Financial Forecasting Using Machine Learning

Research · June 2016

DOI: 10.13140/RG.2.1.1186.0083

1 author:

Solomon Addai

African Institute for Mathematical Sciences South Africa

1 PUBLICATION **0** CITATIONS

SEE PROFILE

Financial Forecasting Using Machine Learning

Solomon Addai (solomonaddai@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Prof. Ronnie Becker
African Institute for Mathematical Science, South Africa

19 May 2016

Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

Abstract

Accurate predictions of the direction of the returns on stock/index are of paramount interest to market dealers and investors. Researchers have shown significant interest in developing different techniques to help generate high forecasting accuracy of stock returns or prices movement. In this essay, five different techniques are considered and applied to predict the movement of the daily returns on the Standard and Poor 500 stock index (S&P 500). These techniques are Artificial Neural Network (ANN), Logit model, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA) and K-Nearest Neighbourhood classification (KNN). Results show that ANN performed relatively better in the classifications of the up and down movements of the returns than the other techniques. Specifically, ANN achieved approximately 61 percent in predicting the returns using opening prices of stocks/indices.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

Solomon Addai, 19 May 2016

Contents

1. Introduction

The financial sector has become one of the most important subjects of human life in recent years. Investors desire to invest their assets in the most viable manner. However, they want to achieve this at the lowest risk of loss. Information on the trend and behaviour of the market is therefore relevant. This essay seeks to predict the direction of the daily returns on Standard and Poor 500 (S&P 500) index, which is a New York Stock Market index and develop trading strategies based on the results.

Many efforts have been made by researchers to forecast the S&P 500 index which is regarded as one of the most influential stock indices in the financial world (Chan, 2009). Some uses linear forecasting techniques with the assumption that there exist a linear relationship between return and the factors affecting it.They resort to this approach due to its simplicity in terms of computation. However, the usefulness of a model does not only depend on its simplicity, but also on how accurate it can predict the underlying phenomenon. Other researchers however hold the view that financial data is noisy and nonlinear and as such, nonlinear forecasting techniques are the most appropriate [Bensaïda (2014), Kantz and Schreiber (2004)]. Notwithstanding, both techniques assume that past information has the ability of predicting the future return on stocks or indices (Enke and Thawornwong, 2005). This assumption of predicting the market with past information is not achievable according to Malkiel and Fama (1970) in the Efficient Market Hypothesis (EMH); with the argument that current stock prices "fully reflect" available information and as such, there is no chance of earning excess returns with the given information. Amidst all these different views on the the prediction of stocks/indices trend, studies have shown that, there has been numerous successes in predicting the market with different approaches.

Kutsurelis (1998) achieved 93.3 percent accuracy in predicting a market rise, and an 88.07 percent in predicting a market drop in S&P 500 using Artificial Neural Networks. Desai and Bharati (1998a), Desai and Bharati (1998b), Thawornwong et al. (2003), and many other researchers made predictions of the financial market using different approaches. The various analysis used by the various researchers are classified into "technical" and "fundamental". Agrawal et al. (2013) explained technical analysis as the predictions made using data generated by "market activity, past prices, and volume" whiles fundamental analysis involves an "in-depth" analysis with variables which are not only dependent on market activities. ANN is very efficient in predicting the stock market using any of the fundamental or technical approach [Maciel and Ballini (2010),Hanias et al. (2012)].

This study seeks to use ANN to make predictions of the direction of the daily returns on standard and Poor (S&P 500) index using the 'technical analysis' approach and compare its performance with other algorithms for classification in terms of accuracy. The choice of ANN is motivated by the fact that financial time series are noisy, chaotic and nonlinear in nature and hence, predicting its future behaviour requires a nonlinear approach of which ANN is. Specifically, this essay will assess the performance of ANN in classifying the movement or direction (up or down) of the return on the S&P 500 index by comparing its hit rate with other techniques which are: linear discriminant analysis, quadratic discriminant analysis, logistic regression and the k-nearest neighbourhood algorithm. Researchers have taken different dimensions on predicting the returns on stocks with different approaches. Masoud (2014) used neural network to predict the direction of stock prices index movement and assessed the accuracy with the various error measurement techniques. Ou and Wang (2009) in an attempt to predict the changes in the daily closing prices of the Hand Seng index, using open price, low price, high prices, the S&P 500 index, US dollar- Hong Kong dollar exchange rate; reported various level of successes using different techniques, with all techniques achieving more than 80 percent hit-rate. Amin et al. (2014) achieved as high as above 89 percent accuracy in predicting the direction of stock price index in Tehran stocks.

With this background, the results from this essay will be compared with the claims of other researches to observe how well our models are performing. Results from this essay would be useful for risk-averse investors, since it will help them to know when to invest and when to hold their resources.

The rest of this essay is structured as follows: chapter two talks about an introduction to the various classification techniques used in this essay, chapter three describes the various methods used, chapter four presents the results and discussions and conclusion based on our findings comes in chapter five.

2. Machine Learning Algorithms

This Chapter presents an introduction to the various classification techniques used in this essay. These techniques ranges from linear to nonlinear. This essay uses different techniques with the motive that, it can determine the one that relatively performs better. The chapter starts by giving a brief introduction to Artificial Neural Networks and later introduces the other classification algorithms considered.

Let the set $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ be the set of training data, where $X = (x_1, x_2, \cdots, x_n) \in \mathbb{R}$ is input vector and $Y \in \{0, 1, \cdots, j\}$ where j is the number of classes, i.e Y is a categorical variable. The idea of classification then, is to build a classifier $C(x)$ to predict Y using X as the bases (Ou and Wang, 2009). That is, to build a model that will help dissociate between the various classes of Y based on X. Suppose Y has exactly two classes (binary), i.e. $Y \in \{0, 1\}$, then the decision boundary separating the two classes is a hyperplane in the feature space. According to Ou and Wang (2009), for a q-dimensional input space, the hyperplane is the set:

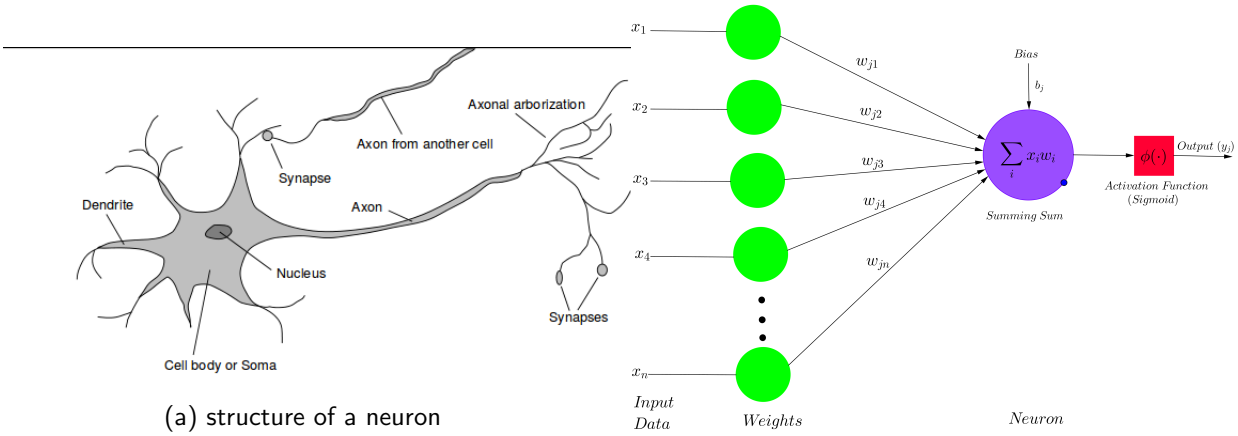
$$\{x : \alpha_0 + \sum_{k=1}^{q} \alpha_k x_k = 0\}. \tag{2.0.1}$$

Also, the hyperplane separating the two regions is given by the sets: $\{x : \alpha_0 + \sum_{k=1}^{q} \alpha_k x_k < 0\}$ and $\{x : \alpha_0 + \sum_{k=1}^{q} \alpha_k x_k > 0\}$. Below, we explain the various classification algorithms used in this essay.

2.1 Artificial Neural Networks

Artificial neural network is a computational model which was first developed by McCulloch and Pitts in 1943 (Clarence N. W, 2001). This principle has natured to a great extent over the past years and especially with the era of high performance computing. Its development was inspired by the activities of the human brain. Researchers thought of how to make a computer mimic the large amount of interconnections and networking that exist between all the nerve cells of the human brain. It mimics the brain in the sense that, it acquires knowledge by learning from an input data, and transforms the data into knowledge, just as the brain does. Neural networks are used for a wide range of purposes, but they all fall under pattern recognition. By recognising both linear and nonlinear patterns, ANN helps in building predictive models which are used for predicting a future phenomenon. Predictive models built by ANN fall into two categories: classification (with categorical outcome) and regression (with a continuous outcome). Artificial Neural Networks can be classified into two broad categories: feedforward and recurrent networks. An artificial neural network is classified as recurrent if the connections between units (neurons) form a direct cycle. On the contrary, a feedforward NN is a class of NN which considers the transfer of information/messages only in the forward direction (i.e not in a cycle). That is to say; input signals travel in one direction, which in a sense implies that the output in a particular layer has no effect on the same layer.

The figure below relates the biological neuron to an artificial neuron:



(a) structure of a neuron

Source: (Nastos et al., 2011)

(b) Artificial Neuron

Figure 2.1a depicts the structure of the biological neuron which processes information. Upon receiving information from another neuron through the dendrites, a neuron transfers the information received to another neuron with the aid of an axon. Figure 2.1b also gives a mathematical model for the neuron. This model, just as the biological neuron, receives an input data, processes it and produce an output. Thus, the formulation of the artificial neuron was with the motive of making it mimic the interconnections that exist in the biological neuron.

2.1.1 Multilayer Perceptrons (MLP). A perceptron model without a hidden layer is only efficient when dealing with a linearly separable problem (Clarence N. W, 2001). To overcome this limitation (handle nonlinear problems), MLP is used. A multilayer differs from a perceptron model in the sense that, it takes into account additional layer(s) known as hidden layer(s). A multilayer perceptron is also known as a multilayer feedforward network. For a continuous function defined on a compact domain, a single hidden layer feedforward can approximate; irrespective of the underlying activation function (Hornik et al., 1989). A Multilayer perceptron has the property that, it takes only one input layer and one output layer. However, MLP can take more than one hidden layer, depending on the learning algorithm or the underlying phenomenon to be modelled. With the exception of the input layer, each neuron within each layer computes the sum of a received input information, and gives a scaled real value (which may be within the interval [0,1] or [-1,1], depending on the activation function) as the output.

The figure below is a MLP with an input layer, one hidden layer and an output layer.

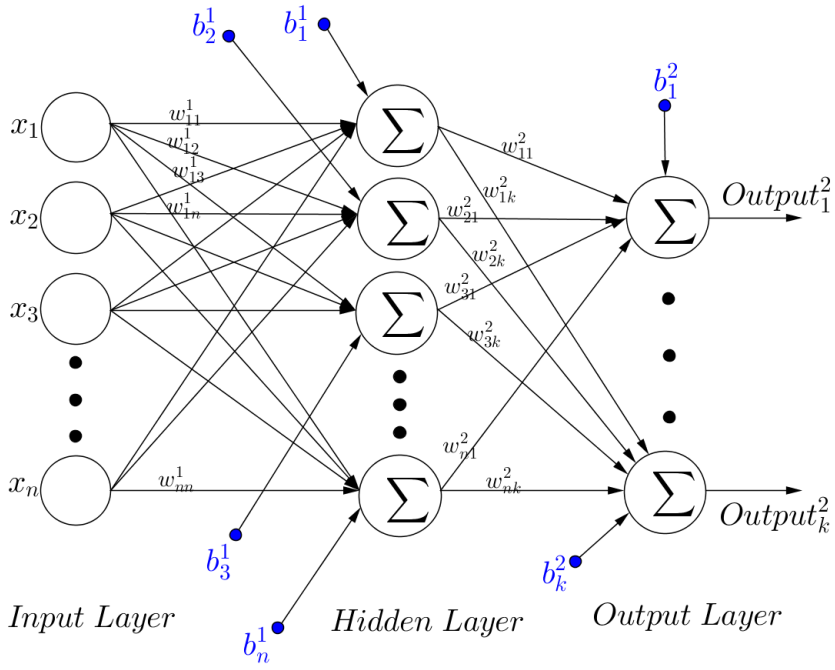


Figure 2.2: Multilayer neural network architecture

In figure 2.2, w_{in}^k represents the weight of the dendrite from the i^{th} neuron of the $k-1$ layer to the n^{th} neuron of layer k; b_i^k is the bias of the i^{th} neuron within the layer k.

Mathematically,

$$output_{(n)}^{(l)} = \varphi^{(l)}\left(\sum_k output_k^{(l-1)} w_{(in)}^{(l)} + b_{(n)}^{(l)}\right). \tag{2.1.1}$$

Each neuron n in the l^{th} layer receives activations $output_k^{(l-1)} w_{kn}^{(l)}$ from the previous layer, activates it, and passes the activated $output_{(n)}^{(l)}$ to the neurons of the layer that follows it. For simplicity, let $output_{(n)}^{(l)}$ be y_n^l and $(output_k^{(l-1)} w_{(in)}^{(l)})$ be x_k^l. It follows then that, the output from the perceptron y_n can be written in the form:

$$y_k(x_i) = \varphi(z_k), \tag{2.1.2}$$

where $z_k = \sum_n x_k^{(l)} w_{kn}^{(l)} + b_k^{(l)}$, $\varphi()$ is an activation function. For a sigmoid activation function given by:

$$\varphi(z) = [1 + e^{-z}]^{-1}, \tag{2.1.3}$$

the derivative has the following property:

$$\begin{aligned}
\varphi'(z) &= -(1+e^{-z})^{-2}(-e^{-z}) \\
&= e^{-z}(1+e^{-z})^{-2} \\
&= \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right).
\end{aligned}$$

Hence, the derivative of the sigmoid function is given by:

$$\varphi'(z) = \varphi(z)[1 - \varphi(z)]. \tag{2.1.4}$$

2.1.2 Terminologies used in neural network architecture.

a) **Input Layer:** the first layer in the interconnections of the NN architecture which distributes input signals to the hidden layer. That is, the layer that receives the set of inputs (vector of inputs; which are real values) which are passed through the network to produce an output. Alternatively, the input layer can be thought of as the layer from which input data is fed into the network.

b) **Weights:** In NN, weights are real-valued numbers assigned as coefficients to the input variables to describe the contributions (thus, the strength) that the input makes to the output.

c) **Hidden Layer:** the hidden layer contains neurons which communicate with other neurons of the next hidden layer if any or the output layer otherwise (Nastos et al., 2011). Using activation/transfer functions, each neuron in the hidden layer sums and processes inputs received and sends the resultant to the output layer.

d) **Transfer/Activation function:** this is a function which may be linear or non-linear and its role in the NN is to provide a non-linear decision boundary with the computed combinations of the weighted inputs. Below are the most commonly used activation functions and their properties:

Table 2.1: functions

Function	Equation	Range of values
Logistic	$\varphi(y) = \frac{1}{1+e^{-y}}$	$(0, 1)$
Hyperbolic tangent	$\varphi(y) = \frac{e^{y}-e^{-y}}{e^{y}+e^{-y}}$	$(-1, 1)$

e) **Output Layer:** this layer is made up of neurons, known as output neurons with the sole task of returning the network's output and comparing it with the target output. It takes only one neuron when dealing with a regression model; and takes one or more neuron(s) when dealing with classifications.

2.1.3 Learning of NN. Learning involves the process of weight adjustment of the interconnections between the neurons of an ANN, given an external input. The main goal of the learning process is to minimize the error summed over all trained cases. Learning of ANN can be classified into two: supervised and unsupervised. In a supervised learning process, a number of data pairs (input values - desired output/target values) are used in training the network. An attempt is made by the network at each training, to compute the desired/target output, using the set of given inputs. The computed output from the network is compared to the target output. The difference between the computed output and the target value is the error. Given that such error has occurred, the network attempts to minimize the error by continuously adjusting the weights of its connections in an iterative manner; until minimum error is achieved. There are many supervised learning algorithms; but the most commonly used ones are the delta rule and the generalization of the delta rule which is popularly known as the backpropagation algorithm (Clarence N. W, 2001). Unlike the supervised learning, in unsupervised learning, there are no explicit target values. Weight adjustments are made based on some parameters and modification of network parameters are based on size (Veri and Baba).

Let $y(x_i)$ be the output for input x_i and t_i be the target output. Then according to Bullinalia (Accessed April 2016), the least squares error (E) is given by:

$$E_i(w,b) = \frac{1}{2}\sum_{i=1}^{n}\Big(t_i - y(x_i)\Big)^2 \tag{2.1.5}$$

with the cross entropy function defined as:

$$E_i(w,b) = -\frac{1}{n}\sum_{p}\sum_{i}\left[t_i^p \log\Big(y^p(x_i)\Big) + \Big(1-t_i^p\Big)\log\Big(1-y^p(x_i)\Big)\right] \tag{2.1.6}$$

where $y(x_i) = \varphi(z_i) = \varphi\big(\sum_k w_{jk}x_j + b_j\big)$ and $\varphi(\cdot)$ is the sigmoid activation function.

2.1.4 Backpropagation Algorithm. In backpropagation, the focus is on understanding how changes in the weights and the biases in NN affects the cost function (error). In effect, partial derivatives of the cost function with respect to the weights and biases are computed.

The backpropagation technique is the generalization of the gradient decent algorithm which is used for updating the weights of the inputs to the neurons of a single-layer NN. In delta rule, the attempt is to minimize the error through gradient decent (Delta). Generally, the learning process of an ANN with error backpropagation takes the following steps (Veri and Baba):

a) Initialize values for weights, learning rate (η) and momentum (α), where the learning rate is a parameter that determines the amount of adjustments that need to be made to the weights.

b) Compute the actual output of the hidden layer (using the activation function on the neurons of the layer) with the formula:

   i.
   $$y_j(l) = \varphi\left[\sum_{i=1}^{n} w_{ij}(l)x_{ij}(l) + b_j\right]. \tag{2.1.7}$$

   ii. Compute the actual output of the output layer by applying the activation function on the output from the hidden layer. This is done by using the relation:
   $$y_k(l) = \varphi\left[\sum_{i=1}^{n} w_{ij}(l)x_{jk}(l) + b_j\right]. \tag{2.1.8}$$

c) Backpropagate the error and train the weights (update weights) to minimize the error. This is achieved by:

   i. calculating the output layer's error gradient using the relation:
   $$\delta_k(l) = y_k(k) \times [1 - y_k(l)] \times \varepsilon_k(l), \tag{2.1.9}$$
   where $\varepsilon_k(l) = t_k(l) - y_k(l)$, such that $t_k(l)$ is the desired/target output and $y_k(l)$ is the computed output. The weight between the hidden layer and the output layer is then updated, using the formula:
   $$\Delta w_{jk}(l) = \alpha y_j(l)\delta_k(l). \tag{2.1.10}$$
   This adjustment is iterated into the previous value with the new weight value:
   $$w_{jk}(l+1) = w_{jk}(l) + \Delta w_{jk}(l). \tag{2.1.11}$$

ii. Calculating the hidden layer's error gradient. This is done with the relation:

$$\delta_j(l) = y_j(l) \times [1 - y_j(l)] \times \sum_{k=1}^{n} \delta_k(l) \times w_{jk}(l). \tag{2.1.12}$$

The weights between the input layer and the hidden layer is also computed with the formula:

$$w_{ij}(l+1) = w_{ij}(l) + \Delta w_{ij}(l), \tag{2.1.13}$$

where $\Delta w_{ij}(l) = \alpha \times x_i(l) \times \delta_j(l)$.

The summary and pictorial view of the learning process is displayed in the figure below:

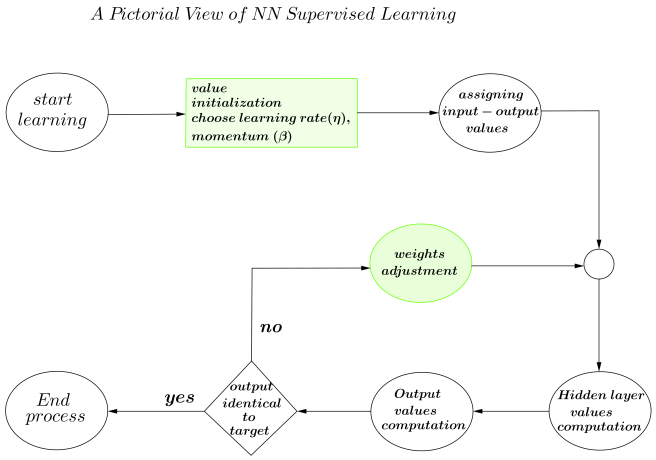


Figure 2.3: A supervised learning process

In figure 2.3, we observe the various stages of how neural network learns using an error backpropagation mechanism. Minimum error is achieved when the computed output is identical to the targeted output.

2.2 Linear and Quadratic Discriminant Analysis

The main objective of discriminant analysis given say two classes; is to identify variables which are good at discriminating between the two distinct classes, develop a function that computes a new variable which is a measure of the difference between the two classes, and based on the identified function, develop a rule that classifies future observations into one of the two distinct classes. The idea of discriminant analysis is based on Bayes theorem. Suppose $P(Y = y|X)$ is a posterior probability of being in class 'a' and $P(X)$ is the prior probability. Then according to the Bayes theorem:

$$P(Y = y|X) = \frac{P(X|Y = y) \cdot P(Y = y)}{P(X)}. \tag{2.2.1}$$

Bayes classification rule is to choose the class where $P(Y|X)$ is maximal. That is, obtaining class posteriors $P(Y = y|X)$ where classification is optimized.

Supposed $P(X|Y) \sim N(\mu, \Sigma)$, where $N(\mu, \Sigma)$ is a multivariate Gaussian distribution with a mean vector μ and a covariance matrix Σ. Then the multivariate density associated with this vector x is defined by:

$$f(x) = \frac{1}{\sqrt{(2\pi)^m |\Sigma|}} \exp\Big[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\Big], \tag{2.2.2}$$

where $|\Sigma|$ is the determinant of Σ and m is the size of x.

For observed sample data with binary response (two classes), say i and k, the multivariate Gaussian density is defined as:

$$P(X|Y=i) = \frac{1}{\sqrt{(2\pi)^m |\Sigma_i|}} \exp\Big[-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\Big]. \tag{2.2.3}$$

Taking the logarithm of this posterior, implies:

$$\log[P(X|Y=i)] = -\frac{1}{2}(x-\mu_i)^T \Sigma^{-1}\mu_i - 2\mu_i^T \Sigma^{-1}x) - \frac{m}{2}\log(2\pi) - \frac{1}{2}|\Sigma| \tag{2.2.4}$$

$$= -\frac{1}{2}\Big(x^T \Sigma^{-1}x + \mu_i^T \Sigma^{-1}\mu_i - 2\mu_i^T \Sigma^{-1}x\Big) - \frac{m}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma|. \tag{2.2.5}$$

Similarly,

$$\log[P(X|Y=k)] = -\frac{1}{2}\Big(x^T \Sigma^{-1}x + \mu_k^T \Sigma^{-1}\mu_k - 2\mu_k^T \Sigma^{-1}x\Big) - \frac{m}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma|. \tag{2.2.6}$$

Taking the ratios of these posteriors and taking the logarithm of the resultant, implies:

$$\log\Big[\frac{P(Y=i|X)}{P(Y=k|X)}\Big] = \log\Big[\frac{P(X|Y=i)\cdot P(Y=i)}{P(X|Y=k)\cdot P(Y=k)}\Big] \tag{2.2.7}$$

$$= \log P(X|Y=i) + \log P(Y=i) - \log P(X|Y=k) - \log P(Y=k). \tag{2.2.8}$$

Let the prior probabilities; $P(Y=i) = r_i$ and $P(Y=k) = r_k$.

Substituting 2.2.4 and 2.2.6 into 2.2.7, implies:

$$\log\Big[\frac{P(Y=i|X)}{P(Y=k|X)}\Big] = -\frac{1}{2}\Big(2\mu_i^T \Sigma^{-1}x - 2\mu_k^T \Sigma^{-1}\mu_i - \mu_k^T \Sigma^{-1}\mu_k\Big) + \log r_i - \log r_k. \tag{2.2.9}$$

Let the constant terms be c_0. Then equation 2.2.9 reduces to:

$$\log\Big[\frac{P(Y=i|X)}{P(Y=k|X)}\Big] = -\Big(\mu_i^T \Sigma^{-1} - \mu_k^T \Sigma^{-1}\Big)x + c_0 \tag{2.2.10}$$

$$= c^T x + c_0, \tag{2.2.11}$$

where c^T is a vector. Observe that 2.2.11 is a linear function. In the derivation of the function, the assumption was that the two classes i and k had the same covariance matrix Σ. This approach is referred to as the **Linear Discriminant Analysis (LDA)**. The discriminant rule of this linear function for future observation is to assign x to class i if $c^T x + c_0 \geq \lambda$ and to class k otherwise, where λ is a chosen cut-off value.

Again, suppose the covariances of the two classes are unequal; thus, the covariance of $P(X|Y=i)$ is Σ_i and $P(X|y=k)$ is Σ_k. Then combining 2.2.4, 2.2.6 and 2.2.7, it follows that:

$$\begin{aligned}\log\left[\frac{P(X|Y=i)P(Y=i)}{P(X|Y=k)P(Y=k)}\right] =& -\frac{1}{2}\Big(x^T(\Sigma_i^{-1}x - x^T\Sigma_k^{-1}x - 2\mu_i^T\Sigma_i^{-1}x + 2\mu_k^T\Sigma_k^{-1}x + \mu_i^T\Sigma_i^{-1}\mu_i - \mu_k^T\Sigma_k^{-1}\mu_k)\Big) - \\ & \log|\Sigma_i| + \log|\Sigma_k| + \log(r_i) + \log(r_k) \\ =& -\frac{1}{2}\Big(x^T(\Sigma_i^{-1} - \Sigma_k^{-1})x\Big) - \Big(\mu_i^T\Sigma_i^{-1} - \mu_k^T\Sigma_k^{-1}\Big)x + c_0 \\ =& x^T c_1 x - c_2^T x + c_0,\end{aligned}$$

where :

$$c_1 = \frac{1}{2}(\Sigma_k^{-1} - \Sigma_i^{-1}),$$

$$c_2 = (\Sigma_i^{-1}\mu_i - \Sigma_k^{-1}\mu_k),$$

$$c_0 = \frac{1}{2}\log\left(\frac{|\Sigma_k|}{|\Sigma_i|}\right) + \frac{1}{2}\Big(\mu_k^T\Sigma_k^{-1}\mu_k - \mu_i^T\Sigma_i^{-1}\mu_i\Big).$$

The above derivation using an unequal covariance assumption is what is referred to as **Quadratic Discriminant Analysis (QDA)**. The discriminant rule of this quadratic function for future observation is: for a chosen cut-off value λ, assign x to the class i if $x^T c_1 x - c_2^T x + c_0 \geq \lambda$. On the other hand, assign x to the class k if $x^T c_1 x - c_2^T x + c_0 < \lambda$.

In general, given a discriminant variable X, with $f_i(x)$ and $f_k(x)$ as the probability density functions for the first and second classes respectively, the classification rule is to:

a) assign an observation to class i if:

$$\frac{f_i(x)}{f_k(x)} \geq \left[\frac{C(i/k)}{C(k/i)}\right]\left[\frac{P_k}{P_i}\right].$$

b) assign an observation to class k if:

$$\frac{f_i(x)}{f_k(x)} < \left[\frac{C(i/k)}{C(k/i)}\right]\left[\frac{P_k}{P_i}\right]$$

,

where $c(i/j)$ is the misclassification cost of an observation for class j to i, and p_i, p_k are the prior probabilities of an observation belonging to class i and j respectively.

2.3 Logit Model

In regression model, the idea is to describe the relationship between a set of predictors (independent variables) and the response variable. Logistic regression model is a special type of regression model in which the response variable is binary in nature (i.e. categorical). The logistic function which models

the probability of belonging to one of the two classes is given by:

$$\pi(x) = P(Y = 1|X = x) = \frac{\exp(\psi_0 + \psi_1 x_1 + \psi_2 x_2 + \cdots + \psi_n x_n)}{1 + \exp(\psi_0 + \psi_1 x_1 + \psi_2 x_2 + \cdots + \psi_n x_n)} \tag{2.3.1}$$

$$= \frac{\exp(\psi_0 + \sum_{k=1}^{n})}{1 + \exp(\psi_0 + \sum_{k=1}^{n})} \tag{2.3.2}$$

$$= \frac{1}{1 + \exp\{-(\psi_0 + \sum_{k=1}^{n})\}}, \tag{2.3.3}$$

where ψ_0 is the intercept and ψ_k for $k = 1, 2, \cdots, n$ are parameters to be estimated. This model tests the relationship between the classes of the response variable and the independent variables (predictors) by estimating the parameters of equation 2.3.1. The ratio of the probability of being in one of the classes to its complement gives what is termed as the *odd ratio*. Thus $\frac{\pi(x)}{1-\pi(x)}$ is referred to as the odd ratio. Taking the log of this ratio, implies:

$$\log\left[\frac{P(Y = 1|X = x)}{P(Y = 0|X = x)}\right] = \log\left(\frac{\pi(x)}{1 - \pi(x)}\right) \tag{2.3.4}$$

$$= \psi_0 + \psi_1 x_1 + \psi_2 x_2 + \cdots + \psi_n x_n. \tag{2.3.5}$$

Equation 2.3.4 is referred to as the *logit*; which transforms the odd ratio into a linear function as shown in equation 2.3.5. The decision boundary is based on these posterior class probabilities. The rule for classifying an observation as belonging to one of the classes is as follows:

assign observations to class 1 if $\log\left[\frac{P(Y=1|X=x)}{P(Y=0|X=x)} > 0\right]$ and to class 0 otherwise.

2.4 k-Nearest Neighbourhood (k-NN)

One of the simplest machine learning algorithms is the nearest neighbour technique which classifies objects based on training examples. Unlike linear and quadratic discriminant analysis, this algorithm does not rely on prior probabilities. Let $X = (x_1, x_2, \cdots, x_m)$ be and m-dimensional vector of an input data. Let also; y be a categorical dependent variable whose values are dependent on the m observations in X. Suppose x' is a point whose class is not known; the task then is to identify the class of x' using k closest neighbours (for k positive integer) in training example. The k-NN algorithm follows the following procedure:

a) Choose a positive integer k.

b) Calculate the distance $d(x, x')$; where $d = \sqrt{\sum_{j=1}^{m}(x_i - x_i')^2}$.

c) Order the computed distances; starting with the shortest.

d) Take the first k distances from the order and find those k-points that corresponds to these distances.

e) Suppose k_j is the number of points belonging to the j^{th} class such that k_j is among the k-points.

f) For classes $j = 1, 2, \cdots, c$, classify x' as belonging to class j if $k_j > k_i \quad \forall j \neq i$.

Shortly put, the k-NN algorithm assigns an object as belonging to a particular class based on the majority votes of its k-nearest members.

3. Data and Methods

3.1 The Data

This section presents the source of the data for this study, the transformations made on the dataset and the variable selection techniques used in this study.

3.1.1 Data Collection. As indicated earlier, this study considers the direction of the return on standard and poor 500 index which is the weighted index of the 500 largest companies in the US stock market.

In research, the availability and the cost of obtaining data are some of the essential factors to consider from the onset. The data used for this study was obtained from yahoo finance. It contains historical information on the opening prices, low prices, high prices, volume traded, closing prices and the adjusted closing prices. The data taken was from January 1, 1999 to December 31, 2008. Data on the six world major indices and the five biggest companies in the S&P 500 index which are believed to have significant impact on return on the index was taken from the same website; for the same time period. Below is the tabulation of these indices/stocks and the market in which the are traded:

Table 3.1: stocks/indices

Stock/Index	Stock Market Traded
Exxol oil (XOM)	New York Stock Exchange (NYSE)
General Electricity (GE)	NYSE
Microsoft (MSFT)	NASDAQ
Procter and Gambler (PG)	NYSE
Johnson and Johnson (JNJ)	NYSE
Dow Jones Industrial Average (DJI)	NYSE
NASDAQ Composite (IXIC)	NASDAQ
Hang Seng Index (HIS)	Hong Kong Stock Market
CAC 40 index (FCHI)	French Stock Exchange
Financial Times Stock Exchange (FTSE)	London Stock Exchange
German Stock index (GDAXI)	Frankfurt Stock Exchange

3.1.2 Data preprocessing. Data preprocessing is the process of transforming and cleaning a dataset into a uniformly distributed set, to improve the data quality and hence improve the predictive performance (Wang et al., 2005). The predictive performance of a model is highly dependent on the data preprocessing stage (Wang et al., 2005). After collecting the data on the dependent variables and the predictors, fitting a model with them directly in their raw form may result in under-performance or over-performance of the model in predicting the desired output. As this study considers the returns on the S&P 500, the returns were computed using the adjusted closing prices. The approach used is the logarithmic approach given by the formula below:

$$R_t = \log(P_t) - \log(P_{t-1}) \tag{3.1.1}$$

$$= \log\left[\frac{P_t}{P_{t-1}}\right], \tag{3.1.2}$$

where R_t is the $\log$-return on the index at time t and P_t is the adjusted closing price at time t. After computing the returns, since the study is focused on the direction of the return instead of the actual

returns, a column is created to record the direction (up/down). Within this column, 1 is assigned when $R_t > 0$ and 0 otherwise. Thus, we assign 1 when positive return is realized and 0 otherwise. In order to have the same number of rows within the data, the first rows of the opening prices column of the various stocks/indices were deleted.

3.1.3 Partitioning of data. For effective and perfect evaluation of the performance of a forecasting algorithm, it is a common practice to partition the dataset into training and testing distinct sets. Mostly practised, the largest set which is used to recognize the patterns in the data is the training set, whiles the testing set serves as a means of evaluating the general ability of the trained algorithm to predict the underlying phenomenon. In this essay, after preprocessing the data, the number of days (observations) used was 2471. With this, 75%, representing 1853 days was taken as the training set, whiles the remaining 25% representing 618 days was taken as the testing set.

3.1.4 Data Normalization. Normalization is the process of data organization with the motive of eliminating redundant data in the dataset and also to overcome data anomalies. There are many methods for data normalization; which includes quantile normalization, $z-score$ approach, etc. (Normalization). This study however, considers the so called $min-max$ normalization approach which linearly transforms the original data to the range $(0, 1)$. This idea is to scale the data to a common range to avoid the influence of outliers in the data in the predictions. Mathematically, this approach is applied as follows: Let X be a data set, X_{min} and X_{max} be the minimum and maximum values of X respectively. Then the formula for normalizing (scaling) X is given by Normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1.3}$$

In $min - max$ normalization, the relationships among the values of the original data is preserved after computing.

3.1.5 Potential Influential Features. The potential features extracted from the various indices/stocks tabulated in Table 3.1 to considered in predicting the direction of the S&P 500 is tabulated below.

Table 3.2: Potential Features

Feature	Explanation
GSPC.Open (t)	Open price of the S&P 500 index in day t
GSPC.Volume (t-1)	Volume of S&P 500 traded in day t-1
GSPC.Adj. (t-1)	Adjusted closing price of S&P 500 for day t-1
JNJ.Open (t)	Open price of Johnson and Johnson company in day t
XOM.Open (t)	Open price of Exxon mobil stock in day t
GE.Open (t)	Open price of General Electric stock in day t
FCHI.Open (t)	Open price of CAC 40 index in day t
GDAXI.Open (t)	Open price of DAX index in day t
FTSE.Open (t)	Open price of FTSE 100 index in day t
IXIC.OPen (t)	Open price of NASDAQ composite index company in day t
MSFT.Open (t)	Open price of Microsoft stock in day t
DJI.Open (t)	Open price of Down Jones index in day t
HIS.Adj. (t-1)	Adjusted closing price of Hang Seng index
PG.Open (t)	Open price Procter and Gambler stock in day t

In Table 3.2, we observe that, the open prices of the various indices/stocks are considered as predictors

except the Hang Seng index, where we consider the adjusted closing price. This is as a result of its market being closed before that of the New York Stock Exchange .

3.1.6 Correlations and Variables selection. In choosing the input predictor variables from the potential feaures, a Pearson correlation test is perform to determine the relationship between the variables. Two variables are said to be correlated when a change in the value of one affects the other. This can be in the positive direction or otherwise. Denoted by r, and always within the range $[-1, 1]$, linear correlation measures the strength of the linear relationship between different variables. An r-value of 1 implies perfectly positively correlated, -1 shows perfect negative linear correlation, whiles a value of zero (0) implies that variables are uncorrelated. Preferably, correlation is deemed significant when its coefficient is greater than 0.5 in absolute terms (thus, when $|r| > 0.5$).

Table 3.3: Correlation Matrix

	ud	SP.V	SP.O	XO.O	GE.O	MS.O	PG.O	JNJ.O	DJI.O	IX.O	H.Adj	FC.O	FT.O	GD.O
ud	1													
SP.V	0.00	1												
SP.O	-0.02	0.10	1											
XO.O	-0.02	0.33	0.81	1										
GE.O	-0.01	-0.39	0.43	0.39	1									
MS.O	-0.04	-0.47	0.22	0.23	0.79	1								
PG.O	-0.02	-0.34	-0.28	-0.30	0.35	0.40	1							
JNJ.O	-0.02	-0.25	0.55	0.66	0.68	0.64	0.08	1						
DJI.O	-0.01	0.36	0.87	0.70	0.09	-0.16	-0.33	0.20	1					
IX.O	-0.01	-0.12	0.80	0.67	0.68	0.45	-0.08	0.65	0.49	1				
H.Adj	0.04	-0.09	0.37	0.25	0.55	0.44	0.16	0.26	0.25	0.27	1			
FC.O	-0.02	0.02	0.90	0.77	0.37	0.26	-0.33	0.57	0.71	0.81	0.10	1		
FT.O	-0.02	0.03	0.94	0.84	0.54	0.40	-0.21	0.67	0.75	0.77	0.41	0.91	1	
GD.O	-0.02	0.30	0.91	0.90	0.31	0.18	-0.33	0.47	0.82	0.74	0.15	0.93	0.90	1

In Table 3.3, we observe that the leading diagonal has one in all the entries. This makes sense according to the interpretation given earlier in this section; because it shows that each variable is perfectly linearly correlated with itself. The benchmark for taking the correlation between variables as being significant is 0.5. Even though there exists significant correlation between some of the predictors, the idea of this technique is to observe the linear correlation between the response variable which is the direction of the return on S&P 500 (up/down) labelled 'ud' and the predictor variables. We observe from the table that no such correlations exist. Different variable reduction technique is then used to reduce the number of predictors since each variable included in the model comes with a cost (Carvalho, Accessed May 2016). This reduction is necessitated by the fact that big models stands the risk of over-fitting and hence models should be built in such a way that they account for the trade-off between the data fitting and the model uncertainties. The variable selection procedure used in this study is the stepwise approach which is more useful in the sense that it combines the usefulness of the forward selection and the backward elimination techniques. In the forward selection, the simplest model is first built and suitable variables are added one at a time until the best model is achieved. On the other hand, the backward elimination method starts with a general model and subsequently drops variables one at a time until the best model is achieved. The stepwise approach which is the combination of the forward and backward selection methods, starts with an empty set of variables and subsequently adds or drops a variable at any point during the variable search process. Thus, in stepwise selection, a variable can be added or dropped. A summary of the stepwise selection process performed in R is as follows:

```
Step:  AIC=3385.31
ud ~ GSPC.Volume + GSPC.Open + MSFT.Open + DJI.Open + FCHI.Open +
    FTSE.Open + GDAXI.Open

              Df Deviance    AIC
<none>             3369.3 3385.3
+ GE.Open      1   3368.9 3386.9
+ PG.Open      1   3369.0 3387.0
+ HIS.Adj      1   3369.2 3387.2
+ IXIC.Open    1   3369.2 3387.2
+ XOM.Open     1   3369.3 3387.3
+ JNJ.Open     1   3369.3 3387.3
- GSPC.Open    1   3374.6 3388.6
- DJI.Open     1   3377.7 3391.7
- MSFT.Open    1   3378.8 3392.8
- GSPC.Volume  1   3379.0 3393.0
- FTSE.Open    1   3379.9 3393.9
- FCHI.Open    1   3381.0 3395.0
- GDAXI.Open   1   3389.8 3403.8
```

Figure 3.1: final stage of stepwise selection procedure

We observe from the output from R, shown in Figure 3.1 that, in the last step of the process, the Akaike Information Criterion (AIC) which gives a measurement of the relative quality of different models using different variables is 3385.31. The variables in the model that achieved this AIC are the volume of S&P 500 traded in day t-1, open prices of S&P 500, MSFT, DJI, FCHI, FTSE and GDAXI. This AIC is used as a selection criteria and the rule is to choose the model with the lowest AIC. In the Figure, we observe that, when the opening prices of general electricity (GE) is added to the model, the AIC increases to 3386.9, which is greater than the previous AIC value of 3385.31, hence not a good model; likewise the open prices of PG, IXIC, XOM, JNJ and the closing price of HIS which all results in high AIC when included in the model. On the other hand, we also observe that, when any of the following variables: the volume of S&P 500 traded in day t-1, open prices of S&P 500, MSFT, DJI, FCHI, FTSE and GDAXI is excluded from the model, it results to a high AIC. Hence based on the AIC analysis, the best predictors are the volume of S&P 500 traded in day t-1, open prices of S&P 500, MSFT, DJI, FCHI, FTSE and GDAXI. Below, we present a summary of the initial model and the final model fitted based on the achieved AIC's, together with the analysis of variance table for the rejected variables.

```
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
ud ~ GSPC.Volume + GSPC.Open + XOM.Open + GE.Open + MSFT.Open +
    PG.Open + JNJ.Open + DJI.Open + IXIC.Open + HIS.Adj + FCHI.Open +
    FTSE.Open + GDAXI.Open

Final Model:
ud ~ GSPC.Volume + GSPC.Open + MSFT.Open + DJI.Open + FCHI.Open +
    FTSE.Open + GDAXI.Open
```

Figure 3.2: Intial and final model fitted in the stepwise procedure

Table 3.4: Anova table for stepwise process showing eliminated variables

	Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
1				2457.00	3368.51	3396.51
2	- JNJ.Open	1.00	0.01	2458.00	3368.51	3394.51
3	- IXIC.Open	1.00	0.05	2459.00	3368.57	3392.57
4	- XOM.Open	1.00	0.13	2460.00	3368.69	3390.69
5	- HIS.Adj	1.00	0.13	2461.00	3368.82	3388.82
6	- PG.Open	1.00	0.12	2462.00	3368.94	3386.94
7	- GE.Open	1.00	0.37	2463.00	3369.31	3385.31

The stepwise selection processes as presented in figure 3.2 and table 3.4 above started by setting a model with all the thirteen potential features and analyzed the contributions of each variable to the model by considering the AIC's. The procedure dropped six of the variables based on the high level of AIC's when those variables included in the model and used the remaining seven; which were eventually considered as the final predictors in this study. Hence the final input variables used in predicting the directional movement of the returns on the S&P 500 index were:

- volume of S& P 500 traded in day t-1
- open price of S& P 500 traded in day t
- open price Microsoft stock traded in day t
- open price Dow Jones index traded in day t
- open price CAC 40 index traded in day t
- open price of FTSE 100 index traded in day t
- open price of DAX index traded in day t

3.2 Neural Network training

The package used for training the NN was **neuralnet** package in R, which was developed by Fritsch and Günther in 2008. This package has flexible inbuilt functions which help in training feed-forward neural networks. It has the flexibility of deciding on the number of hidden layers and hidden neurons, the rate of learning, and the cost function. Another advantage of the **neuralnet** package is its ability to give functions which help to visualize the results emanating from the fitted network. This study uses multi-layer perceptron which to our advantage is the focus of of the **neuralnet** package. In building the network, there is a general formula/logic that needs to be followed in order to achieve the desired results. This is explained below.

a). **Data:** all variables to be included in the model should be in a dataframe.

b). **Hidden:** this is where the number of hidden layers and hidden neurons in each layer are specified. For instance, **hidden** = $c(2,3,1)$ is to say, a network with three hidden layers, with the first, second and third having two, three and one neurons respectively. In the package, the default is one hidden layer, with one hidden neuron. This study however, uses one hidden layer with eight

neurons. There is no specific rule for choosing the number of hidden layers and neurons in each layer.

c). **Threshold:**this is a real-valued number which is the stopping criteria for the partial derivatives of the error functions. It has 0.01 as its default. In this study, upon making different explorations, the best results in terms of accuracy in prediction was achieved at a threshold of 0.08.

d). **Other relevant functions:** other important functions to consider are the error function and the output function. This study uses the Sum of Squared Error (SSE) function which is the default in the neuralnet package as its error function. In addition, the 'linear.output' function is set to false since the interest is not in producing a linear output but a classification output.

3.3 Trainining of other classification algorithms

The training of the logistic model is done in 'R', using the generalized linear model (glm) function. Since the focus is on predicting the ups and down movement of the returns, the **binomial family** of the glm function with the **logit** as its link. Also, in training the linear and quadratic analysis models, **lda** and **qda** functions which are found in a package called **MASS**, is used respectively. On the other hand, **KNN** training is done with **knn** function which is also found in a package called **class**.

3.4 Model performance evaluations

In evaluating the performance of the models, a package in **R**, called **caret** is used to create a confusion matrix, which assesses the general performance of the models. Suppose a table indicating the various predictions made by our model is as follows:

Table 3.5: Cross-tabulation of predicted and actual values

	Reference	
Predicted	**down**	**up**
down	α	β
up	θ	ψ

where $\alpha, \beta, \theta, \psi \in \mathbb{N}$

Then, the following model evaluation criteria can be produced from the cross-tabulation 3.5 using a confusion matrix function:

a) **accuracy**: this measures the percentage of the correct predictions that were made. It is given by the formula:

$$\frac{\alpha + \psi}{\alpha + \psi + \beta + \theta}. \tag{3.4.1}$$

b) **sensitivity:** this is the measurement of the actual positives which were correctly predicted as positives. Its computation is made with the formula:

$$\frac{\alpha}{\alpha + \theta}. \tag{3.4.2}$$

c) **specificity:** this can be thought of as the reverse of the sensitivity measure in the sense that, it measures the proportion of the actual negatives which were correctly classified as such. It is also computed using the formula:

$$\frac{\psi}{\beta + \psi}. \tag{3.4.3}$$

d) **prevalence:** it is the measurement of the rate of the "positive" class in the data. That is, the prevalence of the "positive" class within the data. Mathematically, prevalence rate is calculated with the formula below.

$$\frac{\alpha + \theta}{\alpha + \psi + \beta + \theta}. \tag{3.4.4}$$

e) **detection rate:** this also measures the rate at which the 'postive' class is correctly identified. Its computation is done using the formula:

$$\frac{\alpha}{\alpha + \psi + \beta + \theta}. \tag{3.4.5}$$

f) **detection prevalence:** it is the measurement of the rate of prevalence of the 'positive' class which is given by the formula:

$$\frac{\alpha + \beta}{\alpha + \psi + \beta + \theta}. \tag{3.4.6}$$

g) **no information rate:** this is the measurement of the largest or dominating class in the data. For instance, there were more observations in the 'positive' class than the negative class.

h) **positive predicted values:** this evaluation criteria measures the number observations in the 'positive' class that were correctly predicted expressed as a fraction of the total number of observations in the class. That is, using the formula below:

$$\frac{\alpha}{\alpha + \beta}. \tag{3.4.7}$$

i) **negative predicted rate:** as the name suggests, it is the opposite of the positive predicted values. That is, it measures the proportion of the negative observations which were correctly predicted. It is also computed using the formula below:

$$\frac{\theta}{\theta + \psi}. \tag{3.4.8}$$

There are other statistics that are produced in the output of the confusion matrix. However, in this study, not much attention will be given to those since the overall performance of a model would be measured by the level of accuracy (i.e. the rate of accuracy) in predicting the ups and downs of the return. A better model is the one that achieves relatively high accuracy.

4. Results and Analysis

In this chapter, we present and discuss the results from the various analysis and make inferences based on the results.

4.1 Descriptive Statistics

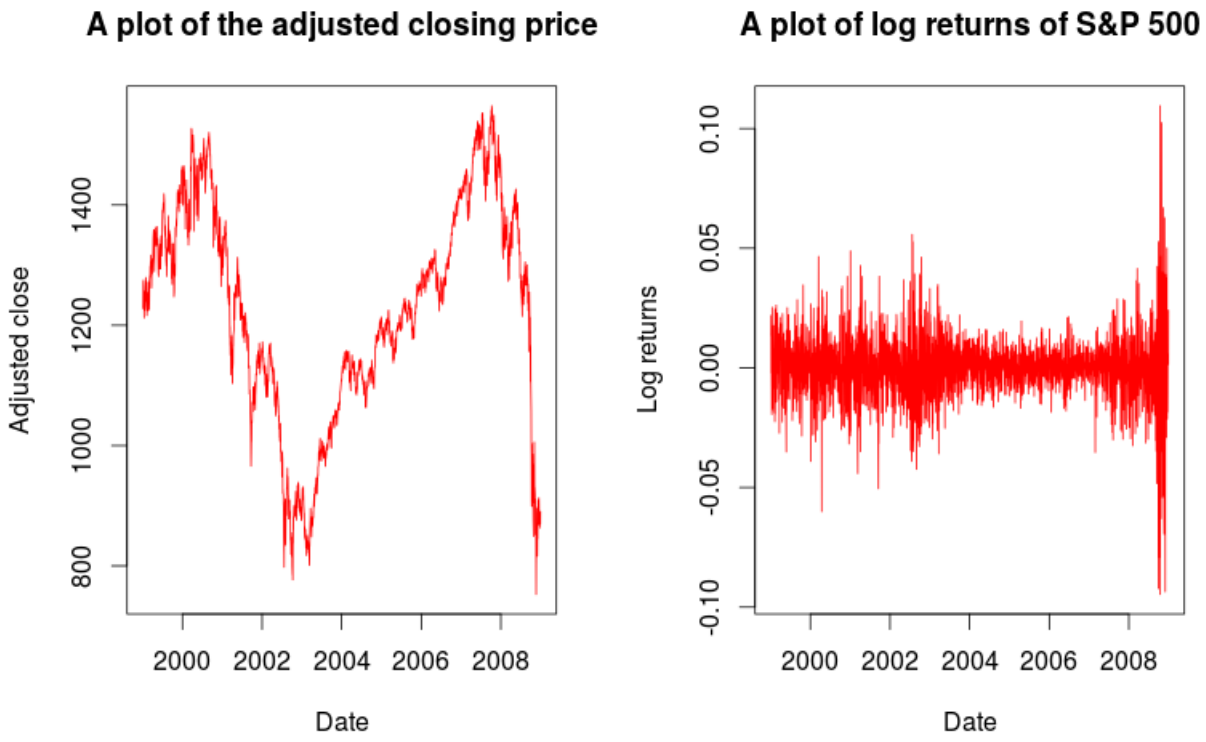
This section provides a statistical description of the behaviour of the various stocks/indices used in this essay. Below is a brief insight of the behaviour of the adjusted closing prices.

Table 4.1: Quantitative description of the various stocks and stock indices

Variable	Mean	Variance	St. Dev.	Min	Max
GSPC.Adjusted	1,224.99	32868.22	181.30	752.44	1,565.15
XOM.Adjusted	40.12	256.34	16.01	21.58	76.92
GE.Adjusted	23.73	20.75	4.56	9.74	36.64
MSFT.Adjusted	22.23	20.45	4.52	14.37	41.26
PG.Adjusted	36.59	100.02	10.00	17.59	57.54
JNJ.Adjusted	40.50	54.93	7.41	22.86	56.71
DJI.Adjusted	10,631.87	1,760,187.53	1,326.72	7,286.27	14,164.53
IXIC.Adjusted	2,281.29	455,805.30	675.13	1,114.11	5,048.62
HIS.Adjusted	2.45	2.88	1.70	0.11	9.85
FCHI.Adjusted	4,584.44	1,002,032.79	1,001.02	2,403.04	6,856.76
FTSE.Adjusted	5,421.28	761,959.88	872.90	3,287.00	6,930.20
GDAXI.Adjusted	5,364.33	2,018,997.48	1,420.91	2,202.96	8,105.69

In table 4.1, it can be deduced that, the Dow Jones Industrial index (DJI), which is made up of the "30 most influential companies in the USA", recorded the highest mean adjusted closing price, with the highest variance as well, making it highly volatile. Although the DJI relatively has a high variance/standard deviation, it has the lowest risk (thus, the ratio of the standard deviation to the mean). Moreover, in terms of volatility, the Hang Seng index which is a Hong Kong stock market index, has the lowest due to its low standard deviation. However, it has the highest risk relatively.

Figure 4.1: Graphical description S&P 500



(a) (b)

Figure 4.1a is a plot of the adjusted closing price S&P 500 with date on the horizontal axis and the adjusted closing prices on the vertical axis. Figure 4.3 on the other hand is the a pictorial view of the log-returns on S&P 500 over the time period considered for this essay.

4.2 ANN results

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 163 131
         1 111 213

               Accuracy : 0.6084142
                 95% CI : (0.5686789, 0.6471085)
    No Information Rate : 0.5566343
    P-Value [Acc > NIR] : 0.005216982

                  Kappa : 0.2124984
 Mcnemar's Test P-Value : 0.221947377

            Sensitivity : 0.5948905
            Specificity : 0.6191860
         Pos Pred Value : 0.5544218
         Neg Pred Value : 0.6574074
             Prevalence : 0.4433657
         Detection Rate : 0.2637540
   Detection Prevalence : 0.4757282
      Balanced Accuracy : 0.6070383

       'Positive' Class : 0
```

Figure 4.2: A confusion matrix showing a NN Classification performance

Figure 4.2, displays a confusion matrix showing the overall performance of the Artificial Neural Network. From the matrix, we observe that, on 163 times, our model predicted 'down' and it was actually down whiles it predicted the upward movements correctly 213 times. The model however, wrongly predicted the movement 242 times. Summing up, the model's prediction accuracy is approximately 61 percent. We are also 95% confident that the accuracy of the model will be in the range $(0.5687, 0.6471)$. Also, although ANN model doesn't seem to be doing well as a classifier; with its 61 percent hit rate, but it achieved approximately 66 percent accuracy in classifying the 'negative' cases (thus, instances where returns truly went up) as shown in the 'negative predicted value' of the displayed results. This in effect gives some level of confidence in the instances when the model will predict that the returns will go up.

4.3 Logistic results

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 109 129
         1 173 207

               Accuracy : 0.5113269
                 95% CI : (0.4711338, 0.5514113)
    No Information Rate : 0.5436893
    P-Value [Acc > NIR] : 0.95093053

                  Kappa : 0.0026292
 Mcnemar's Test P-Value : 0.01334707

            Sensitivity : 0.3865248
            Specificity : 0.6160714
         Pos Pred Value : 0.4579832
         Neg Pred Value : 0.5447368
             Prevalence : 0.4563107
         Detection Rate : 0.1763754
   Detection Prevalence : 0.3851133
      Balanced Accuracy : 0.5012981

       'Positive' Class : 0
```

Figure 4.3: A confusion matrix showing the performance of logistic model

In figure 4.3, we present a summary of the performance of the logistic model. We observe that, the model was correct in predicting that returns will go down whiles it actually went down on 109 occasions. Also, it correctly predicted the upward movements 207 time. On the negative side, the model misclassified the direction of the returns on 302 instances. This limits the overall accuracy of the model in the classification to approximately 51 percent.

4.4 LDA results

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 109 129
         1 173 207

               Accuracy : 0.5113269
                 95% CI : (0.4711338, 0.5514113)
    No Information Rate : 0.5436893
    P-Value [Acc > NIR] : 0.95093053

                  Kappa : 0.0026292
 Mcnemar's Test P-Value : 0.01334707

            Sensitivity : 0.3865248
            Specificity : 0.6160714
         Pos Pred Value : 0.4579832
         Neg Pred Value : 0.5447368
             Prevalence : 0.4563107
         Detection Rate : 0.1763754
   Detection Prevalence : 0.3851133
      Balanced Accuracy : 0.5012981

       'Positive' Class : 0
```

Figure 4.4: A confusion matrix showing the performance of logistic model

Figure 4.4 displays the results from the linear discriminant analysis. The model was successful in classifying the downward returns on 109 different instances, whereas it correctly classified the up returns 207 times. The model was unsuccessful on 302 occasions in the classification. Just as the logistic model, LDA approximately had 51 percent overall prediction accuracy.

4.5 QDA results

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 101 136
         1 181 200

               Accuracy : 0.487055
                 95% CI : (0.4469815, 0.5272528)
    No Information Rate : 0.5436893
    P-Value [Acc > NIR] : 0.9978907

                  Kappa : -0.0472113
 Mcnemar's Test P-Value : 0.0134628

            Sensitivity : 0.3581560
            Specificity : 0.5952381
         Pos Pred Value : 0.4261603
         Neg Pred Value : 0.5249344
             Prevalence : 0.4563107
         Detection Rate : 0.1634304
   Detection Prevalence : 0.3834951
      Balanced Accuracy : 0.4766971

       'Positive' Class : 0
```

Figure 4.5: A confusion matrix showing the performance of quadratic discriminant analysis

In 4.5, we analyse the output of the quadratic discriminant with a confusion matrix. In the matrix, we deduce that the model performed relatively low in classifying the downward directions, compared to ANN, logistic and LDA presented earlier. In all, it correctly predicted the the returns on 302 different instances. The overall hit rate of the QDA is approximately 49 percent which in a sense is low; relative to the other models presented earlier.

4.6 k-NN results

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 122 153
         1 160 183

               Accuracy : 0.4935275
                 95% CI : (0.453413, 0.5337041)
    No Information Rate : 0.5436893
    P-Value [Acc > NIR] : 0.9944483

                  Kappa : -0.0227785
 Mcnemar's Test P-Value : 0.7345042

            Sensitivity : 0.4326241
            Specificity : 0.5446429
         Pos Pred Value : 0.4436364
         Neg Pred Value : 0.5335277
             Prevalence : 0.4563107
         Detection Rate : 0.1974110
   Detection Prevalence : 0.4449838
      Balanced Accuracy : 0.4886335

       'Positive' Class : 0
```

Figure 4.6: A confusion matrix showing the performance of KNN

The results and analysis of the k-Nearest Neighbourhood algorithm is presented in figure 4.6. In the figure, we observe that though the k-NN algorithm seems simple, its overall accuracy in classifying the directions of the returns is relatively higher comparing to that of QDA. Also, it performed better in classifying the downward returns than that of logistic model, LDA and QDA. The overall accuracy of the k-NN algorithm is approximately 49.4 percent.

4.7 Summary of Results

Table 4.2: Performance summary of the various algorithms

Algorithm	Number correctly classified	Accuracy
ANN	376	60.8%
Logistic	316	51.1%
LDA	316	51.1%
QDA	301	48.7%
KNN	305	49.4%

Table 4.6 provides a summary of classification accuracies of the various algorithms, using opening prices of the selected predictors at day t to predict the direction of the return at the end of the same trading day t. Which in a sense, is predicting a time-step ahead.

4.8 Further Analysis

In an attempt to predict the movement (direction) of the return on S & P 500 at day t, using the returns on DJI, FCHI, FTCH, FTSE, GDAXI, GE, IXIC, HIS, JNJ and MSFT at the same day t as the input variables; that is to say, analysing how much influence the various stocks/indices have on the S&P 500 index, the following results were obtained for the various algorithms.

Table 4.3: Performance assessment using returns as predictors

Algorithm	Number correctly classified	Accuracy
ANN	592	95.8%
Logistic	585	94.7%
LDA	569	92.1%
QDA	544	88.0%
KNN	618	91.9%

In table 4.3, we observe that ANN performed relatively better than the other algorithms; achieving 95 percent accuracy in the classifications. Although majority of the algorithms achieved more than 90 percent accuracy, using returns on other stocks/indices at a time 't' to predict the returns on the S&P 500 index at the same time 't' tends to be of less relevance to investors. Investors are mostly interested in knowing the future based on information at hand so as to know how to hedge against possible losses. Hence achieving high predictive accuracy with a normal regression approach tends to be of less interest to investors.

4.9 Discussions

- We have presented in this study, ways of predicting the direction of the returns on S&P 500 index. In an attempt to predict the return on the index at the end of a trading day, given the opening prices of some stocks/indices, we deduced that the highest accuracy achieved was approximately 61 percent. With this, an investor can be confident that the trained model can predict the returns with 61 percent accuracy.
- In addition, the results of all the various algorithms considered in this essay show that, the models performed better in classifying the instances where the returns went 'up' than the 'down' cases. To this effect, the investor can have some level of confidence whenever the models predict an 'upward' trend of the returns.
- He/she (the investor) can develop a daily trading strategy depending on the prevailing market conditions. The table below gives a summary of the possible strategies that the investor can develop under different conditions.

Table 4.4: Daily trading strategies under different conditions

Day	Condition	Trading action
t-1		buy a share of stock at the end of the day (or keep if in ones possession).
t	if returns are predicted to be positive	hold stock throughout day (thus, hold until next trading day).
t	if returns are predicted to be negative	a) sell at the beginning of the trading day and buy it back at the end of the day. b) wait until price reaches the purchased price and sell; else hold until the end of the day to prevent possible loss.

Suppose the investor starts with one share of stock which worth $\$1,244.78$ and invests it over the period of ten years as considered in this study. Then, analysis of the total returns that the investor would probably realize based on the strategies developed in Table 4.4 is as tabulated in Table 4.5 below. It should be noted that, this analysis does not take into account **frictional cost** which measures both the direct and indirect costs involved in financial transactions. That is to say, the analysis does not take into account some costs involved in the sales of the stock such as tax liabilities, brokers commission, exchange fees and other indirect costs.

Table 4.5: Profit/Loss accounting of the various strategies developed (for the ten- year period)

Purchased price	Selling price	Return realized	Actual value of return ($)
Closing price of day t-1	Open price of day t, for open price $\geq$ purchased price	Open price of day t - purchased price	$13,334.03$
Closing price of day t-1	purchased price	purchased price - closing price of day t	940.59
Closing price of day t-1	closing price of day t	closing price of day t - closing price of day t-1	-381.7

Table 4.5 gives the various possibilities of the returns that the investor would realize on the share of stock over the ten-year period. We observe from the table that, the investor would be making profit, trading with our developed strategy as shown in the second and third rows of the table, where he/she trades only when conditions are predicted favourable. On the other hand, the return on the stock over the ten-year period would be negative if the investor is to trade without any knowledge on what would happen at the end of the trading day; as shown in the last row of the table.

Summarizing, the profit/loss accounting of the various trading strategies is as tabulated below:

Table 4.6: Summary of profit/loss accounting

strategy	**profit/loss ($)**
Buying a share for $1244.78 and holding it for the ten-year period.	-381.7
strategy (a): sell at the beginning of trading day and buy back at the end of the day; given that model predicts negative returns.	13,334.03
strategy (b): wait until price reaches the purchased price and sell; else hold till the end of the day (for days when returns are predicted negative).	940.59

As demonstrated in Table 4.6, the instance where one buys a share of stock and keeps it for the ten-year period results in loss on the share (see the second row of Table 4.6). This is a situation that investors always want to avoid. Hence a worry to an investor. On the other hand, having some knowledge on the likely behaviour of the market, that is, using the two strategies developed resulted in additional profit aside holding the share of the stock at the end of the period (see last two rows of Table 4.6). To this effect, our model becomes useful since it helps to develop a strategy to trade only when conditions are predicted favourable (i.e. positive returns would be realized). However, it is limited by its inability to take into account; the cost of transaction involved in the sales of the stock.

5. Conclusion

This study presents different ways of predicting the direction of the returns on the Standard and Poor stock index using different machine learning techniques and develop a trading strategy based on the results. It has been observed that Artificial Neural Network performed relatively better than the other techniques. Although none of the models achieved a 100 percent accuracy, the result from the trained ANN model suggests that investors can develop a trading strategy with the model with some level of confidence. Future works should consider predicting the returns on the index for longer periods and also the actual returns; taking the high and low prices during the day, the volume traded and the other economic indicators such as exchange rates and consumer index into account.

Acknowledgements

My sincere gratitude goes to the Almighty Yahweh for His grace and mercies to make this work a success. Although the journey has not been easy, His grace has been sufficient to bring us this far.

I would also like to express my heart-felt gratitude to my supervisor; Professor Ronnie Becker, whose assistance, coaching and influence on me has helped me come up with this research project.

Next, I would like to express my appreciation to my family back in Ghana, most especially my mother; Madam Cecilia Akomah, who is my role model and Stella Owusu, who is also so dear to my heart.

Also, I would be ungrateful if I fail to appreciate the efforts of all those who made contributions in one way or the other, to make this work a success; most especially Jordan who is doing his Ph.D at the AIMS research center and all the AIMS tutors.

Lastly, I would like to thank the AIMS finance group, Fishoek Seventh Day Adventists church, Elliot, Akindele and the entire AIMS family, for making my stay here; a memorable one.

May the Almighty Yahweh bless you all for your efforts.

References

J. Agrawal, V. Chourasia, and A. Mittra. State-of-the-art in stock prediction techniques. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2(4):1360–1366, 2013.

V. Amin, S. S Hasan, V. Mehrdad, and N. Saber. Predicting direction of stock price index volatility using genetic algorithms and artificial neural network models in tehran stock exchange. 2014.

A. Bensaïda. Noisy chaos in intraday financial data: Evidence from the american index. *Applied Mathematics and Computation*, 226:258–265, 2014.

J. A. Bullinalia. Learning in multi-layer perceptron-back-propagation. online, http://www.cs.bham.ac.uk/~jxb/INC/l7.pdf, Accessed April 2016.

C. M. Carvalho. Model selection, logistic regression and more. online, http://faculty.mccombs.utexas.edu/carlos.carvalho/teaching/Section6.pdf, Accessed May 2016.

E. G. Chan. *Forecasting the S&P 500 index using time series analysis and simulation methods*. PhD thesis, Massachusetts Institute of Technology, 2009.

T. Clarence N. W. *Artificial neural networks: applications in financial distress predictions and foreign exchange hybrid trading systems*. Gold Coast, Wld. : Wilberto, 2001.

Delta. Delta rule. Wikipedia, The free encyclopedia, https://en.wikipedia.org/wiki/Delta_rule, Accessed April 2016.

V. S. Desai and R. Bharati. A comparison of linear regression and neural network methods for predicting excess returns on large stocks. *Annals of Operations Research*, 78:127–163, 1998a.

V. S. Desai and R. Bharati. The efficacy of neural networks in predicting returns on stock and bond indices*. *Decision Sciences*, 29(2):405–423, 1998b.

D. Enke and S. Thawornwong. The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with applications*, 29(4):927–940, 2005.

M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús. *Neural network design*, volume 20. PWS publishing company Boston, 1996.

J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques*. Elsevier, 2011.

M. Hanias, P. Curtis, and E. Thalassinos. Time series prediction with neural networks for the athens stock exchange indicator. *European Research Studies*, 15(2):23, 2012.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8. URL http://dx.doi.org/10.1016/0893-6080(89)90020-8.

H. Kantz and T. Schreiber. *Nonlinear time series analysis*, volume 7. Cambridge university press, 2004.

M. Kuhn. Package "caret" of r language. *Achieved from http://caret.r-forge.r-project.org*.

J. E. Kutsurelis. Forecasting financial markets using neural networks: An analysis of methods and accuracy. Technical report, DTIC Document, 1998.

L. S. Maciel and R. Ballini. Neural networks applied to stock market forecasting: an empirical analysis. *Journal of the Brazilian Neural Network Society*, 8(1):3–22, 2010.

B. G. Malkiel and E. F. Fama. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417, 1970.

N. Masoud. Predicting direction of stock prices index movement using artificial neural networks: The case of libyan financial market. *British Journal of Economics, Management & Trade*, 4(4):597–619, 2014.

Mathbits.com. Correlation coefficient. online, http://mathbits.com/MathBits/TISection/Statistics2/correlation.htm, Accessed April 2016.

P. Nastos, A. Paliatsos, I. Larissi, and K. Moustris. *Air Quality and Bioclimatic Conditions within the Greater Athens Area, Greece-Development and Applications of Artificial Neural Networks*. INTECH Open Access Publisher, 2011.

Normalization. statistics. Wikipedia, The free encyclopedia, https://en.wikipedia.org/wiki/Normalization_%28statistics%29, Accessed April 2016.

P. Ou and H. Wang. Prediction of stock market index movement by ten data mining techniques. *Modern Applied Science*, 3(12):p28, 2009.

A. F. Sheta, S. E. M. Ahmed, and H. Faris. A comparison between regression, artificial neural networks and support vector machines for predicting stock market index. *Soft Computing*, 7:8, 2015.

S. Thawornwong, D. Enke, and C. Dagli. Neural networks as a decision maker for stock trading: a technical analysis approach. *International Journal of Smart Engineering System Design*, 5(4):313–325, 2003.

J. Veri and M. S. Baba. Intelligent decission support system for prediction of indonesia stock exchanges.

L. Wang, K. Chen, and Y. S. Ong. *Advances in Natural Computation: Pt. 1: First International Conference, ICNC 2005, Changsha, China, August 27-29, 2005, Proceedings*, volume 1. Springer Science & Business Media, 2005.