**‘Is AI the Future of Investing?’**

An investigation into the predictive power of Machine Learning models for future stock prices.



CO4015 Computer Science Project

Author: Osato Osagie

May 2021

School of Informatics

# **Acknowledgments**

Firstly, I want to acknowledge and give thanks to my supervisor, Dr. Huiyu Zhou, and Dr. Emmanuel Tadjouddine, for their continued guidance and support throughout this project. Also, for providing rapid responses and feedback during the COVID-19 pandemic. Also, I would like to thank Dr. Irek Ulidowski for his advice given concerning the project during the interview.

# **Abstract**

In this project, I aim to be able to accurately predict future stock prices on the London Stock Exchange (LSE). This will be accomplished by combining historical data from the stock with technical indicators. To accomplish this, I will employ a variety of regression models, including Long short-term memory (LSTM), Random Forest, and Ridge Regression.

**Keywords** – Regression, Machine Learning, Technical Analysis, Algorithmic Trading, Data mining.

# **Declaration**

Text, letter

Description automatically generated

Table of Contents

[Acknowledgments 2](#_Toc71160292)

[Abstract 3](#_Toc71160293)

[Chapter 1 7](#_Toc71160294)

[Introduction 7](#_Toc71160295)

[1. Aims 9](#_Toc71160296)

[1.1 Objectives 9](#_Toc71160297)

[**1.1.1 Changed objectives** 10](#_Toc71160298)

[1.2 Resources & Tools 11](#_Toc71160299)

[Chapter 2 11](#_Toc71160300)

[Literature Review 11](#_Toc71160301)

[2.1. Impact of AI on Trading and Investing 12](#_Toc71160302)

[2.2 Stock Market Prediction 13](#_Toc71160303)

[**2.2.1 Fundamental Analysis** 13](#_Toc71160304)

[**2.2.2 Technical Analysis** 14](#_Toc71160305)

[Chapter 3 16](#_Toc71160306)

[Methodology 17](#_Toc71160307)

[3.1 Technical Indicators 17](#_Toc71160308)

[3.2 Stock Dataset 22](#_Toc71160309)

[3.3 Data Preparation 23](#_Toc71160310)

[**3.3.1 Missing values** 23](#_Toc71160311)

[**3.3.2 Scaling of Features** 24](#_Toc71160312)

[**3.3.3 Splitting the data** 25](#_Toc71160313)

[3.4 Models 26](#_Toc71160314)

[3.4.1 Supervised Learning 26](#_Toc71160315)

[**3.4.1.1 Regression** 26](#_Toc71160316)

[3.4.2 Random Forest 27](#_Toc71160317)

[**3.4.2.1 Drawbacks** 29](#_Toc71160318)

[3.4.3 Ridge Regression 29](#_Toc71160319)

[**3.4.3.1 Simple Linear Regression** 30](#_Toc71160320)

[**3.4.3.2 Regularisation** 32](#_Toc71160321)

[**3.4.3.3 Ridge Regression** 32](#_Toc71160322)

[**3.4.3.4 Drawbacks** 34](#_Toc71160323)

[3.4.4 Long Short-Term Memory (LSTM) 34](#_Toc71160324)

[**3.4.4.1 Drawbacks** 39](#_Toc71160325)

[3.5 Model Evaluation 40](#_Toc71160326)

[3.5.1 R Square/Adjusted R Square 40](#_Toc71160327)

[3.5.2 Mean Square Error (MSE)/Root Mean Square Error (RMSE) 41](#_Toc71160328)

[3.5.3 Mean Absolute Error (MAE) 42](#_Toc71160329)

[Chapter 4 42](#_Toc71160330)

[Implementation and Evaluation 42](#_Toc71160331)

[4.1 Modules and Packages 42](#_Toc71160332)

[4.2 Streamlit User Interface (UI) 45](#_Toc71160333)

[**4.2.1 Query Parameters** 45](#_Toc71160334)

[**4.2.2 Company Description** 46](#_Toc71160335)

[**4.2.3 Closing Price** 48](#_Toc71160336)

[**4.2.4 Data Correlation** 49](#_Toc71160337)

[**4.2.5 Descriptive Statistics** 50](#_Toc71160338)

[**4.2.6 Models** 50](#_Toc71160339)

[**4.2.7 Profit and loss calculations** 58](#_Toc71160340)

[Chapter 5 59](#_Toc71160341)

[Testing 59](#_Toc71160342)

[5.1 General Electric (Defensive Stock) 60](#_Toc71160343)

[5.2 HSBC (Cyclical Stock) 61](#_Toc71160344)

[5.3 IBM (Income Stock) 62](#_Toc71160345)

[5.4 Amazon (Growth Stock) 62](#_Toc71160346)

[5.5 Summary 63](#_Toc71160347)

[Critical Appraisal 64](#_Toc71160348)

[9.1 What Went Well? 64](#_Toc71160349)

[9.2 What could be improved 64](#_Toc71160350)

[**9.2.1 Frontend** 64](#_Toc71160351)

[**9.2.2 Prediction Algorithms** 64](#_Toc71160352)

[9.3 What would you do differently? 65](#_Toc71160353)

[**9.3.1 Frontend Improvement** 65](#_Toc71160354)

[**9.3.2 Sentiment Analysis (SA)** 65](#_Toc71160355)

[**9.3.2 Portfolio optimization** 67](#_Toc71160356)

[**9.3.3 Algorithmic Trading** 67](#_Toc71160357)

[9.4 Benefits and Risks to Society 67](#_Toc71160358)

[**9.4.1 Benefits to Society** 67](#_Toc71160359)

[**9.4.2 Risks to Society** 68](#_Toc71160360)

[9.6 Commercial Context 68](#_Toc71160361)

[**9.6.1 Advertisements** 68](#_Toc71160362)

[**9.6.2 Subscription Model** 68](#_Toc71160363)

[9.7 Personal Development 69](#_Toc71160364)

[**9.4.2 Python Programming** 69](#_Toc71160365)

[**9.4.2 Machine Learning applications in finance** 69](#_Toc71160366)

[**9.4.3 Time management** 69](#_Toc71160367)

[**9.4.3 Summary** 70](#_Toc71160368)

[Conclusion 71](#_Toc71160369)

[Bibliography 71](#_Toc71160370)

[Appendix A 74](#_Toc71160371)

[Appendix B 75](#_Toc71160372)

[Appendix C 76](#_Toc71160373)

[Appendix D 76](#_Toc71160374)

[Appendix E 77](#_Toc71160375)

[Appendix F 77](#_Toc71160376)

[Appendix G 79](#_Toc71160377)

[Appendix H 82](#_Toc71160378)

[Appendix I 85](#_Toc71160379)

# **Chapter 1**

## **Introduction**

The term ‘stock market’ does not refer to a single market. It refers to several stock exchanges located all over the world. Traders and investors can use these stock exchanges to purchase public company's shares and sell them there too. In this project, I will primarily concentrate on companies listed on the London Stock Exchange (LSE).

The stock prices of these companies constantly fluctuate under the laws of economics. A stock is a small ownership interest in a publicly-traded company. Stock prices reflect investors’ and market analysts’ expectations for the company’s potential income.

When an investor believes a company will perform well, they jack up the price by creating demand for stocks in that company. On the other hand, those traders who don’t believe in a company’s future will bid the price down by selling their holdings, thus creating an excess supply. Sellers aim to receive a high price for each share hopefully more than their initial payment, whilst buyers seek to acquire shares at the cheapest price, which allows them to make a profit when they sell the stock.

Investing in stocks is deemed as a reliable method of achieving long-term profits that outperform inflation. On average, the returns outperform many of the other investments like bonds and commodities. According to research, the total market value of all companies trading on the LSE as of February 2021 was 3.67 trillion GBP [1].

There are two ways to make a profit on the stock market. Investors can either trade stocks or decide to hold them. Trading requires investors to buy and sell stocks regularly, capitalizing on small price changes. Whereas value investing requires investors to purchase and let their stocks appreciate over time. Value investors hold stocks and are sometimes rewarded with regular dividend payments.

Investors have previously found ways to obtain insight about the businesses listed on the market for as long as markets have existed to increase their investment returns. However, owing to the size market and the pace at which transactions are conducted, this is not possible today. Simple statistical analysis of financial data can reveal certain trends, but in recent years, AI technologies are increasingly being used by investment firms to search for trends in massive amounts of real-time financial information.

AI is the ability of machines to behave like humans and learn autonomously. For instance, a machine might display learning and problem-solving abilities without the use of hard-coded software containing detailed instructions [1].

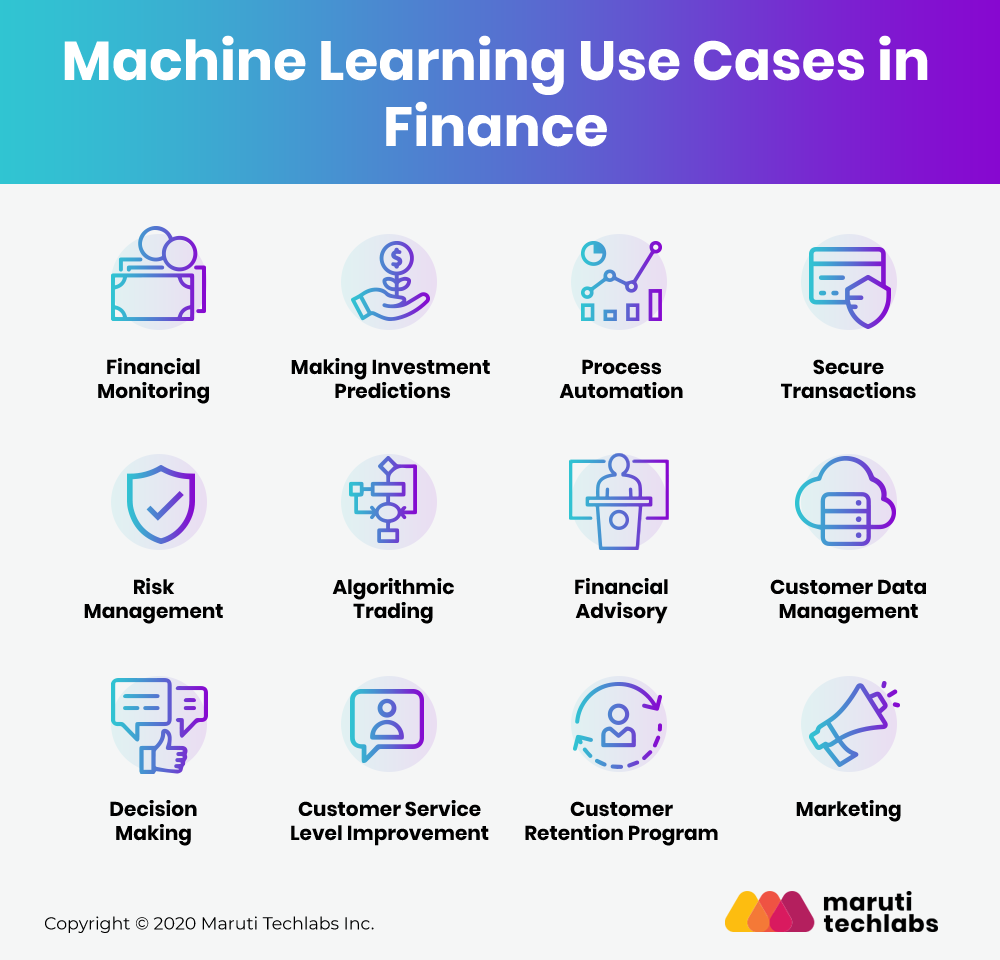


Figure 1- An image showing the main use cases of Machine Learning in Finance [2]

Machine Learning (ML) is a subfield of AI, that enables machines to learn from historic data or experiences without being explicitly programmed. Figure 1 shows the different use cases for Machine Learning in finance. The project I will be building focuses on using ML to make investment predictions. Using ML to make investment predictions is advantageous as it can lead to better predictions of stock prices, fewer errors, and greater efficiency for the investor. To do this, ML algorithms extract key insights from the dataset, learns from it then apply several techniques to accurately predict the result.

This project is motivated by the stock market's revolutionization caused by algorithmic trading. According to a UK research firm Coalition, computerized trades make up roughly 45 percent of cash equities trading profits. Furthermore, at least 1,300 hedge funds, according to Wired [3], have used some type of machine learning for the majority of their trading activities, indicating a recent increase in the use of Artificial Intelligence (AI) in trading. This suggests that, while humans continue to play an important role in trading and investing, AI is becoming increasingly important.

AI can solve massive trading and investment difficulties. These difficulties are frequently related to forecasting, analysis, and optimization. However, there are currently no free and reliable tools for investors to use to support their investment decisions. As a result, I decided to investigate ways in which I could create a trading algorithm to maximise profits or, at the very least, attempt to accurately predict the future price of a stock.

## **Aims**

In this project, I will create a web application that an investor can use to help them make investment decisions. This tool will provide several advantages to investors, including:

* Discovering patterns- detect and replicate historical and replicating trading patterns which are often concealed from investors.  Normal humans cannot process as much data or recognize patterns as quickly as technology.
* Sentiment-Based trading - AI can forecast the movement of stocks and other traders by evaluating media headlines, comments on social media, blog posts, and other data. This is done through sentiment analysis, which is the process of categorizing people's text-based viewpoints (or feelings).
* Trading prediction based on historical data - Using supervised ML models, we can predict the future prices of stocks.

## **Objectives**

To meet the aims this project sets out to achieve, I have broken it down into several tasks. These objectives are as follows:

1. Obtain real-time and historical equity data from Alpha Vantage API.
2. Clean data and form data sets with the obtained data.
3. Build python functions to calculate technical indicators from the obtained dataset.
4. Train an LSTM, Random forest, and Ridge regression model to predict the S&P 500.
5. Test various models to find which one works best for predicting Standard and Poor’s 500 (S&P 500).
6. Fine-tune model parameters to have as low bias as possible while also having low variance on the training data.
7. Split datasets into training and test data, and train models with data.
8. Tests the models with data and measure accuracy. Metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R2) are used.
9. Evaluate which features have the most impact on stock prices.
10. Visualize results using a line chart showing the predicted prices versus the actual prices of the stock.
11. Implement a user-friendly interface for the prediction tool.

### **1.1.1 Changed objectives**

Following the feedback from my interim report to ‘add more up-to-date technologies, some objectives have been changed. These include:

* Train a Ridge Regression, Random Forest, and Long Short-Term Memory models to predict the S&P 500.

## **Resources & Tools**

This project is built using the Python programming language [16]. I chose to use Python because it is easy and flexible to use. Python is also versatile and has a robust collection of libraries that make machine learning tools easily available to use. Also, Python has a diverse pack of visualisation options available which makes it ideal for creating graphs and charts.

The Python code will be written using Jupiter notebook [17]. It is a web application that allows users to create documents containing live code and visualizations. In addition, I have used various Python libraries to access tools that have enabled me to build my project.

The main library used in my project is Scikit-learn [18]. It offers a consistent Python framework for a variety of supervised and unsupervised algorithms. This library's stack contains the following items:

* NumPy: Base n-dimensional package
* Matplotlib: Comprehensive 2D/3D plotting
* Pandas: Data structures and analysis

# Chapter 2

## **Literature Review**

This chapter will discuss the current literature that will be used to establish the context for this project. Such literature is based on topics relating to techniques used to forecast the future price of a stock. I will also go into detail about the current state of AI technology development in investing and how companies around the world are utilising this technology.

These developments need to be taken on further into our development contexts. Technological progress enables more efficient production of more and better goods and services, on which prosperity is based.

## **2.1. Impact of AI on Trading and Investing**



Figure 2-Photo credit: [3]

Before commencing with the development of the System, I did some background research on other institutions currently using AI in finance to evaluate the current technology development.

“Machine Learning is progressing much faster, and financial institutions are among the early adopters.” Intelenet Global Service's vice president of global business growth, Anthony Antenucci, recently said [5]. Companies all over the world are pioneering advanced products and technology that are using AI to make buying and selling stocks more computerized and effective. Bloomberg reported in September 2017 that Japan's third-largest lender is going to use AI in the equities sector through automated processes for institutional clients [6]. In the United States, Merrill Lynch is currently testing a stock platform to find value in small-cap stocks that analysts would otherwise ignore [7].

Currently, there are companies already using AI for smarter trading. Aquan, a UK data science company [8] uses its platform to democratize trading by encouraging statisticians from different backgrounds to develop algorithmic trading strategies to aid in the resolution of investment problems. As a result, investment clients will benefit from data science without having to invest in costly in-house expertise. Aquan has had a significant industry impact, they graduated from Techstars in 2018 and were recently named the 2019 Europa Awards’ Hottest Fintech n Europe.

Similarly, EquBot’s[9] exclusive trading software, which is affiliated with IBM, blends AI with an active exchange-traded fund (ETF). The business centralizes the investment process by collecting and analyzing data from different sources (newspaper articles, social media posts, financial statements) from various parts of the world to "develop a cause-and-effect comprehension of economies, businesses, and managerial staff". EquBot has also had a significant impact on the industry; most recently, they launched the AI-powered Foreign Equity ETF, which aims to invest in established international markets outside of the United States.

The current state of AI technology development in investing demonstrates that as AI and deep-learning models become more sophisticated, Wall Street's key figures will be forced to invest in technological advances to compete. The average investor, on the other hand, does not have free access to this technology. This project's goal is to provide investors with a free AI tool to help them make investment decisions.

## **2.2 Stock Market Prediction**

There are two separate trading schools of thinking, all driven by the need to forecast market fluctuations and profit: fundamental, and technical analysis. This project will be focusing on the latter.

### **2.2.1 Fundamental Analysis**

The study of economic factors that affect the price of a stock is known as fundamental analysis. A balance sheet and an income statement are examples of such causes. Throughout the year, these reports are published quarterly. Fundamental analysis is commonly used to predict long-term market fluctuations because it depends on forecasts that are published based on a longer period.

Fundamental analysis can help you identify the company that outperforms its competitors. Furthermore, it is very effective because it determines the actual value of the stock. This analysis can also be used to identify growth prospects in a specific industry.

Although fundamental analysis offers several benefits, it is very time-consuming as many documents need to be considered while conducting the analysis.

### **2.2.2 Technical Analysis**

Chart, histogram

Description automatically generated

Figure 3- Relative Strength Index

Technical research attempts to predict what other stakeholders are thinking based on available knowledge about stock prices and volume. To forecast future prices, technical analysts use a range of metrics derived from the experience of stock price and volume.

Overall, the main concept of technical analysis is trend. According to technical analysts, stock price trends are triggered by a discrepancy between supply and demand for stocks, which is reflected in the bid and ask prices. Scientific analysts strive to derive trends from the chaotic data of stock prices. Since it depends on visual analysis, technical analysis is mostly qualitative. Figure 3 shows an example of such stock charts. It represents one year’s worth of historical stock price data for the IT sector in 2016.

In this proposed project, I will implement three prediction models: Ridge Regression, Random Forest, and Long Short-Term Memory (LSTM), with the goal of evaluating investment opportunities on various stocks in the LSE and determining whether they are profitable based on their historic prices. These systems will be trained using 65% of the historical data from the previous two years, and tested to see which one produces better results using the remaining 35% of the historical data.To assess the performance of each model, various evaluation metrics will be used.

ML is a technique that has been used in a variety of areas, including science and medical research, such as using machine learning models to determine whether someone has cancer, and financial research, such as forecasting stock market movements. Using data from several global stock markets, Jasic and Wood (2004) [10] created an artificial neural network to predict regular stock market index returns. The main goal is to encourage efficient trading. Short-term stock market index return forecasts are provided using a framework focused on univariate neural networks with untransformed data inputs. The Standard and Poor’s 500 Index (S&P 500), the German DAX index, the Japanese TOPIX Index, and London’s Financial Times Stock Exchange Index are all include in the analysis (FTSE All Share). The S&P 500, DAX, and FTSE Index samples range from January 1, 1965 to November 11, 1999. Since data from years was not available, the TOPIX sample spans the years from January 1, 1969 to November 11, 1999. The neural network’s prediction efficiency is compared to that of a benchmark linear autoregressive model, and the prediction improvement is verified when applied to the S&P 500 and DAX indices.

Enke and Thawornwong (2005) [10] test the predictive relationships for a variety of financial and economic variables using a machine learning knowledge gain technique. A ranking of the variables is obtained by calculating the information gain for each model variable. Only the most important variables are held in the forecasting models after a threshold is calculated. The capacity of neural network models for level estimation and classification to provide an accurate forecast of future values is investigated. The generalizability of multiple models is also improved using a cross-validation technique. S&P data from March 1976 to December 1999 were used to compare the models. The findings show that trading strategies led by classification models produce higher risk-adjusted profits than buy-and-hold, other neural network models, and linear regression models.

The following research employs a stochastic time efficient neural network model to discover the predictive relationships between a variety of financial and economic variables (Liao and Wang, 2010) [10]. Investors are assumed to make investment decisions based on historical stock market data, which is weighted according to how close it is to the present. The greater the effect of historical data on the predictive model, the closer the data is to the present. The model's efficacy is assessed utilizing data of each trading day over an 18-year period, spanning December 19, 1990 to June 7, 2008.

Several conclusions can be drawn from the studies listed above. First, artificial neural networks are effective at forecasting numerical stock market index values. Although each study demonstrated that ANN can be used effectively, single method application can have its limitation. Secondly, the generalizability of the studies needs to be improved. Some of the studies discussed above assess their ML system using a single stock market and/or time. This is a flaw because they fail to consider how effective their systems will be in various scenarios.

Our solution will improve on the research mentioned above because LSTMs outperform traditional feed-forward neural networks and RNNs. This is due to their ability to recall patterns selectively over long periods of time. In addition, our solution will include technical indicators to get better results, as opposed to previous study which only focused on the financial information.

# **Chapter 3**

# **Methodology**

This chapter will go over the methodology that will be used to create accurate models for forecasting the share prices of companies in the LSE. There are several components of the methodology used to create a functional model that will be discussed in depth. The following sections must be completed to create an efficient model for this project:

* Technical Indicators
* Stock Dataset
* Train Model
* Test Model
* Evaluate Model

Diagram

Description automatically generated

Figure 4-Training Structure of Model

## **3.1 Technical Indicators**

Although it is impossible to predict whether the stock will increase or decrease in value, we can, however, make educated guesses and forecasts based on the information we have now and in the past about any stock. Technical Analysis refers to an educated guess based on past stock price movements and patterns.

Technical Analysis (TA) can be used to forecast the price direction of a stock, but it is not always accurate. A few other investors have criticized TA, claiming it's just as effective at forecasting the future as Astrology. However, some investors believe in it and have had long and glorious investing careers.

In this project, the ML models we'll be using will make use of TA to help them make more informed predictions. In TA, technical indicators are used to determine whether analyze the momentum of a stock. For example, if investors are overbuying or over-selling the stock. I will be using technical indicators as features in my models to get better results. Technical indicators are calculated to evaluate brief market volatility, they are also useful for helping value investors identify when to buy a stock and when to sell it.

I will be using momentum technical indicators. These indicators measure the rate at which a security’s price moves, and there are several metrics that can be used to do so including:

* Relative Strength Index (RSI) [11]- this is a popular momentum indicator that can be used to determine whether investors are buying or selling a particular stock when demand unjustifiably drives the price of a stock upwards, it is said to be overbought. This is commonly considered an indicator that the stock is overvalued and is likely to decline in price. When a stock’s price falls dramatically below its true worth, it is oversold. Because of panic sales, this is the outcome. The RSI scale runs from 0 to 100, and a reading of more than 70 suggests that the stock is overbought, whereas a reading of less than 30 implies that the stock is oversold.

**Formula:**

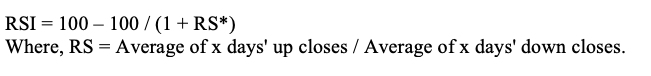


Figure 5- RSI Formula

**Code:**

The code for RSI calculation is shown in appendix A. I started off by identifying the up days (days where the stock price went up) and the down days (days where the stock price went down). I created a column that identifies the price change, then used a condition that set the value of up and down days based on the price change. In addition, I ensured the values for down days were absolute by modifying the column and calculating the Exponential Moving Average of both the UP and Down columns. The final step was to calculate the Relative Strength metric and pass that through to the RSI calculation.

* Stochastic Oscillator [12] – this tracks the price’s momentum. In certain cases, momentum shifts before the price shifts. It calculates the closing price in relation to the low-high range for a given time span.

**Formula:**

Graphical user interface, text, application

Description automatically generated

Figure 6- Stochastic Oscillator Formula

**Code:**

The code I used to calculate the Stochastic Oscillator is shown in appendix B. To calculate Stochastic Oscillator, I use the rolling lambda function. With this function, I specify a 14-period window, and what measurement to apply to each window. After the maximum and minimum values are obtained, I then pass it through my formula and apply the results to the main data frame.

* Williams %R [13] varies from -100 to 0. It shows a sell signal when its value is greater than -20, and a buy signal when the value is less than -80.

**Formula:**

Text

Description automatically generated

Figure 7- Williams %R Formula []

**Code:**

The code I used to calculate the Stochastic Oscillator is shown in appendix C.

* Price Rate of Change (ROC) [14]- this indicator compares the price of a stock currently to that of a predetermined period. An upward change in price causes the indicator to move into positive territory and vice versa.

**Formula:**

Graphical user interface, text, application

Description automatically generated

Figure 8 -ROC Formula []

**Code:**

The code I used to calculate the Stochastic Oscillator is shown in Appendix E.

* Moving Average Convergence Divergence (MACD) [15]- this indicator depicts the connection between two security price moving averages. By subtracting the 26-period exponential moving average (EMA) from the 12-period EMA the MACD is estimated. The "signal line," a nine-day EMA of the MACD, is then plotted on top of the MACD line. When the MACD begins to decline underneath the SingalLine, it indicates a sell signal. As it rises above the SignalLine, it indicates a buying opportunity.

**Formula:**



Figure 9-MACD Formula []

**Code:**

Appendix D contains the code I used to calculate the MACD. I used the stock's closing price to calculate the MACD. After obtaining this column, I then applied the transform method along with the specified Lambda function. Then I proceeded to calculate the EMA by calling the Exponential Moving Weight (emw) function and specifying how many periods to look back. I used the default definition provided by the formula and specified 26 & 12.

After calculating the 26-period EMA and the 12-period EMA, I took the difference between both to obtain the MACD. In addition to the MACD, I also wanted to calculate the EMA of the MACD, so I applied the same ewm function, but in this case, I specified a span of 9. Finally, both the MACD and MACD\_EMA were added to the main data frame.

## **3.2 Stock Dataset**

Table

Description automatically generated

Figure 10- real-time and historical time series data

The figure above shows data was obtained from Alphavantage API [19]. The columns represent the following:

* Open- the starting price of the stock for each day
* Close- the stock's closing price for that trading day
* High - top price for that trading day
* Low- cheapest price for that trading day
* Adjusted Close- amended close price that reflects the value of a stock after accounting for any corporate actions.
* Volume- the number of shares of a stock exchanged on a stock exchange in that day.
* Dividend amount- the amount of dividend paid in that day.

A picture containing graphical user interface

Description automatically generated

Figure 11-Correlation heatmap of obtained data

For the modelling I used Open and High columns combined with technical indicators as they have the highest correlation with the closing price which is the variable we are trying to predict. Although there are several more highly correlated columns such as Low and Adjusted close, adding them to the model will not improve the prediction accuracy.

## **3.3 Data Preparation**

Data preparation is a critical step in ML as data is frequently imperfect, untrustworthy, and/or deficient in specific habits or patterns, as well as rife with errors. Pre-processing data is a tried-and-true way of addressing such problems. It entails transforming data into a meaningful format.

### **3.3.1 Missing values**

A picture containing text

Description automatically generated

Figure 12- check for missing values in dataset

Even though the dataset used in this project was from a reliable source, I still checked the data for missing values. Some rows had missing values after the technical indicators were calculated, so they had to be removed. I did this by deleting the row with missing values or by removing the entire column if the missing values were greater than 75 percent.

This method is only recommended when the data set contains enough samples. It is critical to ensure that there is no additional bias after the data has been deleted. Removing the data will result in information loss, which will result in the predicted output not yielding the expected results.

### **3.3.2 Scaling of Features**

In the dataset, I used Sciki-learn’s [18] MinMaxScaler to scale the features. This means that the range of the variables was limited, thus allowing comparison on common ground. For example, in the dataset pictured above the ‘Open and Volume columns do not have the same scale, and this will cause some issues on the ML models.

Text

Description automatically generated with medium confidence

Figure 3- Euclidean Distance (ED) Formula

The majority of ML models are based on ED, so, therefore, it is easy to see that the Open column in the data will be dominated in ED, and we must try to avoid this. No all models are based on ED,but scaling of the features are still important. The reason for this is because without feature scaling, the algorithm may run for a very long time.

### **3.3.3 Splitting the data**

Text

Description automatically generated

Figure 4-model split code snippet

Before building any ML model, the data set needs to be split into two separate sets:

1. Training Set- this is the data split used to fit our model, which observes, learns, and optimises its parameters based on this data.
2. Test Set- a sample of data that our models have never seen before. It is used to evaluate the model's performance by simulating scenarios that may occur when the model is used in real life.

This step is important because the model we use are nothing more than estimation techniques that learn the historical patterns in the data. To avoid overfitting, I made every effort to ensure that the data sets for testing and training were as similar as possible. One method for achieving this as precisely as possible to choose the subsets at random. However, due to the nature of our data, it is not possible to randomly select subsets of the data, so I split the data into a 65:35 ration. This means that 65% of the data is used to train the model, and 35 percent is used for testing as shown above in figure 14.

Figure 14 depicts the separation of independent and dependent variables into several variables. The terms ‘X test' and ‘X train' refer to the training and testing portions of our features, while ‘y train' and ‘y test' refer to the training and testing portions of our dependent variable.X\_train- the training part of the matrix of features.

## 

# 3.4 Models

## **3.4.1 Supervised Learning**

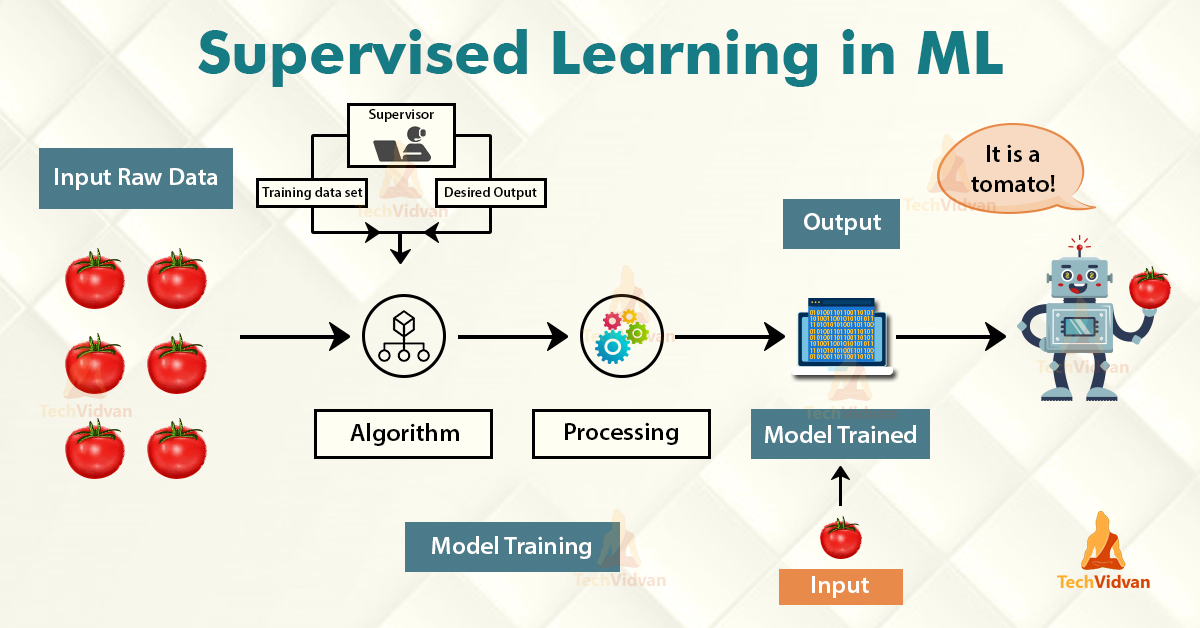


Figure 5- Supervised learning in ML

There are two main types of learning in ML: supervised and unsupervised. In unsupervised learning the model isn’t supervised, instead it is allowed to explore knowledge on its own. We accomplish this by giving the model an “unlabelled” dataset that does not specify which category or value is the “right” response. When we use supervised learning, we give the model a “labelled” dataset that tells it what the “right” value is.

### **3.4.1.1 Regression**

Regression is a supervised learning technique that aids in the discovery of variable correlations and allows one to forecast a continuous output variable using one or more feature variables. Prediction, forecasting, time series modelling, and evaluating the cause-effect relationship between variables are all common applications. In my project, a single output value is produced for the price of a stock using the training data provided. This a probabilistic interpretation that is determined by taking into account the strength of association between the input variables.

## **3.4.2 Random Forest**

Random Forest is a ML ensemble approach that is common due to its versatility, simplicity, and often high-quality performance. In this project, I use Random Forest algorithm to develop a regression model that attempts to predict closing prices of stocks based in the LSE on a range of technical indicators.

Random Forest is made up of Decision Trees, which are the basic building blocks. In essence, Decision Trees are a flowchart framework in which each node tests a different attribute of an entity. Consider the following scenario: I have an individual who will represent our object. We then put this person’s qualities to the test. One test would be to determine whether they are male or female. In our tree, the test will be a “Decision Node” and each of the potential outcomes, “Male” or Female” will be a leaf node. Out “Root Node” will be the first “Decision Node” in our Decision Tree.

**Key Words**

* **Root Node**- The entire population or sample is represented by this node. It is the beginning of the decision tree.
* **Splitting**- The act of dividing a node into a number of sub-nodes
* **Decision Node**- A node that splits into more sub-nodes
* **Leaf/Terminal Node**- this node does not split
* **Pruning-** removing sub-nodes from a decision node.
* **Branch** - a small tree within another tree.
* **Parent Node-** a node split into sub-nodes
* **Child Node**- the sub-nodes of a node that has been split

Diagram

Description automatically generated

Figure 6-Bagging (Bootstrap Aggregation) Flow

Ensemble Learning models are models that uses decisions from multiple models combined to increase the model’s overall efficiency. Ensemble Learning Is perfectly summed up by the adage that two heads are better than one. To get a better understanding of what the true response is, we combine the effects of the several models. Bagging is used for decision trees to increase model consistency in terms of reducing variance and improving accuracy, which removes the problem of overfitting.

In ensemble ML, bagging takes multiple poor models and aggregates the predictions to find the strongest one. The weak models specialise in specific areas of the feature space, allowing bagging leverage predictions to come from any model to achieve the highest level of accuracy.

Random Forest is typically more accurate than single decision trees for several reasons:

1. Instability: Even minor changes to the input data can have a significant impact on the decision tree’s overall structure.
2. They are always insufficiently reliable. With similar data, several other predictors perform better.
3. Knowledge benefit in decision trees is skewed in favour of attributes with more levels when data contains categorical variables of different number of levels.
4. Calculations can become extremely complicated, particularly when multiple values are unknown and/or multiple values are unknown and/or multiple outcomes are related.

The aforementioned are some of the reasons why Random Forest is superior to Decision Tree because it can help solve some of their flaws. There is no ideal model, as there is for everything. Just because something has flaws doesn’t mean it’s worthless; it simply means we need to be aware of them and keep an eye out for them while we use it.

### **3.4.2.1 Drawbacks**

Although Random Forest comes with its advantages, are also some drawbacks which are discussed below:

* Random forest models are similar to black boxes, they lack model interpretability.
* It isn’t advantageous to use Random Forest on large datasets as the trees the size of the trees can consume a large amount of memory.
* Because Random Forest has a tendency to overfit, you should tune the hyperparameters.

## **3.4.3 Ridge Regression**

### **3.4.3.1 Simple Linear Regression**

Chart

Description automatically generated

Figure 7-Simple linear regression example

Simple linear Regression is a mathematical technique for extracting a formula to predict the

values of one Y variable from the values of another variable X when both variables have a causal relationship. X is called the independent variable and Y is called the dependant variable. It is called ‘simple’ because it only examines the relationship between two variables. It is linear because when the independent variable increases (or decreases), the dependent variable increases (or decreases) in a linear fashion.

Chart, scatter chart

Description automatically generated

Figure 8-Simple linear regression formula

As shown in figure 10, the goal is to obtain a relationship (model) between the X (Number of Years of Experience) and the Y (Salary) variable. Once the coefficients m and b are obtained, we will have obtained a simple linear model. This trained model can be later used to predict any salary based on the number of years of experience an employee has.

A picture containing diagram

Description automatically generated

Figure 9-Least Sum of Squares

We need to find the Least Sum of Squares to get our model parameters. Least squares fitting is a method for determining the curve or line that best fits a range of points. The best fit curve or line is calculated using the sum of the squares of the offsets (residuals). The coefficients m and b are obtained using the least squares process, as shown in figure 12.

### **3.4.3.2 Regularisation**

Regularisation techniques are used to avoid networks overfitting. This is when a model provides great results on the training data but performs poorly on the testing set.

### **3.4.3.3 Ridge Regression**

Diagram, schematic

Description automatically generated

Figure 10-Overfitting Data Example

Ridge regression is advantageous because it avoids overfitting. Ultimately, we want a model that can generalise patterns. works best on the training and testing data. Ridge regression works by applying a penalizing term (reducing the weights and biases) to overcome overfitting. As shown in figure 13, least sum of squares is applied to obtain the best fit line. Since the line passes through 3 training dataset points, the sum of squared residuals = 0. However, for the testing dataset, the sum of residuals is large, so the line has a high variance. Variance means that there is a difference in fir (or variability) between the training dataset and the testing dataset. This regression model is overfitting the training dataset.

Diagram, schematic

Description automatically generated

Figure 11-Ridge Regression Example

Ridge regression works by attempting to increase the bias to improve variance (generalisation capability). This works by changing the slope of the line as shown in figure 14. Although the model performance might be slightly poorer on the training set, but it will perform consistently well on both the training and testing datasets. Due to the slope being reduced with ridge regression penalty, the model becomes less sensitive to changes in the independent variable.

A picture containing line chart

Description automatically generated

Figure 12- L2 regularisation

Ridge regression applies a factor of sum of squares of coefficients to the optimisation goal, which is known as L2 regularisation as shown in figure 15. Here, α (alpha) is the parameter that balances the importance of minimising RSS (Residual Sum of Squares) vs Minimising the number of square coefficients. α can have a variety of values.

As alpha increases, the regression line's slope decreases, and the line becomes more horizontal. Furthermore, the model becomes less susceptible to changes in the independent variable.

### **3.4.3.4 Drawbacks**

* Ridge regression increases bias.
* When building a Ridge regression model, you need to pick the perfect alpha (hyper parameter).
* The model's interpretability is poor.

## **3.4.4 Long Short-Term Memory (LSTM)**

Neural networks (NN) are a set of algorithms based on how the brain works. Neurons in our brain that can identify objects around us analyse the data we see when our eyes are open. NNs can analyse large amounts of data and identify and output trends from it.

NNs are not natural, like the neurons in our brains, so therefore they are called Artificial Neural Network (ANN). ANNs, like adults and children, can learn by example. They are typically created for a specific purpose, such as recognising voices or identifying patterns.

A picture containing pencil

Description automatically generated

Figure 13-Feed Forward ANN

Icon

Description automatically generated

Figure 14-Recurrent Neural Network [12]

Vanilla networks (feedforward neural networks) map a fixed size input (such as an image) to a fixed size output (classes or probabilities). Feedforward networks have the disadvantage of having no time dependence or memory impact. A recurrent neural network (RNN) is a type of artificial neural network (ANN) that is designed to understand the temporal dimension by providing a memory (feedback loop).

A picture containing text, clock

Description automatically generated

Figure 15-RNN architecture [13]

RNNs have a temporal loop in which the secret layer not only outputs anything, but also feeds itself. Time has been added as an extra. RNN will remember what happened in the previous time stamp, so it's ideal for text sequences.

Diagram

Description automatically generated

Figure 16-RNN math

A RNN takes an input of x and produces an output of o. The performance o is independent. The input x by itself, on the other hand, is dependent on the entire background of inputs fed to the network in previous time steps. Figure 19 depicts the two equations that control the RNN.

A LSTM is a type of RNN. In several ways, LSTMs outperform traditional feed-forward neural networks and RNNs. This is due to their ability to recall patterns selectively over long periods of time. This is since LSTMs store information in a memory like that of a machine. LSTM has several abilities, including reading, writing and deleting data from memory. Its memory is identical to a gated cell, in the sense that the cell decides whether the information is erased based on the value the information is given. This value is a ‘weight’ which shows how useful the data is. LSTM will learn how useful information is over time.

Diagram, schematic

Description automatically generated

Figure 17- an illustration of RNN with three gates [10]

As shown in figure 16, the input, forget, and output gates are all present in an LSTM. These gates decide whether new input should be allowed (input gate), whether it should be deleted (forget gate), or whether it should have an effect on the output at the current timestep (output gate). Each gate consists of a sigmoid neural net layer along with a pointwise multiplication operation. Sigmoid output ranges from 0 to 1, where 0 does allow data to flow and 1 allows everything to flow.

Diagram

Description automatically generated

Figure 18-Vanishing gradient problem

Since they avoid the vanishing gradient problem, LSTM networks outperform traditional RNNs. When an error must propagate across all previous layers, a vanishing gradient is the product shown in figure 21. The network weights are no longer changed as the gradient decreases. The gradients of the loss function approach zero as more layers are added, making the network difficult to practise.

With the recent advances in data science, it has been discovered that Long Short Term Memory networks, also known as LSTMs, are the most powerful solution for almost all of these sequence prediction problems. LSTM network has even been trained to write movies [14]. The LSTM network was trained with corpus of dozens of sc-fi screenplays and movies dating back from the 1980s and 90s.

### **3.4.4.1 Drawbacks**

* Because they could solve the problem of vanishing gradients, LSTMs became popular. However, they fail to completely remove it.
* They necessitate a significant investment of resources and time to be properly trained and prepared for real-world applications.
* Overfitting is a problem with LSTMs, and the dropout algorithm is difficult to use to combat it. Dropout refers to a regularisation technique that leaves out input and recurrent connections to LSTM units from activation and weight updates while training a network.

## **3.5 Model Evaluation**

Text

Description automatically generated

Figure 19- Example Evaluation Metrics

In data science, model evaluation is critical. It assists you in understanding the quality of each model, thus allowing you to determine which works best with the current and future data. Various models can be used to evaluate the regression model, I will use the following for this project:

## **3.5.1 R Square/Adjusted R Square**

R Square is a measure of how well the model explains the variance in the predictor variables. R Square is named after the square of the Correlation Coefficient (R). The calculation for R-Squared can be seen in figure 30. When the R-squared is lower, it means that there is a good match between the predicted and actual values. R-squared can be calculated using the Scikit-learn library's r2 score() function.

Diagram, text

Description automatically generated

Figure 20-R Square formula

R-squared is a useful metric, however, it fails to account for the problem of overfitting. This has therefore led to the introduction of Adjusted R-squared, which penalises the addition of new independent variables to the model and adjusts the measuring system to avoid overfitting.

From the sample model shown in figure 29, the model can explain 61 percent of the dependent variability, and the Adjusted R-square is similar to R-square, indicating that the model is quite stable.

## **3.5.2 Mean Square Error (MSE)/Root Mean Square Error (RMSE)**

In this project, we include the Mean Square Error to assess the absolute best for the fit. The formula for this calculation is shown below. MSE can be calculated using the Scikit-learn library's Mean square error() function. Figure 31 depicts the formula for calculating MAE.

Text

Description automatically generated

Figure 21-Mean Square Error Formula

For two reasons, the Root Mean Square Error (RMSE) is the preferred metric of evaluation in this project. To begin with, MSE is sometimes too large to compare. Second, because it is on the same level as the prediction, the square root is easier to interpret. Both can range from 0 to infinity, and the lower the score the better the model.

## **3.5.3 Mean Absolute Error (MAE)**

MAE can be compared to the other two evaluation metrics discussed in chapters 3.5.1 and 3.5.2. It is on the same level as the prediction, just like RMSE. MAE, on the other hand, is linear and thus intuitive. It can range from 0 to infinity, and the lower the score the better the model. MAE can be calculated using the Scikit-learn library's Mean absolute error() function. Figure 32 depicts the formula for calculating MAE.

Text

Description automatically generated

Figure 22-MAE Formula

# **Chapter 4**

## **Implementation and Evaluation**

In this project, I have implemented three different ML models to predict the price of a company’s stock. In this chapter, I will give an in-depth description of how each of them was implemented.

## **4.1 Modules and Packages**

The ML models in this project have been implemented using Python 3. Various Python packages were used to implement this project, including the following:

* NLTK[20] - The Natural Language Toolkit (NLTK) is a leading platform for developing Python programs that work with human language data. It offers simple interfaces to over 50 corpora and lexical resources, including WordNet, as well as a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, as well as wrappers for industrial-strength NLP libraries.
* Scikit-Learn[18] - Scikit-learn is a free and open-source machine learning library that can perform both supervised and unsupervised learning. It also includes tools for model fitting, data pre-processing, model selection and evaluation, and a variety of other utilities.
* Matplotlib[21] - Matplotlib is a Python library that allows you to create static, animated, and interactive visualisations.
* Pandas[22] - Pandas is an open source data analysis and manipulation tool built on top of the Python programming language that is fast, powerful, flexible, and simple to use.
* Numpy[23] - NumPy is a Python library that is used to work with arrays. It also includes functions for working with linear algebra, the Fourier transform, and matrices.
* Seaborn[ 24] – Seaborn is a Python library for creating statistical graphics. It is built on top of matplotlib and closely integrates with Panda’s data structures. Seaborn assists you in exploring and comprehending your data. Its plotting functions work on data frames and arrays containing entire datasets, performing the necessary semantic mapping and statistical aggregation domestically to produce informative plots. Its dataset-oriented, declarative API enables us to focus on what the different elements of your plots signify instead of how to draw them.
* AlphaVantage[ 19]- Alpha Vantage API provides access to historical and real-time data for a variety of markets. The API allows me to access the data directly in python, from there I can manipulate the data or store it for later use. Alpha Vantage provides its service at no fee. They permit 5 requests per minute and 500 requests per day.
* Yfinance[ 25]- Yfinance provides a dependable, threaded, and Pythonic method for downloading historical market data from Yahoo! Finance.
* Plotly[26]- Plotly is a Python graphing library that is interactive, open-source, and browser-based. It is a high-level, declarative charting library built on top of plotly.js, containing over 30 chart types, including Scientific charts and 3D graphs.
* Tensorflow [27]- TensorFlow is a complete open-source ML platform. It has an extensive, adaptable ecosystem of tools, libraries, and community resources that allow researchers to push the boundaries of ML and developers to quickly create and deliver ML-powered applications.
* Streamlit [28]- It is an open-source Python library that makes it easy to create and share beautiful, custom web apps for ML and data science.

## **4.2 Streamlit User Interface (UI)**

In this project, I use Streamlit to create a custom web application for the machine learning models I created.

### **4.2.1 Query Parameters**

Graphical user interface, application, Teams

Description automatically generated

Figure 23-Example Query Parameters

Query Parameters are used to obtain data from Alpha Vantage and build the ML models. The user inputs the following parameters before any calculations are done within the application:

* **Start Date**- the data obtained from the Alpha Vantage API's beginning period
* **End Date**- the data obtained from the Alpha Vantage API's end period
* **Stock Symbol**- a string of characters (usually letters) that represents publicly traded companies on an exchange.
* **Number of Days**- the trading window we are trying to predict.
* **Model**- ML model used to train data.

### **4.2.2 Company Description**

Graphical user interface, text

Description automatically generated

Figure 24-Example Company Description

Based on the stock symbol selected by the user, the company name and description are retrieved from the yfinance library using Python. Because the description is often long and contains unnecessary information NLTK- the Natural Language Toolkit is applied to summarize and shorten the text [28]. This is done in the following steps[33]:

Step one: Import the libraries needed:

The main libraries need for this feature are:

1. from nltk.corpus import stopwords- a data set containing pre-determined stop words
2. from nltk.tokenize import word\_tokenize, sent\_tokenize - tokenizer for words and sentences.

Step 2: Stop Words are being removed and kept in a different word array. Any word such as ‘is’ or ‘for’ that adds nothing to the a sentence's meaning As an example,

“Osato is one of the best programmers for freelance work “, a few words can be removed to reduce the word count whilst preserve the meaning as follows: [“Osato”, “one”, “best”, “programmers”, “freelance”, “work”].

Step 3: A frequency table of words created. After removing the stop words, a Python dictionary will keep track of Count how many times every word occurs in the responses.

Step 4: using the words contained in every sentence and a frequency table every sentence is assigned a value. Secondly, we track the score assigned to each sentence using a dictionary. Later, the dictionary is iterated through to create a summary.

# Text tokenization

1. haltWords = set(stopwords.words("english"))
2. phrases = word\_tokenize(message)

# Making a frequency table to keep track of

# each word's score

1. frequencyTable = dict()

Step 5: To compare the sentences in the feedback, a score is assigned. Finding the average can be a good criterion for comparing scores.

1. sumOfValues = 0
2. for i in sentenceWorthiness:

    sumOfValues += sentenceWorthiness [i]

1. average = int(sumOfValues / len(sentenceWorthiness))

Code: Text Summarizer is fully implemented in Python is shown in appendix I.

### **4.2.3 Closing Price**

Chart

Description automatically generated

Figure 25- Example Closing Price Chart

In the web application, the selected stocks closing price is also analysed. Figure 35 shows trendline used to indicated whether the behaviour of a stock has been positive or negative overtime. Although stocks generally fluctuate over the day, the overall behaviour persists over time. So therefore, understanding trends enables predictive analysis.

I used NumPy's polynomial fitting function, polyfit(), to identify the best adjusting first degree polynomial and its fitting error in order to identify patterns. The slope and offset of a first-degree polynomial are its two parameters. If the slope is greater than zero, there is a positive trend in the data, otherwise vice versa.

### **4.2.4 Data Correlation**

Graphical user interface, application

Description automatically generated with medium confidence

Figure 26-Correlation Matrix using Seaborn’s heatmap functionality

Identifying and reducing highly correlated features in a dataset is one of the quickest ways to strengthen a model. Correlated features will add noise and inaccuracy to a model, making it more difficult to achieve the anticipated result.

Figure 36 may appear a little perplexing at first glance. It is, however, relatively simple. On both axes, each feature (variable) is listed, and their relationships with other variables are coloured. The darker the colour, the more highly correlated those variables are and should not be paired in the same model. For this project, only variables that are highly correlated with Closing Price are considered.

The opening price of the stock and its highest price are the two features used to build the ML models in this project. When selecting features, the principle of Occam's razor was used, because after two features were chosen, the number of features had no effect on the model's performance.

### **4.2.5 Descriptive Statistics**

A picture containing text, monitor, black, screen

Description automatically generated

Figure 27-Descriptive Statistics

Descriptive statistics are an important part of machine learning because they provide us with a deeper understanding of the data; ignoring these insights often leads to incorrect conclusions. It also presents the data in a meaningful way, as illustrated in figure 37, allowing for easier interpretation of the data.

Measures of central tenancy (mean) are used in the web application to describe the dataset by identifying a central position. Furthermore, variability measures (interquartile range) are used to quantify the amount of spread or variability within the data.

### **4.2.6 Models**

#### **4.2.6.1 Ridge Regression**

The code for the implementation of this model can be found in Appendix F. The following steps were taken to create the model:

1. Obtain stock data from Alpha Vantage-

Table

Description automatically generated

Figure 28- Vanguard 500 Index Fund ETF

In this step the relevant stock data is obtained from Alpha Vantage. The dataset is already clean, so therefore no data cleansing is required at this stage.

1. Calculate Technical indicators-

Table

Description automatically generated

Figure 29- Technical Indicators

In this step, the following Technical indicators are calculated and added to the dataset:

* Stochastic Oscillator
* Williams %R
* Moving Average Convergence Divergence (MACD)
* Price Rate of Change

The data is cleaned after the Technical indicators are calculated, and any null values are removed.

1. Determine the trading window we want to forecast - Because the primary goal of this project is to forecast future stock prices, we must change the data so that the target stock price today is tomorrow's price, depending on how far into the future we want to forecast. This will be the ML model's target variable. The code below is used to accomplish this.

Graphical user interface, text, application

Description automatically generated

1. Creating Feature and Scale Data - To build the ML model, we only use two feature variables: the stock's opening price and its highest price. These are chosen from the dataset, along with the technical indicators calculated in step 2. The dataset's other columns are all removed. The data is scaled to numbers between zero and one using Scikit-MinMaxScaler. To accomplish this, the code below is used.

Text

Description automatically generated

1. Create the training and test datasets - we will train our model on the training set and then test it on the test set. This is referred to as the holdout-validation method. Because it is not possible to select subsets of our data at random, I divided it in a 65:35 ratio. As shown in the code snippet below, 65 percent of the data is used to train the model, while 35 percent is used for testing.

Text

Description automatically generated

1. Create regression model- The first step is to load the Scikit-Learn library and instantiate the algorithm that will be used which is called Ridge. Second, the model is fitted to the training data, which helps the model learn and predict. Once the model has been built on the training set, predictions on the test set can be made to provide a final, unbiased performance measure of the entire model. The code below is used to accomplish this.

Graphical user interface, text, application

Description automatically generated

1. **Model Evaluation-** Once the model is built, it must be evaluated to determine how well it represents the data and how well it will perform in the future. We will use Apple stock data from the last two years to evaluate this model. The model will also be used to forecast 1, 30, and 60 days in the future to see how it performs as time goes on.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 30- Evaluation Metrics for Ridge Regression After 1 Day

Graphical user interface, application

Description automatically generated

Figure 31-Evaluation Metrics for Ridge Regression After 30 Day(s)

A screenshot of a computer

Description automatically generated with medium confidence

Figure 32- Evaluation Metrics for Ridge Regression After 60 Day(s)

Figures 40, 41 and 42 show that when we use our ridge regression model to predict what the price of a stock will be in a day, the model can explain around 95 percent of the dependent variability, but when the number of days is increased to 30, this drops to 20 percent, indicating a poorer fit between predicted and actual values. After 60 days the R-Squared turns negative, which means that the model does not follow the trend of the data. The adjusted R Square is roughly the same as R Square, indicating that the model is quite robust.

#### **4.2.6.2 Random Forest**

The code for implementing the Random forest model can be found in Appendix G. Steps 1–6 for constructing this model is the same as those in Chapter 4.2.6.1. However, building the model is slightly different because a different algorithm from the Scikit-Learn library is used called RandomForestRegressor.

Text

Description automatically generated

The code snippet above demonstrates how the model is constructed. It has a total of 1000 estimators. This means that there are 1000 decision trees split at different features to produce the most diverse results possible. After the model has been constructed, it is fitted to the training data.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 33-Evaluation Metrics for Random forest After 1 DayA screenshot of a computer

Description automatically generated with medium confidence

Figure 34-Evaluation Metrics for Random forest After 30 Day

Graphical user interface, application

Description automatically generated

Figure 35-Evaluation Metrics for Random forest After 60 Day

Figures 43, 44 and 45 show that when we use our Random forest model to predict what the price of a stock will be in a day, the model can explain around 41 percent of the dependent variability, but when the number of days is increased to 30, this drops to 12 percent, indicating a poorer fit between predicted and actual values. After 60 days the R-Squared turns negative, which means that the model does not follow the trend of the data. The adjusted R Square is roughly the same as R Square, indicating that the model is quite robust.

#### **4.2.6.3 Long short-term memory (LSTM)**

The steps for creating the LSTM model differ slightly from those for the previous two models. This is primarily because LSTM expects the data fed into the model to be in a specific format, typically a 3D array. This means we need to reshape our arrays from 1D to 3D using the code shown below.

Text

Description automatically generated

Additionally, when building this model, I Keras which is Tensor Flow 2’s high-level API. TensorFlow 2 is a complete machine learning framework that is open source. It's a differentiable programming infrastructure layer. It brings together four main abilities:

* Using the CPU, GPU, or TPU to efficiently perform low-level tensor operations.
* The gradient of arbitrary differentiable expressions can be efficiently computed.
* Scaling computing to many devices.
* Programs ("graphs") can be exported to browsers and mobile phones.

Keras provides the required technology for creating high-performing ML solutions. The three main modules imported from Keras in my project are:

* Dense- this is used to add a layer of fully connected neural networks.
* LSTM- this is used to add a layer of Long Short-Term Memory.
* Dropout- this is used to add dropout layers to avoid overfitting.

The following arguments are passed to the LSTM layer::

* 150 units, which is the output space's dimensionality.
* Return\_sequences=True which decides if the last output in the output should be returned.
* Input which is the shape of the training set

Text

Description automatically generated

Dropout layers are defined by a value of 0.3, which means that 30% of the layers will be removed. When the model is compiled, the popular Adam optimizer is used. Furthermore, the loss is set to ‘mse'. As a result, the mean of the squared errors will be calculated. Finally, the model is configured to run every 20 epochs with a batch size of 32. The code snippet for this is shown below.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 36-Evaluation Metrics for LSTM After 1 Day

Graphical user interface, application

Description automatically generated

Figure 37-Evaluation Metrics for LSTM After 30 Day

Graphical user interface, application

Description automatically generated

Figure 38-Evaluation Metrics for LSTM After 60 Day

Figures 46, 47, and 48 show that when we use our Random forest model to predict what the price of a stock will be in a day, the model can explain around 86 percent of the dependent variability, but when the number of days is increased to 30, this drops to 38 percent, indicating a poorer fit between predicted and actual values. After 60 days, the model is still able to explain 5 percent.

#### **4.2.6.3 Overall Evaluation**

Overall, Random forest was the poorest model the Random forest model was the one with the lowest performance as the R-Squared was very low even after just one day at 41 percent and after 60 days the R-Squared was negative, which suggests that which means that the model does not follow the trend of the data. The best performing models on the other hand LSTM and Ridge regression, but I believe LSTM is better because as the number of days increases to 60 the R-squared value did not drop to below zero.

### **4.2.7 Profit and loss calculations**

Graphical user interface

Description automatically generated

Figure 39-Profit calculation for AAPL stock using Ridge Regression for 1 Day

A user can calculate how much money they will make in a given number of days by using the query parameters shown in chapter 4.2.1. To begin, the current share price is determined, so that a user can determine how many shares they can purchase with their initial investment amount. The initial number of shares purchased is then multiplied by the price in the given number of days when the shares are sold. Finally, the initial investment amount is subtracted from the current share price to determine how much profit a user has made from their investment.

### **4.2.7 Profit and loss calculations**

After evaluating the models produced in this project, I decided to implement a trading strategy using them. The algorithm used in this section is very simple, it is based upon predicting the closing price of the stock and if this price is more than a predefined threshold, investors buy the stock else they sell. The code for this algorithm is referenced in appendix J.

Chart

Description automatically generated

Figure 40- Algorithm performance on test data

The trading algorithm is applied to Apple Inc data between 2015-01-01 to 2021-05-06. As shown in figure 44, the algorithm seems to perform well on the 35 percent of test data it has never seen before. According to its estimations, our investor would be £10 up in profit from an original £10 investment. However, this is still a poor return because it is across 500 days.

# **Chapter 5**

## **Testing**

In this chapter, I'll look at the best model from the previous chapter for predicting company stock prices in the past. Because the LSTM model performed the best, I will use it to make predictions.

To ensure objectivity, the strategy model must be tested on a variety of different types of stocks. Stocks are classified into various categories based on the characteristics of specific types of companies, the products they offer, and whether they are more or less likely to be popular at specific times:

* Defensive- these stocks perform well even when the economy suffers, and they provide goods and services that people require regardless of their financial situation. This includes companies that manufacture pharmaceuticals, food, and energy.
* Cyclical- the performance of these stocks is determined by the state of the economy. A bank, for example, suffers greatly during a recession when people begin to withdraw their savings or cannot afford their mortgage.
* Income- these stocks refer to shares in well-known companies. These companies, like a wife, are usually more stable and reliable. These businesses have established suppliers, so you can bet they'll be around for a while. BT and Shell are two examples of this.
* Growth- These stocks are typically very volatile and have had little time to mature. Amazon and ASOS are two examples of such stocks.

## **5.1 General Electric (Defensive Stock)**

Graphical user interface

Description automatically generated

Figure 41- LSTM prediction Results for GE After 1 Day

* LSTM achieved an R-squared of 0.566
* LSTM achieved an Adjusted\_Rsquared of 0.561
* LSTM achieved an MAE of 0.046
* LSTM achieved an RMSE of 0.052

## **5.2 HSBC (Cyclical Stock)**

A screenshot of a computer

Description automatically generated with medium confidence

Figure 42- LSTM prediction Results for HSBC After 1 Day

* LSTM achieved an R-squared of 0.969
* LSTM achieved an Adjusted\_Rsquared of 0.969
* LSTM achieved an MAE of 0.029
* LSTM achieved an RMSE of 0.035

## **5.3 IBM (Income Stock)**

A screenshot of a computer

Description automatically generated with medium confidence

Figure 43-LSTM prediction Results for IBM After 1 Day

* LSTM achieved an R-squared of 0.917
* LSTM achieved an Adjusted\_Rsquared of 0.916
* LSTM achieved an MAE of 0.024
* LSTM achieved an RMSE of 0.034

## **5.4 Amazon (Growth Stock)**

A screenshot of a computer

Description automatically generated with medium confidence

Figure 44-LSTM prediction Results for Amazon After 1 Day

* LSTM achieved an R-squared of 0.879
* LSTM achieved an Adjusted\_Rsquared of 0.877
* LSTM achieved an MAE of 0.052
* LSTM achieved an RMSE of 0.069

## **5.5 Summary**

The tests show that the LSTM model I created has a high level of performance and accuracy for the various types of stocks that were tested. The General Electric stock has a low R-squared of 56 percent and a high R-squared of 96 percent for HSBC, which is outstanding. The predicted and actual values are both very similar, as shown by the line graphs.

# **Chapter 6**

## **Critical Appraisal**

In this chapter, I will provide a quick overview of my project as well as a critical assessment of the work that has been completed. There will also be a review of the social, sustainability, commercial, and economic background, as well as a brief assessment of my own personal growth during the project.

## **6.1 What Went Well?**

Overall, we can say that this project was a success. All the goals I set out to achieve in Chapter 1 have been met. I developed the models in this project to a satisfactory level and tested them on a variety of stocks.

In addition to meeting the initial objectives, I was able to incorporate an additional feature, which is the trading algorithm discussed in Chapter 4.2.8. A user guide has also been created to assist users in navigating the tool and thus improving their experience.

## **6.2 What could be improved**

### **6.2.1 Frontend**

The front-end of my project is one area where I can make significant improvements. I used the Streamlit framework to create it. While this framework provided numerous benefits to the project, it also had some drawbacks. For example, because the customisation of the frontend components is limited, I was constrained in the design of the web application produced.

Furthermore, because the framework is still in its early stages, some of the features are still in beta. This means they are untrustworthy and cannot be relied on. Streamlit's scalability is also limited, which means that the web application will be unable to accommodate growth.

### **6.2.2 Prediction Algorithms**

The prediction algorithms developed in this project, in my opinion, still have a lot of room for improvement. Some of the things that could be improved in the future are the technical indicators used, the buy/sell algorithm/hyperparameters, and the model architecture.

One major flaw in the algorithms developed for this project is that they only consider past price movement, with no regard for future fundamentals. This is a flaw because the stock market is one of the most complex systems ever devised by humanity, and it cannot be consistently beaten. These systems can be improved by considering additional factors. For example, using sentiment analysis on new articles to improve model prediction accuracy.

## **6.3 What would you do differently?**

### **6.3.1 Frontend Improvement**

If I were to do this project again, I would use Flask in conjunction with HTML and CSS rather than Streamlit. Flask is a backend framework that allows users to create and deploy applications. The main reason for this change is that Flask does not limit me to only data applications. Furthermore, Flask has been thoroughly tested and is not in beta, which means that its features are more reliable and trustworthy.

### **6.3.2 Sentiment Analysis (SA)**

Diagram

Description automatically generated

Figure 45- Stock Sentiment Analysis

Another thing I would do differently is adding Sentiment Analysis as an objective of this project. SA refers to the process of textual contextual mining that identifies and brings out subjective information from different sources including Twitter and news headlines. This allows an investor to efficiently monitor the conversations going on about a particular asset. An examination of ex-President Donald Trump's 14,000 tweets, for example, revealed that tweets move the stock market [29]



Figure 46- Tweet from Instagram influencer Kylie Jenner

Figure 46 depicts a similar example of Tweets influencing stock prices. Kylie Jenner, a social media influencer, made a negative tweet about SNAP in February 2018. With 39 million followers, the share price fell by 7%, and SNAP's market value dropped by $1.3 billion [30]. This further shows support for SA in stock market prediction.

### **6.3.2 Portfolio optimisation**

Portfolio optimisation is another tool that I would add to the web application. This refers to the process of creating a portfolio of assets, which allows the investor to maximise returns with minimal risk. This can be implemented using Modern Portfolio Theory (MTP) [], it is a mathematical process that enables investors to maximise their returns for a given risk level.

### **6.3.3 Algorithmic Trading**

Algorithmic trading is the use of algorithms to execute trades automatically. These algorithms typically include parameters such as timing, price, quantity, and other restrictions. This provides investors with a competitive advantage. If I had the chance to do this project again, I would include this. Despite the fact that I have already implemented a training algorithm, it is very simple and will not allow traders to optimise their returns.

## **6.4 Benefits and Risks to Society**

### **6.4.1 Benefits to Society**

Society will gain countless hours of productivity through the use of AI technology in investing. Humans will be able to spend their time doing more productive things, such as fundamental analysis once the stress of data analysis is removed. Furthermore, AI technology in trading eliminates mistakes and human error. By identifying and capitalising on trading opportunities, investors will be able to increase their profits.

### **6.4.2 Risks to Society**

As a result of AI, our workforce will undoubtedly evolve. Humans are at risk of losing their jobs to machines, which also makes it difficult for them to rediscover their passion with new responsibilities that require their uniquely human abilities. According to PWC research, AI will replace 7 million existing jobs in the UK between 2017 and 2037. This uncertainty, as well as changes in how some people will make a living, could be difficult.

Will machines become super-intelligent to the point where humans lose all control? This is a matter of some debate, but we do know that when new technology is introduced, there are always unforeseen consequences. AI's unforeseen consequences will almost certainly pose a challenge to all of us.

Another risk is that AI will cross any ethical or legal boundaries. It will have a negative impact on society if it chooses to achieve the desired goal in a destructive (yet efficient) manner.

## **6.6 Commercial Context**

### **6.6.1 Advertisements**

My web application can be monetised to generate revenue for me. Ads within the application could be one way to accomplish this. These advertisements could take the form of banners or videos. I would be compensated each time a user clicked on an advertisement from my application via Google Ads.

### **6.6.2 Subscription Model**

Another way to monetize the application is to incorporate a subscription model into it. This is where the customer is charged a recurring fee to use the app under the subscription model. Payments are typically made on a monthly or annual basis, which is convenient for both personal customers and the business market.

## **6.7 Personal Development**

### **6.7.1 Python Programming**

Before starting this project, I had little experience with the Python programming language. This project has allowed me to understand and implement optimised ML algorithms, which has aided in the development of my technical skills. This project has also introduced me to several other libraries that I will be able to use in future projects, including:

* Streamlit
* NLTK
* Scikit-learn

### **6.7.2 Machine Learning applications in finance**

Through completing this project, I have been able to develop my knowledge of the different use cases of ML in finance. ML algorithms are used for the following:

* Fraud detection
* Automation of trading activities
* Providing financial advice to investors.

### **6.7.3 Time management**

Diagram

Description automatically generated

Figure 47- 4 Quadrants of Effectiveness

I was able to improve my time management skills while working on this project. Proper time management was essential for increasing personal productivity and ensuring the project's success. I used a technique known as the Covey Time Management Matrix [32], which employs a quadrant system, as shown in fig 47, to help me categorise my tasks throughout the project based on:

* Urgency: tasks that require immediate action.
* Importance: tasks with high significance.

### **6.7.4 Summary**

The skills obtained from this project are very relevant to my future career. As a future Fintech Software Engineer, having a good understanding of Python programming language and the different applications of ML in finance is very relevant to my career. In addition, time good time management skills are also relevant in and out of the workplace.

Overall, I believe the skills learnt in this project, has helped me improve as an individual and further prepared me for a career in Fintech.

# **Conclusion**

The purpose of this paper was to develop a suitable model for predicting future LSE stock prices. LSTM, Random forest, and Ridge regression models were investigated and evaluated, and after extensive testing, LSTM was selected as the best option. The model did not disappoint me, as I was able to achieve a relatively high R-squared, MAE and RMSE for all of the stocks presented.

# **Bibliography**

[1] Statitsa. (2021) Market value of all companies trading on the London Stock Exchange (LSE) from January 2015 to March 2021 [online] Available at: <https://www.statista.com/statistics/324578/market-value-of-companies-on-the-london-stock-exchange/>

[2] Use Cases of AI and Machine Learning in Finance [online] Available at: <https://marutitech.com/ai-and-ml-in-finance/>

[3] Wired. (2016) The Rise of the Artificially Intelligent Hedge Fund [online] Available at: <https://www.wired.com/2016/01/the-rise-of-the-artificially-intelligent-hedge-fund/>

[5] ITProToday. (2018) How AI Trading Systems Will Shake Up Wall Street. [online] Available at: <https://www.itprotoday.com/machine-learning/how-ai-trading-systems-will-shake-wall-street>

[6] Bloomberg. (2017) Mizuho Is Said to Offer AI Trading Before MiFID Overhaul [online] Available at: <https://www.bloomberg.com/news/articles/2017-09-20/mizuho-is-said-to-offer-ai-trading-service-before-mifid-overhaul>

[7] The Cybersecurity Observatory. Artificial Intelligence is transforming Investment Strategies [online] Available at: <https://www.cybersecobservatory.com/2017/06/06/artificial-intelligence-transforming-investment-strategies/>

[8] Auquan [online] Available at: <https://www.auquan.com/>

[9] Equbot [online] Available at: <https://equbot.com/>

[10] Scholarworks. (2020) Machine Learning Stock Market Prediction Studies: Review and Research Directions [online] Available at: <https://scholarworks.lib.csusb.edu/cgi/viewcontent.cgi?article=1435&context=jitim>

[11] Investopedia. (2021) Relative Strength Index (RSI) [online] Available at: <https://www.investopedia.com/terms/r/rsi.asp>

[12] Investopedia (2021) Stochastic Oscillator [online] Available at: <https://www.investopedia.com/terms/s/stochasticoscillator.asp>

[13] Investopedia (2021) Williams %R Definition and Uses [online] Available at: <https://www.investopedia.com/terms/w/williamsr.asp>

[14] Investopedia (2020) Price Rate of Change Indicator (ROC) [online] Available at: <https://www.investopedia.com/terms/p/pricerateofchange.asp>

[15] Investopedia (2021) Moving Average Convergence Divergence (MACD) [online] Available at: <https://www.investopedia.com/terms/m/macd.asp>

[16] Python 3.9.5 Documentation [online] Available at: <https://docs.python.org/3/>

[17]Jupyter Notebook 6.2.0 Documentation [online] Available at: <https://jupyter-notebook.readthedocs.io/en/stable/>

[18] Scikit-learn 0.21.3 Documentation [online] Available at: https://scikit-learn.org/0.21/ documentation.html

[19] Alphavantage Documentation [online] Available at: <https://www.alphavantage.co/documentation/>

[20] NLTK 3.5 Documentation [online] Available at: https://www.nltk.org/

[21] Matplotlib 3.3.2 Documentation [online] Available at: https://matplotlib.org/users/ index.html

[22] Pandas Documentation [online] https://pandas.pydata.org/docs/ user\_guide/index.html

[23] Numpy 1.19 Documentation [online] https://numpy.org/doc/stable/

[24] Seaborn [online] Available at: <https://seaborn.pydata.org/introduction.html>

[25] yfinance 0.1.59 Doumentation [online] Available at: <https://pypi.org/project/yfinance/>

[26] Plotly Documentation[online] Available at: <https://plotly.com/python/>

[27] TensorFlow 2.4.1 Documentation [online] Available at: <https://www.tensorflow.org>

[28] Streamlit Documentation [online] Avialable at: <https://docs.streamlit.io/en/stable/>

[29] Barron’s (2019) Yes, Trump’s Tweets Move the Stock Market. But Not for Long [online] Available at: <https://www.barrons.com/articles/donald-trump-twitter-stock-market-51567803655>

[30] DailyFX (2020) Using Sentiment Analysis to Examine Stocks [online] Available at: <https://www.dailyfx.com/education/understanding-the-stock-market/stock-market-sentiment-analysis.html>

[31] Investopedia (2021) Mordern Portfolio Theory (MPT) [online] Available at: <https://www.investopedia.com/terms/m/modernportfoliotheory.asp>

[32] Bytestart Covey’s Matrix: the simple secret to great time management [online] Available at: <https://www.bytestart.co.uk/coveys-matrix-time-management>

[33] GeekforGeeks (2020) Python | Text Summarizer [online] Available at: <https://www.geeksforgeeks.org/python-text-summarizer/>

# **Appendix A**

**Indicator Calculation: Relative Strength Index (RSI)**

# calculating the Relative Strength Index  
  
def calc\_RSI(data\_dated):  
 # Calculate the 14 day RSI  
 n = 14  
  
 # First make a copy of the data frame twice  
 up\_df, down\_df = data\_dated[['Ticker', 'change\_in\_price']].copy(), data\_dated[['Ticker', 'change\_in\_price']].copy()  
  
 # For up days, if the change is less than 0 set to 0.  
 up\_df.loc['change\_in\_price'] = up\_df.loc[(up\_df['change\_in\_price'] < 0), 'change\_in\_price'] = 0  
  
 # For down days, if the change is greater than 0 set to 0.  
 down\_df.loc['change\_in\_price'] = down\_df.loc[(down\_df['change\_in\_price'] > 0), 'change\_in\_price'] = 0  
  
 # We need change in price to be absolute.  
 down\_df['change\_in\_price'] = down\_df['change\_in\_price'].abs()  
  
 # Calculate the EWMA (Exponential Weighted Moving Average), meaning older values are given less weight compared to newer values.  
 ewma\_up = up\_df.groupby('Ticker')['change\_in\_price'].transform(lambda x: x.ewm(span=n).mean())  
 ewma\_down = down\_df.groupby('Ticker')['change\_in\_price'].transform(lambda x: x.ewm(span=n).mean())  
  
 # Calculate the Relative Strength  
 relative\_strength = ewma\_up / ewma\_down  
  
 # Calculate the Relative Strength Index  
 relative\_strength\_index = 100.0 - (100.0 / (1.0 + relative\_strength))  
  
 # Add the info to the data frame.  
 data\_dated['down\_days'] = down\_df['change\_in\_price']  
 data\_dated['up\_days'] = up\_df['change\_in\_price']  
 data\_dated['RSI'] = relative\_strength\_index

# **Appendix B**

**Indicator Calculation: Stochastic Oscillator**

# Claclulting the Stochastic Oscillator  
def stochastic\_Oscillator(data\_dated):  
 # Calculate the Stochastic Oscillator  
 n = 14  
  
 # Make a copy of the high and low column.  
 low\_14, high\_14 = data\_dated[['Ticker', '3. low']].copy(), data\_dated[['Ticker', '2. high']].copy()  
  
 # Group by symbol, then apply the rolling function and grab the Min and Max.  
 low\_14 = low\_14.groupby('Ticker')['3. low'].transform(lambda x: x.rolling(window=n).min())  
 high\_14 = high\_14.groupby('Ticker')['2. high'].transform(lambda x: x.rolling(window=n).max())  
  
 # Calculate the Stochastic Oscillator.  
 k\_percent = 100 \* ((data\_dated['4. close'] - low\_14) / (high\_14 - low\_14))  
  
 # Add the info to the data frame.  
 data\_dated['low\_14'] = low\_14  
 data\_dated['high\_14'] = high\_14  
 data\_dated['k\_percent'] = k\_percent

# **Appendix C**

**Indicator Calculation: Williams %R**

# calculating williams R%  
def calc\_williams\_r(data\_dated):  
 # Calculate the Williams %R  
 n = 14  
 # Make a copy of the high and low column.  
 low\_14, high\_14 = data\_dated[['Ticker', '3. low']].copy(), data\_dated[['Ticker', '2. high']].copy()  
 # Group by symbol, then apply the rolling function and grab the Min and Max.  
 low\_14 = low\_14.groupby('Ticker')['3. low'].transform(lambda x: x.rolling(window=n).min())  
 high\_14 = high\_14.groupby('Ticker')['2. high'].transform(lambda x: x.rolling(window=n).max())  
 # Calculate William %R indicator.  
 r\_percent = ((high\_14 - data\_dated['4. close']) / (high\_14 - low\_14)) \* - 100  
 # Add the info to the data frame.  
 data\_dated['r\_percent'] = r\_percent

# **Appendix D**

**Indicator Calculation: Moving Average Convergence Divergence (MACD)**

def calc\_macd(data\_dated):  
 # Calculate the MACD  
 ema\_26 = data\_dated.groupby('Ticker')['4. close'].transform(lambda x: x.ewm(span=26).mean())  
 ema\_12 = data\_dated.groupby('Ticker')['4. close'].transform(lambda x: x.ewm(span=12).mean())  
 macd = ema\_12 - ema\_26  
  
 # Calculate the EMA  
 ema\_9\_macd = macd.ewm(span=9).mean()  
  
 # Store the data in the data frame.  
 data\_dated['MACD'] = macd  
 data\_dated['MACD\_EMA'] = ema\_9\_macd

# **Appendix E**

**Indicator Calculation: Price Rate Of Change**

def calc\_price\_rate\_of\_change(data\_dated):  
 # Calculate the Price Rate of Change  
 n = 9  
  
 # Calculate the Rate of Change in the Price, and store it in the Data Frame.  
 data\_dated['Price\_Rate\_Of\_Change'] = data\_dated.groupby('Ticker')['4. close'].transform(  
 lambda x: x.pct\_change(periods=n))

# **Appendix F**

**Ridge Regression Model**

# Building ridge regression model  
def pricePrediction\_LR(symbol, days, starting\_date, end\_date):  
  
 # obtain stock data  
 stock\_df = stock\_data(symbol, starting\_date, end\_date)  
  
 # obtaining technical indicators  
 stochastic\_Oscillator(stock\_df)  
 calc\_williams\_r(stock\_df)  
 calc\_macd(stock\_df)  
 calc\_price\_rate\_of\_change(stock\_df)  
  
 # set the trading window we are trying to predict  
 stock\_df\_targeted = trading\_window(stock\_df, days)  
  
 stock\_df\_targeted.reset\_index(inplace=True)  
 stock\_df\_targeted = stock\_df\_targeted.dropna()  
  
 stock\_df\_targeted\_scaled = stock\_df\_targeted.copy()  
 # drop unused columns in dataset  
 stock\_df\_targeted\_scaled.drop(  
 ['Ticker', '4. close', '7. dividend amount', '3. low', '5. adjusted close', '6. volume', '8. split coefficient',  
 'low\_14', 'high\_14', 'MACD\_EMA'], axis=1, inplace=True)  
  
 # scale feature and target  
 target\_price = stock\_df\_targeted\_scaled.filter(['Target'])  
 stock\_df\_targeted\_scaled = sc.fit\_transform(stock\_df\_targeted\_scaled.drop(columns=['date','Target']))  
 target\_price = y\_sc.fit\_transform(target\_price)  
  
 # Creating Feature and Target  
 X = stock\_df\_targeted\_scaled[:, :6]  
 y = target\_price  
  
 # splitting data into training and testing  
 split = int(0.65 \* len(X))  
 X\_train = X[:split]  
 y\_train = y[:split]  
 X\_test = X[split:]  
 y\_test = y[split:]  
  
  
# building and evaluating the model  
 regression\_model = Ridge(alpha=1)  
 regression\_model.fit(X\_train, y\_train)  
 last\_element = X\_test[len(X\_test) - 1]  
 original\_prices= stock\_df\_targeted['4. close'].values  
 current\_price= original\_prices[len(original\_prices) - 1]  
 last\_element=last\_element.reshape(1, -1)  
  
  
 # using model to predict the entire dataset  
 predicted\_prices = regression\_model.predict(X)  
  
 # using the model to predict the test data  
 eval\_predict = regression\_model.predict(X\_test)  
  
 # predicting future price  
 predicted\_price= regression\_model.predict(last\_element)  
  
 # calculating profits  
 profitInXDays= y\_sc.inverse\_transform(predicted\_price)  
 share\_amount= investment\_amount/current\_price  
 profit=round((share\_amount \* profitInXDays[0][0])-investment\_amount,2)  
  
 # getting predicted prices for for the test data  
 Predicted = []  
 for i in predicted\_prices:  
 Predicted.append(i[0])  
 # getting the original prices  
 close = []  
 for i in stock\_df\_targeted\_scaled:  
 close.append(i[0])  
 # adding predicted and actual prices to a data frame  
 df\_predicted = stock\_df\_targeted[['date']]  
 df\_predicted['Close'] = close  
 df\_predicted['Prediction'] = Predicted  
  
 # evaluating the prediction metrics  
 RMSE.append(math.sqrt(mean\_squared\_error(y\_test, eval\_predict)))  
 Rsquared.append(r2\_score(y\_test, eval\_predict))  
 Mae.append(mean\_absolute\_error(y\_test, eval\_predict))  
 adj\_Rsquared.append(1 - (1-r2\_score(y\_test, eval\_predict))\*(len(y\_test)-1)/(len(y\_test)-X.shape[1]-1))  
  
 # appending information to web page  
  
 interactive\_plot(df\_predicted, "Original Vs. Prediction")  
 st.info("in {} day(s) the price of this stock will be £{}".format(days, round(profitInXDays[0][0], 2)))  
 st.info("You would make £{} in {} day(s)".format(profit, days))  
  
 unscaled\_y\_test = y\_sc.inverse\_transform(y\_test)  
 y\_test\_predicted = y\_sc.inverse\_transform(eval\_predict)  
 unscaled\_y\_test = [item for sublist in unscaled\_y\_test for item in sublist]  
 y\_test\_predicted= [item for sublist in y\_test\_predicted for item in sublist]  
 data = {'Close': unscaled\_y\_test,  
 'Prediction': y\_test\_predicted}  
  
 st.header("Trading Algorithm")  
 # Create DataFrame  
 df = pd.DataFrame(data)  
 trade\_algorithm(df)  
 plot\_trades(unscaled\_y\_test, y\_test\_predicted, sells, buys)  
 compute\_earnings(buys, sells)

# **Appendix G**

**Random Forest**

# randomForest Model  
def pricePrediction\_RandomForest(symbol, days, start\_date, end\_date):  
  
 # obtain stock data  
 stock\_df = stock\_data(symbol, start\_date, end\_date)  
  
 # obtaining technical indicators  
 stochastic\_Oscillator(stock\_df)  
 calc\_williams\_r(stock\_df)  
 calc\_macd(stock\_df)  
 calc\_price\_rate\_of\_change(stock\_df)  
  
 # set the trading window we are trying to predict  
 stock\_df\_targeted = trading\_window(stock\_df, days)  
  
 # drop any row with null values  
 stock\_df\_targeted.reset\_index(inplace=True)  
 stock\_df\_targeted = stock\_df\_targeted.dropna()  
  
 stock\_df\_targeted\_scaled = stock\_df\_targeted.copy()  
  
 # dropping columns that were not used in the data  
 stock\_df\_targeted\_scaled.drop(  
 ['Ticker', '4. close', '7. dividend amount', '3. low', '5. adjusted close', '6. volume', '8. split coefficient',  
 'low\_14', 'high\_14', 'MACD\_EMA'], axis=1, inplace=True)  
  
 # scale feature and target  
 target\_price = stock\_df\_targeted\_scaled.filter(['Target'])  
 stock\_df\_targeted\_scaled = sc.fit\_transform(stock\_df\_targeted\_scaled.drop(columns=['date','Target']))  
 target\_price = y\_sc.fit\_transform(target\_price)  
  
 # Creating Feature and Target  
 X = stock\_df\_targeted\_scaled[:, :6]  
 y = target\_price  
  
 # splitting data into training and testing  
 split = int(0.65 \* len(X))  
 X\_train = X[:split]  
 y\_train = y[:split]  
 X\_test = X[split:]  
 y\_test = y[split:]  
  
  
# Building and fitting the model  
 rf = RandomForestRegressor(n\_estimators=1000)  
 rf.fit(X\_train, y\_train.ravel())  
  
  
 last\_element = X\_test[len(X\_test) - 1]  
 original\_prices = stock\_df\_targeted['4. close'].values  
 current\_price = original\_prices[len(original\_prices) - 1]  
 last\_element = last\_element.reshape(1, -1)  
 # predicting future price  
 predicted\_price = rf.predict(last\_element)  
 predicted\_price = predicted\_price.reshape(1, -1)  
  
 # making model prediction on the whole dataset  
 pred\_rf = rf.predict(X)  
 # making predictions on the test data  
 eval\_predict= rf.predict(X\_test)  
  
  
 # calculating profits  
 profitInXDays = y\_sc.inverse\_transform(predicted\_price)  
 share\_amount = investment\_amount / current\_price  
 profit = round((share\_amount \* profitInXDays[0][0]) - investment\_amount, 2)  
  
 # getting predicted prices for for the test data  
 Predicted = []  
 for i in pred\_rf:  
 Predicted.append(i)  
 # getting the original prices  
 close = []  
 for i in stock\_df\_targeted\_scaled:  
 close.append(i[0])  
 # adding predicted and actual prices to a data frame  
 df\_predicted = stock\_df\_targeted[['date']]  
 df\_predicted['Close'] = close  
 df\_predicted['Prediction'] = Predicted  
  
  
 # evaluating the prediction metrics  
 RMSE.append(math.sqrt(mean\_squared\_error(y\_test, eval\_predict)))  
 Rsquared.append(r2\_score(y\_test, eval\_predict))  
 Mae.append(mean\_absolute\_error(y\_test, eval\_predict))  
 adj\_Rsquared.append(1 - (1 - r2\_score(y\_test, eval\_predict)) \* (len(y\_test) - 1) / (len(y\_test) - X.shape[1] - 1))  
  
 # plottig actual vs predicted graph  
 interactive\_plot(df\_predicted, "Original Vs. Prediction for ")  
  
 # appending information to web page  
 st.info("in {} day(s) the price of this stock will be £{}".format(days, round(profitInXDays[0][0], 2)))  
 st.info("You would make £{} in {} day(s)".format(profit, days))

# trading algorithm

unscaled\_y\_test = y\_sc.inverse\_transform(y\_test)  
 y\_test\_predicted = y\_sc.inverse\_transform(eval\_predict.reshape(1, -1))  
  
 unscaled\_y\_test = [item for sublist in unscaled\_y\_test for item in sublist]  
 y\_test\_predicted= [item for sublist in y\_test\_predicted for item in sublist]  
  
 data = {'Close': unscaled\_y\_test,  
 'Prediction': y\_test\_predicted}  
  
 st.header("Trading Algorithm")  
 # Create DataFrame  
 df = pd.DataFrame(data)  
 trade\_algorithm(df)  
 plot\_trades(unscaled\_y\_test, y\_test\_predicted, sells, buys)  
 compute\_earnings(buys, sells)

# **Appendix H**

**LSTM Model**

# LSTM model  
def pricePrediction\_LSTM(symbol, days, start\_date, end\_date):  
 # obtain stock data  
  
 stock\_df = stock\_data(symbol, start\_date, end\_date)  
  
 # obtaining technical indicators  
 stochastic\_Oscillator(stock\_df)  
 calc\_williams\_r(stock\_df)  
 calc\_macd(stock\_df)  
 calc\_price\_rate\_of\_change(stock\_df)  
  
 stock\_df.reset\_index(inplace=True)  
  
 # set the trading window we are trying to predict  
 stock\_df\_targeted = trading\_window(stock\_df, days)  
  
 stock\_df\_targeted\_ = stock\_df\_targeted.copy()  
 stock\_df\_targeted.drop(  
 ['Ticker', '4. close', '7. dividend amount', '3. low', '5. adjusted close', '6. volume', '8. split coefficient',  
 'low\_14', 'high\_14', 'MACD\_EMA'], axis=1, inplace=True)  
  
 stock\_df\_targeted.dropna(inplace=True)  
 training\_data\_X = stock\_df\_targeted.iloc[:, 1:7].values  
 training\_data\_y = stock\_df\_targeted.iloc[:, 7:].values  
  
 stock\_df\_targeted\_scaled = sc.fit\_transform(stock\_df\_targeted.drop(columns=['date']))  
  
 X = sc.fit\_transform(training\_data\_X)  
 y = y\_sc.fit\_transform(training\_data\_y)  
  
 # Convert the data into array format  
 X = np.asarray(X)  
 y = np.asarray(y)  
  
 # Split the data  
 split = int(0.65 \* len(X))  
 X\_train = X[:split]  
 y\_train = y[:split]  
 X\_test = X[split:]  
 y\_test = y[split:]  
  
 # Reshape the 1D arrays to 3D arrays to feed in the model  
 X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))  
 X\_test = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1))  
  
 # Create the model  
 inputs = keras.layers.Input(shape=(X\_train.shape[1], X\_train.shape[2]))  
 x = keras.layers.LSTM(150, return\_sequences=True)(inputs)  
 x = keras.layers.Dropout(0.3)(x)  
 x = keras.layers.LSTM(150, return\_sequences=True)(x)  
 x = keras.layers.Dropout(0.3)(x)  
 x = keras.layers.LSTM(150)(x)  
 outputs = keras.layers.Dense(1, activation='linear')(x)  
  
 model = keras.Model(inputs=inputs, outputs=outputs)  
 model.compile(optimizer='adam', loss="mse", metrics=['mean\_squared\_error', 'mae'])  
 model.summary()  
  
 # Train the model  
 history = model.fit(  
 X\_train, y\_train,  
 epochs=20,  
 batch\_size=32,  
 validation\_split=0.2  
 )  
  
 predicted = model.predict(X)  
 eval\_predict = model.predict(X\_test)  
  
 last\_element = X\_test[len(X\_test) - 1]  
 original\_prices = stock\_df\_targeted\_['4. close'].values  
 current\_price = original\_prices[len(original\_prices) - 1]  
 last\_element = last\_element.reshape(1, -1)  
  
 # predicting future price  
 predicted\_price = model.predict(last\_element)  
 predicted\_price = predicted\_price.reshape(1, -1)  
  
 # calculating profits  
 profitInXDays = y\_sc.inverse\_transform(predicted\_price)  
 profitInXDays= round(profitInXDays[0][0], 2)  
 share\_amount = investment\_amount / current\_price  
 profit = round((share\_amount \* profitInXDays) - investment\_amount, 2)  
  
  
 test\_predicted = []  
  
 for i in predicted:  
 test\_predicted.append(i[0])  
  
 close = []  
 for i in stock\_df\_targeted\_scaled:  
 close.append(i[0])  
  
 df\_predicted = stock\_df\_targeted[['date']]  
 df\_predicted['Close'] = close  
 df\_predicted['Prediction'] = predicted  
  
 # interactive\_plot(df\_predicted, "Original Vs. Prediction for " )  
 scores = model.evaluate(X, y, verbose=0)  
  
# add evaluations to list  
 RMSE.append(math.sqrt(mean\_squared\_error(y\_test, eval\_predict)))  
 Rsquared.append(r2\_score(y\_test, eval\_predict))  
 Mae.append(mean\_absolute\_error(y\_test, eval\_predict))  
 adj\_Rsquared.append(1 - (1 - r2\_score(y\_test, eval\_predict)) \* (len(y\_test) - 1) / (len(y\_test) - X.shape[1] - 1))  
  
 # Plot the data  
 interactive\_plot(df\_predicted, "Original Vs Prediction")  
 # appending information to web page  
  
 st.info("in {} day(s) the price of this stock will be £{}".format(days, round(profitInXDays, 2)))  
 st.info("You would make £{} in {} day(s)".format(profit, days))  
  
  
 unscaled\_y\_test = y\_sc.inverse\_transform(y\_test)  
 y\_test\_predicted = y\_sc.inverse\_transform(eval\_predict)  
 unscaled\_y\_test = [item for sublist in unscaled\_y\_test for item in sublist]  
 y\_test\_predicted = [item for sublist in y\_test\_predicted for item in sublist]  
 data = {'Close': unscaled\_y\_test,  
 'Prediction': y\_test\_predicted}  
  
 st.header("Trading Algorithm using Testing Data")  
 # Create DataFrame  
 df = pd.DataFrame(data)  
 trade\_algorithm(df)  
 plot\_trades(unscaled\_y\_test, y\_test\_predicted, sells, buys)  
 compute\_earnings(buys, sells)

# **Appendix I**

**NLTK Text Summarizer [33]**

def summarise\_text(string\_summary):  
 # text to sum up  
 message = string\_summary  
  
 # Text tokenization  
 haltWords = set(sw.words("english"))  
 phrases = word\_tokenize(message)  
  
 # Making a frequency distribution table to record each word's score  
 frequencyTable = dict()  
 for word in phrases:  
 word = word.lower()  
 if word in haltWords:  
 continue  
 if word in frequencyTable:  
 frequencyTable[word] += 1  
 else:  
 frequencyTable[word] = 1  
  
 # Creating a dictionary to keep the score  
 # of each sentence  
 sentences = sent\_tokenize(message)  
 sentenceWorthiness = dict()  
  
 for i in sentences:  
 for word, freq in frequencyTable.items():  
 if word in i.lower():  
 if i in sentenceWorthiness:  
 sentenceWorthiness[i] += freq  
 else:  
 sentenceWorthiness[i] = freq  
  
 totalOfValues = 0  
 for i in sentenceWorthiness:  
 totalOfValues += sentenceWorthiness[i]  
  
 # Average value of a sentence from the original text  
  
 average = int(totalOfValues / len(sentenceWorthiness))  
  
 # Storing sentences into our summary.  
 textsummary = ''  
 for sentence in sentences:  
 if (sentence in sentenceWorthiness) and (sentenceWorthiness[sentence] > (1.2 \* average)):  
 textsummary += " " + sentence  
 return textsummary

# **Appendix J**

**Trading Algorithm**

# trade algorithm  
def trade\_algorithm(df\_predicted):  
 x=0  
 for actual , predicted in zip( df\_predicted['Close'], df\_predicted['Prediction']):  
   
 predicted\_price=predicted  
 price\_today=actual  
  
 delta= predicted\_price- price\_today  
  
 if delta > thresh:  
 buys.append((x,price\_today))  
 elif delta <= thresh:  
 sells.append((x,price\_today))  
 x +=1