Fundamental Analysis for Stocks using Extreme Gradient Boosting

Gumani Thanyani Rodney

Student No: 11640483

Supervisor: Dr Wilbert Chagwiza
Co-supervisor: Mr Tlou Kubjana

This dissertation is presented as partial fullfilment of the requirements of the degree:

Master of Science in Applied Mathematics
at the University of Venda
Department of Mathematical and Computational Sciences

14 June 2022

Declaration

*I, **Gumani Thanyani Rodney**, declare that this dissertation titled: **Fundamental Analysis for Stocks using Extreme Gradient Boosting**, submitted in partial fulfilment of the requirements for the conferral of the degree **Master of Science in Applied Mathematics**, from the University of Venda, is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.*

I further declare that I have not previously submitted this work, or part of it, for examination at University of Venda for another qualification or at any other higher education institution.

[illegible]

Mr TR. Gumani

14/06/2022

Date

Abstract

When it comes to stock price prediction, machine learning has grown in popularity. Accurate stock prediction is a very difficult activity as financial stock markets are unpredictable and non-linear in nature. With the advent of machine learning and improved computational capabilities, programmed prediction methods have proven to be more effective in stock price prediction. Extreme gradient boosting(XGBoost) is the variant of the gradient boosting machine. XGBoost, an ensemble method of classification trees, is investigated for the prediction of stock prices based on the fundamental analysis. XGBoost outperformed the competition and had higher accuracy. The developed XGBoost model proved to be an effective model that accurately predicts the stock market trend, which is considered to be much better than conventional non-ensemble learning techniques.

Keywords: *Stock Prediction, Machine Learning, XGBoost, Fundamental Analysis, Classification.*

Acknowledgments

This research would not have been possible without the assistance of the DSI-NICIS National e-Science Postgraduate Teaching and Training Platform (NEPTTP). Opinions expressed and conclusions reached are those of the author and should not be attributed to the NEPTTP.

I'd like to express my appreciation to the officials and staff members who assisted me with my research project.

I would like to express my gratitude to my supervisors, Dr. W Chagwiza and Mr. TL Kubjana, for their assistance and encouragement in carrying out this project. I also want to thank Dr. S Moyo for advising me and pushing me to finish the research project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Stock price forecasting has been a popular area of study for many years due to the potential for large profits. Forecasting the stock market is a difficult task, owing to its close-to-random-walk behavior. Stock investing is a popular investment strategy. However, if investors do not have enough information and expertise, their investments may result in losses. An astute trader would forecast the stock price and buy or sell a stock before its value rose or fell. Though it is difficult to replace the expertise of an experienced trader, an accurate model methodology can directly result in high earnings for financial institutions, implying a clear relationship between the accuracy of the prediction model and the revenue generated by using a specific algorithm. If we can forecast stock prices more accurately, society's capital will be allocated to the right place, avoiding resource waste, allowing the stock market price to grow in a desirable manner and people to invest more comfortably, avoiding blind investment habits.

Machine learning can be defined as information obtained through knowledge extraction. Machines do not need to be explicitly programmed; instead, they are trained to make data-driven decisions. Instead of writing code for each specific problem, data is fed into generic algorithms, and logic is built around that data. When a machine improves its performance based on previous experiences, it is said to have truly learned. Machine learning is a sub-field of computer science that seeks to develop algorithms that generate models of naturally existing systems based on examples of their endpoints. When developing machine learning models, statistics is commonly used. It is also referred to as data-driven learning or learning by example. In contrast, other types of systems rely solely on static application programs.

Figure 1.1 depicts the Cross Industry Standard Process for Data Mining(CRISP-DM).

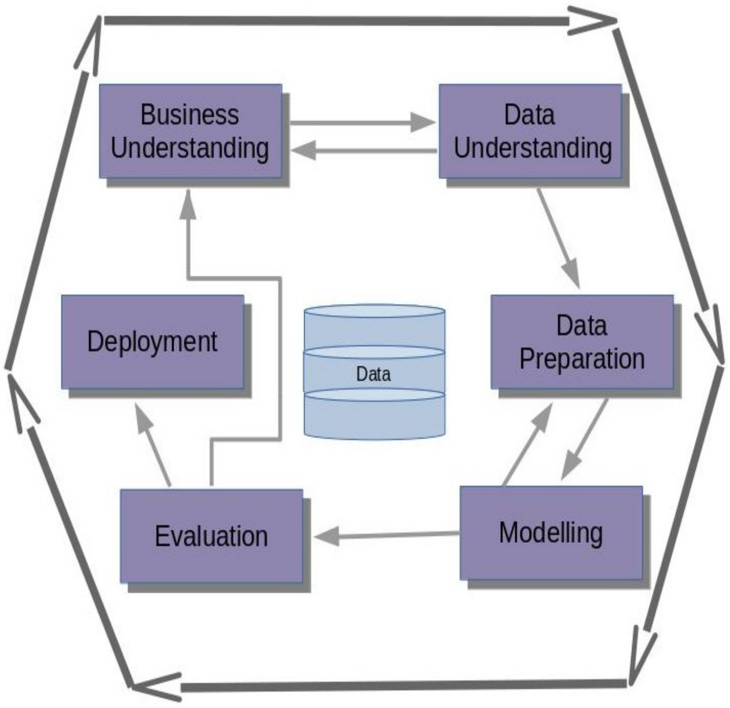


Figure 1.1: A diagram for the Cross Industry Standard Process for Data Mining(CRISP-DM)

The CRISP-DM process model serves as the foundation for a data science process. The business understanding phase focuses on comprehending the project's objectives and requirements. The following stage is data understanding. It drives the focus to identify, collect, and analyze data sets that can help you achieve the project goals, adding to the foundation of business understanding. The data preparation stage is responsible for preparing the final data sets for modelling. Here, we build and evaluate various models using various modelling techniques. The evaluation phase considers which model best meets the needs of the business and what steps should be taken next. The algorithms are used to appropriately build a predictive model, preparing the system for the evaluation stage. Experimentation, testing, and tuning take place during the modelling and evaluation stage. The overarching goal is to optimize the algorithm in order to achieve the required machine learning outcome while also maximizing system performance. We get the output in the deployment stage, which can be thought of as a non-deterministic query that must be deployed further in order to make decisions based on the output.

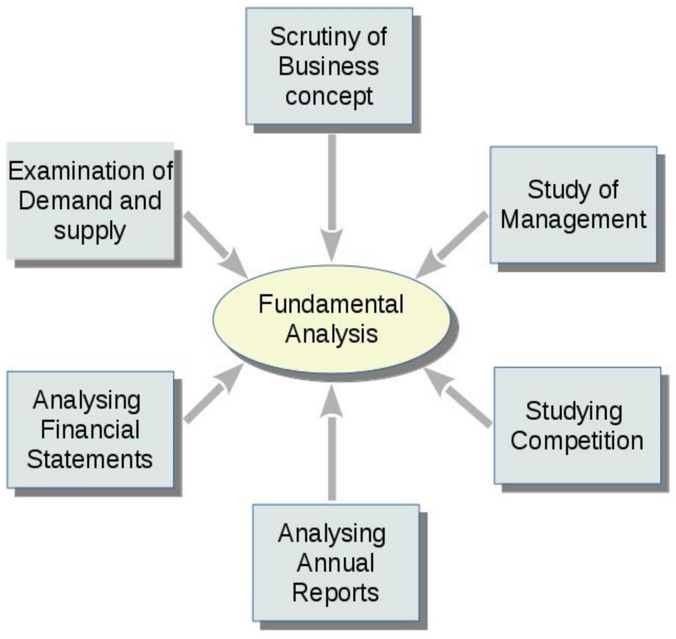
Many previous studies on predicting stock values have been conducted, but there are numerous disagreements about the reliability of such forecasts. Successful stock price forecasts offer limited guidance to investors looking to invest in a company or stock index, but price predictability studies can help with market price discovery functions. Long-term stock price fluctuation appears to be largely unpredictable, but on the other hand, different trends can be found in short-term fluctuations and thus generate disproportionate returns by capturing those rapidly shifting trends. It is important to note that there is no guarantee that stock price prediction models developed in the past will remain accurate in the future, which is why stock price prediction research is ongoing.

Another well-known model is the Black Scholes model. A mathematical model for calculating the price of an option contract. The model, in particular, calculates how financial instruments change over time. It was the first option pricing model that was widely used. The theoretical value of options is calculated using current stock prices, expected dividends, the option's strike price, expected interest rates, time to maturity, and expected volatility. Stock shares or futures contracts, for example, will have a lognormal distribution of prices based on a random walk with constant deviation and uncertainty, according to Black Scholes. Based on this assumption and other important variables, the equation calculates the price of a European-style call option.

The use of five variables is required by the Black-Scholes equation. The inputs are volatility, the price of the financial commodity, the option's target price, the time until the option expires, and the risk-free interest rate. In theory, options sellers can use these variables to set reasonable prices for the options they sell. In addition, the Black Scholes model predicts that the price of popular trading assets will move in a geometric brownian motion with a constant deviation and price fluctuations. When applied to a stock option, the model considers the stock's constant price variation, the payback period, the strike price of the option, and the time until the option expires.

When it comes to approaching the markets, fundamental and technical analysis are two primary components that are at opposite ends of the spectrum. Investors and traders use both to research and forecast future price movements. Both have supporters and detractors, just like any investment strategy or philosophy. Technical analysis is used by traders to identify opportunities by examining statistical trends such as price and volume movements in a stock. The basic idea is that all recognized fundamentals have been priced in, so there is no need to pay close attention to them. The goal of technical analysts is not to calculate the underlying value of a security. They instead use stock charts to identify patterns and trends that indicate how a stock will perform in the future.

Fundamental analysis is used to evaluate stocks with the goal of calculating their intrinsic value. Fundamental analysis is regarded as one of the most effective methods of valuing companies. Fundamental analysis is a technique for estimating the intrinsic value of a stock by examining a company's internal and external variables. It forecasts the movement of a stock price on the assumption that the stock price reflects the intrinsic value of the stock. The primary goal of the fundamental analysis is to reveal the company's true present value. One of the primary goals of fundamental analysis is to forecast future income, dividends, and risk in order to determine the true value of a stock. Finding a profitable company is not enough; one must also look for businesses that are worth more than most investors believe. The process of fundamental analysis is depicted in Figure 1.2 below.



Scrutiny of Business concept
Examination of Demand and supply
Study of Management
Fundamental Analysis
Analysing Financial Statements
Studying Competition
Analysing Annual Reports

Figure 1.2: Aspects of fundamental analysis

Ensemble methods aim to improve model accuracy by combining multiple models rather than using a single model. The combined models improve the accuracy of the results significantly. As a result, ensemble techniques have become increasingly popular. Bagging, Boosting, and Stacking are the three main types of ensemble methods. Boosting is an ensemble technique that involves the addition of new models to correct errors made by existing models. From Friedman (2001), models are added until no further improvements can be made. Gradient boosting is a method in which new models are created that predict the residuals or errors of previous models, which are then combined to make the final prediction. According to Chen and Guestrin (2016), XGboost is an efficient and scalable variant of the gradient boosting machine (GBM), which has recently won several machine learning competitions due to features such as ease of use, parallelization ease, and impressive predictive accuracy. The architecture of XGBoost is depicted in Figure 1.3.

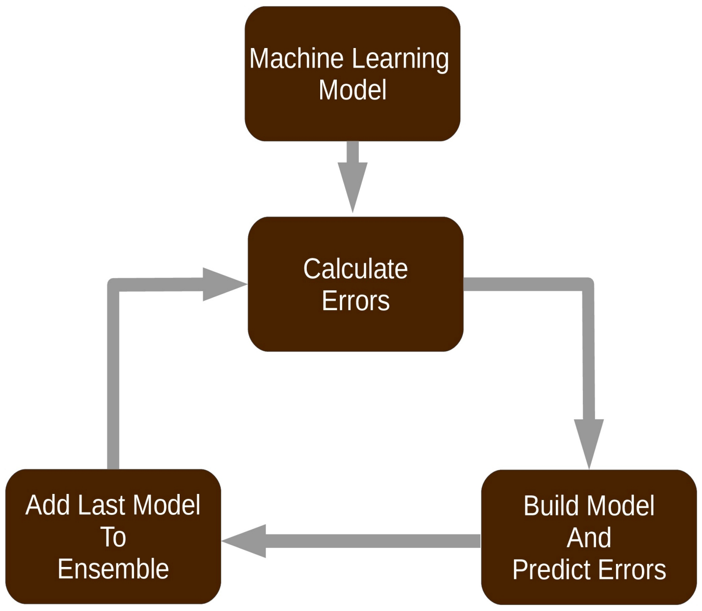


Figure 1.3: The process of XGBoost algorithm

1.2 Problem Statement

Stockbrokers who execute trades and advise customers frequently rely on their experience, technical analysis, or fundamental analysis when selecting stocks. These approaches are subjective and, as a result of their limitations, are typically short-sighted. Incorrect investment can easily result in large losses for investors with trade money at stake, especially if they continue to make poor decisions. As a result, having a method that can guide you on the most likely future price prediction is desirable as a basis for making any investment decision. The use of fundamental analysis serves as the foundation for forecasting potential changes in stock prices. Machine learning techniques can be used to evaluate stock prices over time and gain information, which can then be used to predict future stock prices.

We'd like to know how the feature selection method improves prediction model performance. Based on the abundance of previous works, we can conclude that stock price data is riddled with noise, and there are also correlations between features, making price prediction notoriously difficult. That is also the main reason why most previous works introduced the feature engineering component as an optimization module.

Unlike previous works, our evaluation will focus on the effectiveness of newly added features extracted from the financial domain, rather than the common evaluation of data models such as training costs and scores. We will discuss some aspects of the financial domain. While we only obtained a few specific findings from previous works, the raw data must be processed into usable features.

Because the stock market follows a random walk pattern, traditional machine learning and deep learning methods produce average results. Most previous works have not evaluated the algorithm for prediction accuracy and they suffer from overfitting. According to Bosco and Khan (2018) Previous stock prediction methods, such as the use of artificial neural networks and convolution neural networks, resulted in an average error loss of 20%. If an efficient algorithm for predicting an individual stock price can be developed, investment rates and business prospects on the stock market will improve. In this study, we look at how effective the extreme gradient boosting machine (XGBoost) is at predicting stock prices with a low percentage of error. When using XGBoost, you can control overfitting. XGBoost is said to be a powerful machine learning algorithm in terms of speed and accuracy. Successful stock price forecasting could assist investors in making more profit from their investments.

1.3 Research Questions

Three research questions were addressed: What is the role of feature engineering in improving model prediction accuracy? What are the implications of financial domain findings for prediction model design? And how does XGBoost fare when compared to other machine learning algorithms?

1.4 Research Aims and Objectives

1.4.1 Aim

The main aim of the study is to compare the predictive performance of extreme gradient boosting machines and the performance of other machine learning techniques that have been used in the previous studies.

1.4.2 Objectives

1. To develop the appropriate extreme gradient boosting models for stock price prediction;
2. To increase the prediction accuracy by the use of feature engineering; and
3. To come up with features that are significant in stock prediction.

1.5 Significance of the Study

The research will help investors, the general public, traders, brokers, and others make better investments and more accurate forecasts of potential returns. It will also allow for a better understanding of the predictive ability of financial ratios and their role in forecasting stock returns for academic purposes. A thorough understanding of the company's financial factors, as well as a forecast of potential profitability, would be provided by fundamental analysis at the business level .E (2009).

1.6 Structure

The thesis is divided into five chapters, the first of which is an introduction in which we introduce the concepts and background of the study. In Chapter 2, related papers on stock prediction are reviewed, with a focus on the research methods of the papers. The methodology of the classification models included in the study is covered in Chapter 3. The study's findings are presented in Chapter 4. Chapter 5 is the conclusion and discussion.

Chapter 2

Literature Review

2.1 Introduction

Accordingly, machine learning techniques aim to learn and interpret patterns in large amounts of data. Given a set of distinguishing characteristics, several well-known machine learning algorithms can be used to categorize a problem. The research looks into an XGBoost-based stock price prediction process. The related research is primarily concerned with the prediction model and feature selection for the prediction model of existing machine learning algorithms. This section will go over the literature on these two topics.

2.2 Prediction model

The improvement of prediction models has always been one of the most important research directions in stock price prediction. The majority of stock price forecasting methods rely on an econometric or machine learning model. These two models have been fine-tuned over time to be better suited to processing financial time series data in the volatile stock market.

Ding et al. (2018) proposed an improved ARIMA-GARCH model based on differential information in terms of econometric models by Zhang et al. (2016). Using the estimated differential information of the dependent variable lag and taking stock price change trend information into account improves the ability to forecast the course of price change.

Many academics are attempting to solve the problem by using modern digital technology rather than traditional prediction models to more reliably forecast stock price as machine learning technology advances. Maknickas and Maknickiene (2019) used a recursive neural network (RNN) to build a stock price prediction model and optimized the selection of RNN parameters such as the number of neurons and iterations. Wang et al. (2016) forecasted potential stock price increases and decreases using the LSTM model and various market analysis metrics. According to the results, LSTM outperformed both the traditional machine learning model and the non-time series model. Xu et al. (2019) focused on financial time series data preprocessing, including interpolation, wavelet de-noising, and data normalization, and experimented with various LSTM model parameter combinations. According to the researchers, the optimized model had low computational complexity and significantly improved prediction accuracy. Vo et al. (2019) compared the effects of LSTM, Bi-directional LSTM (Bi-LSTM), and gated recurrent unit on stock price prediction. They discovered that by reading the data one more time backward, the Bi-LSTM model improved prediction accuracy, especially when forecasting sequential data such as financial time series.

Furthermore, current literature combines statistical econometric models with machine learning models or employs more than two types of machine learning models to forecast stock price. When compared to a single model, these models typically outperform. Achkar et al. (2018) combined models using the back propagation algorithm-multi-layer perception (BPA-MLP) and the LSTM-RNN. Using stock price data from Facebook, Google, and Bitcoin, they discovered that the LSTM-RNN model outperformed the BPA-MLP model. Bao et al. (2017) used a wavelet transform to reduce noise in the original time series stock data before forecasting the future with an LSTM model. According to the findings, the integrated model outperformed other related models. Chan Phooi M'ng and Mehralizadeh (2016) proposed a wavelet principal component analysis-neural network (WPCA-NN) prediction model that combined wavelet transform, principal component analysis, and artificial neural network to de-noise, removing random noise in stock price series. According to the findings, the WPCA-NN outperformed traditional prediction approaches. Kim and Kim (2019) proposed an LSTM-CNN model based on feature combination using stock time series and stock trend graphs as input features. According to the findings, the LSTM-CNN model outperformed the single model in forecasting stock prices.

2.3 Feature Selection

Feature selection and feature engineering are useful for enriching data sets and extracting valuable information from them, which can improve prediction accuracy. Previous research has produced a variety of useful features from the original data set, such as stock movement graphs and major economic and political events, to improve stock price prediction results. Stock investor emotion is one of the most commonly used features in stock price prediction models.

The fluctuation of stock prices is frequently influenced by investors' emotions for a stock and the overall stock market, Nassirtoussi et al. (2014). As a result, prior research relied on natural language processing (NLP) technology to analyze stock social media documents and extract investor emotions, adding new dimensions to stock price prediction models. The proper noun system, a modern approach for identifying essential nouns, was proposed by Schumaker and Chen (2009). Among the seven types of nouns they identified were date, place, money, organization, percentage, individual, and time. According to the findings, the proper noun system outperformed the bag-of-words and called object identification schemes. The financial text disclosed by companies on trading days was compiled by Kraus and Feuerriegel (2017), who dealt with business disclosures using sequence modeling. They then combined RNN and LSTM models to forecast stock price. The results showed that including financial text in the equation significantly improved prediction accuracy. Zhou et al. (2018) used the bag-of-words model to derive five emotional qualities of stock market investors: disgust, excitement, sorrow, and fear. The K-means model outperformed the baseline models, which included one that only used financial time series as data. Linear Discriminant Analysis (LDA) is a relatively unknown but intriguing stock price forecasting strategy. Jin et al. (2013) used LDA to extract topics from the text, and the representative topics were used as input features for the prediction model, resulting in improved prediction accuracy.

Feature dimensionality reduction is one of the most important research directions in feature extraction. Xie X and Y (2020) proposed a dual dimension reduction model of joint mutual knowledge to improve the Principal Component Analysis (PCA) algorithm. The model first screened a large number of characteristics using shared information, and then used the complex correlation coefficient and cumulative variance contribution rate to calculate the number of PCA principal elements for secondary dimension reduction. In terms of prediction, the improved approach outperformed the conventional feature reduction model. Hagenau et al. (2013) attempted to depict text with more expressive characteristics. They extracted English word stems from dictionaries, then calculated the interpretive power of features using Chi-square and Bi-normal-separation, retaining only those with high interpretive power. The findings demonstrated that using the feature dimension reduction approach to reduce overfitting in machine learning models could improve prediction accuracy while reducing overfitting. Huang et al. (2010) extracted nouns, verbs, and compound phrases from the text and then converted synonyms for each word using a thesaurus to achieve the dimension reduction goal.

2.4 Other Related Works

Econometric models and machine learning models are the two most commonly used approaches in stock price prediction. Traditional machine learning models, on the other hand, take a single period of data as a sample and ignore a large amount of implicit knowledge that emerges over time, making it difficult for econometric models to deal with nonlinear time series problems, Baek and Kim (2018). XGBoost is a new emerging technology that can effectively process time-series and multi-period data. A combination of multiple models typically outperforms a single model, and this is quickly becoming the dominant trend in stock price prediction.

Most existing research employs standard text feature extraction methods to incorporate new text features for market price prediction (such as bag-of-words, named entity recognition, and LDA). Although these features can partially reflect investor emotions, they cannot typically represent document semantic content, meaning, or other information in social media, Nassirtoussi et al. (2014).

Shen et al. (2012) proposed a forecast algorithm that uses the temporal relationship between global equity markets and various financial items to forecast the next day's stock trend using support vector machine (SVM). They then used the same algorithm with a different regression algorithm to forecast actual market growth and built a simple trading model with its algorithm contrasted by the SVM algorithm. The benchmark model 1 is a simple model in which the profit is determined by the trend during the testing period. A major drawback of SVM for direction forecasting is that the input variables are located in a high-dimensional feature space with hundreds to thousands of dimensions. The ability to measure variables necessitates a large amount of memory and computing time. In fact, a stock market contains a variety of stocks, which increases the computational complexity of the variables. As a result, direct dimension reduction is critical in order to have a reliable and discriminative representation prior to classification.

SVM was used to predict the time series path. Kim (2003) trained SVM on the daily time series of the Korean stock market and achieved a 56% hit rate. Huang et al. (2005) attempted to forecast weekly NIKKEI 225 index movements using SVM. The Huang et al. (2005) study achieved a 73 percent hit rate with SVM and a 75 percent hit rate with the combined model. In the approach to back-propagation by Huang et al. (2005), SVM also outperformed neural networks. Schumaker and Chen (2009) tried to forecast the movement of the S and P 500 indexes 20 minutes after the news was published.

To predict Jordanian Stock Exchange, Alkhatib et al. (2013) used k-nearest neighbor (kNN) and non-linear regression methods. Alkhatib et al. (2013) anticipated KNN to be an efficient algorithm, and predictions were nearly identical to the stock price. An efficient expectation model can be evaluated based on how accurately it predicts stock price based on the data provided, in order to determine whether or not the returns are gradually increasing. The KNN approach has a dense area in that large memory is required for large training sets, and estimation accuracy degrades rapidly as the number of parameters increases.

ANNs have been described as a powerful method for non-parametric information representation in a variety of distinct contexts where the output is a non-linear function of the inputs. A variety of studies on neural systems foreseeing stock market developments have recently been conducted. Soni (2011) stated the validity of ANNs in stock market record expectations and provided a link between the Fama model, the French model, and the ANN model from the perspective of Chinese stock market forecast. They claim that ANNs outperform linear models in predicting financial markets in terms of predictive power.

Hanias et al. (2012) predicted the Athens stock index using neural networks and back-propagation. Hanias et al. (2012) used a configuration with one hidden layer made up of seven hidden neurons and achieved a respectable forecast execution up to nine days ahead of the mean squared error (MSE). De Faria et al. (2009) distinguished neural network prediction of execution, the neural networks outperformed the adaptive exponential smoothing method in the forecasting movement, in the Brazilian stock market from the adaptive exponential smoothing technique. They also claimed that the neural network outperforms the adaptive exponential smoothing technique and that the right directional hit rate was 60%. Many analyses have been conducted, and the results show that a consistent rate of results is maintained in all markets. Finally, De Faria et al. (2009) concluded that a good prediction output can be used to develop beneficial speculation techniques, and that De Faria et al. (2009)'s model is ideal for developing a decision support system for the Brazilian market.

According to the paper by Saini and Singh (2014), the back propagation algorithm, in conjunction with the ANN, is used to forecast daily returns on the stock market as well as the weather in Dehradun. As a result, the use of soft computing techniques in stock exchanges was the primary focus. Patel and Yalamalle (2014) attempted to forecast the movement of companies on the stock market within the context of the National Securities Exchange using ANN techniques. Historical data is used to build a model, and the process's output determines the model's accuracy. Tsai and Wang (2009) conducted research and attempted to predict stock prices using ensemble learning, which is made up of decision trees and artificial neural networks. They created a dataset using Taiwanese stock market data, including fundamental, technical, and macroeconomic indexes. The F-score performance of Decision Trees + Artificial Neural Networks trained on Taiwan stock exchange data was 77%. F-score performance of single algorithms was up to 67 percent".

Decision trees are effective and well-known tools for classification and forecasting. In data mining, a decision tree is a statistical model that can be used to describe both classifiers and regression models. They are also useful for investigating data to obtain a snapshot of the relationships between a significant number of variable competitor inputs and goal output. Typically, the goal variable is absolute, and the decision tree model is used to either quantify the value of providing a record for each classification, or to classify the record by assigning it to the most likely class. Panigrahi and Mantri (2015) proposed a decision tree rough-set based hybrid method with a hierarchical hidden Markov model to forecast future stock market developments. A hybrid structure based on decision tree rough collection exists to forecast trends in the Bombay Stock Exchange alongside the Hierarchical Hidden Markov Model. Potential patterns based on income from prices and benefit are also shown. Accounting income data assist in predicting the present estimate of potential earnings when arriving at the midpoint of several years.

Chen (2011) established a few operating rules at Taiwan Stock Exchange Corporation to improve the predictive accuracy of the financial distress model, for which they initially gathered data from 100 firms. They extracted the relevant variables using principal component analysis, as well as a 37-ratio empirical experiment that included financial and non-financial ratios. Thus, the decision tree classification methods and linear regression techniques were used to apply the financial prediction model. The experiments produced a satisfactory result, confirming the likelihood and validity of the proposed structures for predicting financial distress in the previously mentioned organizations.

Phua et al. (2003) conducted a study predicting the movement of five major stock indexes: the DAX, DJIA, FTSE-100, HSI, and NASDAQ. They used neural networks and were able to predict the direction of price movement with greater than 60% accuracy by using component stocks as input.

2.5 Summary

We looked at previously published projects and reports on their findings. Then we gave an overview of what had been said, who the key writers were, the dominant theories and hypotheses, the questions being raised, and the methods and methodologies that had proven useful in stock price prediction.

Chapter 3

Research Methodology

3.1 Introduction

In this thesis, we will perform classification analysis using XGBoost, which is the chosen method of supervised machine learning. Based on the values of predictor variables (x), we will be able to predict a binary target variable (y). A classification model's goal is to create a mathematical equation that defines y as a function of the x variables. Following that, this equation can be used to predict the outcome (y) using new values for the predictor variables (x).

3.2 Machine Learning Algorithm

Gradient boosting involves adding models sequentially until no more changes can be made. Gradient boosting is a method that generates new models to predict the residuals or errors of previous models, which are then combined to make the final prediction. It is called gradient boosting because it employs a gradient descent algorithm to minimize loss while introducing new models. Instead of practising on a newly sampled distribution, the weak learner trains on the residual errors of the strong learner. At each iteration, the pseudo-residuals are computed, and these pseudo-residuals are fitted to a slow learner. The contribution of the weak learner to the strong learner is then determined, not on the newly distributed sample based on its results, but using a gradient descent optimization method. The computed contribution is the one that minimizes the overall error of the strong learner.

3.3 Proposed Method

Extreme Gradient Boost is used for supervised learning problems, where we use the training data x_i to predict a target variable y_i.

In supervised learning, the model typically refers to the mathematical framework that is used to predict y_i from x_i data. The predictive value may have different interpretations depending on the task at hand, either regression or classification. The coefficients denoted by β will be the undetermined parameters that we must learn from the data.

Model training's role is to find the best parameters that fit the x_i training data and y_i labels. To train the model, we must first define the objective function, which determines how well the model fits the training data. The fact that objective function is divided into two parts, training loss and regularisation term, is a distinguishing feature.

$$Obj(\beta) = L(\beta) + \Omega(\beta) \tag{3.3.1}$$

where L represents the loss function and Ω represents the regularization term The training loss quantifies how accurate our model is in relation to the training data. L is also known as the mean squared error, and it is given by

$$L(\beta) = \sum_{i} \frac{1}{n}(y_i - \hat{y}_i)^2 \tag{3.3.2}$$

Where n is the number of records in the dataset. We know that a tree ensemble model is made up of a collection of classification and regression trees (CART). Our model can be expressed as follows:

$$\hat{y}_i = \sum_{k=1}^{K} \frac{1}{K} f_k(\mathbf{x}_i), f_k \in \Psi \tag{3.3.3}$$

where K represents the number of trees The set of all possible classification and regression trees is denoted by Ψ. The to-be-optimized objective function is given by

$$Obj(\beta) = \sum_{i}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{3.3.4}$$

Now in tree boosting, the equation below is the objective function

$$Obj = \sum_{i=1}^{n} L(y_i, \hat{y}^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \tag{3.3.5}$$

The additive training allows us to correct what we've already learned while also adding one new tree at a time. The prediction value is written at step t as $\hat{y}_i^{(t)}$. Therefore,

$$\begin{cases} \hat{y}_i^{(0)} & = 0 \\ \hat{y}_i^{(1)} & = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\ \hat{y}_i^{(2)} & = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\ & \vdots \\ \hat{y}_i^{t} & = \sum_{k=1}^{t} f_k(x_i) = \hat{y}^{(t-1)} + f_t(x_i) \end{cases}$$

If we use mean squared error as our loss function, the objective function is now written as

$$Obj^{(t)} = \sum_{i=1}^{n} (y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^{t} \Omega(f_i) = \sum_{i=1}^{n} \left[2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2 \right] + \Omega(f_t) \tag{3.3.6}$$

Taking the loss function's second order Taylor series expansion, we get the following

$$Obj^{(t)} = \sum_{i=1}^{n} \left[L(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{3.3.7}$$

where g_i and h_i are defined as

$$g_i = \partial_{\hat{y}_i^{(t-1)}} L(y_i, \hat{y}_i^{(t-1)}) \tag{3.3.8}$$

$$h_i = \partial^2_{\hat{y}_i^{(t-1)}} L(y_i, \hat{y}_i^{(t-1)}) \tag{3.3.9}$$

Specifically, the objective function at step t now becomes

$$\sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{3.3.10}$$

We must define the regularization term, which is the tree's complexity $\Omega(f)$. $f(x)$ is denoted as $f_t(x) = w_{q(x)}$, $w \in R^T, q : R^d \to R^+$. Here, w represents the leaf weights or scores, q is a function that assigns each data point to the corresponding leaf, and T represents the number of leaves. The term ”regularization” in XGBoost is defined as

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{3.3.11}$$

After determining the regularization term or complexity mentioned above, we can write the objective function as follows:

$$Obj^{(t)} \approx \sum_{i=1}^{n}\left[g_i w_{q(x_i)} + \frac{1}{2}h_i w_{q(x_i)}^2\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{3.3.12}$$

$$= \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right) w_j^2\right] + \gamma T \tag{3.3.13}$$

where $I_j = \{i|q(x_i) = j\}$ and by defining $G_j = \sum_{i\in I_j} g_i$ and $H_j = \sum_{i\in I_j} h_i$, then we have

$$Obj^{(t)} = \sum_{j=1}^{T}\left[G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2\right] + \gamma T \tag{3.3.14}$$

w_j are independent to each other in the above equation. $G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2$ has a quadratic form. Then the best objective reduction we can get is:

$$w_j^* = -\frac{G_j}{H_j + \lambda} \tag{3.3.15}$$

$$Obj^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T \tag{3.3.16}$$

To learn the model’s tree structure, we will try to optimize one tree level at a time. We are attempting to divide a leaf into two pieces, and the score it receives is given by

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma \tag{3.3.17}$$

If we have $Gain$ that is less than γ, we should avoid adding that branch. In tree-based models, this is how pruning techniques work.

3.4 Other Methods

In this section, we present the methodology of various methods that we will compare in terms of performance to XGBoost. Logistic Regression, Stochastic Gradient Descent, k-Nearest Neighbor, and Supervised Vector Machines will be used.

3.4.1 Logistic Regression

Logistic regression is a classification algorithm that assigns observations to one of several classes. Given a data with (X, Y), where X is a value matrix with m examples and n features and Y is a vector with m examples. The goal is to train the model to predict which class future values will belong to. We begin by creating a weight matrix with a random initialization. Then we multiply it by the number of features.

$$a = w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n \tag{3.4.1}$$

The above equation is to be substituted into the following link function:

$$\hat{y}_i = \frac{1}{1 + e^{-a}} \tag{3.4.2}$$

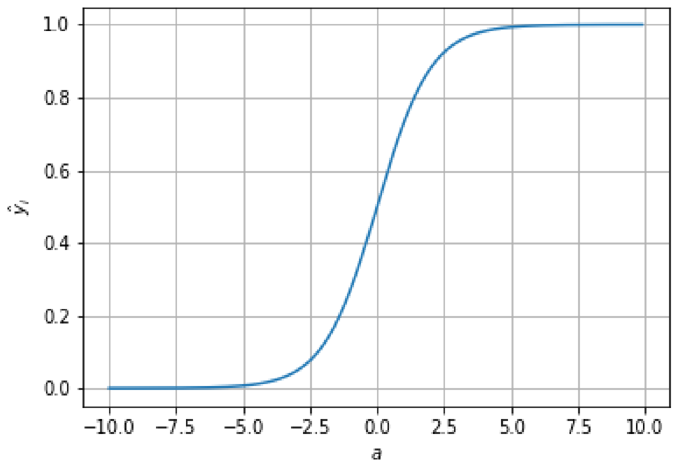


Figure 3.1: Graph of the link(sigmoid) function.

After that what follows is to calculate the cost of the iteration and the cost function is given by:

$$Cost(w) = \frac{1}{m}\sum_{i=1}^{m} y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i) \tag{3.4.3}$$

The derivative of the cost is calculated, according to which of the weights are being updated. Now we have gradient and update formulas

$$dwj = \sum_{i=1}^{n}(\hat{y}-y)x_j^i \tag{3.4.4}$$

$$w_i = w_j - (\alpha \times dw_j) \tag{3.4.5}$$

Given the values of x and w, logistic regression calculates the likelihood of a specific set of data points belonging to either of those classes. In Logistic regression, we know that probability is always greater than 0 but less than 1. Therefore, the sigmoid link function is given by

$$\frac{1}{1+e^{-(w_0+w_1x_1+...+w_nx_n)}} \tag{3.4.6}$$

The odds of an event occurring in Logistic regression is given as

$$\frac{p}{1-p} = e^{w_0+w_1x_1+...+w_nx_n} \tag{3.4.7}$$

By applying the Log transformation we obtain

$$log(\frac{1}{1-p}) = w_0 + w_1x_1 + ... + w_nx_n \tag{3.4.8}$$

After carefully considering the conditional probability of obtaining an output equal to the sigmoid function and assuming that our sample has a Bernoulli distribution, the logistic regression model's cost function is derived as follows:

$$p(y|X;W) = \sum_{i=1}^{n}(h_W(X))^y + (1-(h_W(X))^{1-y}) \tag{3.4.9}$$

As gradient descent minimizes error, the minus sign at the beginning of the function ensures that we try to minimize the negative of likelihood rather than maximizing the value.

3.4.2 Stochastic Gradient Descent

Gradient descent can be used to minimize costs by first calculating the gradients of the cost function and then updating existing parameters in response to the gradients. Stochastic gradient descent randomizes the dataset and updates the weights and parameters for each individual training example. The following formula will be used to model stochastic gradient descent. This iteration is performed after each backpropagation of the model until the cost function approaches its point of convergence.

$$w_0 = w_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} h_w(x^{(i)}) - y^{(i)} \tag{3.4.10}$$

$$w_1 = w_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}).x^{(i)} \tag{3.4.11}$$

The gradient of the loss function with respect towards each weight is multiplied by the learning rate and subtracted from the vector.

3.4.3 K-Nearest Neighbors

This algorithm saves all available cases and classifies new cases based on the majority vote of its K neighbors. It calculates the distance between each data point and the test data and then determines the likelihood that the points are similar to the test data. The points with the highest probabilities are used to classify them. The distance function can be either Euclidean, Minkowski, or Hamming. Here we focus on the first two distance functions.

Euclidean Distance Function

The Euclidean distance is simply the shortest distance between two points, regardless of their dimensions. The Euclidean distance is the most commonly used method for calculating the distance between two points. The Euclidean distance between two plane points with coordinates (x, y) and (c, d) is given by

$$distance = \sqrt{(x-c)^2 + (y-d)^2} \tag{3.4.12}$$

The algorithm will find the k-nearest neighbors of the data point for a given value of K. and then assign the class to the data point based on the class with the most data points among the K neighbors. Following the calculation of the distance, the input x is assigned to the class with the highest probability:

$$P(y = j|X = x) = \frac{1}{K} \sum I(y^{(i)} = j) \tag{3.4.13}$$

Minkowski Distance Function

The Minkowski distance is a generalization of the Euclidean distance that measures the distance between two points in normed vector space. Minkowski distance between two points is given as:

$$\sqrt[p]{(x_1 - y_1)^p + (x_2 - y_2)^p + ... + (x_n - y_n)^p} \quad (3.4.14)$$

When p is equal to 2, Minkowski is the Euclidean distance.

3.4.4 Support Vector Machines

The Support Vector Machine(SVM) basically aids in the classification of data into two or more categories by using a boundary to distinguish similar categories. SVM works better with binary classification. At first, we are given the data that will be separated by SVM. The given dataset is represented by $x \in R^D$. The point is mapped on x, which is a complex feature space.

$$f(x) \in R^M$$

Following feature space transformation, f(x) can be defined as follows for each input feature mapped to a transformed basis vector:

$$f(x) : R^D \mapsto R^M$$

The line that separates the points into their classes is called decision boundary. This is done by using the equation of the hyperplane, which is given by

$$H : w^T(x) + b = 0, \quad (3.4.15)$$

where b is the intercept. We need to know the distance between the hyperplane and the data points to minimize the errors in the classification of those data points. The distance between a hyperplane and a given point vector (x_0) is easily expressed as:

$$d_H(f(x_0)) = \frac{|w^T(f(x_0)) + b|}{||w||_2} \quad (3.4.16)$$

Since the aim is to get the shortest possible distance, then we have

$$w^* = arg_w max[min_n d_H(f(x_n))] \quad (3.4.17)$$

On correct prediction, the product of a predicted and true label would be greater than zero; otherwise, it would be less than zero.

$$y_n[w^T f(x) + b] = \begin{cases} \geq 0 & \text{if correct} \\ < 0 & \text{if incorrect} \end{cases}$$

In SVM, a kernel is used to predict the outcome. By definition, the kernel avoids the clear and specific mapping required for linear learning algorithms to learn a decision boundary. The kernel function is given by

$$w : f \rightarrow g$$

which satisfies

$$k(x, x\prime) = (w(x), w(x\prime))g$$

Since the kernel is symmetric in nature and positive semi-definite, we can write it as

$$\sum_m \sum_n r_m r_n k(x_m, x_n) \geq 0 \tag{3.4.18}$$

These values $f^T(x_m)f(x_n) = k(x_m, x_n)$ can be substituted into Equation 3.4.18 based on the kernel's definition. After substitution and using the kernel definition in our dual form,

$$max_{\beta_i} h(\lambda_n, \beta_n) = \sum_n \beta_n + \frac{1}{2} \sum_n \beta_m \beta_n y_m y_n k(x_m, x_n)$$
$$\beta_n, \lambda_n \geq 0, \forall n; \sum_n \beta_n y_n = 0; C - \beta_n - \lambda_n = 0$$

Therefore, by using kernels for further prediction:

$$w^T f(x) = \left(\sum_n \beta_n y_n f(x_n) \right)^T f(x) \tag{3.4.19}$$
$$\Rightarrow \sum_n \beta_n y_n f^T(x_n) f(x) \tag{3.4.20}$$
$$\Rightarrow \sum_n \beta_n y_n k(x_n, x) \tag{3.4.21}$$

3.5 Feature Selection

We use permutation feature importance for feature selection. The decrease in a model score caused by randomly shuffled feature values is referred to as permutation feature importance. The scoring argument, which accepts multiple scorers, can be used to specify the score function that will be used to compute importance. Using multiple scorers is more computationally efficient than calling permutation importance several times with a different scorer because it reuses model predictions.

The following is a description of the permutation feature importance algorithm:
Input: m - trained model, X - feature matrix, y - target vector, and $E(y, m)$ - error measure.

1. Estimate the original model error $e^{orig} = E(y, m(X))$(e.g. mean squared error)
2. For each feature $j = 1, ..., p$ do:
   (a) Generate feature matrix X^{perm} by permuting feature j in the data X. This breaks the association between feature j and true outcome y.
   (b) Estimate error $e^{perm} = E(Y, m(X^{perm}))$ based on the predictions of the permuted data.
   (c) Calculate permutation feature importance $FI^j = \frac{e^{perm}}{e^{orig}}$. Alternatively, the difference can be used: $FI^j = \frac{e^{perm}}{e^{orig}}$.
3. Sort features by descending FI.

Tree-based models provide an alternative measure of feature importance based on the mean decrease in impurity. The decision tree splitting criterion quantifies impurity (Gini, Entropy or Mean Squared Error). When the model is overfitting, however, this method may place a high value on features that are not predictive on unseen data. In contrast, permutation-based feature importance avoids this issue because it can be calculated on previously unseen data.

Furthermore, the importance of impurity-based features for trees is skewed, favoring high cardinality features over low cardinality features such as binary features or categorical variables with a limited number of possible categories. In permutation-based feature importances, such a bias does not exist. Furthermore, the permutation feature importance can be computed as a model prediction performance metric and applied to any model class. In our case, we use the XGBoost classifier to compute the permutation importance of the features. We must contend with the deceptive values of highly correlated features. When two features are correlated and one is permuted, the model can still access the permuted feature through the correlated feature. As a result, even if both features are important, they will have a lower importance value.

3.6 Model Evaluation

Accuracy on training data is critical, but so is obtaining a genuine and approximate result on unseen data, otherwise the model is useless. So, in order to build and deploy a generalized model, we must evaluate the model on various metrics, which allows us to better optimize the effectiveness, fine-tune it, and achieve a better result. There is no need for multiple metrics if one metric is perfect. Understanding the benefits and drawbacks of evaluation metrics is important because each evaluation metric fits on a different set of a dataset.

3.6.1 Confusion Matrix

Confusion matrix is the evaluation metric for machine learning classification problems with two or more classes as output. It is a table consisting of four different predicted and true values.

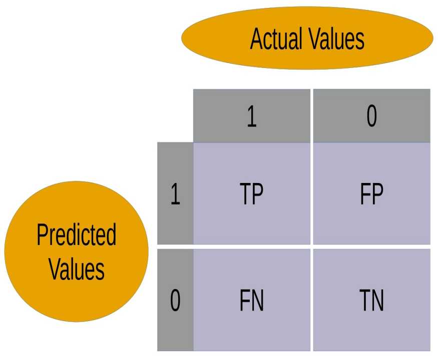


Figure 3.2: Confusion matrix

It is particularly useful for calculating Recall, Precision, F1 measure, Accuracy, and, most notably, AUC-ROC curves. True Positive is when the classifier predicted positive and it is true. True Negative is when the classifier predicted negative and it is true. False Positive(Type 1 Error) is when the classifier predicted positive and it is false. False Negative(Type 2 Error) is when the classifier predicted negative and it is false.

3.6.2 Recall

Recall is the percentage of positives that you correctly identified out of all positives. Equation below can be explained by stating how many of the positive classes we correctly predicted.

$$Recall = \frac{TP}{TP + FN}$$

3.6.3 Precision

Precision informs us about the likelihood of making a correct classification of positive class. It is calculated by dividing the number of True Positives by the total number of positive calls.

$$Precision = \frac{TP}{TP + FP}$$

3.6.4 Accuracy

Accuracy is one metric that indicates the percentage of correct predictions made by our model. Accuracy is defined by:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

3.6.5 F1 measure

The harmonic mean of the model's precision and recall is used to calculate the F1 score. As a result, the F1 score is a better metric to use if you want to strike a balance between Precision and Recall.

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

3.6.6 ROC/AUC Curve

Another common tool for evaluation is the receiver operator characteristic(ROC). It plots the sensitivity and specificity of a model for each possible decision rule cut-off between 0 and 1. A threshold can be used to convert probability outputs to classifications in classification problems with probability outputs. The ROC curve plots the False positive rate versus the True positive rate for each possible threshold. The fraction of negative instances that are incorrectly classified as positive is referred to as the False Positive Rate. The fraction of positive instances that are correctly predicted as positive is referred to as the True Positive Rate.

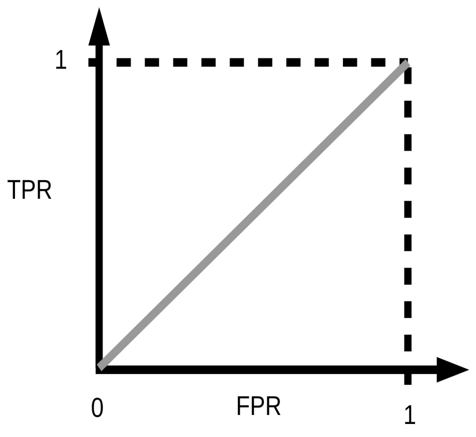


Figure 3.3: The axes of TPR and FPR with the TPR=FPR line

The model is good if its ROC curve is at the upper triangle in the figure above. One way to compare classifiers is to measure the area under the curve for ROC.

3.7 Matthews Correlation Coefficient (MCC)

MCC is calculated by the following formula:

$$MCC = \frac{TN \times TP - FN \times FP}{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}$$

MCC, like most correlation coefficients, has a range of -1 to 1. Where 1 represents the best agreement between actuals and predictions and 0 represents no agreement at all. In other words, the prediction is based on chance in relation to the actuals.

3.8 Data

The dataset from Chagwiza (2018) includes over 200 financial indicators for all stocks in the US stock market. The financial indicators are from the Financial Modelling Prep API and can also be found in the annual 10-K filings of publicly traded companies. The data was collected between 2014 and 2018. We train an XGBoost model to learn to distinguish between stocks that are and are not suitable for trading. Stocks that belong to class 1 are stocks that one should buy at the start of year 2019, and sell at the end of year 2019. The target variable is the dataset's last column named "Class," which represents the class of each stock; if the value of a stock increases, class=1; if the value of a stock decreases, class=0. This dataset was created in order to determine whether or not it is possible to classify a stock's future performance based on financial data. There are 225 variables in the data.

The features are: Revenue, Revenue Growth, Cost of Revenue, Gross Profit, R&D Expenses, SG&A Expense, Operating Expenses, Operating Income, Interest Expense, Earnings before Tax, Income Tax Expense, Net Income - Non-Controlling int, Net Income - Discontinued ops, Net Income, Preferred Dividends, Net Income Com, EPS, EPS Diluted, Weighted Average Shs Out, Weighted Average Shs Out (Dil), Dividend per Share, Gross Margin, EBITDA Margin, EBIT Margin, Profit Margin, Free Cash Flow margin, EBITDA, EBIT, Consolidated, Income, Earnings Before Tax Margin, Net Profit Margin, Cash and cash equivalents, Short-term investments, Cash and short-term investments, Receivables, Inventories, Total current assets, Property, Plant & Equipment Net, Goodwill and Intangible Assets, Long-term investments, Tax assets, Total non-current assets, Total assets, Payables, Short-term debt, Total current liabilities, Long-term debt, Total debt, Deferred revenue, Tax Liabilities, Deposit Liabilities, Total non-current liabilities, Total liabilities, Other comprehensive income, Retained earnings (deficit), Total shareholders equity Investments, Net Debt, Other Assets, Other Liabilities, Depreciation & Amortization, Stock-based compensation, Operating Cash Flow, Capital Expenditure, Acquisitions and disposals, Investment purchases and sales, Investing Cash flow, Issuance (repayment) of debt, Issuance (buybacks) of shares, Dividend payments, Financing Cash Flow, Effect of forex changes on cash, Net cash flow / Change in cash, Free Cash Flow, Net Cash/Marketcap, priceBookValueRatio, priceToBookRatio, priceToSalesRatio, priceEarningsRatio, priceToFreeCashFlowsRatio, priceToOperatingCashFlowsRatio, priceCashFlowRatio, priceEarningsToGrowthRatio, priceSalesRatio dividendYield, enterpriseValueMultiple, priceFairValue, ebitperRevenue, ebtperEBIT, niperEBT, grossProfitMargin, operatingProfitMargin, pretaxProfitMargin, netProfitMargin effectiveTaxRate, returnOnAssets, returnOnEquity, returnOnCapitalEmployed and nIperEBT.

The variables were reduced to 78 after cleaning the data. The number of variables was reduced to 71 after feature selection. Variables in the dataset are not correlated with one another. If we find correlation, we handle the correlated variables by removing one of the perfectly correlated features. The data set is skewed negatively against the target variable classes. Because skewed data can impair the classification abilities of our machine learning model, it is preferable to transform the skewed data to normally distributed data. Normalization is the function considered in this study for such transformation.

3.9 Summary

In machine learning, classification employs a mathematically provable set of methods to do the analysis that would take people hundreds of hours to complete. XGBoost model is to be compared with other four models. For feature selection, permutation importance was chosen to guide us on the importance of the features. All metrics to be used to evaluate the classification models in this study have been outlined.

Chapter 4

Results

4.1 Introduction

Here we discussed the process of configuring the parameters of the XGBoost algorithm and its high interpretability. The results of the XGBoost model on the data were then displayed and compared to the previous methods.

4.2 Tuning Model Parameters

The most powerful machine learning algorithms are known for automatically tuning thousands or millions of parameters to detect patterns and regularities in data. This parameters in XGBoost are the decision variables used at each node and the numerical thresholds used to determine whether to take the left or right branch when trying to generate predictions. The more design decisions and adjustable hyper-parameters an algorithm has, the more flexible and powerful it is. The maximum depth of the tree, the number of trees to grow, the number of variables to consider when building each tree, the minimum number of samples on a leaf, the fraction of observations used to build a tree, and a few others are examples of hyper-parameters.

RandomSearchCV was used to perform a grid search on all parameters in order to find the best parameters. Although it often takes longer to execute, random search is excellent for discovery and obtaining hyper-parameter combinations that you would not have guessed intuitively. The tuning parameters are shown in the table below. We adjusted n_estimators and used the validation set to check the absolute error of the stock price's highs and lows.

Parameters	Initialization value	Search space
min_child_weight	1	$[1, 2, 3, 4, 5, 6]$
$gamma$	0	$[0.0, 0.1, 0.2, 0.3, 0.4]$
$subsample$	0.4	$[0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$
max_depth	3	$[3, 4, 5, 6, 7, 8, 9, 10, 11]$
$learning\_rate$	0.01	$[0.01, 0.03, 0.05, 0.07, 0.1]$
$colsample\_bytree$	0.6	$[0.6, 0.7, 0.8, 0.9]$

Table 4.1: The details of tuning parameters.

The absolute error of the validation set did not decrease after 40 iterations, so the optimal value of n estimators was set to 40 to avoid over-fitting. The next parameter, gamma, is adjusted, and n_estimators is set to 40. Table below displays the absolute error of the validation set for various gamma values. We chose the optimal value of gamma to be 0.4,

gamma	Absolute Error
0	0.2576
0.1	0.2727
0.2	0.2778
0.3	0.2803
0.4	0.2576
0.5	0.2753

Table 4.2: The absolute errors of validation set with their respective gamma values.

since we have 0 and 0.4 which are corresponding to the smallest absolute error. We have all the hyper-parameters and the optimal values are shown in the table below. Using the

Parameters	Optimal Value
$n\_estimators$	40
$gamma$	1.5
min_child_weight	1
$learning\_rate$	0.01
max_depth	5
$colsample\_bytree$	0.8
$subsample$	0.6

Table 4.3: Optimal values of parameters.

best parameters, the absolute error of stock prices on the validation set is 0.2291 and on the test set is 0.2298. It can be seen that after configuring the parameters, the model's performance improved. As a result, parameter adjustment is beneficial for improving accuracy. For each feature in the dataset, importance scores are calculated and ranked. The amount that each attribute point of separation improves the performance measure, weighted by the set of measurements for which the node is responsible, is used to calculate importance for a single decision tree. The greater a feature's improvement to the point of separation, the more important the feature. Figure 4.1 shows the first 20 features of the stock price with highest importance scores.

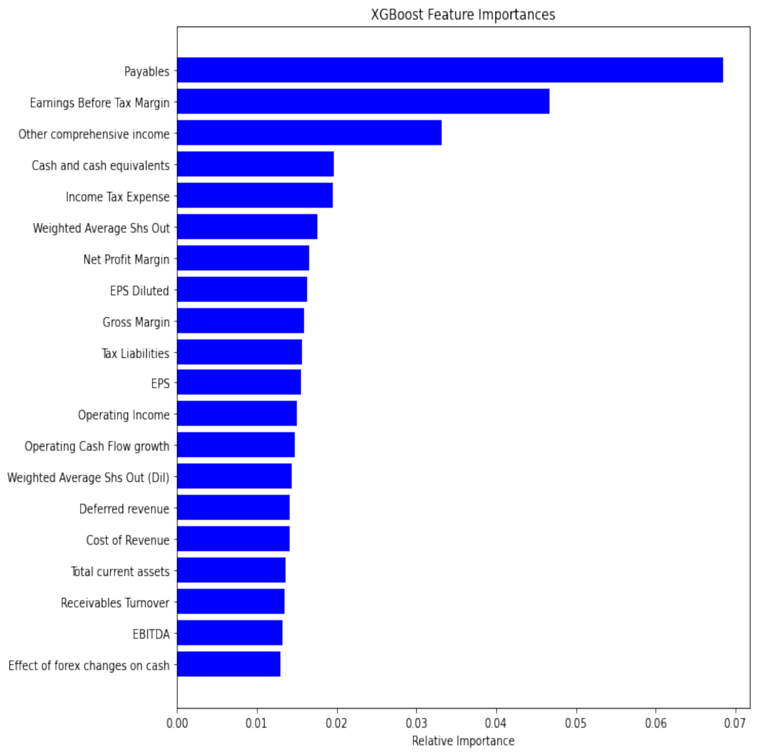
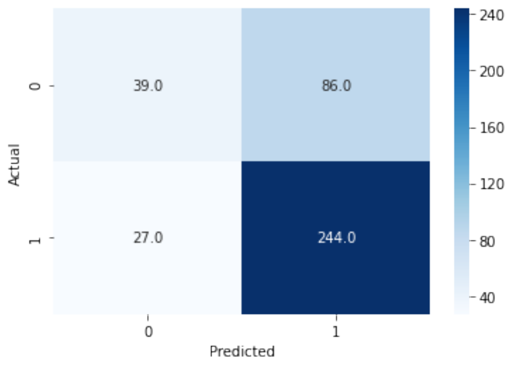


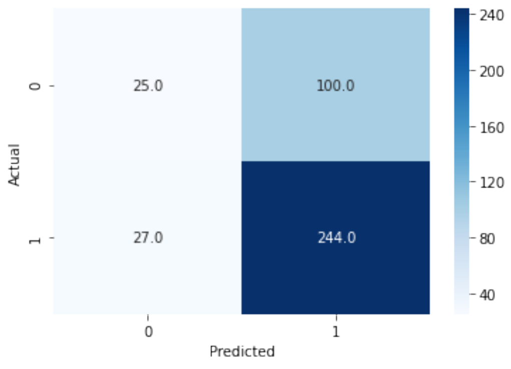
Figure 4.1: Top 20 features with the highest importance scores

4.3 Performance Comparison

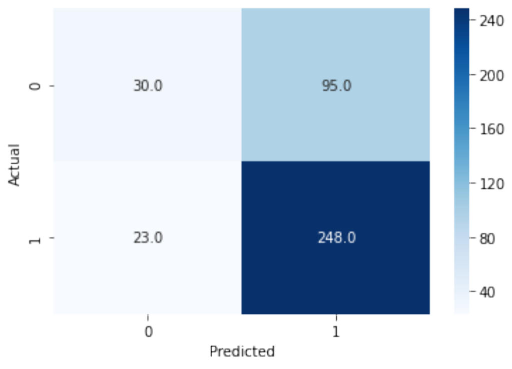
In this study, we trained five models on the training set: Logistic Regression, Stochastic Gradient Descent, SVM, KNN, and XGBoost, and the parameters were optimized based on performance on the validation set. To begin, we count the number of correct and incorrect classifications. A confusion matrix is used to evaluate a classification model's performance on a set of test data for which the true values are known. The confusion matrix is used to calculate most performance measures such as precision and recall. The confusion matrix is provided in Figure 4.2. Models were compared by their precision, recall, f1 and accuracy scores. Figure 4.3 shows the plot for those metrics. In Figure 4.3a, KNN is leading with 74.45%, followed by XGBoost which is 73.98%, SGD is 73.00%, Logistic Regression is 71.01% and the last one is SVM with a precision of 68.43%. This means that KNN has higher probability of producing positive classification. In Figure 4.3b, SVM has the perfect



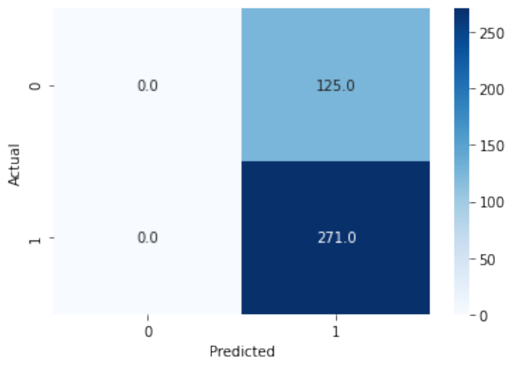
(a) XGBoost



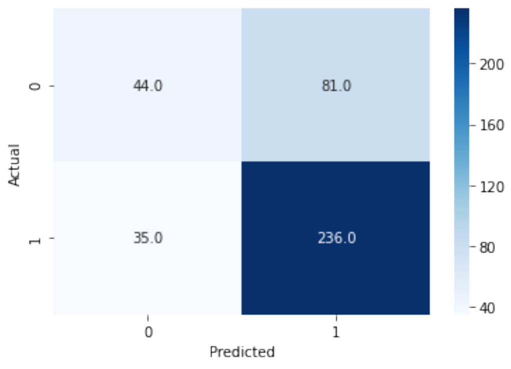
(b) Logistic Regression



(c) Stochastic Gradient Descent

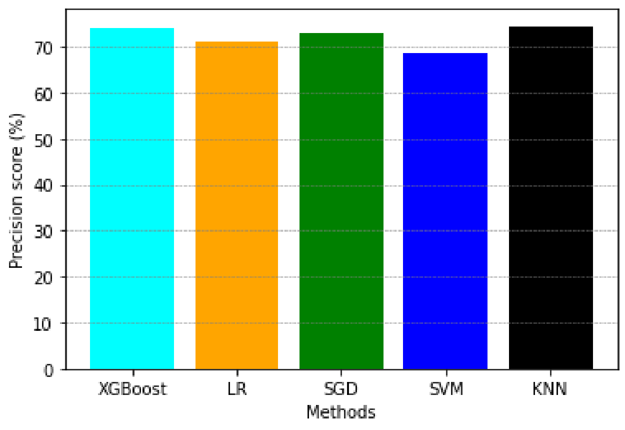


(d) SVM

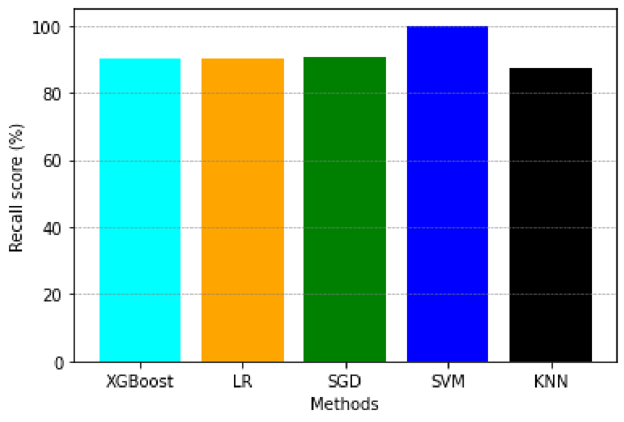


(e) KNN

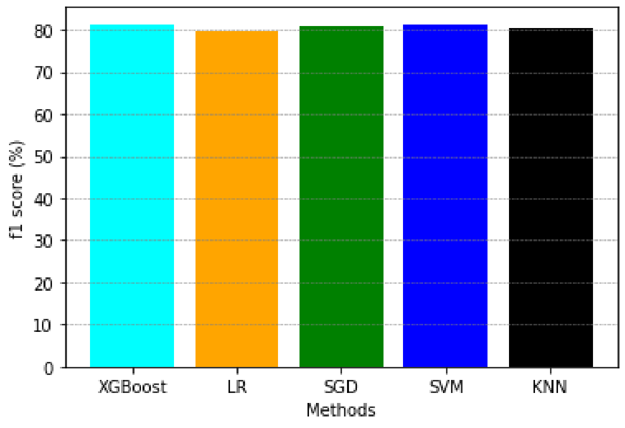
Figure 4.2: Confusion matrix



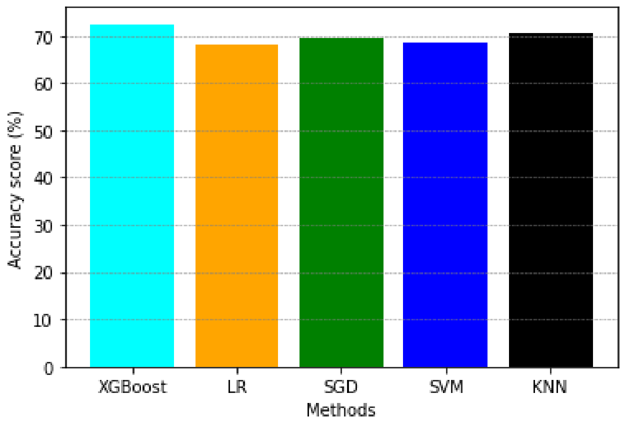
(a) Precision



(b) Recall



(c) F1 score



(d) Accuracy

Figure 4.3: Comparisons of the models

score of 100%. This means that false negative of the SVM classifier is zero. SGD has a recall of 90.77%, for Logistic Regression is 90.41%, XGBoost is 90.35% and the last one is KNN with the recall of 87.08%. F1 scores are shown in Figure 4.3c. XGBoost is leading with the f1 score of 81.35%, followed by SVM with 81.25%. SGD has f1 score of 80.92%, KNN is 80.27% and the last one is the Logistic Regression with f1 score of 79.55%. Accuracy is detailed in Figure 4.3d, XGBoost is leading again with the accuracy score of 72.46%. KNN's accuracy is 70.71%, Stochastic Gradient Descent has 69.70%, followed by SVM with 68.43% and the last one is Logistic Regression.

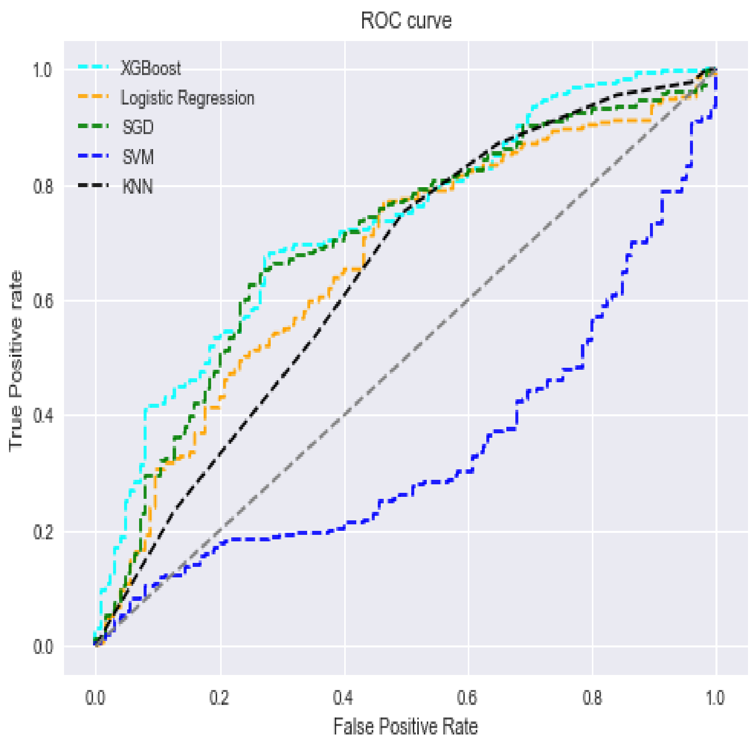


Figure 4.4: ROC curves of the models

Figure 4.4 displays the ROC curves of the models. The Area Under Curve(AUC) of XGBoost ROC curve is the highest of them all. The AUC of XGBoost is 73.04%, for SGD is 70.29%, Logistic Regression is 66.78%, KNN is 64.87% and lastly we have SVM with 34.21%. This means that XGBoost was better when it comes to classifying positive class in the dataset.

Lastly, we check the Mathews Correlation Coefficient(MCC) of all the classifiers. In Table 4.4 shows the MCC values. XGBoost has better MCC, both XGBoost, Logistic Regression, SGD, and KNN have MCC which is greater than zero. There is some agreement between the actual and predicted y values. The MCC for SVM means that there is random prediction with respect to the true y values.

Algorithm	MCC
XGBoost	0.2649
Linear Regression	0.1444
Stochastic Gradient Descent	0.2346
SVM	0.0
KNN	0.2592

Table 4.4: Mathews Correlation Coefficient with their respective models.

4.4 Summary

To tune the hyperparameters, cross-validation function of XGBoost was used. Tuning hyperparameters helps to improve the algorithm's performance by minimizing the absolute error on cross-validation. Confusion matrices revealed that XGBoost and KNN performed better than others in classification. The task was to compare classification models and get the best model. XGBoost is the classification model that performed better than other chosen models in this study.

Chapter 5

Conclusion and Discussion

When it comes to stock price prediction, machine learning has grown in popularity. This paper investigates and contrasts various classification techniques for stock price prediction. The use of machine learning techniques (from the scikit-learn library) in the implementation envisions the use of an application to carry out training and testing exercises for various stock prices. The use of significance features improved the performance of the classification techniques used in this study.

The following conclusions were drawn from the data examined and the classification techniques tested: The XGBoost classification technique has the highest accuracy of 72.46% and the highest F1 score of 81.35%; it is clear from Figure 4.4 that the AUC for the XGBoost ROC curve is higher than that for the ROC curves of the other classification techniques. As a result, we can conclude that XGBoost performed better in classifying the positive class in the dataset; XGBoost has the best MCC of 0.2649, indicating that the agreement between actual and predicted values is better than with other techniques. XGBoost outperformed all four machine learning classification techniques evaluated.

Stock prices will most likely fluctuate. Factors such as sale, commodity, demand, products, recession, and investor sentiment. A very accurate prediction of a stock's future price could result in much-needed returns or profit. XGBoost outperformed the competition and had higher accuracy. The developed XGBoost model proved to be an effective model that accurately predicts the stock market trend, which is considered to be much better than conventional non-ensemble learning techniques. Stock forecasting is critical because the stock market provides capital, it is a measure of a company's liquidity, and it has the capacity to satisfy the company's short-term obligations as well as finance operations. Stock markets exist to serve the larger economy. Investing in the stock market allows individuals to earn a return on their profits while also allowing businesses to spread their risks and reap significant rewards. We conclude that the XGBoost model outperformed the algorithms in related works in terms of stock price prediction accuracy. Despite the fact that our proposed solution yielded a satisfactory result, this study has more research potential in the future. A distributed processing environment should be considered for large data sets. This allows for a high level of correlation among the variables, which will ultimately make the model's output more efficient.

References

R. Achkar, F. Elias-Sleiman, H. Ezzidine, and N. Haidar. Comparison of bpa-mlp and lstm-rnn for stocks prediction. In *2018 6th International Symposium on Computational and Business Intelligence (ISCBI)*, pages 48–51. IEEE, 2018.

K. Alkhatib, H. Najadat, I. Hmeidi, and M. K. A. Shatnawi. Stock price prediction using k-nearest neighbor (kNN) algorithm. *International Journal of Business, Humanities and Technology*, 3(3):32–44, 2013.

Y. Baek and H. Y. Kim. Modaugnet: A new forecasting framework for stock market index value with an overfitting prevention lstm module and a prediction lstm module. *Expert Systems with Applications*, 113:457–480, 2018.

W. Bao, J. Yue, and Y. Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7):e0180944, 2017.

F. Black and M. Scholes. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, pages 637–654, 1973.

J. Bosco and F. Khan. Stock Market Prediction and Efficiency analysis using Recurrent Neural Networks. *Technical Computer Science*, pages 2–7, 2018.

W. Chagwiza. Financial modelling prep: Stock fundamental analysis, 2018. URL https://financialmodelingprep.com/api/v4/financial-reports-json?symbol=AAPL&year=2018&period=FY.

J. Chan Phooi M'ng and M. Mehralizadeh. Forecasting east asian indices futures via a novel hybrid of wavelet-pca denoising and artificial neural network models. *PloS one*, 11(6):e0156338, 2016.

M.-Y. Chen. Predicting corporate financial distress based on integration of decision tree classification and logistic regression. *Expert Systems with Applications*, 38(9):11261–11272, 2011.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International conference on knowledge discovery and data mining*, pages 785–794, 2016.

E. De Faria, M. P. Albuquerque, J. Gonzalez, J. Cavalcante, and M. P. Albuquerque. Predicting the Brazilian Stock Market through neural networks and adaptive exponential smoothing methods. *Expert Systems with Applications*, 36(10):12506–12509, 2009.

C. Ding, J. Duan, Y. Zhang, X. Wu, and G. Yu. Using an arima-garch modeling approach to improve subway short-term ridership forecasting accounting for dynamic volatility. *IEEE Transactions on Intelligent Transportation Systems*, 19:1054–1064, 03 2018. doi: 10.1109/TITS.2017.2711046.

J. .E. Fundamental Analysis Strategy and the Prediction of Stock Returns. *International Research Journal of Finance and Economics*, pages 95–108, 2009.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

M. Hagenau, M. Liebmann, and D. Neumann. Automated news reading: Stock price prediction based on financial news using context-capturing features. *Decision Support Systems*, 55(3):685–697, 2013.

M. P. Hanias, P. G. Curtis, and E. Thalassinos. Time series prediction with neural networks for the Athens Stock Exchange indicator. *European Research Studies*, 15(2), 2012.

C.-J. Huang, J.-J. Liao, D.-X. Yang, T.-Y. Chang, and Y.-C. Luo. Realization of a news dissemination agent based on weighted association rules and text mining techniques. *Expert Systems with Applications*, 37(9):6409–6413, 2010.

W. Huang, Y. Nakamori, and S.-Y. Wang. Forecasting stock market movement direction with support vector machine. *Computers & operations research*, 32(10):2513–2522, 2005.

F. Jin, N. Self, P. Saraf, P. Butler, W. Wang, and N. Ramakrishnan. Forex-foreteller: Currency trend modeling using news articles. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1470–1473, 2013.

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

T. Kim and H. Y. Kim. Forecasting stock prices with a feature fusion lstm-cnn model using different representations of the same data. *PloS one*, 14(2):e0212320, 2019.

M. Kraus and S. Feuerriegel. Decision support from financial disclosures with deep neural networks and transfer learning. *Decision Support Systems*, 104:38–48, 2017.

A. Maknickas and N. Maknickiene. Support system for trading in exchange market by distributional forecasting model. *Informatica*, 30(1):73–90, 2019.

A. K. Nassirtoussi, S. Aghabozorgi, T. Y. Wah, and D. C. L. Ngo. Text mining for market prediction: A systematic review. *Expert Systems with Applications*, 41(16):7653–7670, 2014.

S. Panigrahi and J. Mantri. Epsilon-SVR and decision tree for stock market forecasting. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 761–766. IEEE, 2015.

M. B. Patel and S. R. Yalamalle. Stock price prediction using artificial neural network. *International Journal of Innovative Research in Science, Engineering and Technology*, 3(6): 13755–13762, 2014.

P. K. H. Phua, X. Zhu, and C. H. Koh. Forecasting stock index increments using neural networks with trust region methods. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 1, pages 260–265. IEEE, 2003.

M. Saini and A. Singh. Forecasting stock exchange market and weather using soft computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(5):877–882, 2014.

R. P. Schumaker and H. Chen. Textual analysis of stock market prediction using breaking financial news: The AZFin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):1–19, 2009.

S. Shen, H. Jiang, and T. Zhang. Stock market forecasting using machine learning algorithms. *Department of Electrical Engineering, Stanford University, Stanford, CA*, pages 1–5, 2012.

S. Soni. Applications of ANNs in stock market prediction: a survey. *International Journal of Computer Science & Engineering Technology*, 2(3):71–83, 2011.

K.-H. Tsai and J.-C. Wang. External technology sourcing and innovation performance in LMT sectors: An analysis based on the Taiwanese Technological Innovation Survey. *Research Policy*, 38(3):518–526, 2009.

N. N. Vo, X. He, S. Liu, and G. Xu. Deep learning for decision making and the optimization of socially responsible investments and portfolio. *Decision Support Systems*, 124:113097, 2019.

J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

L. X. Xie X and Z. Y. Application of mutual information and improved pca dimensionality reduction algorithm in stock price forecasting. *Computer Engineering and Applications, in Chinese*, 2020.

W. Xu, H. Peng, X. Zeng, F. Zhou, X. Tian, and X. Peng. Deep belief network-based ar model for nonlinear time series forecasting. *Applied Soft Computing*, 77:605–621, 2019.

K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE signal processing letters*, 23(10):1499–1503, 2016.

Z. Zhou, K. Xu, and J. Zhao. Tales of emotion and stock in china: volatility, causality and prediction. *World Wide Web*, 21(4):1093–1116, 2018.