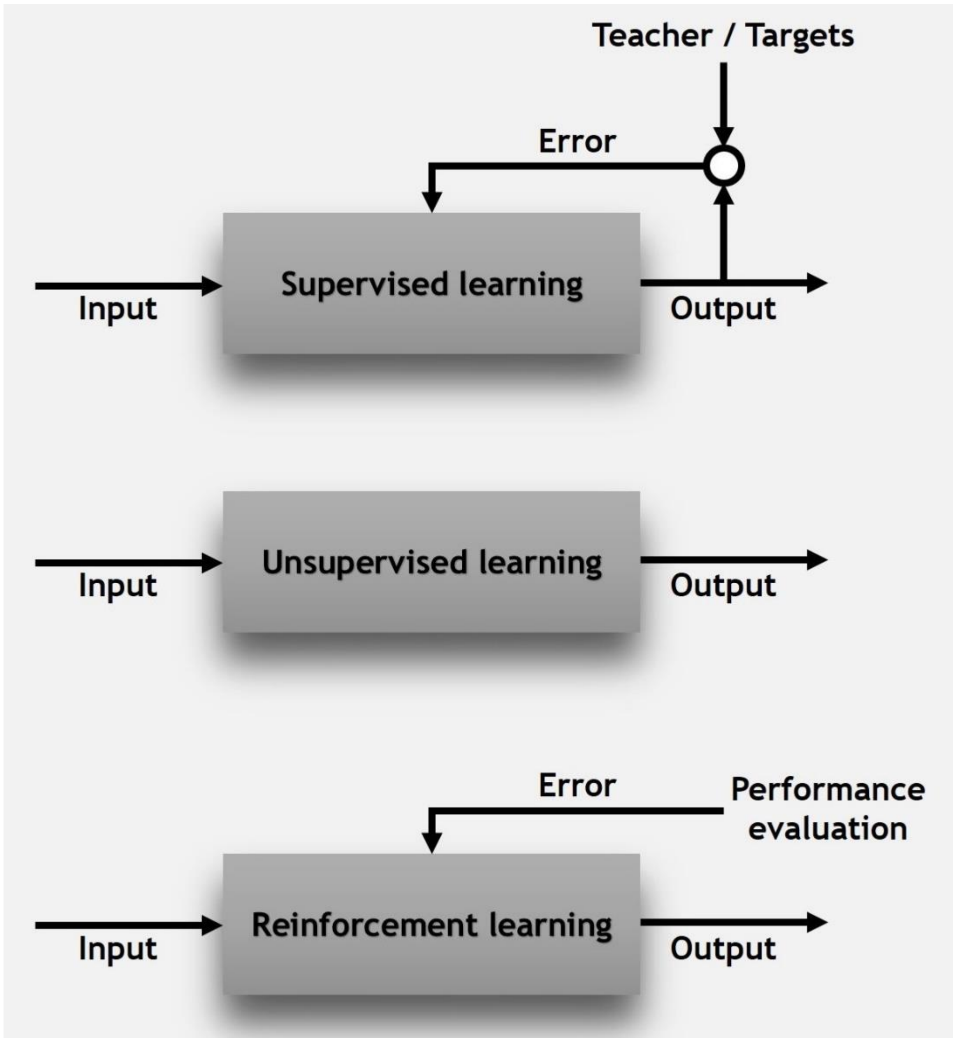


Figure 2.10 Most common learning processes

Source: Wang, Shouyi & Chaovalitwongse, Wanpracha and Babuska et al. (2012)

2.7.8 Performance Metrics

Measuring the performance of a model is a crucial step that takes place more than once during the whole process. Different models usually show different forecasting accuracy for the same problem which is dependent on their architecture, data and many more. There have been proposed many different metrics however there is no measure that is universally accepted by academics and practitioners for a given problem. Usually, an accuracy measure is representation of the forecasting error which is computed as the difference between the predicted and expected value. The four most common measurements are:

1. Mean Absolute Error (MAE)

   The MAE metric aims to avoid the offsetting of positive and negative prediction errors by calculating the absolute difference between the actual and the predicted values of the time series in the same period. It is a popular method of performance estimation which is expressed as:

$$MAE = N^{-1} \sum_{t=1}^{N} |e^{t}|$$

   Where t = time period

   N = total of error terms

   e_t = difference between every actual and predicted value

2. Mean Squared Error (MSE)

   Mean squared error, another popular accuracy measurement, aims to avoid offsetting like MAE but differentiates by calculating the average of the squared error of the forecast. Its formula is expressed as:

$$MSE = \frac{\sum (e_t)^2}{N}$$

   Where t = time period

   N = total of error terms

   e_t = difference between every actual and predicted value

3. Mean Absolute Percentage Error (MAPE)

   When performance comparison between different forecasting models and for different time series needs to be done metrics like MAE and MSE proposed above are of low credibility because there can be huge variations in the scale of the observations of the series and as a result large values will affect the metric significantly (Chatfield, 1988).

Mean absolute percentage error targets the relative measures by the comparison and percentage error of each forecast unit.

$$MAPE = \frac{1}{N}\sum_{t=1}^{N} | \frac{A_t - F_t}{A_t} |$$

Where t = time period

N = total of error terms

A_t = Actual value when time is t

F_t = Forecasted value when time is t

4. Root Mean Squared Error (RMSE)

RMSE is characterised as a great method to compare the error of different models for a particular variable. Its good performance lies on the RMSD which practically is the standard deviation of the actual and predicted values. The purpose of the RMSD is to aggregate the magnitudes of the forecasting errors for various times into a single measure of the model's prediction power and is represented as:

$$RMSE = \sqrt{\left\{\frac{1}{N}\sum_{t=1}^{N} |e_t|^2\right\}}$$

Where t = time period

N = total of error terms

e_t = difference between every actual and predicted value

2.8 Support Vector Machine (SVM)

2.8.1 Overview

Support Vector Machines (SVMs) are supervised learning methods that generates input – output mapping functions from a set of labelled data and are used for classification or regression analysis problems. Basak,Pal and Patranabis et al. (2007) refer to SVMs as learning machines implementing the structural risk minimization inductive principle to obtain good generalization on a limited number of learning patterns. Structural risk minimization (SRM) involves simultaneous attempt to minimize the empirical risk and the VC (Vapnik–Chervonenkis) dimension first introduced by Vladimir Vapnik and Alexey Chervonenkis who attempted to explain the learning theory from a statistical point of view.

Vladimir N. Vapnik and Alexey Ya. Chervonenkis et al. (1963) were the first to introduce the SVM algorithm soon to be followed by Isabelle M. Guyon and Vladimir N. Vapnik et al. (1992) who took it a step further by using a kernel trick which sought to maximise the margin of hyperplanes and as a result contributed in developing a nonlinear classifier. Finally, Boser, Guyon and Vapnik et al. (1992) introduced SVMs with a form close to the one used today before Vapnik et al. (1995) extent the system to the case of regression.

2.8.2 Linearly – Separable Data

In cases where data have linear separation properties, it is fairly easy to find the maximum margin since it is as far away as possible from the outer boundaries (convex hull) of the developed data groups. The maximum margin hyperplane is then the perpendicular bisector of the shortest line between the two convex hulls which can be calculated after applying classical quadratic optimization. Alternatively, a search in the space of every possible hyperplane can be applied in order to identify a set of two parallel planes which manage to divide the points into homogenous groups but stay at the same time as far apart as possible. The search method can easily be described mathematically.

Suppose a n-dimensional space and the following equation:

$$\vec{w} * \vec{x} + b = 0$$

Where w = {w_1, $w_{2,\ \dots\ ,}$ w_n} is a vector of size n, x is a vector and b is the bias parameter

Assuming that only two outputs are possible (-1, 1) a set of w (weights) that define two hyperplanes can be found using:

$$\vec{w} * \vec{x} + b \geq 1$$

$$\vec{w} * \vec{x} + b \leq -1$$

Where the points of one of the two classes are falling above the first hyperplane and the points of the second below the second hyperplane.

Finally, using vector geometry the task is reformulated as:

$$min \frac{1}{2} ||\vec{w}||^2$$

$$s.t.\ y_i(\vec{w} * \vec{x_i} + b) \geq 1, \forall \vec{x_i}$$

Where ||w|| is the Euclidean norm

The function that derives from the above and is responsible for the prediction of the target variable f(x) or else y is:

$$f(x) = sign(\sum_{i=1}^{l} y_i a_i(\vec{x} * \vec{x_i}) + b)$$

Where l is the number of objects in the training dataset and α and b parameters used to approximate the optimal solution.

2.8.3 Nonlinearly – Separable Data

In most of real - world problems, it is a rare thing to come across data that show linear properties. Nine out of ten cases the researcher or analyst will have to deal with data that exhibit nonlinear patterns and he or she is called to address that first in order to gain insights. As was already said in the previous sections though not all models are appropriate to handle non linearities. Isabelle M. Guyon and Vladimir N. Vapnik et al. (1992) implemented in classic SVMs the “kernel trick”, an approach who resulted in a sufficiently good performing Support

Vector Machine for non – linear data. The “kernel trick” technically is a process that enables SVMs to map the problem into a higher dimension space and by doing so it seeks to find potential linearities in the data that are non – linear in the two dimensions. Essentially, the kernel trick in SVMs involves addition of extra features that express mathematical relationships between measured characteristics. The relationship in mathematical formula terms is:

$$K\left(\vec{x}_i \vec{x}_j\right) = \varphi(\vec{x}_j) * \varphi(\vec{x}_j)$$

Where φ represents the mapping in the new multidimensional space

In the case of non – linear separable data it is essential to handle the error terms in order to define how much tolerable the system should be. Assuming that ξ represents the maximum error permitted then:

$$min \frac{1}{2} ||\vec{w}||^2 + C \sum_{i=1}^{l} \xi_i$$

$$s.t.\ y_i(\vec{w} * \vec{x_i} + b) \geq 1 - j, \forall\, \vec{x_i} \xi_i \geq 0, i = 1 \dots l$$

Mentioned already in previous chapters the kernel trick allows the usage of different kernels for the sake of better separation of data that present non – linearities.

There are a lot of kernels that have been developed throughout the years in order to fit as many types of problems as possible under a universal system. The most common and popular are listed below:

- Linear Kernel

  The linear kernel does not require any data transformations. Instead it can be imagined as a transformation of the data to themselves. Its purpose is to address problems that already show linear properties and is described as:

$$K\left(\vec{x}_i, \vec{x}_j\right) = \vec{x}_i * \vec{x}_j$$

- Polynomial Kernel

This type of kernel comes with a degree d, where d is a natural number and adds a simple non – linear transformation. The formula describing this kernel is shown below:

$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i * \vec{x}_j + 1)^{\alpha}$$

When d = 1, the kernel is the linear plus 1. For different d values it can be mathematically proven that polynomial kernels satisfy Mercer's condition.

- Radial Basis Function (RBF)

  There are many similarities between the RBF kernel of the Support Vector Machines and the one of the Neural Networks. This is the most widely used type of kernel due to its very good performance in a variety of problems and also a good benchmark for many learning tasks. Here the support vectors are the centers of the radial basis functions. A considerable drawback of this method is that RBF is influenced by outliers due to the fact that it employs Euclidean distance something that was dealt in the work of Chen et al. (2004) who proposed the *M*-estimator-based robust kernels, which are robust versions of RBF kernels. Its formula follows:

$$K(\vec{x}_i, \vec{x}_j) = \tanh{(\vec{x}_i * \vec{x}_j - \delta)}$$

As with model selection there is not a straightforward approach to define the best SVM kernel for a specific task. The most common way to go about with this problem is the typical trial – error approach in order to validate several SVMs with their corresponding error metrics.

2.8.4 Support Vector Machine – Classification

Support Vector Machines are relatively new methods, more known for their data classification properties which are widely used for both linear and non – linear separable data. To better describe the concept behind SVM one can picture a surface that defines a boundary between the points of a given dataset with labelled data (supervised learning), each point representing examples plotted in multidimensional space. The previous represents a simple SVM intended for two dimensional problems and is more often met as a “binary classification” problem. The objective of the SVM is to identify a flat boundary, often met under the term hyperplane, which results in relatively homogeneous partitions of the data on either side of the hyperplane. The examples that end to be closest to the margin hyperplane are called support vectors. The larger the margin of the hyperplane the better the data are classified thus the prediction power of the final model constructed increases.

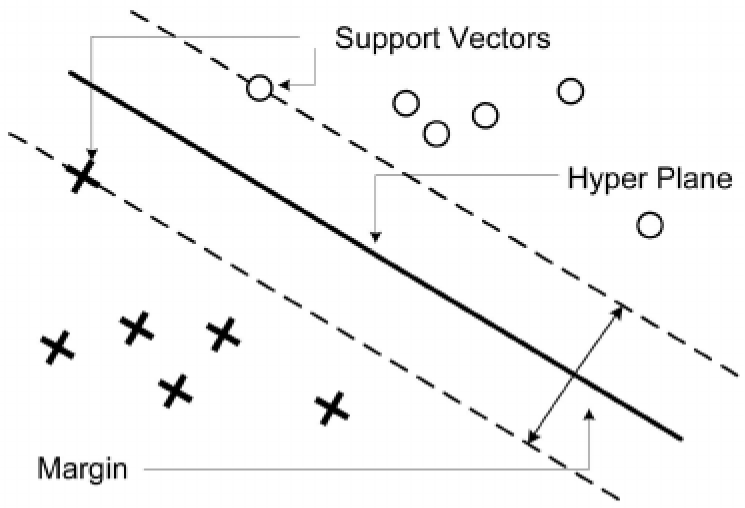


Figure 2.11 A two - dimensional classification

Source: Dacheng, Xiaoou, Xuelong and Xindong et al. (2006)

Apart from the binary classification, SVMs with multiclass classification capabilities have been constructed as well. Some typical ways to achieve that have been proposed throughout the years with the most classical to be from Weston, Jason, and Chris Watkins et al. (1999) who proposed a maximisation of dual objective functions for multiple separating hyperplanes. Alternatively, another popular method is the one-against-all classifiers who employees C separate SVMs with the last one being trained with data from class C who are labelled as positive while the rest of the data classes are considered as negative. The mathematics for both cases were known for a long time but their popularity burst when SVMs prediction performance was discovered and widely used after that for many problems. Although it would be highly interesting to understand mathematically how Support Vector Machines manage to solve such a variety of tasks their complexity prohibits such a thing given the purpose of this paper.

2.8.5 Support Vector Machine – Regression

Support Vector Machines with either classification or regression properties are two very hot research topics. Especially for the latter, due to its experimental good performance it is a widely used method for time series prediction and financial related problems. Support Vector Regression (SVR) operates under the same principles as classification SVMs with a couple of significant differences though. Firstly, since the output of the prediction is not any more a binary value but numeric, thus with infinite possibilities, the actual forecasting becomes very difficult. In addition, and based on the previous declaration, a term ε, representing the margin tolerance, is set. Despite the differences though the main objective remains the same: Minimisation of the error or alternatively maximisation of the margin.

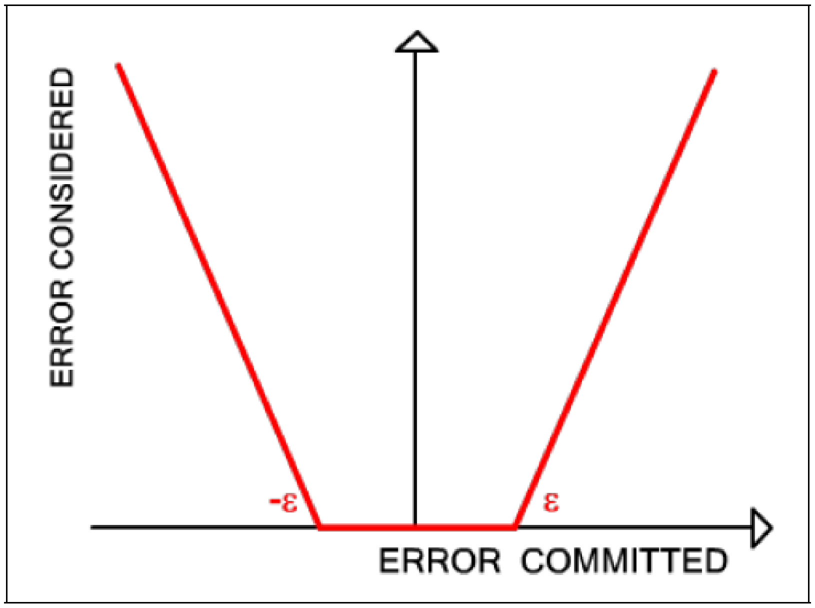


Figure 2.12 Error function

Source: Parrella et al. (2007)

Up until the threshold, the error is considered 0, after the error it becomes calculated as ”error-epsilon”.

In the case of the SVR the quadratic optimisation problem becomes:

$$\min \frac{1}{2}||\vec{w}||$$

$$\begin{cases} y_i - (\vec{w} * \varphi(\vec{x}) + b) \leq \varepsilon \\ (\vec{w} * \varphi(\vec{x}) + b) - y_i \leq \varepsilon \end{cases}$$

Where $\varphi(x)$ is the kernel function described in the previous section, w represents the margin and x_i, y_i are the input and output variables respectively. As in classification, a boundary can be set to account for the tolerance preferred:

$$min\frac{1}{2}||\vec{w}|| + C\sum_{i=1}^{l}(\xi_i + \xi_i{}^*)$$

$$s.t. \begin{cases} y_i - (\vec{w} * \varphi(\vec{x}) + b) \leq \varepsilon + \xi_i \\ (\vec{w} * \varphi(\vec{x}) + b) - y_i \leq \varepsilon + \xi_i{}^* \\ \xi_i, \xi_i{}^* \geq 0, i = 1 \ldots l \end{cases}$$

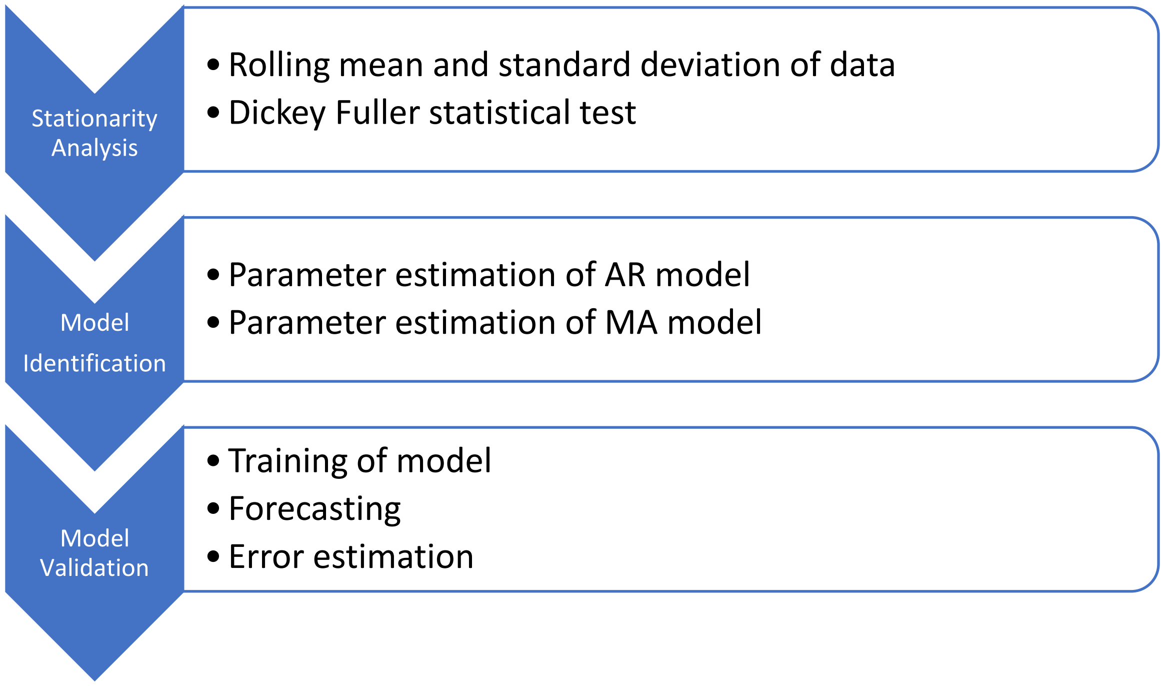
Finally, the formula under which SVR will operate in order to make its prediction is presented:

$$f(x) = \sum_{i=1}^{l} \theta_i \varphi(x, x_i) + b$$

Chapter 3. Implementation

3.1 Implementation of ARIMA

One of the most popular methods of stock index forecasting is the ARIMA model. As already mentioned before, ARIMA is a statistical model that seeks to produce accurate predictions by applying univariate time series analysis in historical data. Although this method is widely studied and used, this paper seeks to use the Box – Jenkins model more as a benchmark compared to the rest developed. The implementation of ARIMA was done using Python, an interpreted high - level programming language that includes a vast and comprehensive standard library thus making it an ideal tool for data analysis, and the process can be categorized as follows:


Stationarity Analysis
Rolling mean and standard deviation of data
Dickey Fuller statistical test
Model Identification
Parameter estimation of AR model
Parameter estimation of MA model
Model Validation
Training of model
Forecasting
Error estimation

The full code for the data analysis and ARIMA implementation is included in Appendix C.

3.1.1 Stationarity Analysis

A stationary process is defined as a stochastic one whose probability distribution does not change shifted over time. Consequently, mean and variance will not change if a process is stationary, thus this paper assumes for practical reasons that a time series is considered to be stationary if the above statistical properties remain constant over time. The importance behind stationarity is that, if a time series is proved to be one, then it is highly likely that it will follow the same trend in the future allowing predictions with high accuracy. A traditional approach taking the autocorrelation and autocorrelation plots of the data and examine their movement.

Dickey D. and Fuller W. in 1979 proposed a formal statistical test that defined if the degree of differentiating of the data is sufficient or should be continued further. The null hypothesis of the test is that the time series can be represented by a unit root, that it is not stationary (has some time-dependent structure). The alternate hypothesis (rejecting the null hypothesis) is that the time series is stationary.

- **Null Hypothesis (H0)**: If failed to be rejected, it suggests the time series has a unit root, meaning it is non-stationary. It has some time dependent structure.
- **Alternate Hypothesis (H1)**: The null hypothesis is rejected; it suggests the time series does not have a unit root, meaning it is stationary. It does not have time-dependent structure.

We interpret this result using the p-value from the test. A p-value below a threshold (such as 5% or 1%) suggests we reject the null hypothesis (stationary), otherwise a p-value above the threshold suggests we fail to reject the null hypothesis (non-stationary).

- **p-value > 0.05**: Fail to reject the null hypothesis (H0), the data has a unit root and is non-stationary.
- **p-value <= 0.05**: Reject the null hypothesis (H0), the data does not have a unit root and is stationary.

Both ways have been used to perform the analysis of the data and provide more accurate conclusions.

The first step of the basic ARIMA time series analysis of the historical data mandates that a plot of the raw data, in this case the daily closing prices of the market, over time should be performed. The aforementioned plot is shown below:

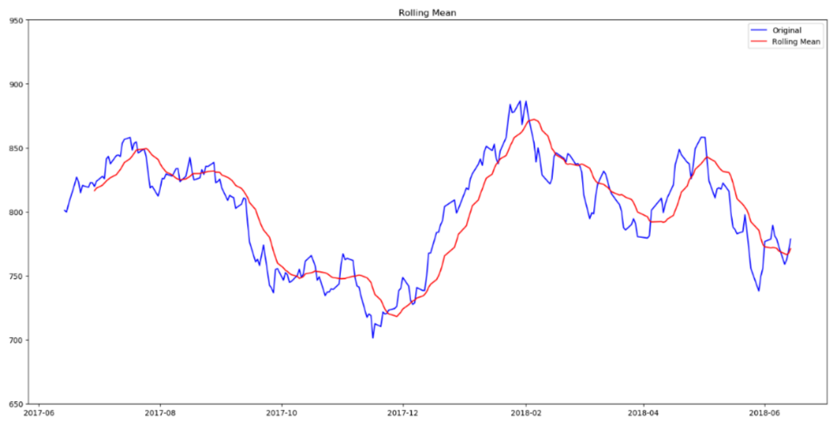
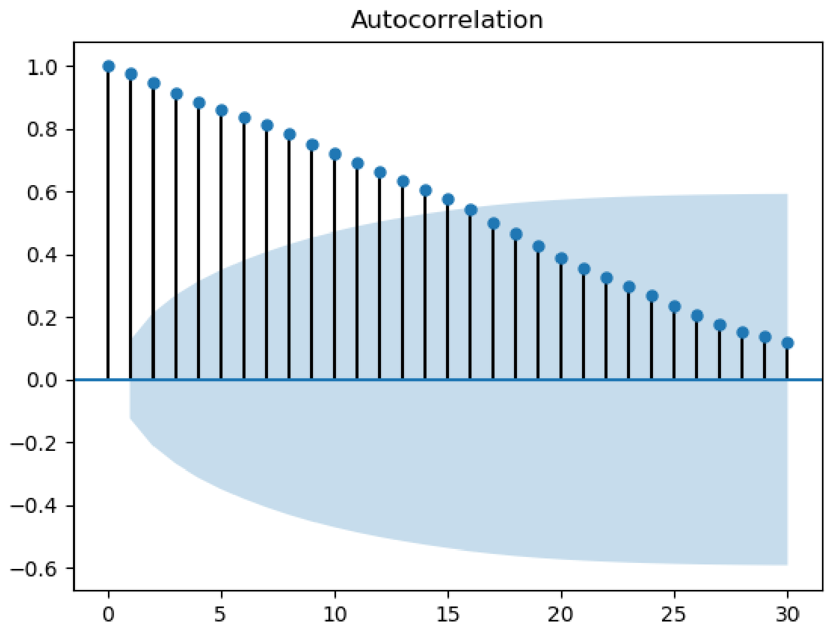


Figure 3.1 Rolling mean of raw data

Looking at the plotted data it is observed that they have a downward trend until November and then go up until February where it starts decreasing once again. In addition, seasonality can be present as well since prices are extremely volatile in specific periods during the year. In any case one should proceed to identification of (non) stationarity.

According to Figure 3.2 it is obvious that the ACF does not decay quickly to zero a sign of non - stationarity in time series since a constant mean and standard deviation over time is assumed.



.

Figure 3.2 ACF between 0-30 lags

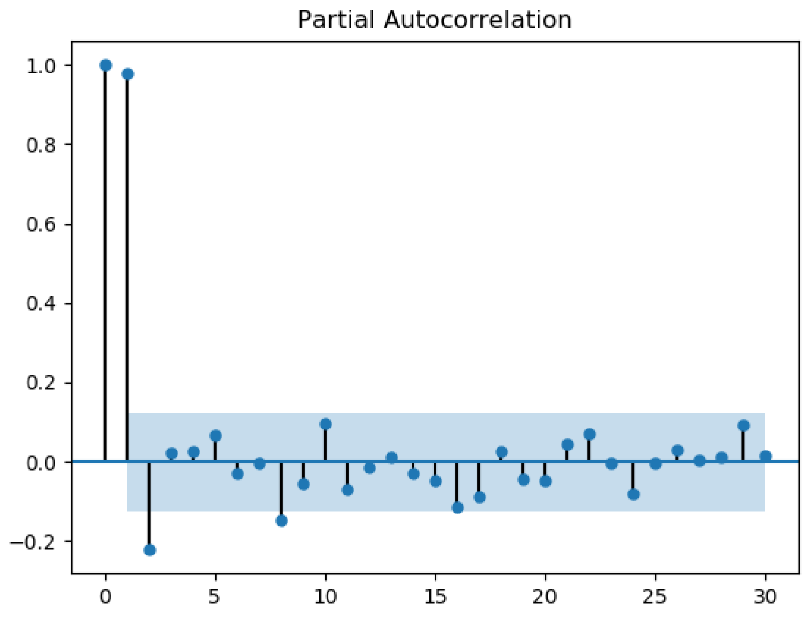
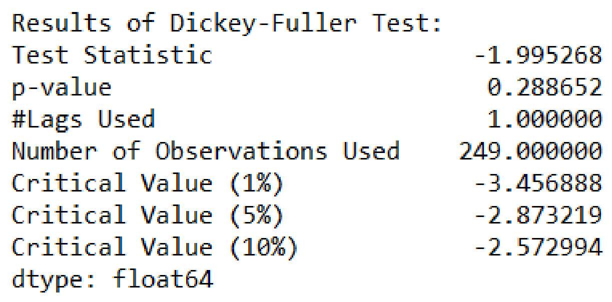


Figure 3.3 PACF between 0-30 lags

As explained before, the Dickey – Fuller test is a valuable indication for the stationarity of a time series. The test was also implemented in Python and uses the raw data and lags are set automatically to minimize AIC criterion. Results follow:



```
Results of Dickey-Fuller Test:
Test Statistic                   -1.995268
p-value                           0.288652
#Lags Used                        1.000000
Number of Observations Used     249.000000
Critical Value (1%)              -3.456888
Critical Value (5%)              -2.873219
Critical Value (10%)             -2.572994
dtype: float64
```

Figure 3.4 Dickey – Fuller test of raw data

The test results comprise of a **Test Statistic** and some **Critical Values** for difference confidence levels. If the 'Test Statistic' is less than the 'Critical Value', we can reject the null hypothesis and say that the series is stationary with corresponding confidence. Alternatively, one can consider the p-value on a 95% confidence. In this case, it is apparent that the p-value is larger than 0.05 a fact that forces us not to reject the null hypothesis of non - stationarity.

3.1.2 Model Identification

If a time series is found to be non – stationary the standard Box and Jenkins analysis includes differentiating the data by calculating and using their natural logarithm instead. Following that approach the same process as above has been conducted but this time for the differentiated data. Initially, the differentiated data are plotted and the result in Figure 3.5 suggests that the differentiated data have an overall constant trend and so does the rolling mean and standard deviation.

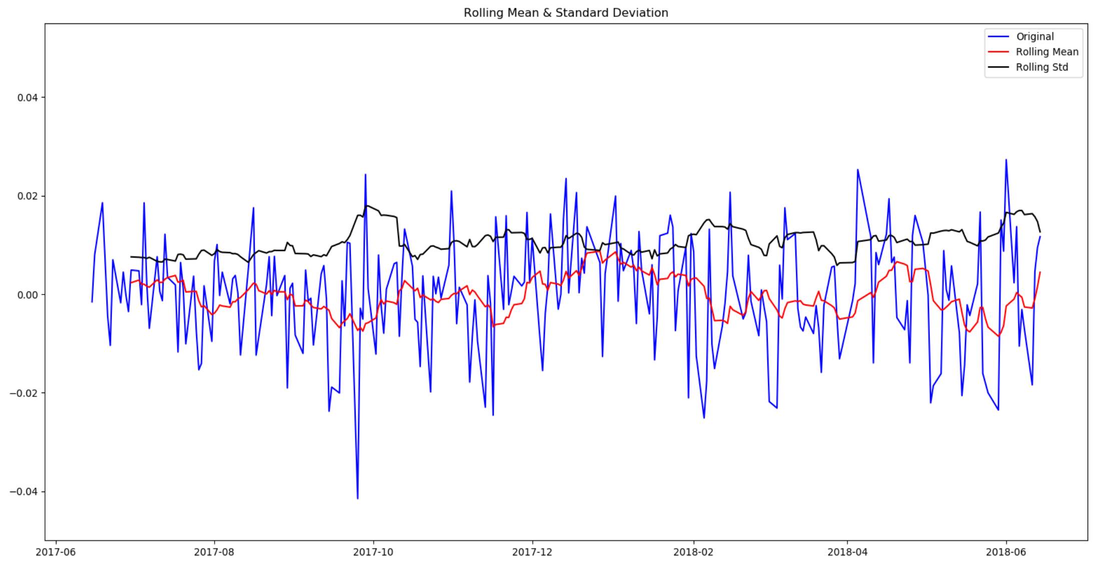


Figure 3.5 Rolling mean and standard deviation of transformed data (1st difference)

The analysis requires once more to plot the auto – correlation of first degree differentiated data and, as before, observe its movement (Appendix A, Figure A1). It can be concluded that the function decays quickly to zero. In addition, and according to the new Dickey – Fuller test results (see Appendix A, Figure A2), p-value < 0.05 and thus the null hypothesis of non – stationarity can be rejected with a 95% level of confidence.

All the results taken from the auto correlation functions and the Dickey – Fuller test are evidence that the forecasting process should be done with data differentiated once, since they show stationary properties and then conversion back to the original scale can take place.

After establishing stationarity on the data (d=1), the p and q parameters of ARIMA must be found. This can be done by observing the shape of ACF and PACF plots of the differentiated data.

In the next plot (Figure 3.6), the two dotted lines on either side of 0 are the confidence intervals. These can be used to determine the ‘p’ and ‘q’ values as:

1. **p** – The lag value where the **PACF** chart crosses the upper confidence interval for the first time. If someone notices closely, in this case p=1.

2. **q** – The lag value where the **ACF** chart crosses the upper confidence interval for the first time. If someone notices closely, in this case q=1.

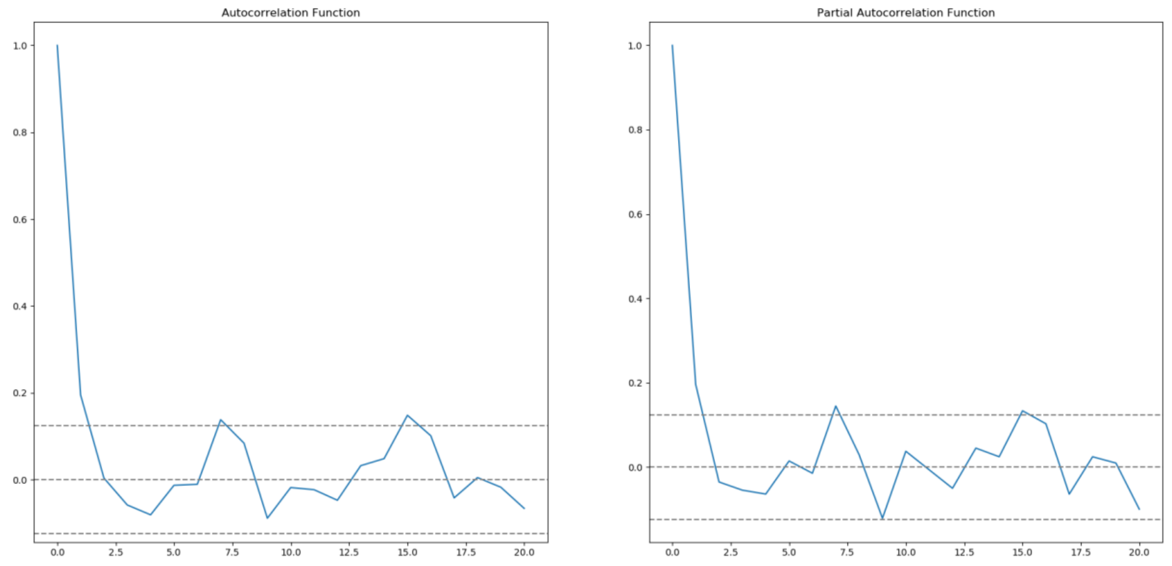


Figure 3.6 PACF and ACF Corellograms

3.1.3 Model Validation

After establishing the parameters of ARIMA it is required to check its relative performance before someone proceed to the actual forecasting. In order to do that, the dataset that contains the historical data was divided in two distinct smaller datasets. The first dataset was named “Train” and includes 67% percent of the original data while the second one ranges from 02/02/2018 – 06/14/2018 (33% of original data) is named “Test” and the forecasted values were calculated for the latter and compared with the expected prices. The results of the comparison for the aforementioned period can be visualized on figure 3.7 where it is apparent that the model has a good prediction accuracy comprising of a mean absolute error of approximately 7.567558.

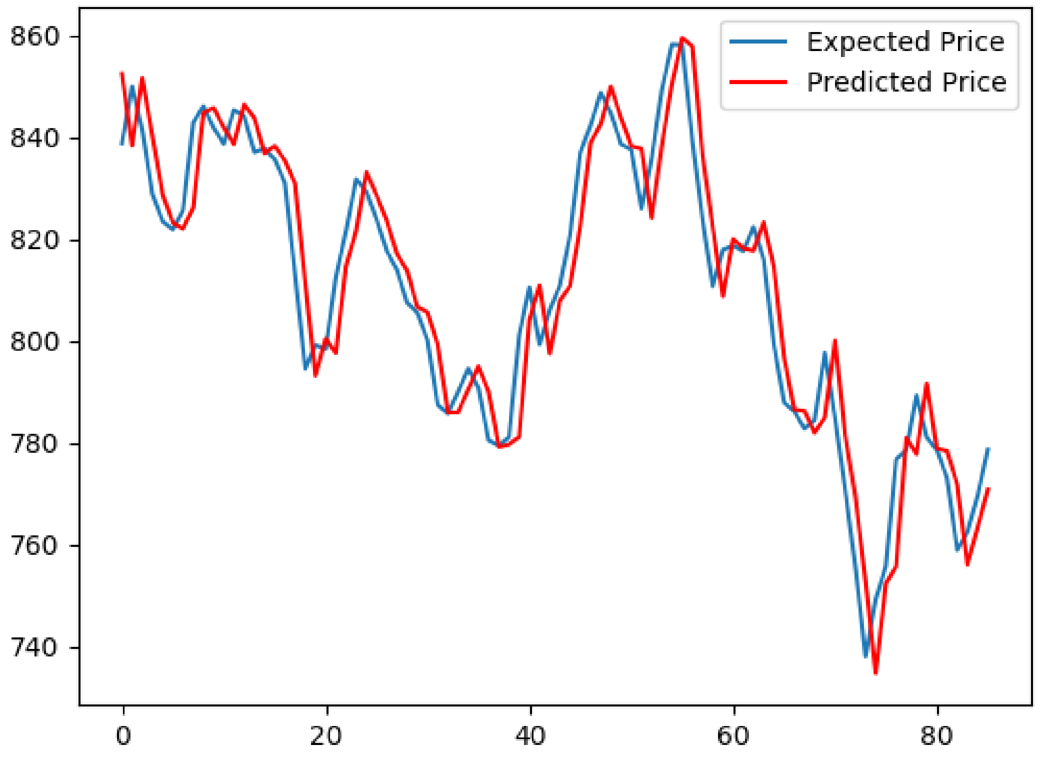


Figure 3.7 ARIMA (1,1,1) - Expected vs predicted prices

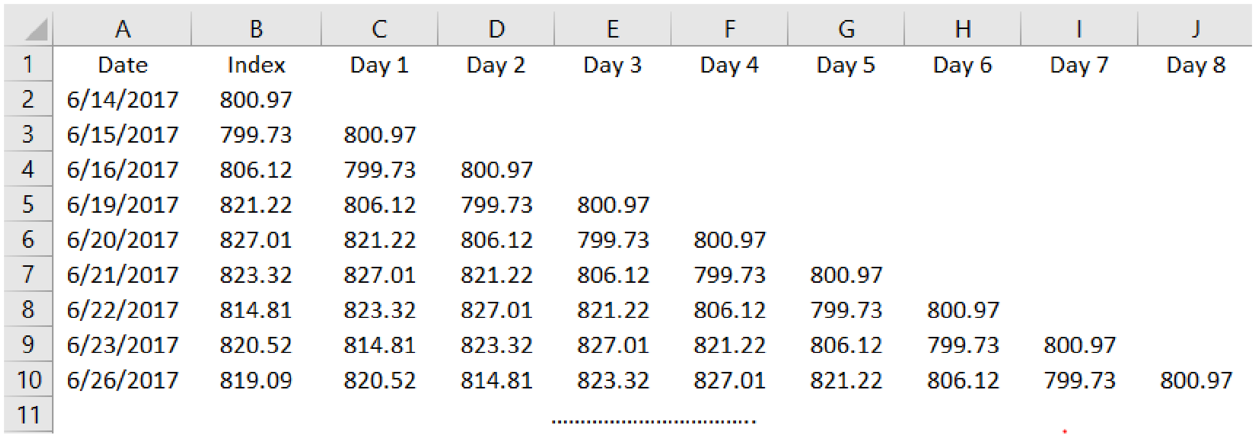
3.2 Implementation of ANN

One other model that has been widely used to forecast future values of a stock market index is the ANN in this research it has been implemented by using IBM SPSS Modeller. This particular platform was designed by IBM in order to help practitioners in the analysis of data and by doing so improve decision making and business performance. It is a tool that delivers complete and consistent information, easy to use, employs state of the art analysis methods and algorithm and its toolkit includes a variety of methods thus making it ideal for almost every kind of problem. It is used to build predictive models through its interface which also provides users the ability to leverage statistical and data mining algorithms without programming. Its main merit though lies in the simplicity that it offers in the analysis stage where it gets rid of unnecessary actions such as data transformations through automated processes. IBM SPSS

Modeller has been deployed in many organisations to boost their performance and its applications include a variety of industries such as telecommunications, retail, healthcare, banking (fraud detection), education and insurance (risk management).

3.2.1 Data Pre - Processing

As it was mentioned in the previous sections lagging terms usually affect a lot the value of forecast for future dates. Consequently, it is mandatory to experiment with varying number of inputs while constructing the model in order to get the optimal result. More specifically there should be identified the appropriate number of inputs that will fed as inputs in the ANN. These inputs apart from the indexing and target features include the lagging terms as well and this study uses two to eight lagging terms when constructing the model. The rationale behind the maximum number of eight lags is that sometimes news that break into the market on Monday can affect the incline or decline of the index for the whole week. Different datasheets have been constructed depending on the number of inputs before proceeding to the modelling process. The above process is more known as “windowed method” and was proposed by Refenes et al. (1992). What it practically does is removal of noise and identification of regularities in the dataset with the historical values. An example of a datasheet with ten inputs and eight lagging terms is presented in Figure 3.8.



	A	B	C	D	E	F	G	H	I	J
1	Date	Index	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8
2	6/14/2017	800.97								
3	6/15/2017	799.73	800.97							
4	6/16/2017	806.12	799.73	800.97						
5	6/19/2017	821.22	806.12	799.73	800.97					
6	6/20/2017	827.01	821.22	806.12	799.73	800.97				
7	6/21/2017	823.32	827.01	821.22	806.12	799.73	800.97			
8	6/22/2017	814.81	823.32	827.01	821.22	806.12	799.73	800.97		
9	6/23/2017	820.52	814.81	823.32	827.01	821.22	806.12	799.73	800.97	
10	6/26/2017	819.09	820.52	814.81	823.32	827.01	821.22	806.12	799.73	800.97
11					....					

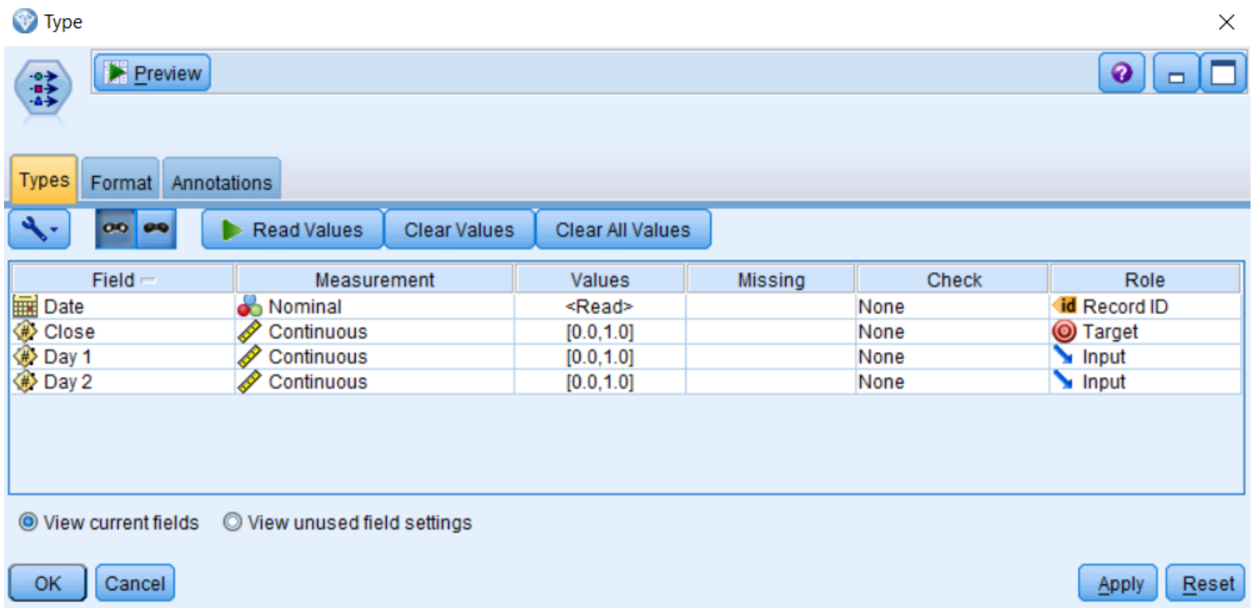
Figure 3.8 Eight lagged input of the model

The previous figure shows the data table constructed when considering eight lag terms and a total of ten variables in the model. The first column represents the time period t while the second are the actual values of the stock market index for the corresponding dates. The columns labelled Day “X” are the lag terms and have the same value the index had “X” days ago. For example, at 6/26/2017, the 1st lag has the value the index had the day before, the 2nd lag has the value the index had two day ago etc. This pattern continuous for the whole range of the time period studied and finally all of the input data have been normalised before inserted into the model in order to reduce training time and improve performance (Principe et al., (1999) by having data scaled in comparable range and by doing so avoiding getting stuck in a local optima. In summary there have been constructed and modelled seven different worksheets in order to establish the most appropriate combination of inputs.

3.2.2 File Import

The first thing of the model preparation is to import the data that are going to be used. IBM SPSS Modeller has several tabs near the bottom of the screen with all its modelling tools. By choosing the “Sources” tab there are displayed several nodes that are responsible for the information loading into the system. Since the pre - processed data are in “xlsx” format the “Excel” node is chosen and dragged in the stream (Appendix B, Figure B1).

The next stage of the model building includes specification of the type and role of each variable of the data. The node responsible for this task is called “Type” and is located in the tab “Field Ops”. Depending on the input variables someone wishes to use in the model, the setting of their types should be set. In Figure 3.9 is shown a capture of the modeller window where four variables have been used. The first variable labelled as “record ID” includes the dates from 06/14/207 to 06/14/2018 and is used for indexing purposes, the second variable is the target or in other words is the dependent variable of the forecasting process and finally the rest of the variables are classified as “Input” and represent the “Day 1” and “Day 2” features of the dataset (independent variables). In this step the type of the variables is also selected depending if they are nominal, continuous, ordinal or categorical. Finally, the source of data is connected to the new node via a straight line.



Field	Measurement	Values	Missing	Check	Role
Date	Nominal	<Read>		None	Record ID
Close	Continuous	[0.0,1.0]		None	Target
Day 1	Continuous	[0.0,1.0]		None	Input
Day 2	Continuous	[0.0,1.0]		None	Input

View current fields [✓] View unused field settings

Figure 3.9 Specification of the type and role of variables

3.2.3 Data Partitioning

The common approach for data portioning when it comes to model building is to divide them in three distinct datasets:

1. **Training data**

   Training data are used to capture underlying patterns in past values of the time series. It is the dataset used to "teach" the neural network of how it should operate by using the input/output pairs. It is usually the dataset with the most observations.

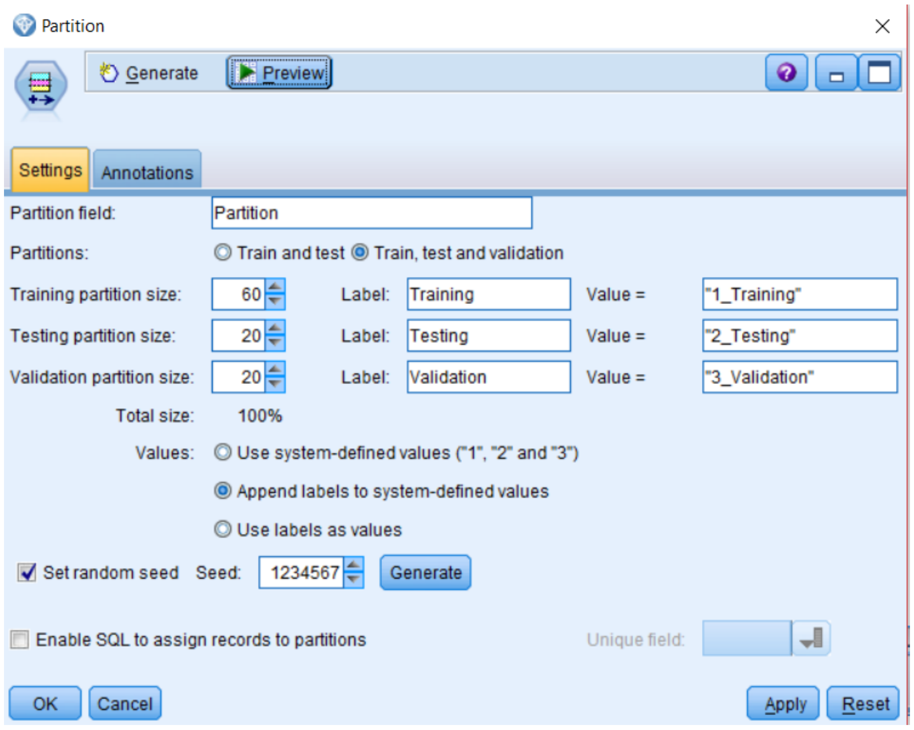
2. **Validation data**

   Set of examples used to tune the parameters of the classifier used and in cases such as the Multilayer Perceptron (MLP) provides estimation of the number of hidden layers needed and the stopping point of the algorithm in order to avoid overfitting.

3. **Testing data**

This part of the data is used to estimate how well the final model can predict. The system predicts values for every day in the dataset and then compares it with the actual value in order to derive an accuracy metric and an estimation of the model performance.

The “Partition” node is inserted into the stream and connected with the “Type” node. Editing process of the node is required in order to define that three datasets. By right clicking the node and choosing edit the properties will show up. Figure 3.10 shows the settings tab where the training dataset field is assigned to accumulate for 60% of the initial dataset while validation and testing accumulate for 20% respectively.



Partition field: Partition

Partitions: ○ Train and test [✓ Train, test and validation]

Training partition size:	60	Label:	Training	Value =	"1_Training"
Testing partition size:	20	Label:	Testing	Value =	"2_Testing"
Validation partition size:	20	Label:	Validation	Value =	"3_Validation"

Total size: 100%

Values: ○ Use system-defined values ("1", "2" and "3")
[✓ Append labels to system-defined values]
○ Use labels as values

[✓] Set random seed Seed: 1234567 Generate

☐ Enable SQL to assign records to partitions Unique field:

Figure 3.10 Setting of the Partition Node

3.2.4 Construction of the Artificial Neural Network

IBM SPSS Modeller includes various modelling techniques and algorithms that can be used for forecasting purposes. All of these techniques are located in the tab labelled “Modelling”. In this case the node “Neural Net” is picked and added into the current stream where it is connected with the partition node sequentially. In order to specify the properties of the new node, it is right clicked and the edit option is chosen. In the new window, under the tab “fields” index is set as the target variable and the lag terms used as the predictors. In the next option the network architecture is defined and IBM SPSS Modeller provides with two of the most largely used, Multilayer Perceptron (MLP) and Radial Basis Function (RBF). Both of them have been monitored in this study in order to identify the best performing and the comparison between them is presented in the next section. When it comes to the MLP there are two ways to define the number of the hidden layers. Literature review in previous chapters revealed that for MLP the majority of problems require either one or two hidden layers and that a bigger number of layers can very often lead to complex models without significant increase in performance and even overfitting results. Similarly, the software provides the user with the power to choose between one or two hidden layers but also allows an automated estimation of the most appropriate number of the hidden layers by computing itself all the possibilities before choosing the best. Therefore, the hidden units in the model developed here is also set to be chosen automatically for both MLP and RBF.

According to neural network theory a stopping rule is mandatory in order to acquire a well performing model that does not have overfitting properties. Training time is changed to half an hour per component model and it is set to run. Figure 3.11 shows the model constructed for the two lag terms used as inputs and similarly are developed the models for all of the rest lagging terms. Finally, two nodes, “Analysis” and “Table” are used to record the output of the model after the execution and provide performance and statistical measures.

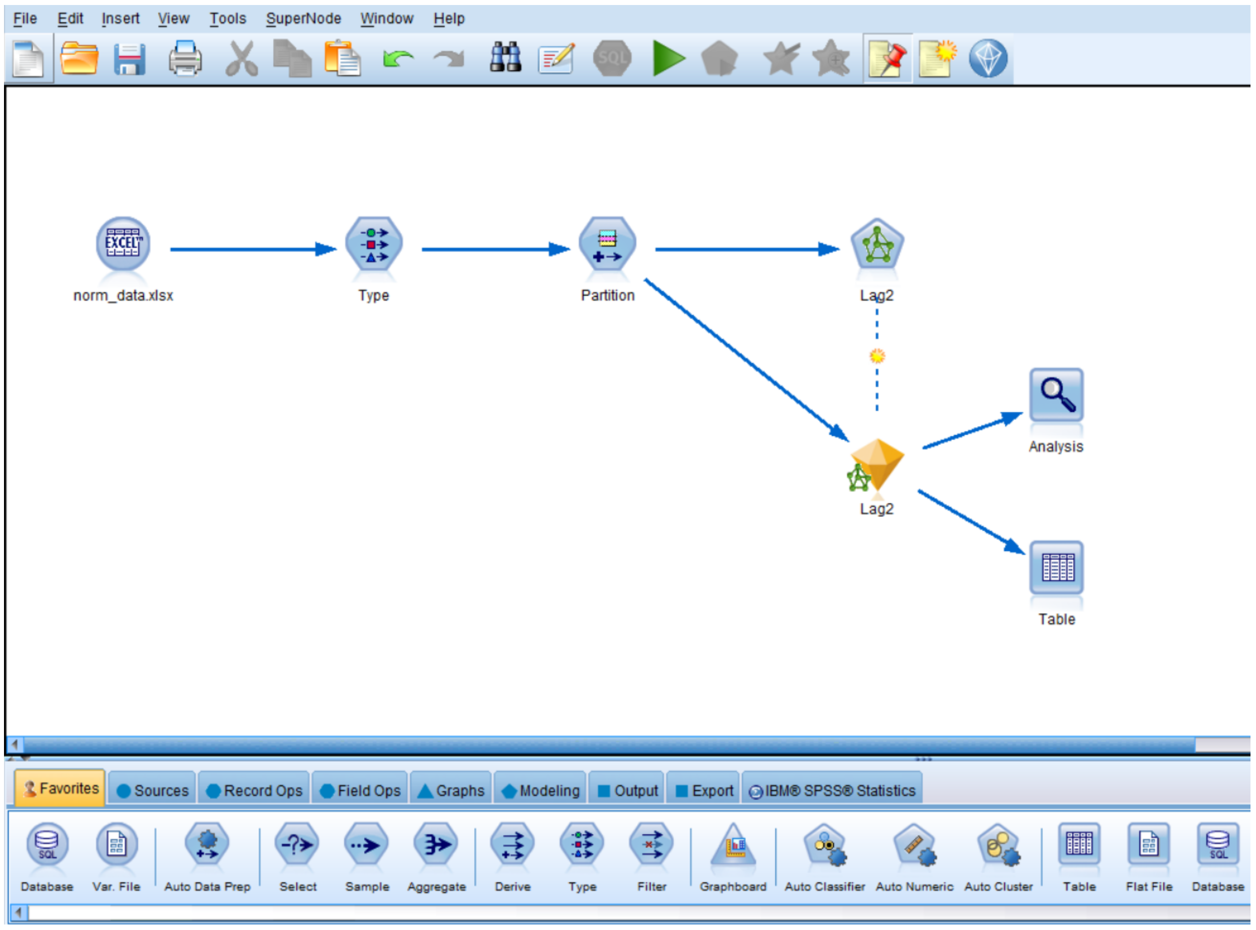


Figure 3.11 Diagrammatic representation of the ANN

3.2.5 ANN Results

At this section the results from the neural network are presented. These results come from two different architectures, Multilayer Perceptron and Radial Basis Function. For each one has been used varying inputs (lagged terms) and the outputs have been gathered in order to estimate the best performing model. In the two tables below, the first column describes all the models tested which are noted as [x, y, z] where the first parameter describes the number of inputs, the second the number of hidden neurons and the third the number of hidden layers. For instance, when a model is referred as [7,4,1] it is meant that it has been constructed with seven inputs, four hidden neurons and one hidden layer. SPSS Modeller provides with a variety of results such as statistics, topography of the models computed and an index for the network quality. However, the Mean Absolute Error (MAE) has been preferred as criterion for choosing the best prediction model to conduct the forecast.

Table 3.1 shows the performance of the models constructed with their corresponding MAE after implementing the MLP algorithm. The model with the least mean absolute error is the last one which uses all of the lag terms in a network with one hidden layer and seven hidden neurons hence it is chosen as the best of all the eight when simulating with multilayer perceptron.

Table 3.1 MAE and Quality of Neural Network with MLP

Model	Mean Absolute Error (MAE)	Neural Network Quality	Next Day Forecast (Normalised)
[2,1,1]	0.041	96.4%	0.421
[3,1,1]	0.043	96.3%	0.413
[4,2,1]	0.041	96%	0.418
[5,2,1]	0.041	96.2%	0.418
[6,6,1]	0.042	96.4%	0.427
[7,4,1]	0.044	94.5	0.412
[8,7,1]	0.038	95.8%	0.413

Similarly, the results of RBF have been produced and gathered for all the different inputs. Table 3 depicts the output of all the eight models in an identical structure and rationale as before. In this case, the model denoted as [2,10,1] which uses two lagged terms and 10 hidden neurons has a MAE of 0.053 thus it manages to outperform the rest seven.

Table 3.2 MAE and Quality of Neural Network with RBF

Model	Mean Absolute Error (MAE)	Neural Network Quality	Next Day Forecast (Normalised)
[2,10,1]	0.053	94.3%	0.325
[3,10,1]	0.059	92.9%	0.290
[4,10,1]	0.061	90.5%	0.288
[5,10,1]	0.063	91.5%	0.306
[6,10,1]	0.066	90.5%	0.323
[7,10,1]	0.073	88.8%	0.320
[8,10,1]	0.063	88.7%	0.373

Finally, in order to identify which of the two, Multilayer Perceptron or Radial Basis Function, will lead to better prediction of the Greek stock market index a comparison between them is required. Therefore, model [8,7,1] of the MLP and model [2,10,1] of RBF are considered. It can be observed that the first has a significantly better accuracy than the latter hence the most suitable for the forecasting problem in hand.

3.3 Implementation of Support Vector Machine

Another method useful for forecasting purposes is the Support Vector Machine, a powerful classifier used in many applications. Due to the nature of the problem studied here and in order to effectively compare it with the rest of the models developed earlier, a special case of SVM has been used more known as Support Vector Regression which was deployed using Python. Support Vector Machines are very specific class of algorithms, characterized by usage of kernels, absence of local minima, sparseness of the solution and capacity control obtained by acting on the margin, or on number of support vectors.

Initially the dataset has been imported and split in a train and test set according to the basic process of model evaluation. Consequently, the data have been normalised in order to allow the method to learn from data with a common scale, hence improve the performance of the outcome.

Support Vector Machine and in extent SVR can adopt different kernel functions in order to identify (non) linearities in the data and produce a model appropriate for forecasting. Earlier it was further analysed that it seeks to divide the space that the data points lie with as large margin as possible. The most often used kernels referred to earlier are: 1) Radial Basis Function, 2) Linear and 3) Polynomial all of whom have been tested in this research.

As mentioned in the previous chapter, in SVM's the parameter estimation for the different kernels used is a crucial step towards the right model in order to obtain more accurate classification of estimation of the value in the case of SVR. All of the kernels require identification of a cost parameter but especially for the case of the Radial Basis kernel the γ parameter needs to be acquired as well. In this research and for the purpose in hand the grid search and cross validation method has been used, a method suggested and accepted by many researchers due to its capability to identify "good" values. The cross validation included ten folds and the range of the grid was set from 2^{-5} to 2^{-15} for cost and 2^{-15} until 2^{-3} for the γ parameter (Hsu et al., 2003). The full code in python for the SVR deployment can be found in Appendix C.

In summary the optimal values found for all four kernels are:

Table 3.3 Optimal Parameters for the SVR

Inputs	RBF	Polynomial	Linear
2	C: 256	C: 256	C: 64

	γ: 0.0625		
3	C: 16384 γ: 0.0009765625	C: 16	C: 128
4	C: 16384 γ: 0.0009765625	C: 32	C: 256
5	C: 32768 γ: 0.00048828125	C: 32	C: 16384
6	C: 16384 γ: 0.00048828125	C: 64	C: 16
7	C: 32768 γ: 0.000244140625	C: 8	C: 32768
8	C:8192 γ: 0.00048828125	C: 8	C: 8

3.3.1 SVR Results

At this section the results acquired from the Support Vector Regression model are presented. In order to recognise the best performing approach three different kernel types have deployed to the SVM and applied with varying inputs and the parameters estimated from grid search. Table 5 summarises the results provided by the simulation process and includes five columns of which the first specifies the inputs used and the rest the kernel type associated with the result mentioned for the corresponding number of lagged terms. As with neural networks before, the mean absolute error (MAE) criterion has been chosen to classify the classes used.

Table 3.4 Mean Absolute Error (MAE) of different kernels and for various number of inputs

Mean Absolute Error (MAE)							
Inputs	2	3	4	5	6	7	8
RBF	8.725404	8.700085	8.768627	8.619631	8.824482	9.243448	9.086053
Linear	8.63404	8.71056	8.789534	8.536055	8.958269	9.219026	9.001292
Polynomial	12.53714	13.06565	13.05174	12.68262	12.47111	13.50146	13.81535

According to Table 3.4 the linear kernel with five inputs used resulted in the smallest MAE hence it is preferred from the rest as the most accurate method to forecast the stock market index with a mean absolute error criterion of 8.536055.

3.4 Comparison of ARIMA, ANN and SVM

In this chapter, three different techniques have been used in order to solve a very complex problem that has been attracting research and market interest for many years, prediction of stock market movement. The aforementioned methods have been ARIMA, ANN and SVM, and the rationale behind that is that these particular systems have repeatedly been classified as some of the best performers. Throughout the analysis of each method separately, different functions were tested and calibration of their respective parameters was also performed in order to optimise the results. After obtaining the results each method had a clear representative that seemed to perform better than the rest. More specifically these representatives were:

- ARIMA (1,1,1) with a MAE of 7.567558

- ANN with MLP architecture, eight inputs and a mean absolute error of 6.98052
- SVM with a linear kernel of five inputs and a mean absolute error of 8.536055

Undoubtedly the [8,7,1] ANN succeeded in performing the most accurate results of all the methods attempted but it has to be mentioned at this point that many more managed to predict with relatively high accuracy as well.

3.5 Forecasting with MLP Neural Network

After establishing the most accurate way for the prediction of the index, the actual forecasting process takes place. Using the [8,7,1] neural network with Multi – layer perceptron architecture and a hyperbolic tangent function for the hidden layers the value of the index for the next day out of sample was predicted. In order to do the prediction a new dataset must be imported into the current stream already constructed but the difference this time is that there is an extra record representing the "06/15/2018" day with its respective predictors but without target. As before the variables types must be defined by onnecting the new data source with the "Type" node. Finally, both are given as input to the model acquired through after the training face which is expected to have "learned" sufficiently enough from the historical data.

The final forecasting result of the model for "06/15/2018" was found to be 777.8393 and the full process can be found in Appendix B.

Chapter 4. Conclusions, Limitations and Recommendations

4.1 Conclusions

It is common knowledge that investments and banking overall have always been an area where forecasting is a crucial operation in the value acquisition game. After the financial crisis of 2008 that hit and still affects many countries in the globe, institutions and public have been very cautious and reluctant when it comes to their financials. As a result, the above triggered an explosion of interest regarding the future profitability of stock movement and the associated with them risk in order to minimize their loses and at the same time increase their profits. Since then many research projects from both academics and market analysts have seen the light of day who evaluated different ways of forecasting methods in the hope that there is a universal optimal performer for all types of problems.

The ARIMA model has been for many years the center of interest when it comes to forecasting research. It is a model known for its ability to identify relationships between historical data with linear properties. Another method that emerged later than ARIMA and attracted a lot of attention, especially recently with the progress of Deep Learning", is Artificial Neural Networks (ANNs). Famous for their non – linear capabilities, ANNs are considered as a mean to address a variety of complex problems and it has been observed that they perform relatively well in many of them. Finally, a very "young" modelling approach has been the Support Vector Machines, who can be used for either classifying or regression purposes and they are considered from many as the best method of all.

In this paper it was attempted to forecast the movement of the Greek stock market index for 06/15/2018, by using ARIMA, ANN, SVM and historical closing prices data ranging from 06/14/2017 to 06/14/2018. For the Artificial Neural Network and Support Vector Machine different architectures have been used in order to be more thorough and effective in mapping the data in hand. More specifically, ANN employed Multi – Layer Perceptron and Radial Basis Function while in SVM different kernels such as linear, radial basis function and Polynomial were used. For all three methods used the Mean Absolute Error metric was calculated in order to identify the best performer. The results show that MLP outperformed RBF in the context of neural network architecture showing a lower MAE and on the other hand linear was the winning kernel from those evaluated with SVM. Comparing MLP with an SVM with linear kernel and ARIMA it is suggested that MLP, with eight inputs, one hidden layer and seven hidden neurons, is the most accurate in the experiment attempted perhaps due to its good ability for non – linear pattern identification. Despite MLP's good performance, it is observed that all processing methods have error metrics that do not deviate much one from another meaning that all of them can be considered for the problem studied here or other similar objectives.

4.2 Limitations & Recommendations

Almost in every forecasting attempt there are limitations, usually unforeseen before the results are actually obtained, that affect the process and its conclusions. One of the majors in this work is that the size of the data used to train the models are not the same. In the case of ARIMA, the train dataset consisted of 67% of the whole while the ANN used 60% and the SVM 80%. Generally speaking, the more data are used to train a model the better forecasting results this model is going to provide. In this case the ANN received the least training of all three but still managed to outperform them allowing us to infer that its superiority lies in more facts than just the Mean Absolute Error. Regardless the result the performance of a model is also a subject of the characteristics of the data used in the experiment. That been said if all these methods are applied with more data or with a completely different dataset the final ranking may be different highlighting once again that there is no best technique than experimentation with various approaches.

Another important thing that this study did not take into consideration is the nature of the stock market. As explained in previous chapters, stock markets are very complicated and volatile something that makes the feature selection itself extremely difficult. Many will argue that important predictors in predicting a stock market's index is the stock price of the biggest companies in this market whether other support that opening price, volume and closing price or even the mood of general public is sufficient but the truth is that there is no straightforward approach but it is on the hand of the researcher to decide.

To expand this study the following are proposed:

- Experimentation with more time series models and comparison with machine learning techniques could lead to interesting results.
- Hybrid approaches, e.g. ARIMA with ANN, can result in improved outcomes since they combine the linear properties of the first with the strong non – linear identification capability of the second thus making this new model a good candidate for a variety of problems.

- More features can be included in the experiment in an attempt to assist the forecasting models capture more relevant patterns in time series and hence improve the overall performance. In the scope it is worth exploring the influence of news that break out and affect its movement in real time as well.
- Bigger datasets can be used in order to improve training phase and the resulting fitting of the methods.
- Application to different historical datasets and comparison with the current study could also reveal interesting insights on the relative performance of the models.

References

Abu-Mostafa, Y., Atiya, A., Magdon-Ismail, M. and White, H. (2001). Introduction to the special issue on neural networks in financial engineering. *IEEE Transactions on Neural Networks*, 12(4), pp.653-656.

Adebiyi, A., Adewumi, A. and Ayo, C. (2014). Comparison of ARIMA and Artificial Neural Networks Models for Stock Price Prediction. *Journal of Applied Mathematics*, 2014, pp.1-7.

Basak, D., Pal, S. and Patranabis, D. (2007). Neural Information Processing-Letters and Reviews. pp.203-224.

Benardos, P. and Vosniakos, G. (2007). Optimizing feedforward artificial neural network architecture. *Engineering Applications of Artificial Intelligence*, 20(3), pp.365-382.

Borah, T., Sarma, K. and Talukdar, P. (2016). *Handbook of Research on Emerging Perspectives in Intelligent Pattern Recognition, Analysis, and Image Processing*. IGI Global, pp.335-366.

Box, G. and Jenkins, G. (1976). *Time Series Analysis: Forecasting and Control*. San Francisco: CA: Holden-Day.

Box, G., Jenkins, G., Reinsel, G. and G. M., L. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Chen, J. (2004). M-estimator based robust kernels for support vector machines. In: *Proceedings of the Seventeenth International Conference on Pattern Recognition (ICPR 2004)*. Cambridge, UK, pp.168–171.

Chen, W., Hsu, S. and Shen, H. (2005). Application of SVM and ANN for intrusion detection. *Computers & Operations Research*, 32(10), pp.2617-2634.

Chen, A., Hsu, Y. and Hu, K. (2018). A hybrid forecasting model for foreign exchange rate based on a multi-neural network. In: *2008 Fourth International Conference on Natural Computation*. IEEE.

Da Silva, I., Hernane Spatti, D., Andrade Flauzino, R., Liboni, L. and Dos Reis Alves, S. (2017). *Artificial Neural Networks*. Cham: Springer.

Dacheng Tao, Xiaoou Tang, Xuelong Li and Xindong Wu (2006). Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7), pp.1088-1099.

De Villiers, J. and Barnard, E. (1993). Backpropagation neural nets with one and two hidden layers. *IEEE Transactions on Neural Networks*, 4(1), pp.136-141.

Deshmukh, S., Senthilnath, J., Dixit, R., Malik, S., Pandey, R., Vaidya, A., Omkar, S. and Mudliar, S. (2012). Comparison of Radial Basis Function Neural Network and Response Surface

Methodology for Predicting Performance of Biofilter Treating Toluene. *Journal of Software Engineering and Applications*, 05(08), pp.595-603.

Dickey, D. and Fuller, W. (1979). Distribution of the Estimators for Autoregressive Time Series With a Unit Root. *Journal of the American Statistical Association*, 74(366), p.427.

Dickey, D. and Pantula, S. (1987). Determining the Order of Differencing in Autoregressive Processes. *Journal of Business & Economic Statistics*, 5(4), pp.455-461.

DING, Y., SONG, X. and ZEN, Y. (2008). Forecasting financial condition of Chinese listed companies based on support vector machine. *Expert Systems with Applications*, 34(4), pp.3081-3089.

Dunis, C. and Feeny, M. (1989). *Exchange rate forecasting*. New York [etc]: Woodhead-Faulkner.

Fama, E. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, 25(2), p.383.

Framework, N., Du, K., Swamy, M. and London, S. (2018). *Neural Networks in a Softcomputing Framework | Ke-Lin Du | Springer*. [online] Springer.com. Available at: https://www.springer.com/us/book/9781846283024 [Accessed 25 Aug. 2018].

Fyfe, C. (2000). *Artificial neural networks and information theory*. University of Paisley.

Graupe, D. (2013). *Principles of Artificial Neural Networks*. Singapore: World Scientific Publishing Company.

Greig, A. (1992). Fundamental analysis and subsequent stock returns. *Journal of Accounting and Economics*, 15(2-3), pp.413-442.

Hansen, J., McDonald, J. and Nelson, R. (1999). Time Series Prediction With Genetic-Algorithm Designed Neural Networks: An Empirical Comparison With Modern Statistical Models. *Computational Intelligence*, 15(3), pp.171-184.

He, Z., Wen, X., Liu, H. and Du, J. (2014). A comparative study of artificial neural network, adaptive neuro fuzzy inference system and support vector machine for forecasting river flow in the semiarid mountain region. *Journal of Hydrology*, 509, pp.379-386.

Heaton, J. (2008). *Introduction to Neural Networks for Java, Second Edition*. St. Louis, Mo.: Heaton research.

Hsu, C., Chang, C. and Lin, C. (2018). *A Practical Guide to Support Vector Classification*. [online] Semanticscholar.org. Available at: https://www.semanticscholar.org/paper/A-Practical-Guide-to-Support-Vector-Classification-Hsu-Chang/05393361e6d9e56ee7dbabb1e5ef6c1c212fc34d [Accessed 25 Aug. 2018].

Jain, A., Jianchang Mao and Mohiuddin, K. (1996). Artificial neural networks: a tutorial. *Computer*, 29(3), pp.31-44.

Kara, Y., Acar Boyacioglu, M. and Baykan, Ö. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems with Applications*, 38(5), pp.5311-5319.

Kim, K. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2), pp.307-319.

Lee, C., Sehwan, Y. and Jongdae, J. (2007). Neural network model versus SARIMA model in forecasting Korean stock price index (KOSPI). *Issues in Information System*, 8(2), pp.372–378.

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4), pp.115-133.

Mehrotra, K., Mohan, C. and Ranka, S. (1997). *Elements of artificial neural networks*. Cambridge, Mass.: MIT Press.

Minsky, M. and Papert, S. (1969). *Perceptron Expanded Edition*.

Minsky, M. and Papert, S. (1969). *Perceptrons*. MIT Press, Cambridge, MA.

Montazer, G., Giveki, D., Karami, M. and Rastegar, H. (2018). *Radial Basis Function Neural Networks : A Review*. [online] Purkh.com. Available at: https://www.purkh.com/index.php/tocomp/article/view/36 [Accessed 25 Aug. 2018].

Murata, N., Yoshizawa, S. and Amari, S. (1994). Network information criterion-determining the number of hidden units for an artificial neural network model. *IEEE Transactions on Neural Networks*, 5(6), pp.865-872.

Networks, A., Karayiannis, N., Venetsanopoulos, A. and US, S. (2018). *Artificial Neural Networks - Learning Algorithms, Performance Evaluation, and Applications | Nicolaos Karayiannis | Springer*. [online] Springer.com. Available at: https://www.springer.com/us/book/9780792392972 [Accessed 25 Aug. 2018].

Networks, N., Dreyfus, G. and Heidelberg, S. (2018). *Neural Networks - Methodology and Applications | Gérard Dreyfus | Springer*. [online] Springer.com. Available at: https://www.springer.com/us/book/9783540229803 [Accessed 25 Aug. 2018].

Parrella, F. (2007). *Online support vector regression*. Master. Department of Information Science, University of Genoa, Italy.

Príncipe, J., Euliano, N. and Lefebvre, W. (2000). *Neural and adaptive systems*. New York: Wiley.

Proceedings of the fifth annual ACM workshop on computational learning theory. (1992). Baltimore, MD: ACM Press.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), pp.386-408.

Schalkoff, R. (1997). *Artificial neural networks*. 1st ed. New York: McGraw-Hill.

Smith, K. and Gupta, J. (2000). Neural networks in business: techniques and applications for the operations researcher. *Computers & Operations Research*, 27(11-12), pp.1023-1044.

Startz, R. (2006). Binomial Autoregressive Moving Average Models with an Application to U.S. Recessions. *SSRN Electronic Journal*.

Sterba, J. and Hilovska, K. (2010). The implementation of hybrid ARIMA neural network prediction model for aggregate water consumption prediction. *Aplimat—Journal of Applied Mathematics*, 3(3), pp.123–131.

Tansel, N., Yang, S., Venkataraman, G., Sasirathsiri, A., Bao, W. and Mahendrakar, N. (1999). *Modelling Time Series data by using Neural Networks and Genetic Algorithms*. pp.1055-1060.

Theory, T., Vapnik, V. and York, S. (2018). *The Nature of Statistical Learning Theory | Vladimir Vapnik | Springer*. [online] Springer.com. Available at: https://www.springer.com/us/book/9780387987804 [Accessed 25 Aug. 2018].

Trenn, S. (2008). Multilayer Perceptrons: Approximation Order and Necessary Number of Hidden Units. *IEEE Transactions on Neural Networks*, 19(5), pp.836-844.

Tsay, R. (2005). *Analysis of financial time series*. John Wiley & Sons.

Turian, J., Bergstra, J. and Bengio, Y. (2009). Quadratic features and deep architectures for chunking. In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Boulder, Colorado, pp.245-248.

Wang, S., Chaovalitwongse, W. and Babuska, R. (2012). Machine Learning Algorithms in Bipedal Robot Control. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(5), pp.728-743.

Weston, J. and Watkins, C. (1998). *Support Vector Machine with more than two groups*. [online] Available at: https://www.researchgate.net/post/Support_vector_machine_with_more_than_two_groups [Accessed 25 Aug. 2018].

Widrow, B., Rumelhart, D. and Lehr, M. (1994). Neural networks: applications in industry, business and science. *Communications of the ACM*, 37(3), pp.93-105.

Wijaya, Y. (2010). Stock Price Prediction: Comparison of Arima and Artificial Neural Network Methods - An Indonesia Stock's Case. In: *2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies*. pp.176-179.

Yao, J. and Poh, H. (1996). Equity Forecasting: A Case Study on the KLSE Index. Singapore: Proceedings of the Third International Conference on Neural Networks in the Capital Markets, pp.341–353.

Zaiyong Tang, de Almeida, C. and Fishwick, P. (1991). Time series forecasting using neural networks vs. Box- Jenkins methodology. *SIMULATION*, 57(5), pp.303-310.

Zaremba, W., Sutskever, I. and Vinyals, O. (2014). Recurrent neural network regularization.

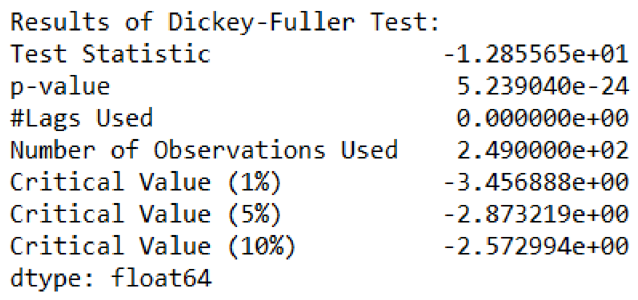
Zhang, D., Jiang, Q. and Li, X. (2004). Application of Neural Networks in Financial Data Mining. *International Journal of Computational Intelligence*, 1(2), pp.116-119.

Zhang, X. (2018). *Introduction to Artificial Neural Network*.

Appendices

Appendix A

Results from Stationarity Analysis for Time Series of 1st Degree Differenced Data



```
Results of Dickey-Fuller Test:
Test Statistic                -1.285565e+01
p-value                        5.239040e-24
#Lags Used                     0.000000e+00
Number of Observations Used    2.490000e+02
Critical Value (1%)           -3.456888e+00
Critical Value (5%)           -2.873219e+00
Critical Value (10%)          -2.572994e+00
dtype: float64
```

Figure A1 Dickey – Fuller Test of 1st Degree Differenced Data

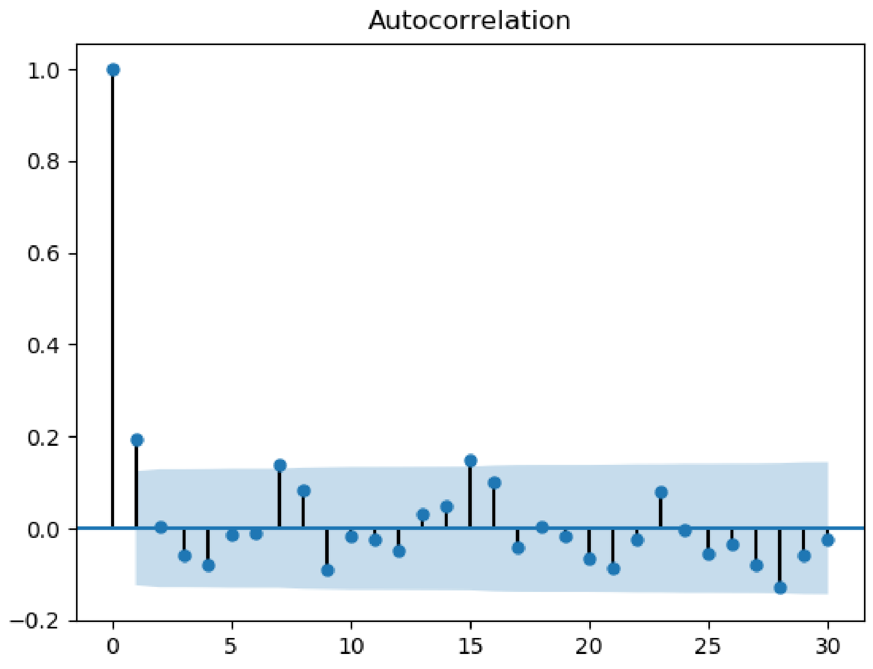


Figure A2 ACF of 1st Degree Differenced Data

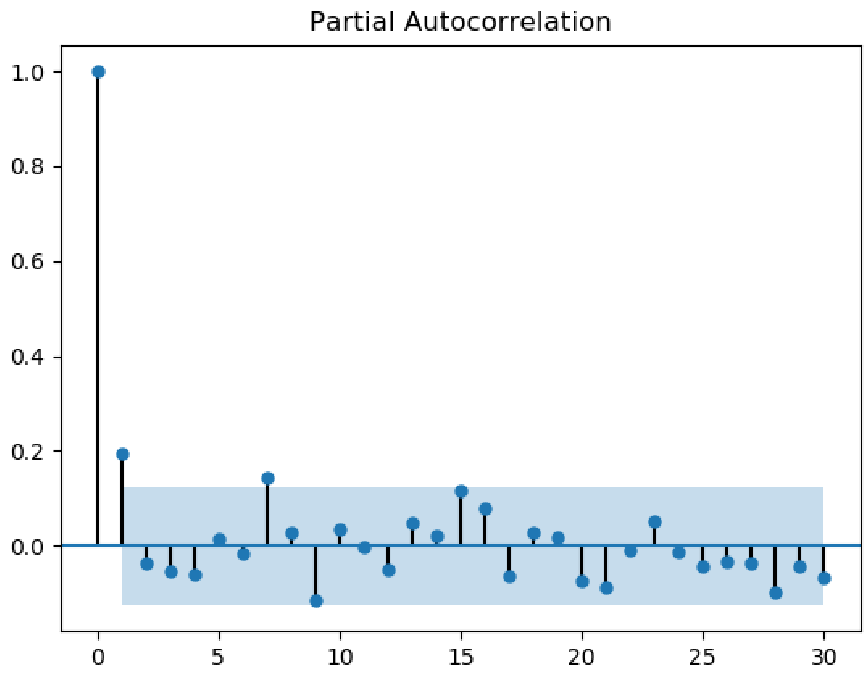


Figure A3 PACF of 1st Degree Differenced Data

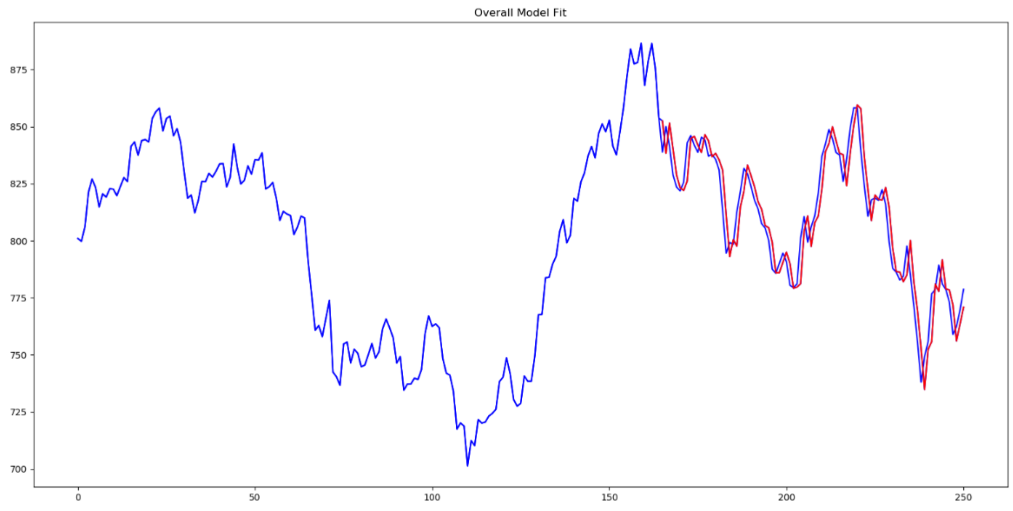


Figure A4 Fitting in the historical data

Appendix B

Setting of Artificial Neural Network in IBM SPSS Modeller 14.2

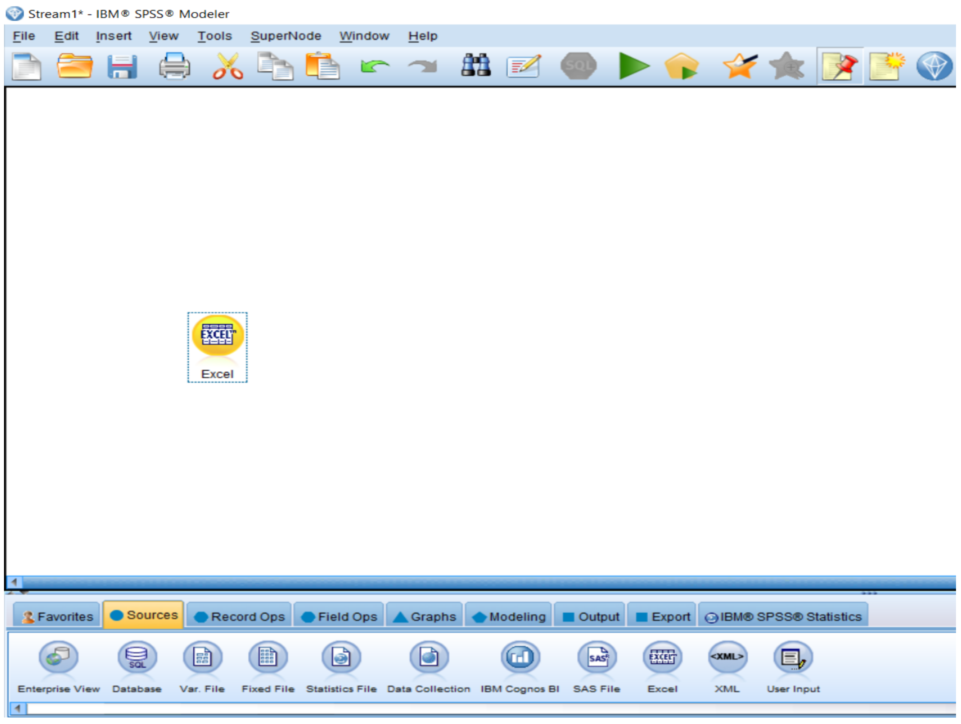
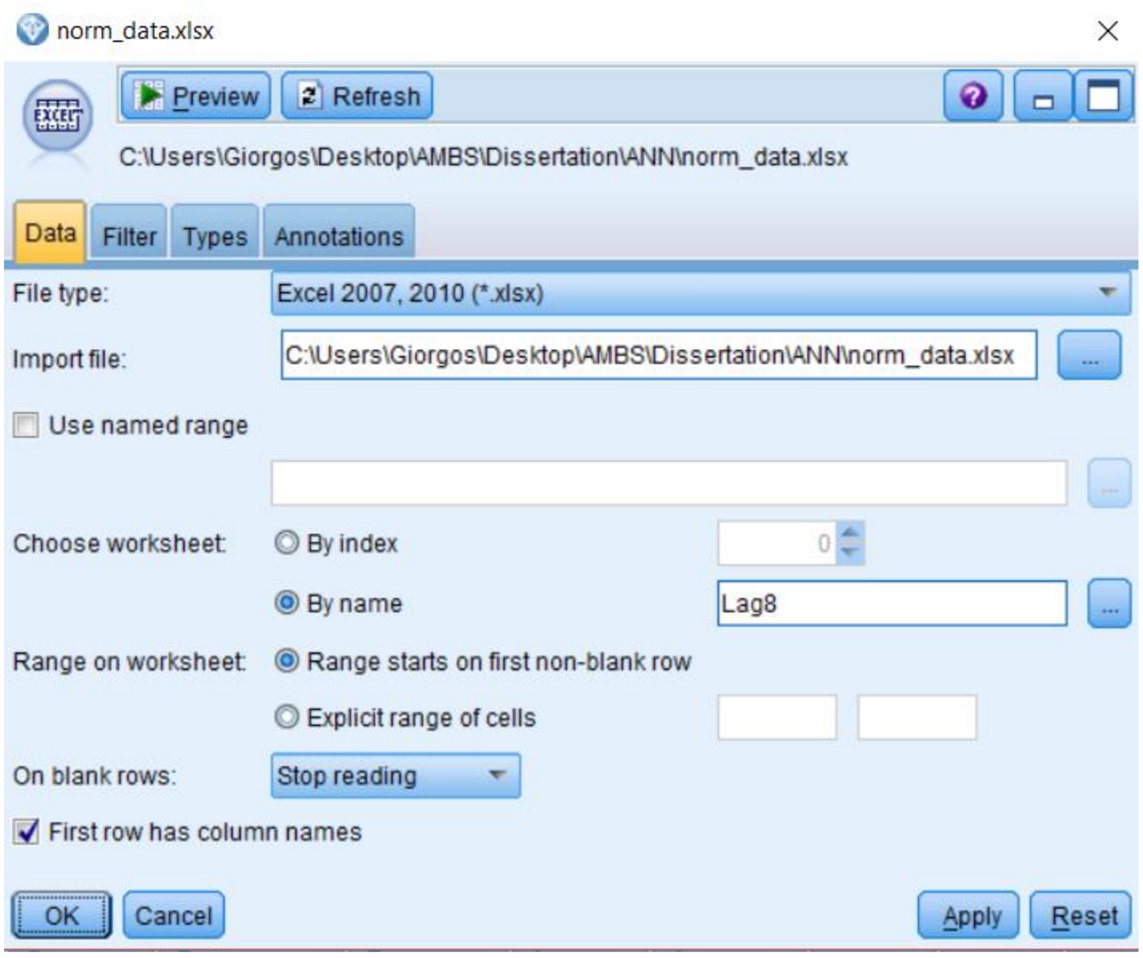


Figure B1 Importing files in IBM SPSS Modeller


norm_data.xlsx

C:\Users\Giorgos\Desktop\AMBS\Dissertation\ANN\norm_data.xlsx

File type: Excel 2007, 2010 (*.xlsx)

Import file: C:\Users\Giorgos\Desktop\AMBS\Dissertation\ANN\norm_data.xlsx

☐ Use named range

Choose worksheet: ○ By index 0

[✓ By name] Lag8

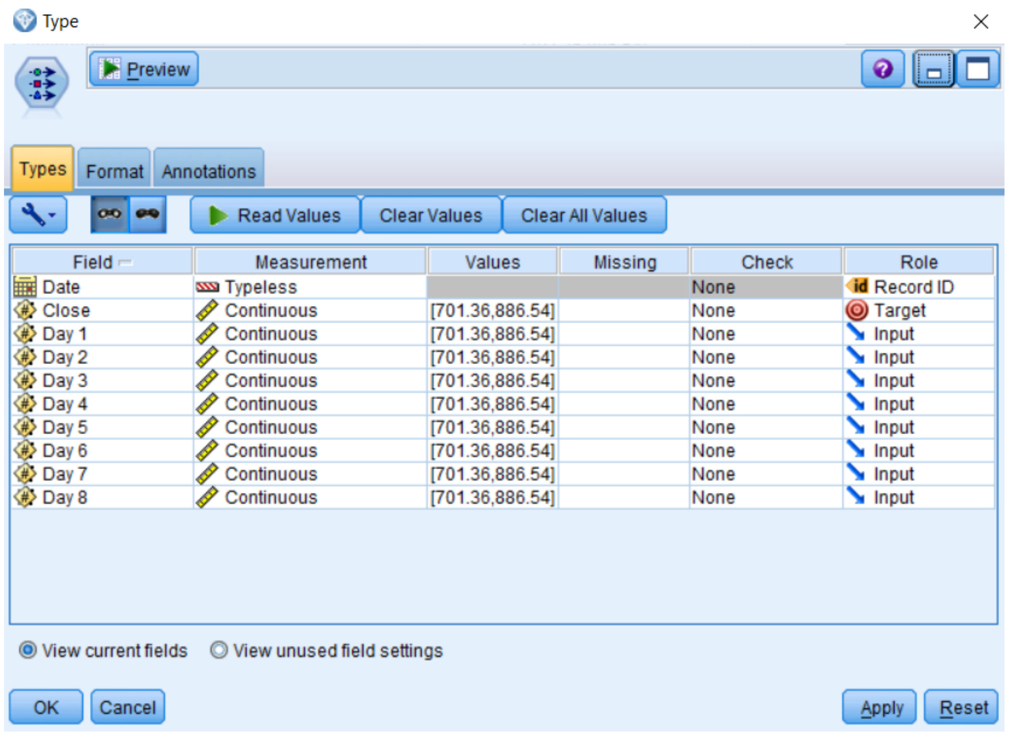
Range on worksheet: [✓ Range starts on first non-blank row]

○ Explicit range of cells

On blank rows: Stop reading

☑ First row has column names

Figure B2 Data Selection in SPSS Modeller 14.2


Type

Preview

Types | Format | Annotations

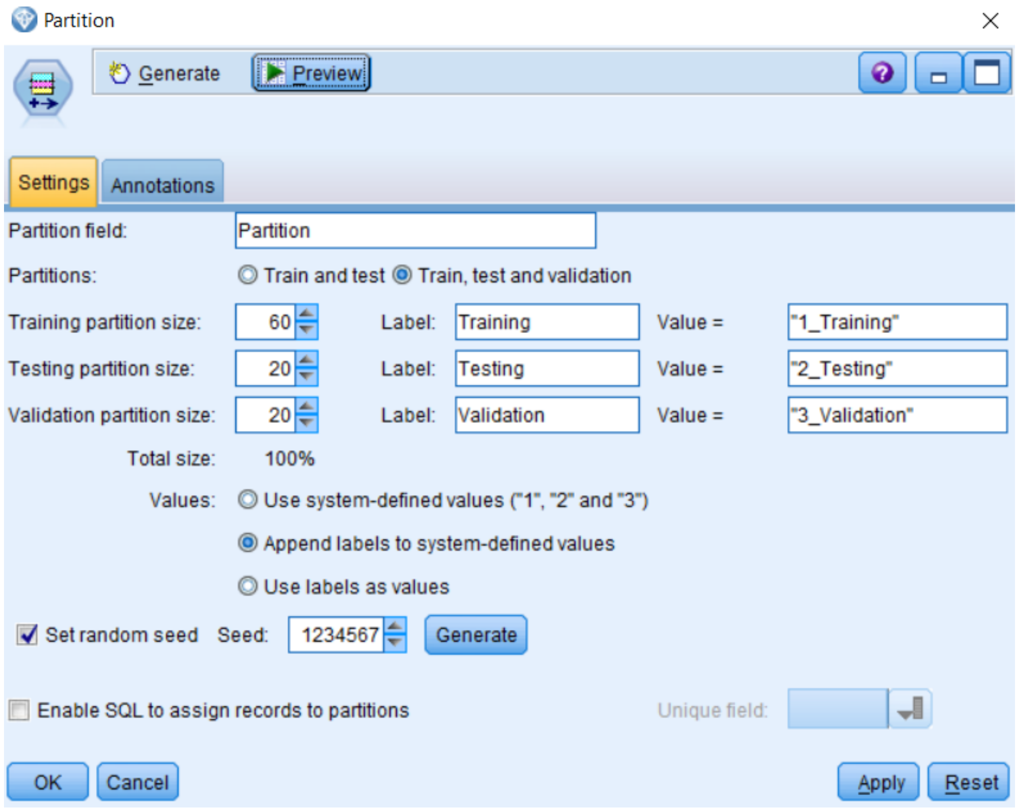
Read Values | Clear Values | Clear All Values

Field	Measurement	Values	Missing	Check	Role
Date	Typeless			None	Record ID
Close	Continuous	[701.36,886.54]		None	Target
Day 1	Continuous	[701.36,886.54]		None	Input
Day 2	Continuous	[701.36,886.54]		None	Input
Day 3	Continuous	[701.36,886.54]		None	Input
Day 4	Continuous	[701.36,886.54]		None	Input
Day 5	Continuous	[701.36,886.54]		None	Input
Day 6	Continuous	[701.36,886.54]		None	Input
Day 7	Continuous	[701.36,886.54]		None	Input
Day 8	Continuous	[701.36,886.54]		None	Input

[✓ View current fields] View unused field settings

OK | Cancel | Apply | Reset

Figure B3 Selecting the Types of the Variables for Eight Inputs


Partition

Generate | Preview

Settings | Annotations

Partition field: Partition

Partitions: Train and test [✓ Train, test and validation]

Training partition size:	60	Label:	Training	Value =	"1_Training"
Testing partition size:	20	Label:	Testing	Value =	"2_Testing"
Validation partition size:	20	Label:	Validation	Value =	"3_Validation"

Total size: 100%

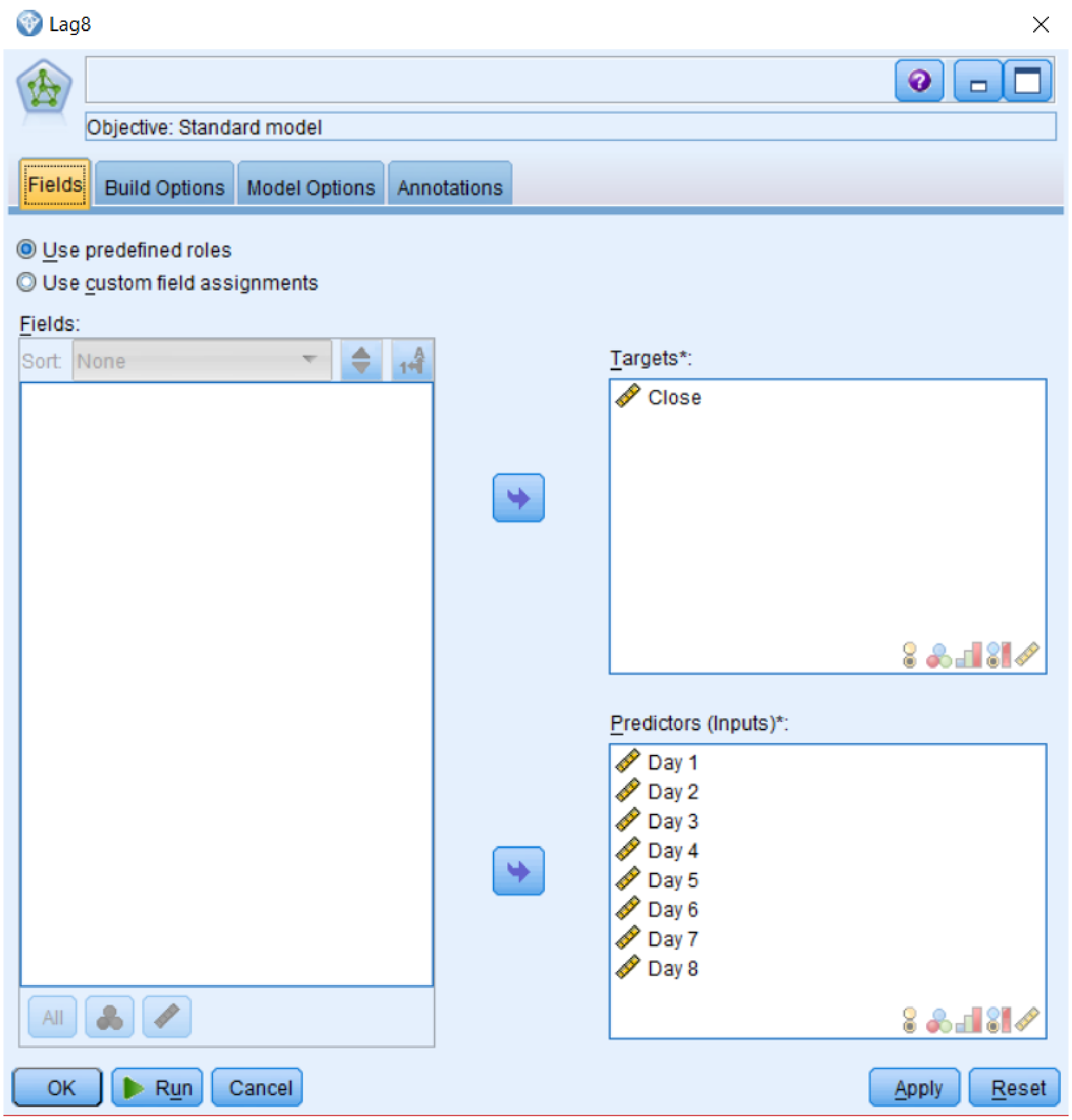
Values: Use system-defined values ("1", "2" and "3")
[✓ Append labels to system-defined values]
Use labels as values

[✓ Set random seed] Seed: 1234567 Generate

Enable SQL to assign records to partitions   Unique field:

OK | Cancel | Apply | Reset

Figure B4 Partitioning of Data


Lag8

Objective: Standard model

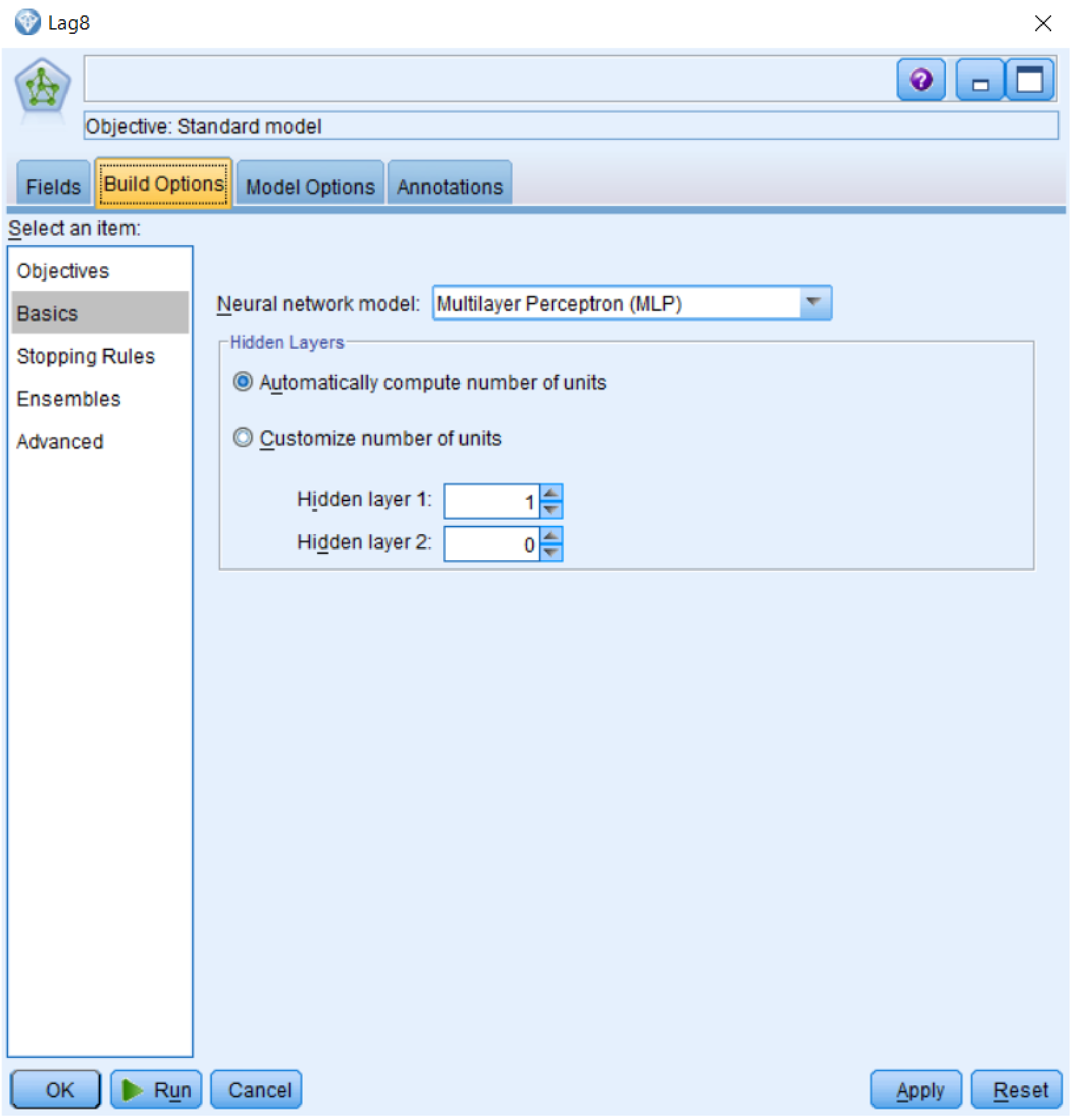
[✓ Use predefined roles]
Use custom field assignments

Fields:
Sort: None

Targets*:
Close

Predictors (Inputs)*:
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day 7
Day 8

Figure B5 Setting of Features for Building an ANN with Eight Inputs


Lag8

Objective: Standard model

Neural network model: Multilayer Perceptron (MLP)

Hidden Layers

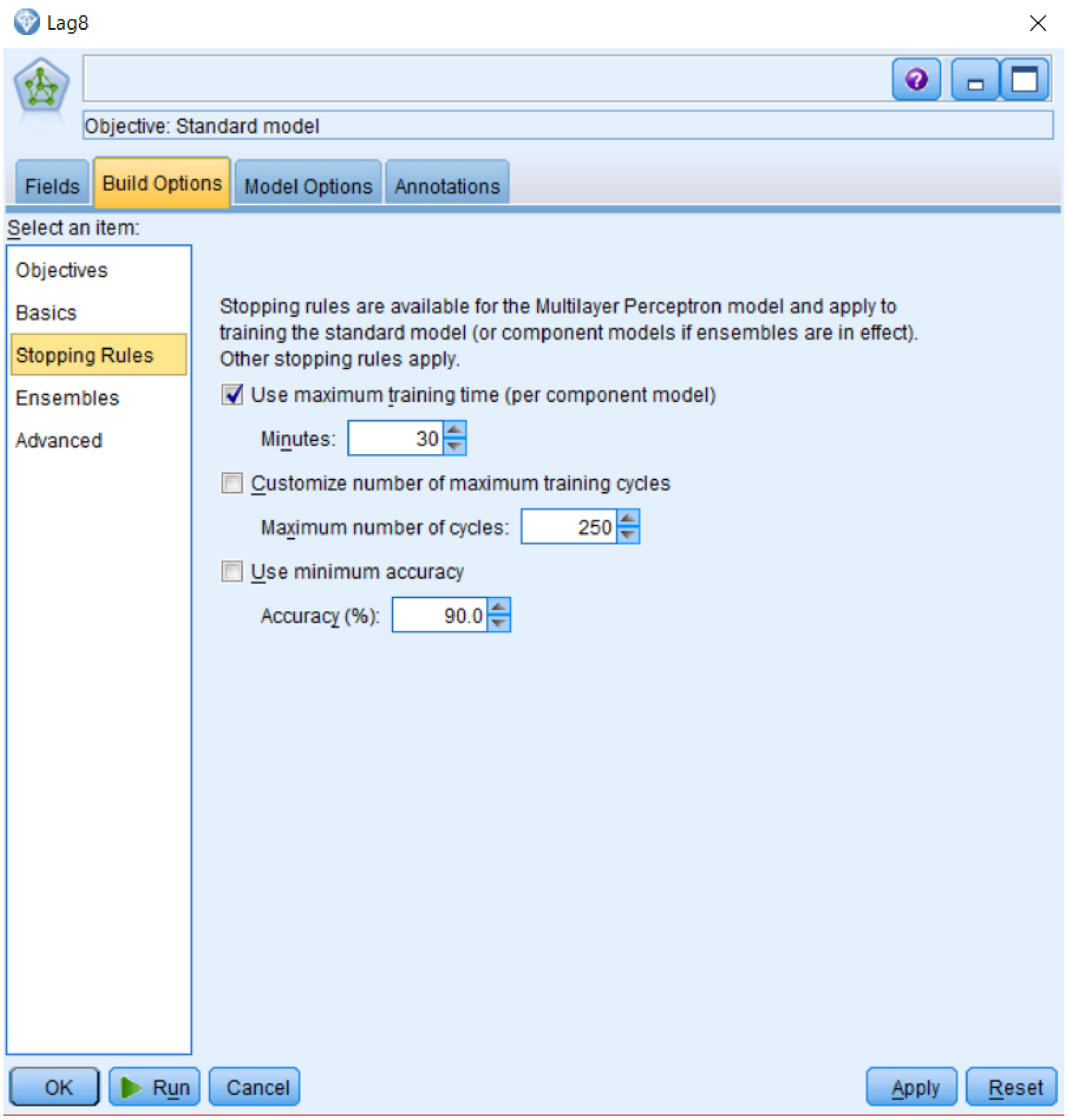
[✓ Automatically compute number of units]

Customize number of units

Hidden layer 1: 1

Hidden layer 2: 0

Figure B6 Setting the Model of the ANN with eight Inputs

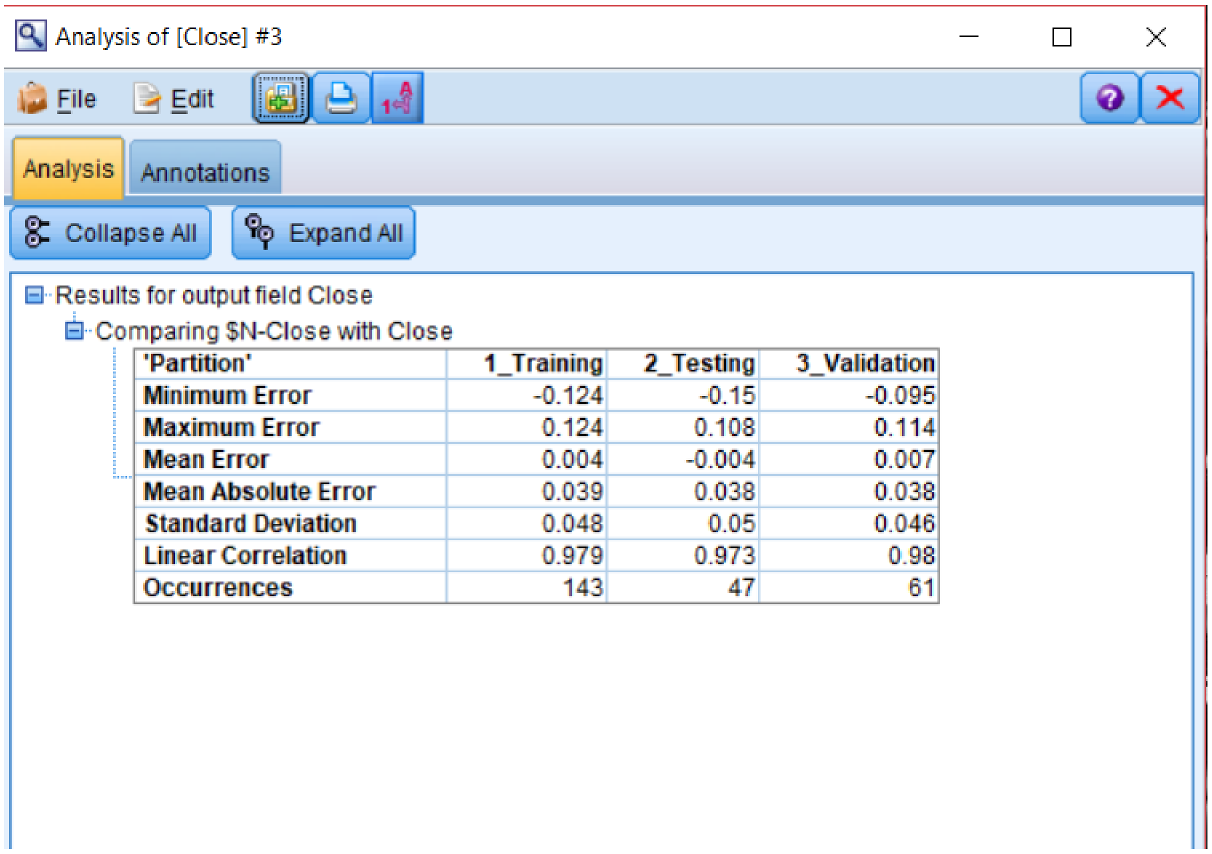


Objective: Standard model

Stopping rules are available for the Multilayer Perceptron model and apply to training the standard model (or component models if ensembles are in effect). Other stopping rules apply.

- [✓] Use maximum training time (per component model)
  - Minutes: 30
- [ ] Customize number of maximum training cycles
  - Maximum number of cycles: 250
- [ ] Use minimum accuracy
  - Accuracy (%): 90.0

Figure B7 Setting the Stopping Rule for the ANN with eight Inputs



Results for output field Close

Comparing $N-Close with Close

'Partition'	1_Training	2_Testing	3_Validation
Minimum Error	-0.124	-0.15	-0.095
Maximum Error	0.124	0.108	0.114
Mean Error	0.004	-0.004	0.007
Mean Absolute Error	0.039	0.038	0.038
Standard Deviation	0.048	0.05	0.046
Linear Correlation	0.979	0.973	0.98
Occurrences	143	47	61

Figure B8 Statistical Results of the eight Inputs ANN

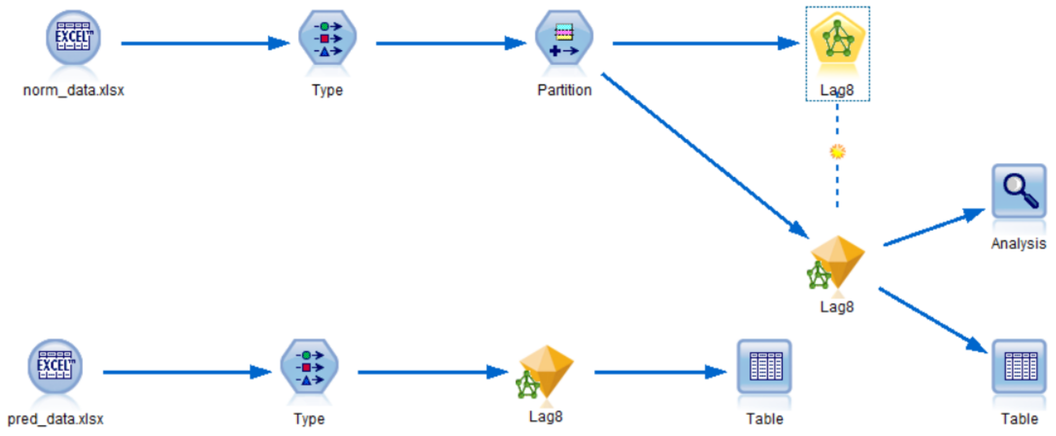
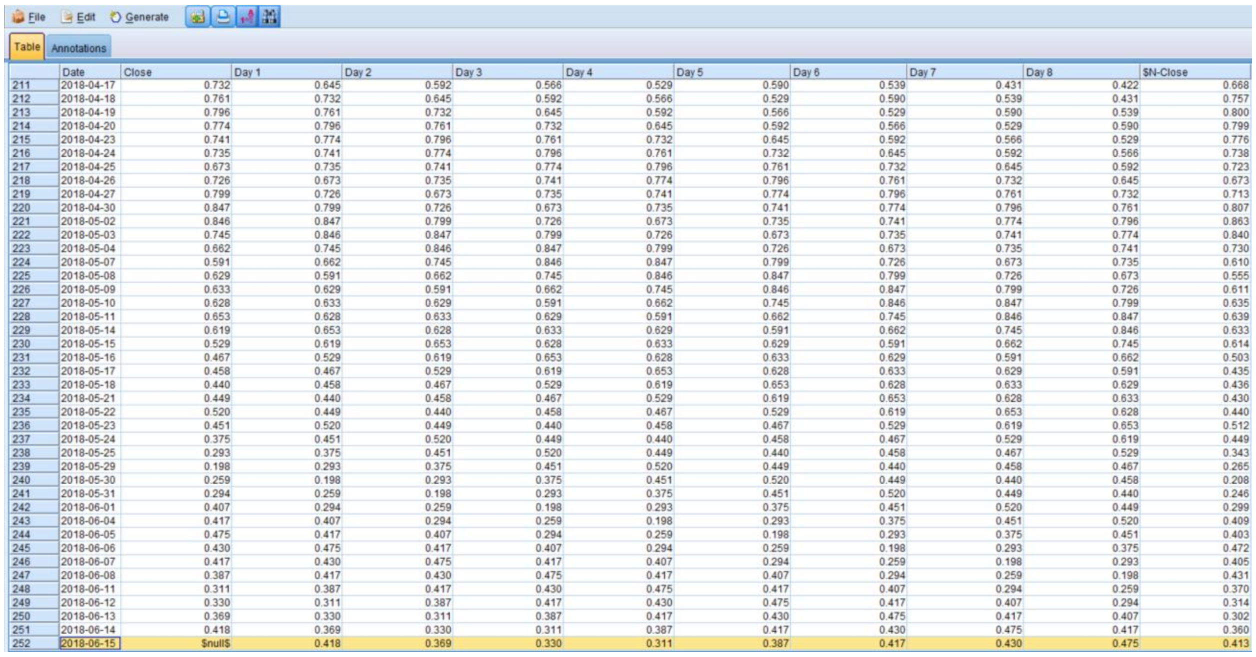


Figure B9 Diagrammatic representation of the forecasting procedure



	Date	Close	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	$N-Close
211	2018-04-17	0.732	0.645	0.592	0.566	0.529	0.590	0.539	0.431	0.422	0.668
212	2018-04-18	0.761	0.732	0.645	0.592	0.566	0.529	0.590	0.539	0.431	0.757
213	2018-04-19	0.796	0.761	0.732	0.645	0.592	0.566	0.529	0.590	0.539	0.800
214	2018-04-20	0.774	0.796	0.761	0.732	0.645	0.592	0.566	0.529	0.590	0.799
215	2018-04-23	0.741	0.774	0.796	0.761	0.732	0.645	0.592	0.566	0.529	0.776
216	2018-04-24	0.735	0.741	0.774	0.796	0.761	0.732	0.645	0.592	0.566	0.738
217	2018-04-25	0.673	0.735	0.741	0.774	0.796	0.761	0.732	0.645	0.592	0.723
218	2018-04-26	0.726	0.673	0.735	0.741	0.774	0.796	0.761	0.732	0.645	0.673
219	2018-04-27	0.799	0.726	0.673	0.735	0.741	0.774	0.796	0.761	0.732	0.713
220	2018-04-30	0.847	0.799	0.726	0.673	0.735	0.741	0.774	0.796	0.761	0.807
221	2018-05-02	0.846	0.847	0.799	0.726	0.673	0.735	0.741	0.774	0.796	0.863
222	2018-05-03	0.745	0.846	0.847	0.799	0.726	0.673	0.735	0.741	0.774	0.840
223	2018-05-04	0.662	0.745	0.846	0.847	0.799	0.726	0.673	0.735	0.741	0.730
224	2018-05-07	0.591	0.662	0.745	0.846	0.847	0.799	0.726	0.673	0.735	0.610
225	2018-05-08	0.629	0.591	0.662	0.745	0.846	0.847	0.799	0.726	0.673	0.555
226	2018-05-09	0.633	0.629	0.591	0.662	0.745	0.846	0.847	0.799	0.726	0.611
227	2018-05-10	0.628	0.633	0.629	0.591	0.662	0.745	0.846	0.847	0.799	0.635
228	2018-05-11	0.653	0.628	0.633	0.629	0.591	0.662	0.745	0.846	0.847	0.639
229	2018-05-14	0.619	0.653	0.628	0.633	0.629	0.591	0.662	0.745	0.846	0.633
230	2018-05-15	0.529	0.619	0.653	0.628	0.633	0.629	0.591	0.662	0.745	0.614
231	2018-05-16	0.467	0.529	0.619	0.653	0.628	0.633	0.629	0.591	0.662	0.503
232	2018-05-17	0.458	0.467	0.529	0.619	0.653	0.628	0.633	0.629	0.591	0.435
233	2018-05-18	0.440	0.458	0.467	0.529	0.619	0.653	0.628	0.633	0.629	0.436
234	2018-05-21	0.449	0.440	0.458	0.467	0.529	0.619	0.653	0.628	0.633	0.430
235	2018-05-22	0.520	0.449	0.440	0.458	0.467	0.529	0.619	0.653	0.628	0.440
236	2018-05-23	0.451	0.520	0.449	0.440	0.458	0.467	0.529	0.619	0.653	0.512
237	2018-05-24	0.375	0.451	0.520	0.449	0.440	0.458	0.467	0.529	0.619	0.449
238	2018-05-25	0.293	0.375	0.451	0.520	0.449	0.440	0.458	0.467	0.529	0.343
239	2018-05-29	0.198	0.293	0.375	0.451	0.520	0.449	0.440	0.458	0.467	0.265
240	2018-05-30	0.259	0.198	0.293	0.375	0.451	0.520	0.449	0.440	0.458	0.208
241	2018-05-31	0.294	0.259	0.198	0.293	0.375	0.451	0.520	0.449	0.440	0.246
242	2018-06-01	0.407	0.294	0.259	0.198	0.293	0.375	0.451	0.520	0.449	0.299
243	2018-06-04	0.417	0.407	0.294	0.259	0.198	0.293	0.375	0.451	0.520	0.409
244	2018-06-05	0.475	0.417	0.407	0.294	0.259	0.198	0.293	0.375	0.451	0.403
245	2018-06-06	0.430	0.475	0.417	0.407	0.294	0.259	0.198	0.293	0.375	0.472
246	2018-06-07	0.417	0.430	0.475	0.417	0.407	0.294	0.259	0.198	0.293	0.405
247	2018-06-08	0.387	0.417	0.430	0.475	0.417	0.407	0.294	0.259	0.198	0.431
248	2018-06-11	0.311	0.387	0.417	0.430	0.475	0.417	0.407	0.294	0.259	0.370
249	2018-06-12	0.330	0.311	0.387	0.417	0.430	0.475	0.417	0.407	0.294	0.314
250	2018-06-13	0.369	0.330	0.311	0.387	0.417	0.430	0.475	0.417	0.407	0.302
251	2018-06-14	0.418	0.369	0.330	0.311	0.387	0.417	0.430	0.475	0.417	0.360
252	2018-06-15	$null$	0.418	0.369	0.330	0.311	0.387	0.417	0.430	0.475	0.413

Figure B10 Prediction output of SPSS for the [8,7,1] MLP neural network

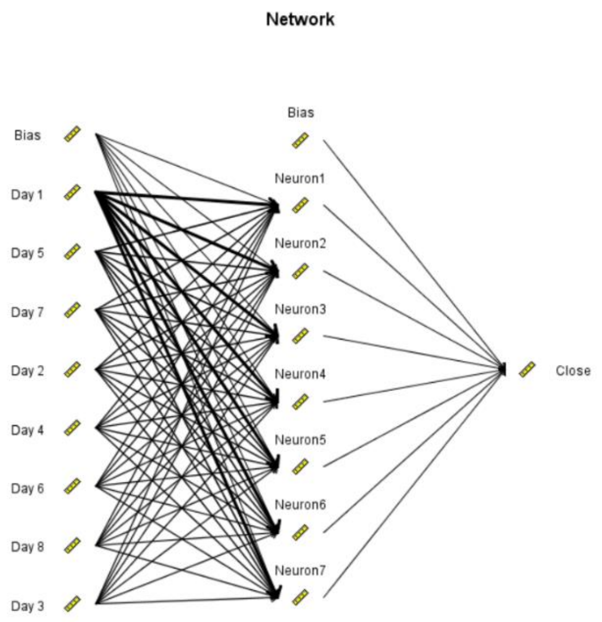


Figure B11 Breakdown of the [8,7,1] MLP model

Appendix C

Code in Python for ARIMA

```
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.stattools import adfuller,acf, pacf
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from pandas import datetime
from matplotlib import pyplot
from sklearn.metrics import mean_absolute_error



def parser(x):
      return datetime.strptime('190'+x, '%Y-%m')

data = pd.read_csv('ASE_Indices.csv', parse_dates=['Date'], index_col='Date')
series=data['Close']
X = series.values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
validation = list()
forecast=list()

series_log = np.log(series)
series_diff = series_log - series_log.shift()
series_diff.dropna(inplace=True)



def test_stationarity(timeseries):

    #Determing rolling statistics
    rolmean = pd.rolling_mean(timeseries, window=12)
    rolstd = pd.rolling_std(timeseries, window=12)

    #Plot rolling statistics:
    plt.plot(timeseries, color='blue',label='Original')
    plt.plot(rolmean, color='red', label='Rolling Mean')
    plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.figure()

    #Perform Dickey-Fuller test:
    print ('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries,autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-
value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print (dfoutput)


def ACF_PACF():

#    ACF and PACF plots:
    lag_acf = acf(series, nlags=20)
    lag_pacf = pacf(series, nlags=20, method='ols')
    lag_acf = acf(series_diff, nlags=20)
    lag_pacf = pacf(series_diff, nlags=20, method='ols')

    #Plot ACF:
    plt.subplot(121)
    plt.plot(lag_acf)
    plt.axhline(y=0,linestyle='--',color='gray')
    plt.axhline(y=-1.96/np.sqrt(len(series_diff)),linestyle='--
',color='gray')
    plt.axhline(y=1.96/np.sqrt(len(series_diff)),linestyle='--',color='gray')
    plt.title('Autocorrelation Function')

    #Plot PACF:
    plt.subplot(122)
    plt.plot(lag_pacf)
    plt.axhline(y=0,linestyle='--',color='gray')
    plt.axhline(y=-1.96/np.sqrt(len(series_diff)),linestyle='--
',color='gray')
    plt.axhline(y=1.96/np.sqrt(len(series_diff)),linestyle='--',color='gray')
    plt.title('Partial Autocorrelation Function')
    plt.tight_layout()

    plot_acf(series, lags=30)
    pyplot.show()
    plot_pacf(series, lags=30)
    pyplot.show()

    plot_acf(series_diff, lags=30)
    pyplot.show()
    plot_pacf(series_diff, lags=30)
    pyplot.show()


def validation_proc():

    for t in range(len(test)):
        model = ARIMA(history, order=(1,1,1))
        model_fit = model.fit(disp=0)
        output = model_fit.forecast()
        yhat = output[0]
        validation.append(yhat[0])
        obs = test[t]
        history.append(obs)
        print('predicted=%f, expected=%f' % (yhat, obs))

    error = mean_absolute_error(test, validation)
    print(model_fit.summary())
    # plot
    li=train.tolist()+validation
    df=pd.DataFrame({'ExpPrice':history,'PredPrice':li})
    ax = df['ExpPrice'].plot(color='b')
    ax = df['PredPrice'].plot(color='b')
    df.loc[df.index >= len(train), 'PredPrice'].plot(color='r', ax=ax)
    plt.title('Overall Model Fit')
    pyplot.figure()

    pyplot.plot(test,label='Expected Price')
    pyplot.plot(validation, color='red',label='Predicted Price')
    plt.legend(loc='best')
    plt.title('MAE:%f' %error)
    pyplot.show()


def forecast_proc():


     model = ARIMA(X, order=(1,1,1))
     model_fit = model.fit(disp=0)
     output = model_fit.forecast(steps=30)
     forecast = output[0]


     plt.plot(forecast, color='red',label='Predicted Price')
     plt.legend(loc='best')
     pyplot.show()


plt.ylim(650,950)
test_stationarity(series)

plt.ylim(-0.05,0.055)
test_stationarity(series_diff)

ACF_PACF()

validation_proc()

forecast_proc()
```

Code in Python for the Support Vector Machine

```
from sklearn.svm import SVR
from pandas import datetime
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import learning_curve
import numpy as np
from sklearn.metrics import mean_absolute_error
```

```
def parser(x):
      return datetime.strptime('190'+x, '%Y-%m')

data = pd.read_csv('data_norm_svr.csv', parse_dates=['Date'],
index_col='Date')
X = data.drop(['Close','Day 8','Day 7','Day 6','Day 5','Day 4','Day 3'],
axis=1)
#X= X.values
y = data['Close'].values.reshape((251,1))
size=200
# Split the dataset in two equal parts
X_train, X_test, y_train, y_test = X[0:size], X[size:], y[0:size], y[size:]

#
############################################################################
# Fit regression model

Cs = [2**-5,2**-4,2**-3,2**-2,2**-
1,2**0,2**1,2**2,2**3,2**4,2**5,2**6,2**7,2**8,2**9,2**10,2**11,2**12,2**13,2
**14,2**15]
gammas = [2**-15,2**-14,2**-13,2**-12,2**-11,2**-10,2**-9,2**-8,2**-7,2**-
6,2**-5,2**-4,2**-3,2**-2,2**-1,2**0,2**1,2**2,2**3]

svr_rbf = GridSearchCV(SVR(kernel='rbf', gamma=0.1), cv=10,
               param_grid={"C": Cs,
                           "gamma": gammas})

svr_lin = GridSearchCV(SVR(kernel='linear'), cv=10,
               param_grid={"C": Cs})

svr_poly = GridSearchCV(SVR(kernel='poly'), cv=10,
               param_grid={"C": Cs})


y_rbf = svr_rbf.fit(X_train, y_train).predict(X_test)
y_lin = svr_lin.fit(X_train, y_train).predict(X_test)
y_poly = svr_poly.fit(X_train, y_train).predict(X_test)

print(svr_rbf.best_params_)
print(svr_lin.best_params_)
print(svr_poly.best_params_)
print(mean_absolute_error(y_test, y_rbf))
```