SOLENT UNIVERSITY SOUTHAMPTON

FACULTY OF BUSINESS, LAW AND DIGITAL TECHNOLOGIES

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MODULE CODE: COM 726

MODULE TITLE: DISSERTATION

STUDENT ID: Q15016595

STUDENT NAME: ERIC ISIBOR

TITLE: Artificial Intelligence Algorithms for Stock Market Prediction

SUPERVISOR: JARUTAS ANDRITSCH

Submitted by Eric Isibor to Solent University Southampton

Authors Declaration.

I hereby declare that this dissertation titled "Artificial Intelligence Algorithms for STOCK Market Prediction" is mine. This project has not been submitted in any university. Resources obtained in the development of this thesis were all referenced.

Name & Signature

Abstract

The Information Technology (IT) field has continued to witness series of innovation and advancement. The Stock market sub-sector has also witness series of large capitalization, it is now a hub of increasing investment for investors. Based on this, many analysts, investors, and domain expert has developed series of statical tools to predict the outcome of the market. However, The stock market has witnessed transition from traditional statistical tools of stock prediction to an era of algorithmic trading built on the foundation of Artificial Intelligence. The artificial Intelligence approach has greatly improved the accuracy of stock market prediction but remain one of the most challenging tasks because the market is categorized has being volatile. The technical indicators for prediction like Open Price, Low Price, High Price, and Closed Price is not static with Time. Hence the outcome of the data is usually non-linear in relationship. The purpose of stock market prediction is to predict the future values of a company's stock. Recent prediction technologies have been developed to perform this task using machine learning by making prediction using the past and current values of stock to predict the future values through a training method. Machine learning uses different models to make prediction easy and simple.

This thesis is focused on deploying a classification model Random Forest Classifier, Sentiment Analysis as the base model for stock market trend while a Regression Model Long Short-Term Memory for prediction of next day close price for 10 global brands (Companies). The Random Forest Classifier was design to provide insight on the direction of the market to advise buyers/traders if to buy or not and also advising the users how confident is the system for every prediction knowing well that the market is very volatile. While the Long Short Term Memory was deployed to learn the dynamics of long term price for making price prediction into the future because the model has a capacity of story long term data which is very important knowing that stock market prices can be voluminous and being released per seconds. While a 3rd level decision support system was deployed to guild users using a sentiment analysis. Sentiment was considered as a factor to provide insight knowing the power of speculations and news headlines within the stock market.

The strength and weaknesses of the application of artificial intelligence were also discussed including the threats and opportunities and associated with this particular study and the unique addition to the body of knowledge using artificial intelligence algorithm in stock market prediction.

Acknowledgement

My gratitude goes to Jesus Christ for his grace, favor, strength to see me through in life and this phase of my master’s degree.
I also acknowledge my supervisor Dr Jarutas Andritsch. Thank you for your support, guidance and your ideas in making this thesis a better one. Your criticism were constructive and it made me a better researcher. Also, to Dr Femi Isiaq, module leader for this thesis. Thank you for your input and time in providing clarification regarding the stricture of this thesis.
To my family, I really want to say a big thank you for your support and encouragement. My parents Elder Edward Isibor Jp. and Deaconess Rose Isibor for your motivation and advice all through my journey of life. I also appreciate my siblings, Mrs Dorothy Tunde-Ajala for your support, Kelvin Isibor for your motivation and ideas. Osasu Isibor for your guidance all through this journey, Osazuwa, Osaigbovo and Edoba Isibor for your encouragement.
Lastly, I want to say thank to Solent University for providing me the platform to study Artificial Intelligence and Data Science. My journey with Solent University has been fruitful and valuable.

Table of Contents

Authors Declaration…………………………………………………………………………..2

Abstract…………………………………………………………………………………………3

Acknowledgement……………………………………………………………………………4

Table of Content ………………………………………………………………………………5

CHAPTER ONE ………………………………………………………………………………..6

1.0 Background and Introduction ………………………………………………………..6

1.1. Problem Definition……………………………………………………………………..7

1.2 Research Questions ……………………..8

1.3. Research Aim ……………………………………………………………………………8

1.4 Value Preposition………………………………………………………………………..8

1.5 Project Specification…………………………………………………………………..8

1.6 Tools & Packages ……………………………………………………………………….8

1.7 Model Suitability ………………………………………………………………………..11

1.8 Criteria for Selection of Company ………………………………………………….12

1.9 Chapter Summary & Conclusion…………………………………………………….12

1.9.1 Thesis Outline…………………………………………………………………………..13

CHAPTER TWO. ………………………………………………………………………………… 14

2.0 Literature Review ……………………………………………………………………… 14

CHAPTER THREE…………………………………………………………………………………16

3.0 Methodology………………………………………………………………………………..16

3.1 First Model-LSTM………………………………………………………………………..16

3.1.1 Data Collection…………………………………………………………………………16

3.1.2 Exploratory Data Analysis…………………………………………………………..18

3.1.3 Data Cleaning ……………………………………………………………………………19

3.1.4 Data Preprocessing & Scaling………………………………………………………

3.1.5 Training & Testing…….……………………………………………………………… 19

3.1.6 Tune & Further Train ……………………………………………………………...21

3.1.7 Model Evaluation & Result………………………………………………………….23

3.2. Second Model Random Forest Classifier ………………………………………..24

3.2.1 Data collection & Preprocessing ……………………………………………………24

3.2.2 Train Random Forest Classifier ………………………………………………………24

3.2.3 Feature Engineering, Tuning & Further Train…………………………………26

3.2.3 Model Performance, Result & Evaluation ………………………..…………………27

3.3. Sentiment Analysis……………………………………………………………………………27

3.3.1 Data Collection ………………………………………………………….…………………27

3.3.2 Data Cleaning……………..…………………………………………………………………28

3.3.3 Train Sentiment Model…………………………………………………………………..29

CHAPTER FOUR……………………………………………………………………………………31.

4.1Front End Development Life Cycle……………………………………………………31

4.2 System Design Flow Chat……………………………………………………………….31

4.3 Front-End Development Tools………………………………………………………….33

4.4 System Design Implementation……………………………………………………….34

4.5 Result of Study…………………………………………………………………………………. 36

4.5.1 Result-Price Prediction………………………………………………………………….36

4.5.2Result-Sentiment Analysis………………………………………………………………37

4.5.3Result – Market Trend……………………………………………………………………38

CHAPTER FIVE ………………………………………………………………………………………39

5.1 Challenges and Limitations……………………………………………………………….39

5.2 Conclusion…………………………………………………………………………………………41

References…………………………………………………………………………………………………43

CHAPTER ONE

1.0 Background & Introduction

The stock market is a market associated with the buying and selling of shares of Public Liability Companies (PLC). Shares are also regarded as equity. Companies use this instrument as a means to raise capital in response to business performance. There can be situations where a company might need more resources in response to business performance, it is expected that they will raise more capital by selling more shares to the public. The ownership of a share in a PLC is a representation of having a stake in a public listed company. The transaction is facilitated by a regulatory body that acts as an intermediary between buyers and sellers of shares.

One of the major challenges of Financial Market Intelligence expert is the inability to accurately predict the prices and behaviors of the market. The market is known to be dynamic and nonlinear *"The stock market is a complex nonlinear dynamic system that is affected by many factors, such as policies, social news events, the operation of companies and the psychological changes of investors*" (Tao et al 2022) For example the future share prices, as to what will be the value of share/stock in the future, will the price go up or down, should investors hold, buy or sell shares within a specified period of time ? These questions are very important to ensure profit margins are maintained. The Efficient Market Hypothesis(EMH) has argued that stock market prices cannot be predicted using any factor or indicators because the principles upheld that the price of a stock is unpredictable either by using the Technical or Fundamental Analysis. Technical analyses are indicators that use historical prices and volume to predict future prices while fundamental analyses are indicators associated with the financial performance of the company to predict future prices of stock. e.g profit. However, a different approach has been adopted to argue against the Efficient Market Hypothesis that stock market price can be predicted. Firstly, emphasis on Market predictions started solely based on assumptions of domain experts e.g This method has been unreliable because it can be subjective and might not address issues relating to environmental factors like sentiment. Secondly, Artificial Intelligence expert delved into algorithmic financial trading to study the behaviors of the market by using Traditional Machine Learning models to automate and predict the interactions of the market. This method seems to have achieved a higher level of accuracy because issues relating to bias were eliminated. Later, a deep learning neural network model was introduced to study the patterns of the market because of its ability to store large historical data. Some school of thought have argued that there is a relationship between sentiment as a result of news articles from media and the prices of stock *"In other words, after a market or stock sector has had a long up move with a sharp price appreciation over a sustained time, this is when it gains the attention of the media".*(Pearson, 2013). When most people receive this information they tend to buy stock and join the trend when prices are high. Hence the assumption that sentiment is regarded as a price indicator.

Based on the above, a deep neural network algorithm will be implemented to predict the future stock price using stock historical data and monitor the trend using tweets about the stock market for sentiment analysis. Part of this project is to build a stock price predictor application that can advise investors on how to deploy their resources. The application will predict the stock price of Ten global brands(Google, Meta, Tesla, United Parcel Service, Coca-Cola, Walmart, Chevron, Pfizer, Visa and Microsoft) using a Long Short Term Memory(LSTM) which is a deep learning model for time series problems. An LSTM model will be perfect for this project because we are dealing with stock price which is time-series data, and this data is generated over time in the form of intervals and sequences.

The companies were selected based because they belong to fortune 500 companies and they represent major players in industries like Finance, Technology, Social networks, E-commerce, Commodity, Automobile, Health, Oil and Gas, and Logistics. Among these companies are Google, Meta, Tesla, United Parcel Service, Coca-Cola, Walmart, Chevron, Pfizer, Visa and Microsoft. These companies' stocks will be used in the research prediction and analysis

1.1 Problem Definition.

- Deploy a deep learning model to anticipate the stock prices of Google, Meta, Tesla, United Parcel Service, Coca-Cola, Walmart, Chevron, Pfizer, Visa and Microsoft.

10 global brands(Google, Meta, Tesla, United Parcel Service, Coca-Cola, Walmart, Chevron, Pfizer, Visa and Microsoft) were selected as a case study for this research. To optimize profit margin for these brands the future price of stock must be defined for potential investors unfortunately the predicted price is unknown. The research problem of price prediction is regarded as a regression problem because the target variable (Price) is continuous in value. This is because the values are not specific, the result can be any value. The system should be able to predict stock prices.

.

- Develop a deep learning model that can advise on the profitability on when to buy stock from Google, Meta, Tesla, United Parcel Service, Coca-Cola, Walmart, Chevron, Pfizer, and Visa. And provide a confidence score for the prediction to buy.

Profitability decisions are taken into consideration based on cost-benefit analysis. The cost-benefit analysis associated with the decision to invest or not is unknown. This research will seek to answer whether it is profitable to buy or not based on if price will ‘go up’ or ‘fall’ This definition is a classification problem. Because the predicted outcome is classed as categorical, it predicts if the user should ‘Buy’ or ‘Sell’ based on the output. And provide a confidence score for the advise.

- A Graphic User Interface System/Platform for prediction.

An interaction between users and the predictive system will be required to determine the success of the research. To solve this problem a graphical user interface will be developed for users to log requests, interact and view the outcome of results based on prediction.

Based on the above problem(s), the research questions and aim are as follows.

1.2 Research Questions

- Can sentiment analysis help the investor predict the trend of a particular stock?
- How effective are Tweets from Twitter about stocks to help in the sentiment analysis of a particular stock?

1.3 Research Aim

The aim of this research is develop an Artificial Intelligence model using a deep learning & traditional machine learning approach for stock price prediction. That can guild investors to know the future values of stock if a stock will rise or fall using 10 global brands as a case study.

1.4 Value Preposition of this Project

The project or system will be of importance based on the following

- To identify risk and opportunities for stock market traders.
- To increase profit margin on Return on Investment (ROI) for Stock market traders.
- To support financial researchers in studying stock market behaviours.

1.5 Project *Specification*

Based on the above, a deep neural network algorithm and application will be implemented to predict the future stock price using stock historical data and tweets about the stock market for sentiment analysis. Part of this project is to build a stock price predictor application that will also include a sentiment analytics dashboard. This is a machine learning application that can predict the stock price of Ten Companies using a Long Short-Term Memory (LSTM) model, this model is a deep learning model for time series problems. While a classification task to advise traders if to buy or not using a confidence score rating for every prediction.

1.6 Tools / Packages will be used:

1. Long-short Term Memory

LSTM which stands for Long Short Term Memory is a deep neural network that can be applied to time series forecasting. I want to implement this deep neural network in this research because the stock market is a time series problem. A time series is a series of values of a quantity obtained through time or successive times, often with intervals between them. LSTM models are good at learning patterns of sequence data. It can learn a function that maps a sequence of past observations as input to an output observation.

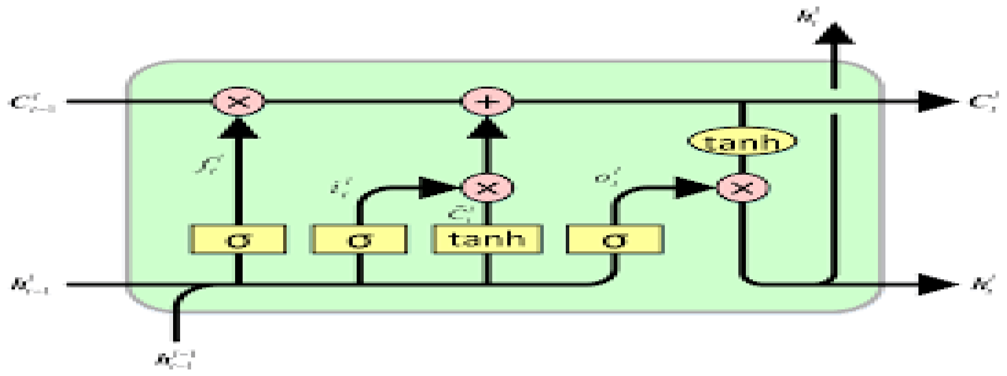


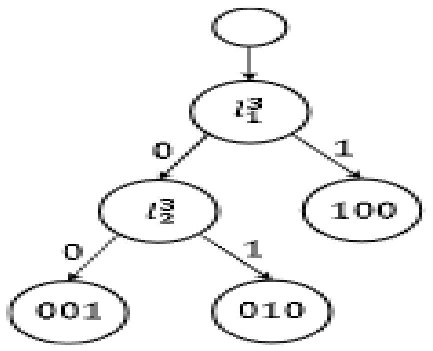
***Figure 1 Simple LSTM architecture**. Source: (versamopoulous 2018)*

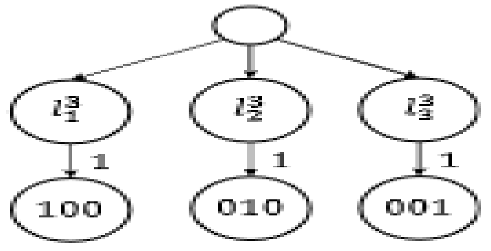
Stacked LSTM are multiple hidden LSTM layers Stacked one on top of another. It has been proven empirically that deep Recurrent neural networks work better than shallow networks on some tasks. How deep we can go on this task depends on our experimentation, amount of data, and access to computational power (GPU and Ram).

The use of market sentiment was also considered in the market analysis to determine the trend of the market.

2. Random Forest Classifier (RFC)

I will be implementing the Random Forest Classifier model to predict market trends and patterns. If the predicted price will 'rise' or 'fall'. This problem is a classification problem because the target outcome is a categorical variable hence we require a classification model for the implementation of market trends. Random Forest Classifier is a supervised learning model that uses a decision tree algorithm to make prediction. '*A Random Forest is a method of ensemble learning where multiple classifiers are generated and their results are aggregated*'(Borges and Neves, 2020). The aggregation of result is made possible by the model randomly picking each decision tree to make prediction for the targeted class.





3. Sentiment Analysis - Twitter.

Expression of public or personal opinion on news articles and headlines regarding a company and its product is a suitable source of data for predicting the changes in the close price of stock market. Sentiment Analysis was deployed to monitor the changes in the interest of stock for the 10 global brands in twitter

4. Python Programming Language (3.9)

The models will be designed using python programming language version 3.9. and the Keras framework. It is a deep learning fame work that is used to build deep neural networks. Keras is a higher API in TensorFlow which is owned by Google for deep learning research and used internally by Google for building deep learning products, which was made open and open source for the community to use.

The visualization and data manipulations will be done using other python libraries and packages like Pandas, NumPy, Matplotlib, and Plotly. These pandas and NumPy packages will be used for data manipulation and transformation, and matplotlib and Plotly will be used for data visualization and creating Visualization for the dashboard. The Yahoo Finance python package will be used for the download of stock data.

5. IDE (Jupyter Notebook & Visual Studio Code)

The Integrated Development Environment is a software tool for programming software and web application. The IDE that will be used is the Jupyter notebook, it is an interactive web-based IDE used for computing, and can be used to develop Machine learning models. I will be using a Jupyter notebook on the cloud called google collaboratory. It is a cloud base notebook that gives the users access to a CPU, GPU, or TPU. I will use an IDE called Visual Studio Code to build the stock price prediction application. This IDE will be used along with other python libraries to develop the graphical interface where I will have my dashboard.

6. Graphical User Interface

To build the Graphical User Interface I will be using the streamlit python library, which is used to create and share beautiful custom web apps for machine learning and data science. It is very lightweight and easy to use. It has a lot of layouts and frameworks for developing an interesting user interface. This will be used to build the front end of our application.

1.7 Model Suitability

LSTM model will be perfect for this project because we are dealing with a stock price that contains time-series data, this data is generated over time intervals in a sequence. LSTM models are good at learning patterns of sequence data. It can learn a function that maps a sequence of past observations as input to an output observation.

To predict if a stock will go up or down, we will need a classification model. In the implementation of a classifier, a Random Forest Classifier is used. This model will help us predict if a stock will rise or fall.

1.8 Criteria for Selection of Company.

10 companies were selected based on the following assumptions

- They must be global brands
- They must be quoted within a major stock exchange market
- They must belong to 1 of fastest growing industry

Based on the above the following companies were identified.

Company	Industry	Nature of Business	LISTED	ticker
Google	Information Technology	To provide global information and make it available to users	NASDAQ	GOOG
VISA	Financial Services	To provide electronic bank transfers using bank cards	London Stock Exchange	V
TESLA	Automobile	Electric vehicle design, green energy storage	TESLA	TSLA
COCA-COLA	Beverage	Market, Sell and Manufacture	New York Stock Exchange	KO

		alcoholic & Non-alcoholic beverage		
United Parcel Services	Supply Chain Management	Parcel delivery service	New York Stock Exchange	UPS
Meta	Technology	Social Networking	NASDAQ	META
Pfizer	Pharmaceutical	Manufactures innovative drugs	New York Stock Exchange	PFE
Chevron	Oil & Gas	Production and Exploration of crude oil	New York Stock Exchange	
Microsoft	Technology	Production of computer software and hardware	NASDAQ	

1.9 Chapter Summary & Conclusion

This chapter introduces what the research will be about. Clarification of how data will be collected, the justification, suitability of the selected models were also provided. The reason for engaging in this study Artificial Intelligence Algorithms for Stock Market Prediction at this present time. The detailed research regarding this project will be covered in the next chapters based on the below structure;

1.9.1 Thesis Outline

This thesis is structured as follows.

Chapter 2 - provides existing literatures regarding artificial intelligence in stock market prediction. A review of the past, current & different approach used by researchers in predicting stock market using intelligence.

Chapter 3 – This chapter provides clarification on the research methodology implored, conceptual framework that was deployed and the justification for technical tools used for the study.

Chapter 4 – The chapters focus on result, performance, limitations of my models and unexpected findings were also discussed.

Chapter 5 – This chapter focus on the summary of the study, possible work that should be done in the future.

CHAPTER TWO

2.0 Literature Review

Predicting the price of a stock and its trend is very important in today's economy. The ability to predict the future price of a company's stock can help investors make informed decisions about whether to buy particular company stock or not. The goal of an investor is to make a profit from their investment, and in other to do this the investor needs to analyze his risk to reward before investing in any business, in this case in a stock market, which can be very volatile. In predicting a stock's future price and trend, the investor needs the historical data of the stock market to make an informed decision if to invest or not. The challenge is that the market is not linear in nature; sometimes it may be random and volatile, leading to bad decisions and loss of money.
Investment companies have been looking for ways to automate the process of analyzing and using this data to predict future stock prices or trends. This has led to research on how to use modern tools and develop new algorithms to predict market prices and trends. Researchers have used both linear and non-linear methods to forecast future market trends and prices. One of the methods used is machine learning algorithm based on historical data to predict future market conditions. But before the use of modern tools and algorithms, there has been a debate if there is a correlation between price and time to predict a stock price.

In 1965, Eugen Farma carried out an analysis to see if there exists any correlation between price and time and he came to the conclusion that there is no correlation between price and time, that price is completely random and as such, it is not possible to predict price using historical data (*fama, 1965)*. This has not stopped researchers to attempt developing ways to predict the future stock price using historical data. Over the years there have been researched papers that have created methods to predict the stock price, one such method is the ARIMA (autoregressive integrated moving average) model, which most models rely on as a baseline. "The ARIMA model is a statistical analysis model that uses time series data to either better understand the data or predict future trends" *(Investopedia)*
There are papers that created methods of predicting stock market price based on the ARIMA method for example *(Ping-FengPai and Chih-ShengLin 2005) and (Khashei and Hajirahimi, 2018)* They applied different techniques but used the ARIMA method has its baseline model.

Since the use of and spread of Artificial Neural Network, a lot of research have been carried out using ANN to predict or forecast the stock market price. A hybrid of combined ANN and traditional machine learning models like SVM and other logistic models have been used by researchers. ANN are good approximators and generalizes well with experience from historical data. The use of ANN with multiple layers is also known as Deep Neural Network. For the past few years, deep learning has been used for stock price prediction Rasel and Hasan(2016) used Artificial Neural Network approach for stock price prediction. They used a three-layer perceptron and a windowing method in their model and a Root Mean Square Error for the

Evaluation. (Erkam 2011,Pp 10389-10397), carried out a study on the effect of ANN on stock market forecasting. (Batres 2015) did a study using Deep Learning model on time series analysis. (Pinheiro, Leonardo, and Dras 2017) from the University of Macquarie carried out a study on the Character-based Neural Language Model for Event-based Trading.

Some researchers have gone ahead to use very recent breakthrough models in Deep Learning for stock price prediction, like combining computer vision and long short-term memory for stock price prediction. (Shah, Gor, and Sagar 2022) did a study on Deep Learning architecture using CNN and LSTM. (Staffini 2022) carried out research on the use of Deep Convolutional Generative Adversarial Network(DCGAN) for forecasting the close price of a stock. His research shows that there are possibilities in using modern deep learning models to forecast stock prices. GANS was introduced by (Goodfellow 2014). GANS is composed of Discriminator and a Generator Network that interact with each other. The discriminator tries to distinguish what instance is real or fake and the generator tries to generate realistic data to confuse the discriminator. In 2018 Zhou X and Zhaol used GANS which uses LSTM as a generator and CNN as a discriminator to forecast the stock market. Their studies show you can effectively improve stock price prediction direction accuracy and reduce forecast error.

CHAPTER THREE

3.0 Methodology

3.1 **First Model - LSTM**

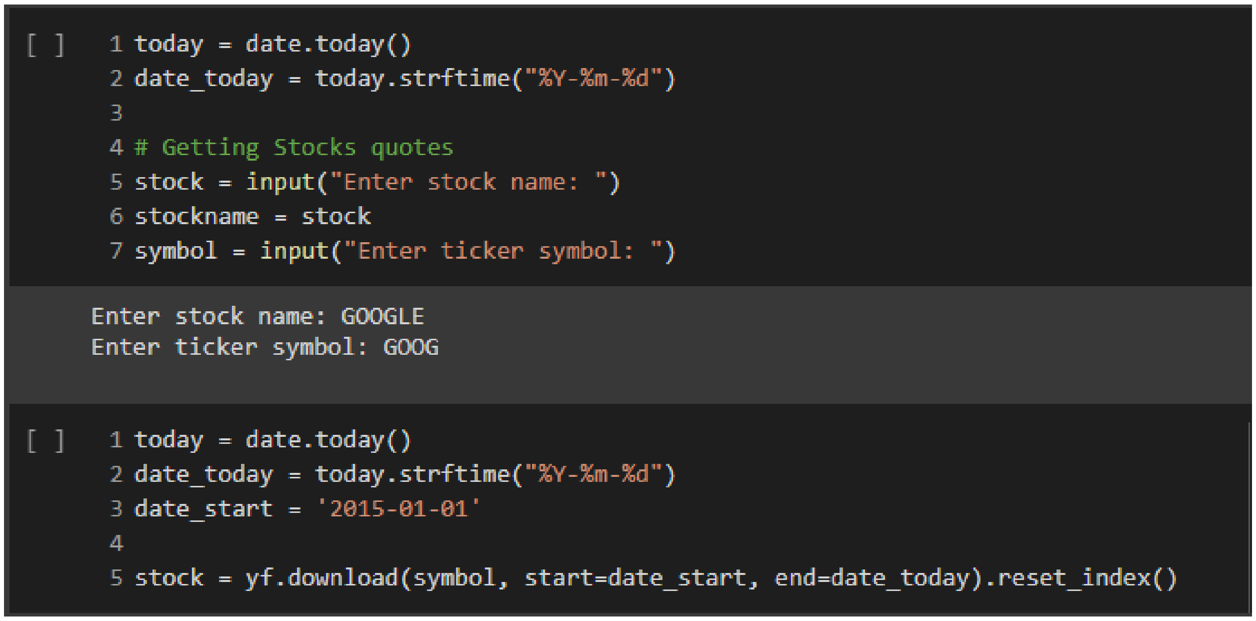
In this study, the aim is to build a machine learning application that can assist in decision-making on which stock to invest in based on the model prediction of price, trend, and sentiment analysis of a stock. The combination of price, trend, and sentiment will determine if a particular stock is good to buy or sell. To achieve these this study will use the following steps and design.

3.1.1 **Data Collection**

In this Project, data was collected using the yahoo Stock Data API. The API is called Yfinance, this API allows users to collect data using the download function of the API, this function expects three important arguments the stock market symbol, for example, the Google stock symbol is GOOG, and it expects the start and end date of the stock data you want to download. In this study, we used data from 2015 to the present date. This is because we want our model to make predictions based on the most recent data. The download data has the following features: Date, Open, High, Low, Close, Adjusted Close, and Volume. The first 5 rows of the data can be printed using Pandas because the data is a Dataframe object.

Description of Collected Data
The data collected has 8 columns and they ae are as follows.

Date – This is the period trading/ transaction for each company
Open – The ‘Open’ column is the open price for the day. It is the starting price for transaction.
High – The ‘High’ column is the highest price of stock for a particular day
Low – The ‘Low’ columns has the lowest price of stock for a specific period of time
Close – This is the closing price for the day
Volume – The ‘Volume’ column has the total transaction in volume for the day.

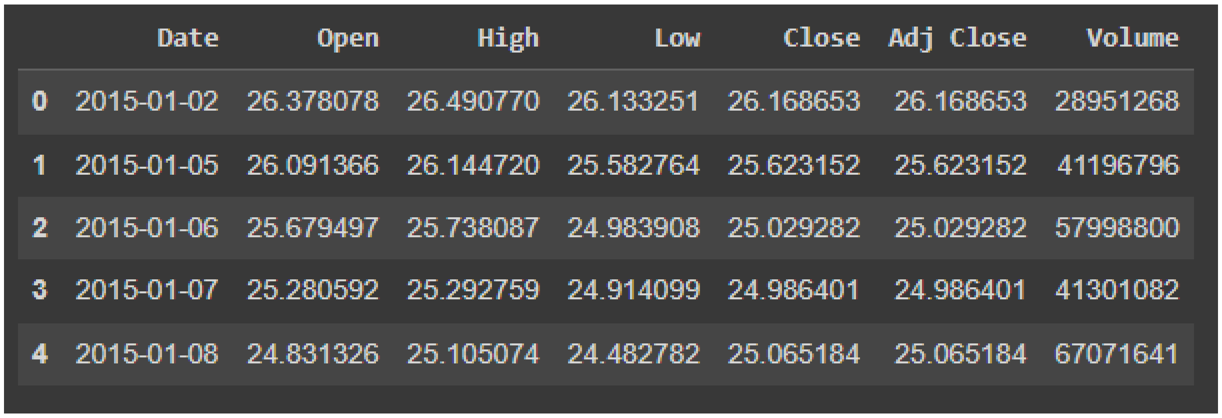


```
[ ]
today = date.today()
date_today = today.strftime("%Y-%m-%d")

# Getting Stocks quotes
stock = input("Enter stock name: ")
stockname = stock
symbol = input("Enter ticker symbol: ")
```

```
Enter stock name: GOOGLE
Enter ticker symbol: GOOG
```

```
[ ]
today = date.today()
date_today = today.strftime("%Y-%m-%d")
date_start = '2015-01-01'

stock = yf.download(symbol, start=date_start, end=date_today).reset_index()
```

Figure 2. Data download code snippet



	Date	Open	High	Low	Close	Adj Close	Volume
0	2015-01-02	26.378078	26.490770	26.133251	26.168653	26.168653	28951268
1	2015-01-05	26.091366	26.144720	25.582764	25.623152	25.623152	41196796
2	2015-01-06	25.679497	25.738087	24.983908	25.029282	25.029282	57998800
3	2015-01-07	25.280592	25.292759	24.914099	24.986401	24.986401	41301082
4	2015-01-08	24.831326	25.105074	24.482782	25.065184	25.065184	67071641

Figure 3. Stock dataframe

This data collected will be used as input to our model for training and testing, but first, we need to process this data and convert them to NumPy arrays because our model expects an array, not a pandas dataframe. It is good practice to explore our data by plotting the close price using a line plot.

3.1.2 Exploratory Data Analysis.
An exploratory data analysis was conducted on the downloaded data to see the relationship between the variables. However, in this case 2 variables were selected for exploration. The variables are DateTime and Close Price. This is because date is the input variable and the target variable is Close Price.

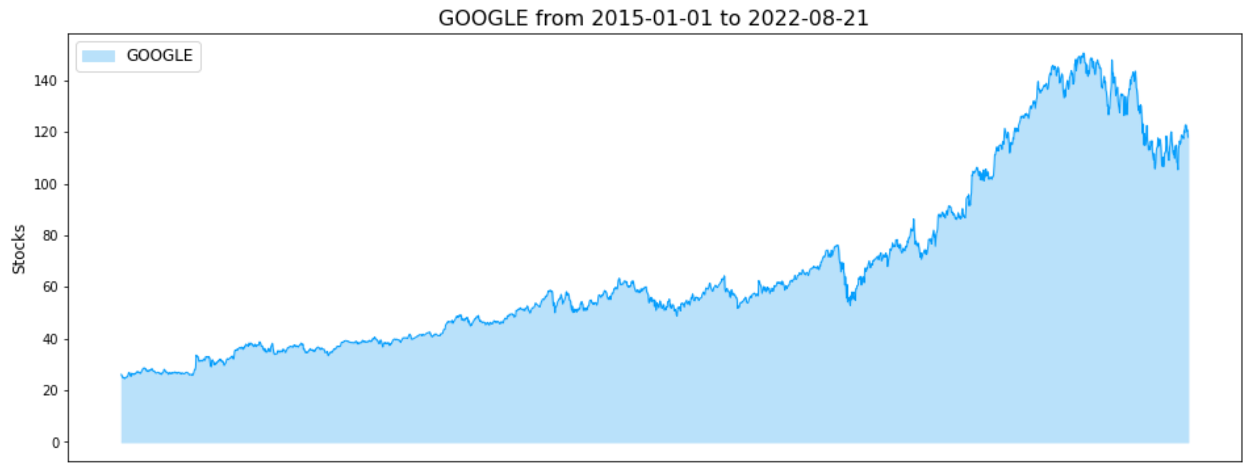
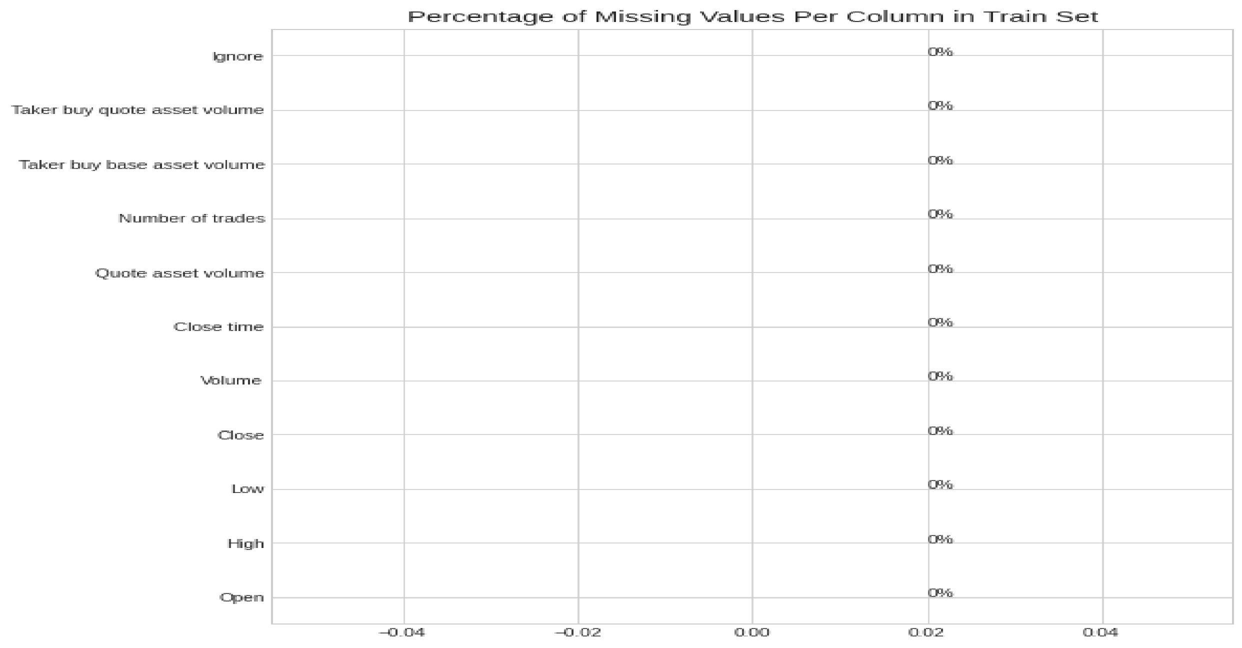


Figure 4. Google trend plot

Based on the above we can see the relationship between data and close price is not static and non-linear. This supports the assumption that the stock market is very volatile and the data is non-linear in nature.

3.1.3 Data Cleaning.

A data integrity check was carried out on the downloaded data and the technique used was to check for missing values on the downloaded data using the isna function. The outcome of missing values was later visualize to see if there was any missing values. Please see below.

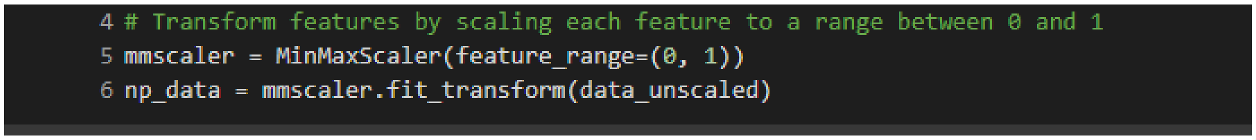


Based on the above findings from my data cleaning technique there are no missing values on the downloaded dataset.

3.1.4 Data preprocessing and Scaling

Deep learning expects the input data as an array and not all the stock features will be used. For our implementation, only the Open, High, Low, Close, and Adjusted Close features will serve as the input. These input features are scaled so that the values will range from 0 to 1. A MinMax Sclar is used to scale the features. To scale means to change the range of the values, it does not change the shape of the distribution. Scaling will improve the training time and may improve training accuracy.

I tried using other features



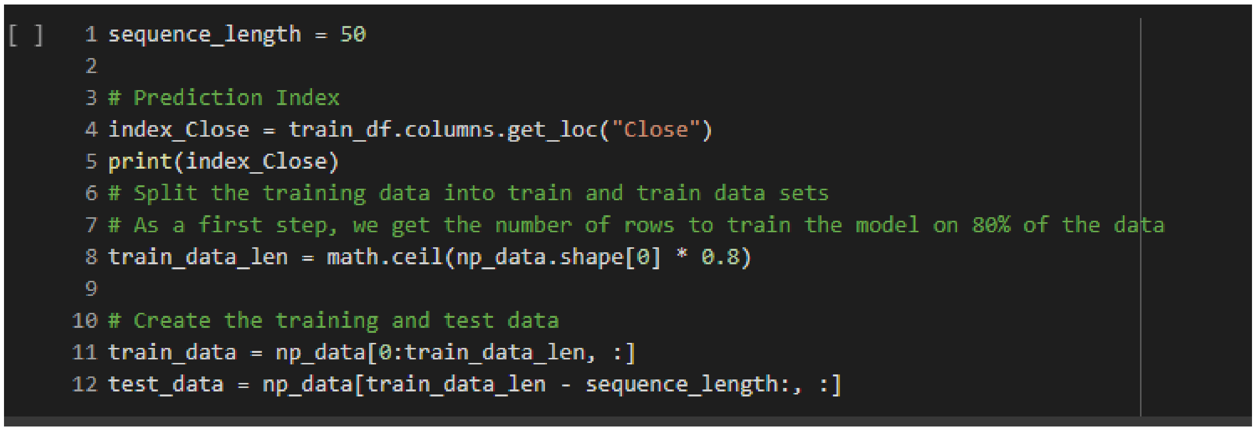
```
# Transform features by scaling each feature to a range between 0 and 1
mmscaler = MinMaxScaler(feature_range=(0, 1))
np_data = mmscaler.fit_transform(data_unscaled)
```

Figure 5. MinMaxScaler function

3.1.5 Training and Testing

A machine learning algorithm is used to find patterns, and learn from the dataset, it also uses data to evaluate and make decisions based on what it has learned. The stock dataset will be split into two sets, one is the training data and the other set is the testing data. The data will be split between 80% training and 20% testing, the training data is a subset of our entire data. The training data is usually larger because we want to train our model with as much data as possible so the model can find and learn meaningful patterns. Machine learning algorithms learn to solve problems based on past data, this data serves as examples for these algorithms to learn from. The more data the algorithm is exposed to the more it learns.

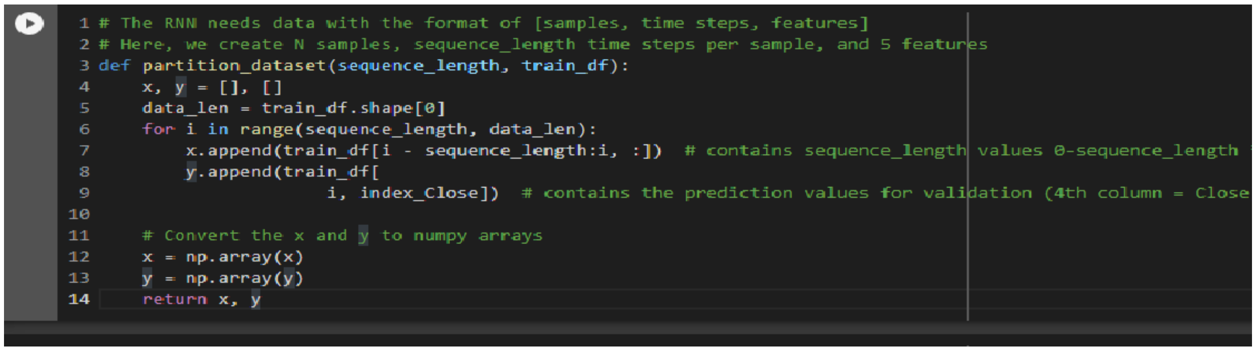
The testing data is used to evaluate the model after the model has been trained with the training data. This data needs to be unseen data that has not been seen during training by the model and it must be large enough for meaningful prediction.



```
[ ]
sequence_length = 50

# Prediction Index
index_Close = train_df.columns.get_loc("Close")
print(index_Close)
# Split the training data into train and train data sets
# As a first step, we get the number of rows to train the model on 80% of the data
train_data_len = math.ceil(np_data.shape[0] * 0.8)

# Create the training and test data
train_data = np_data[0:train_data_len, :]
test_data = np_data[train_data_len - sequence_length:, :]
```

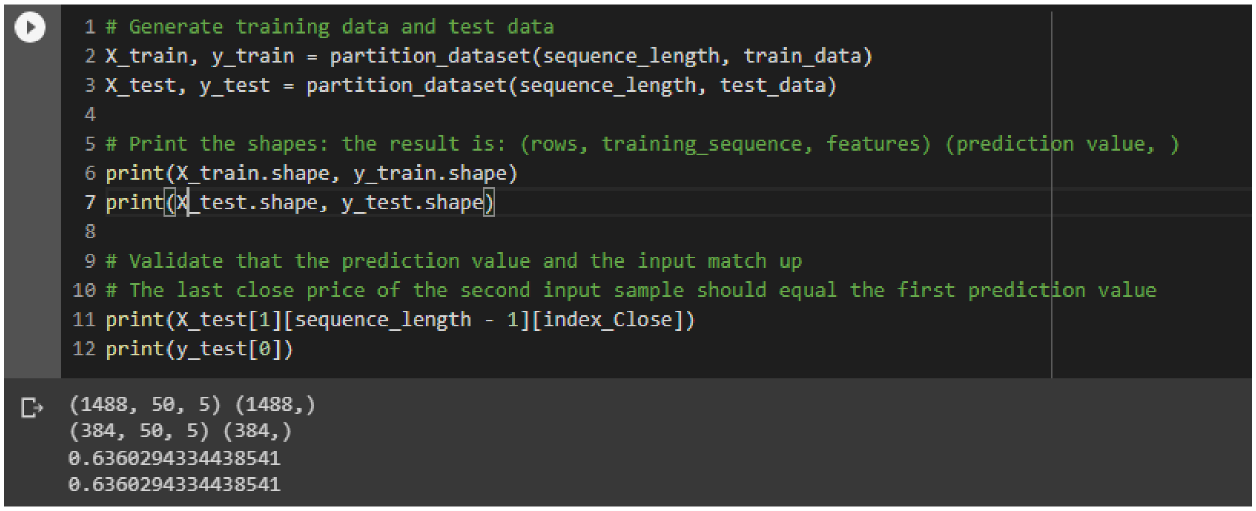
Figure 5. Creating Train and test data split.

The LSTM model needs the input data to be in three dimensions, these dimensions will include Samples, Sequences, and Features. The samples are the market index, the sequence is the time steps, and the third dimension is the features. To prepare the data in this form, I sliced the data into multiple input data sequences using a sliding window. The sliding window goes through the time series dataset and appends multiple sequences of data into our input data list, the target column which is the close price column follows the same sequence and did the same with the test data. This results in a training input of array shape (1488, 50, 5), (1488), and the sequence is hard-coded to be 50, the testing input is of array shape (384, 50, 5), (384). Since this is continuous time series data, the sample size may differ.



```python
# The RNN needs data with the format of [samples, time steps, features]
# Here, we create N samples, sequence_length time steps per sample, and 5 features
def partition_dataset(sequence_length, train_df):
    x, y = [], []
    data_len = train_df.shape[0]
    for i in range(sequence_length, data_len):
        x.append(train_df[i - sequence_length:i, :])  # contains sequence_length values 0-sequence_length
        y.append(train_df[
                    i, index_Close])  # contains the prediction values for validation (4th column = Close

    # Convert the x and y to numpy arrays
    x = np.array(x)
    y = np.array(y)
    return x, y
```

Figure 6. Sliding Window Function.



```python
# Generate training data and test data
X_train, y_train = partition_dataset(sequence_length, train_data)
X_test, y_test = partition_dataset(sequence_length, test_data)

# Print the shapes: the result is: (rows, training_sequence, features) (prediction value, )
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)

# Validate that the prediction value and the input match up
# The last close price of the second input sample should equal the first prediction value
print(X_test[1][sequence_length - 1][index_Close])
print(y_test[0])
```

```
(1488, 50, 5) (1488,)
(384, 50, 5) (384,)
0.6360294334438541
0.6360294334438541
```

Figure 7. Creating Input data of shape (samples, sequences, features).

3.1.5 Tune & Further Train

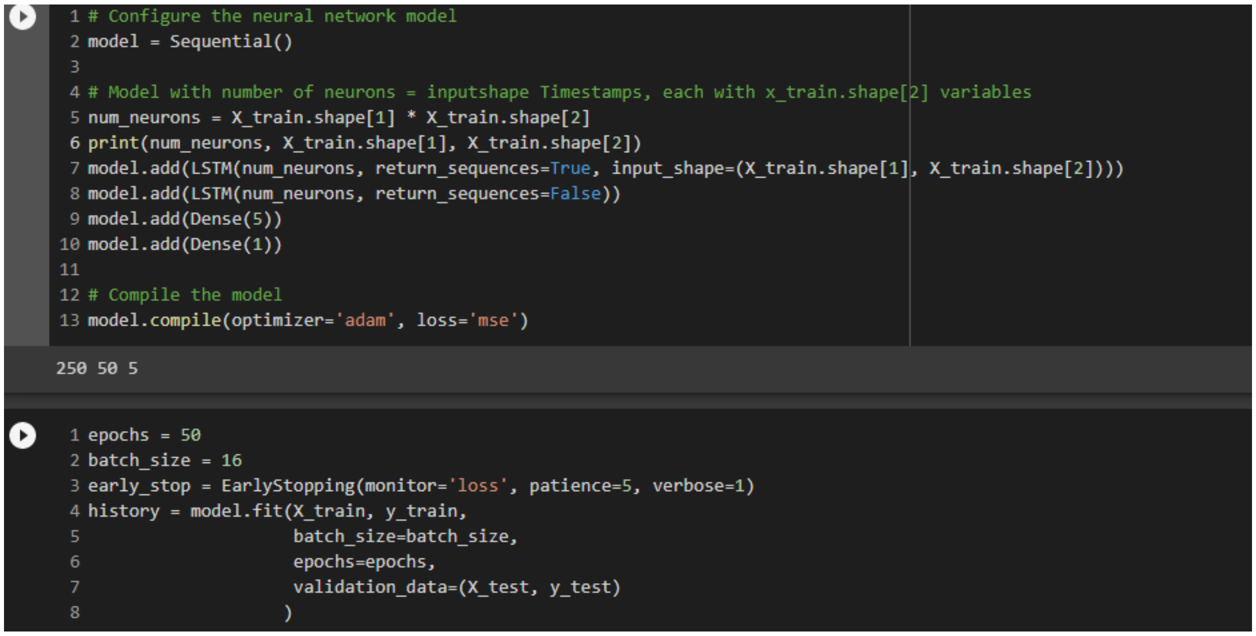
Now that the input data is prepared with the right shape, the model architecture is built using Keras. The architecture is an LSTM architecture that has four layers, the following are the layers with what they accept as inputs and their output:

1. The first layer takes in a mini-batch as input and outputs the whole sequence.
2. The second layer accepts the sequence from the output from the first layer and outputs five values.
3. A Dense layer with five neurons that takes in the five outputs from the second layer as input.
4. The final Dense layer is the prediction layer, it accepts input from the previous Dense layer and outputs a prediction.

These layers are objects of the Keras Sequential function that allows us to stack networks or layers together in a sequential manner. The first layer will take as input the size of the mini-batch, each mini-batch is a matrix that has 50 timesteps and 5 features. Multiplying this will give us 250 neurons in our first input layer. After creating the architecture, the model is compiled and ready to be trained but to compile the model it is necessary to specify the optimizer and lose function.

In this implementation, I used Adam optimizer and Mean Square Error Loss. The Adam optimizer is a gradient descent algorithm used to update the weights of a neural network during training, the update is done during the backpropagation of the network training. There are several optimizer techniques like Stochastic Gradient Descent Mini-Batch Gradient Descent, Root Mean Square Propagation, etc. The Adam optimizer was first introduced by Diederik Kingma and Jimma Ba in their paper "*Adam: A Method for Stochastic Optimization*". It is an adaptive learning rate method that computes individual learning rates for different parameters. According to the papers, it combines the advantages of two stochastic base optimizers; Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). I chose this optimizer because it has be come very popular and it is shown in the paper that it gives better performance than other optimizers.

The training loss of the model is the Mean Squared Error (MSE) it is one of the simplest error functions, it calculates the difference between our prediction and the ground truth, and squares the result, and averages it across the whole dataset. The goal is to reduce this loss during training by so doing our model is generalizing and learning important patterns. After compiling I fit the model to my Training data and trained for 50 epochs with a batch size of 16.



```
# Configure the neural network model
model = Sequential()

# Model with number of neurons = inputshape Timestamps, each with x_train.shape[2] variables
num_neurons = X_train.shape[1] * X_train.shape[2]
print(num_neurons, X_train.shape[1], X_train.shape[2])
model.add(LSTM(num_neurons, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(LSTM(num_neurons, return_sequences=False))
model.add(Dense(5))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mse')
```

```
250 50 5
```

```
epochs = 50
batch_size = 16
early_stop = EarlyStopping(monitor='loss', patience=5, verbose=1)
history = model.fit(X_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    validation_data=(X_test, y_test)
                   )
```

Figure 8. Creating LSTM Architecture and Model Training.

3.1.6 Model Evaluation and result

After training the model, I evaluated the model to see how the model performed with our test data. After running the prediction on the test data I denormalize the result in other for to have the exact prediction prices. I used several evaluation metrics like Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Median Absolute Percentage Error (MDAPE). These metrics will tell us how the model is performing.

I. Mean Absolute Error (MAE): This metric calculates the absolute error of our model, it measures the average magnitude of the error in a set of predictions. It calculates the absolute difference between the prediction and ground truth and averages it. In the evaluation, the prediction shows an MAE of **1.76** this shows on average the prediction distance from the true value is off by 1.76.

II. Mean Absolute Percentage Error (MAPE): This represents the average percentage error of the prediction. The model gave a **1.39% error.** This means our prediction percentage error from the ground truth is 1.39%.

III. Absolute Percentage Error (MDAPE): An error metric used to measure the performance of the regression model. It is the median of all absolute percentage errors. The MPAPE of our model is 0.98%

3.2 **Second Model(Random Forest Classifier)**

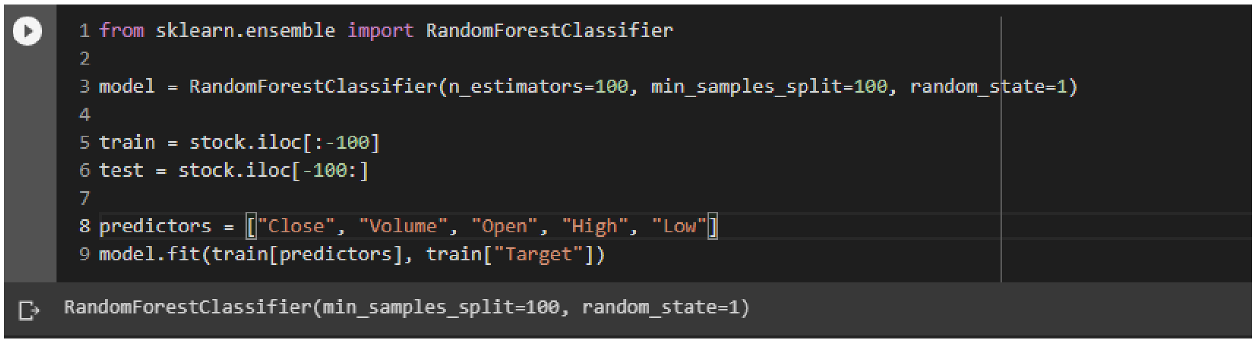
One of the goals of this study is to build a regression model with LSTM and a classifier. It is not enough to predict the price, what if we want to know the direction of the market if the market will go Up or Down? This is important because we want to know the relationship with stock sentiment analysis, we will have a better idea of which direction the market is heading and we can tell if we want to Buy or Sell. My goal with this ML algorithm is to optimize for a Buy signal. We can predict the price of the stock but do not necessarily know if the price will go Up or Down.

3.2.1 Data Collection and Preprocessing

The Data collection of this model follows the same process as model 1. In this implementation, I will be using Random Forest as my classifier and the Google stock for example. I want to predict a binary classification where 0 means the market will go down and 1 means the market will go up. To achieve this I processed the data to have a target value of 0 or 1. I shift my close price by 1 day so I can use tomorrow's price to compare with the close price, if tomorrow’s price is greater than the close price we assign the target to be 1 which is if tomorrow's close price is greater than today's close price the market was positive but if tomorrows close price is less than today's close price we assign 0 to the target which means the price was negative.

3.2.2 Train Random Forest Classifier

Since it's a classification challenge, I decided to use a Random Forest Classifier. It is a supervised learning algorithm that is used both for classification and regression. A forest is made up of different trees, it creates decision trees on randomly selected data from the dataset and selects the best result through voting from the trees. RFC is good at handling overfitting and can easily pick nonlinear relationships.

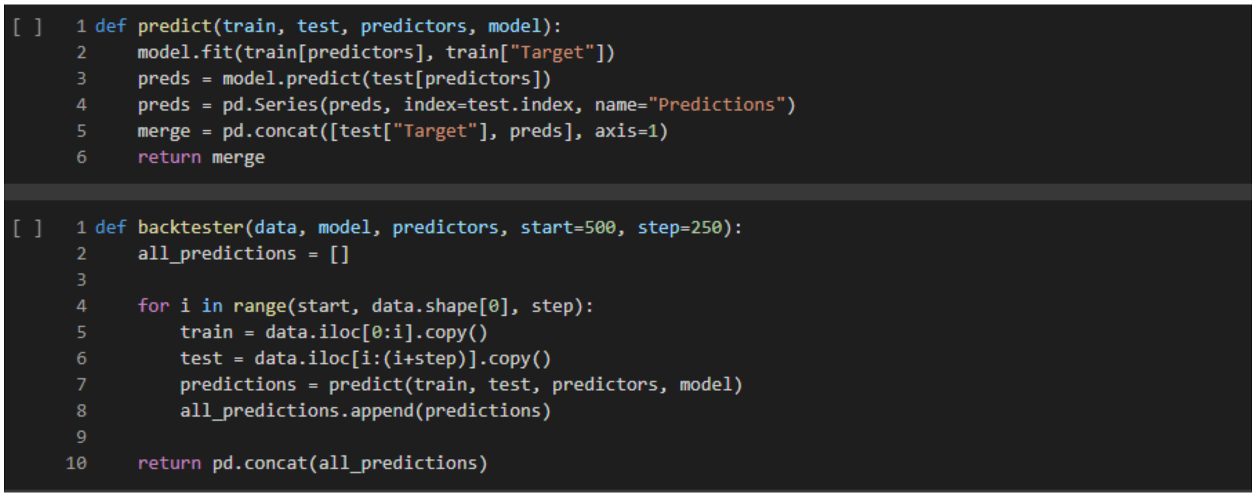


```python
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100, min_samples_split=100, random_state=1)

train = stock.iloc[:-100]
test = stock.iloc[-100:]

predictors = ["Close", "Volume", "Open", "High", "Low"]
model.fit(train[predictors], train["Target"])
```

```
RandomForestClassifier(min_samples_split=100, random_state=1)
```

Figure 9. Creating and Training the RFC and Data split.

After creating the model I split the data into train and test, using the last 100 data samples as my test and the rest as my train samples. I specify the features for our input in a list. The training did not take long compared to the LSTM model. I set my estimators to be 100 which is the number of trees and a min_sample_split of 100. To measure the performance of the classifier model with the test data, I used the precision score metrics and the result wasn't impressive so I decided to create more features. The idea is that when the model predicts that the stock price will go up we want to be to some certain degree confident that it will go up. To achieve this I backtested the data in other to make predictions on the entire dataset, not just the last 100 days. Backtesting only ensures we carry out a prediction on data from before the day we want to predict. This will help give a better estimation of the error.



```
[ ]  def predict(train, test, predictors, model):
         model.fit(train[predictors], train["Target"])
         preds = model.predict(test[predictors])
         preds = pd.Series(preds, index=test.index, name="Predictions")
         merge = pd.concat([test["Target"], preds], axis=1)
         return merge
```

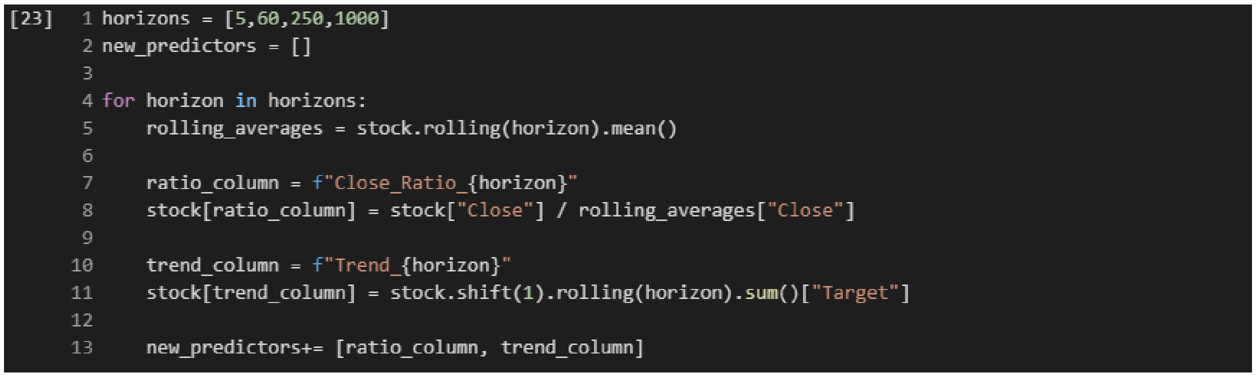
```
[ ]  def backtester(data, model, predictors, start=500, step=250):
         all_predictions = []

         for i in range(start, data.shape[0], step):
             train = data.iloc[0:i].copy()
             test = data.iloc[i:(i+step)].copy()
             predictions = predict(train, test, predictors, model)
             all_predictions.append(predictions)

         return pd.concat(all_predictions)
```

Figure 10. Predict and Backtester function.

The back tester function takes 750 rows of data for training and predicts the next 250 rows, the 250 rows is the number of test data. It loops through the data such that every 750 rows are used to predict the next 250 rows. The predict function allows us to train and test our model using the back tester function. After using the back tester the performance of the model increased from about **70%** to **78%**. Since we are using the precision score it is good to check the count of the buy trades as compared to the sell trades, this will inform us the number of times the model said the market will go up and it actually did. For our Google stock, the market actually went Up **620** as against it going Down **2070** this means that if we had traded that the market will trade Up we would have been accurate 50% of the time with 620 trades.

3.2.3 Feature Engineering, Tunning & Further Training

To improve this model, decided to experiment with some feature engineering to see if our accuracy will improve. I created a variety of rolling averages, that will give more information about the stock. I want to know if the stock price today is higher than the stock price last week, higher than it was three months ago, a year ago, and 5 years ago. I can use all these as input to my model to predict if the stock will go up or down. I calculated the mean close price for the last trading week, the last two days, the last year, and the last four years and find the ratio between today’s closing price and the closing price of the last period this will give the model more information to make a better prediction know if the market. This ratio informs us if the market is oversold or overbought. I also looked at the trend, that is the number of days the price went up in the past week, three months, one year, and 4 years. Adding all these features to our original dataset will increase the features.



```
[23]
horizons = [5,60,250,1000]
new_predictors = []

for horizon in horizons:
    rolling_averages = stock.rolling(horizon).mean()

    ratio_column = f"Close_Ratio_{horizon}"
    stock[ratio_column] = stock["Close"] / rolling_averages["Close"]

    trend_column = f"Trend_{horizon}"
    stock[trend_column] = stock.shift(1).rolling(horizon).sum()["Target"]

    new_predictors+= [ratio_column, trend_column]
```

Figure 11. Feature Engineering.

I used these features as input to another random classifier, this time I increased the number of estimators to 200 and min sample split to 50. After training,

3.2.4 Model Performance & Evaluation

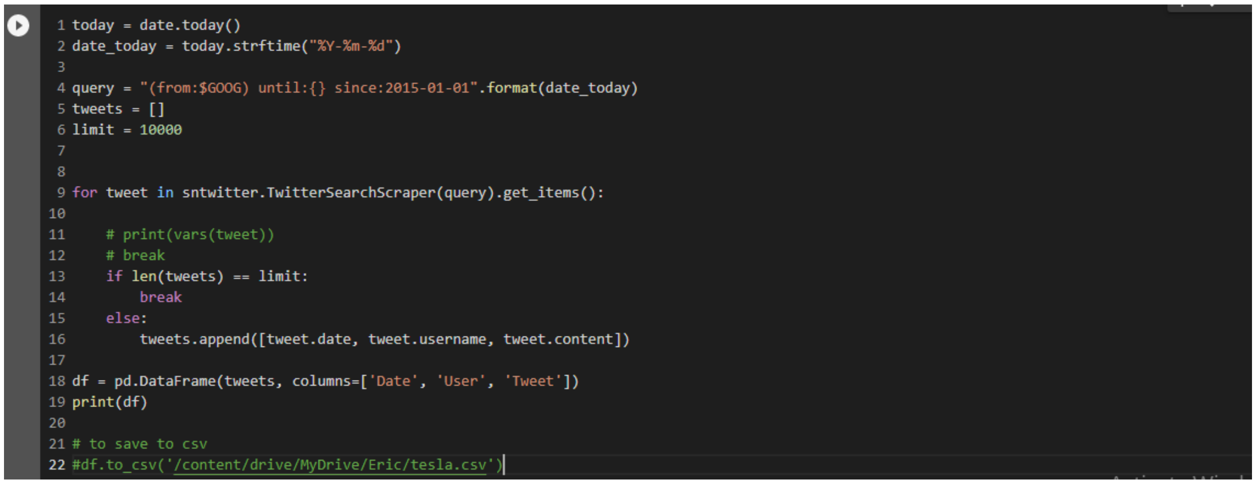
I used the custom predict function but this time I added a threshold to a probability score function. I want only Up trend prediction whose probability is greater than 60%. Backtesting this model on the data gave a precision score of **53%,** improving the model by 3%, this result is not so great but it shows that there was an improvement based on model parameter tuning and feature engineering.

3.3. Third Model - Sentiment Analysis

Market sentiment can play a vital role in stock trends, it refers to the general attitude of investors towards a particular stock market or the financial market as a whole. Sentiment can drive demand and supply which can affect the movement of a stock. When the market is rising it means sentiment is positive or bullish, when the market is falling it means sentiment is negative or bearish. In this study, I am combining a machine learning model predictor with sentiment indicators to determine if the market is right to invest in or not.

3.3.1Data Collection

To achieve this I need a source where I can get data that can be regarded as people's sentiment toward a particular stock and one of the best places to get such data is Twitter. Twitter has an API for scraping data from their platform but when I tried using the API I could only get 7 days' worth of data which to me was not enough based on what I intended to do with the data, so I decided to use a third party API for scraping enough data. I used snscrape, a python library that allows you to scrape data easily from Twitter, you can scrape as much as you want. The API allows you to specify the kind of data you want to scrape with its ‘TwitterSearchScraper’ function. To use this function you need to specify the kind of search query you want, the query in our case is the name of the stock and the start and end date of the tweets we want to scrape. I decided to scrape data from January 1st, 2015 to date, you can also specify the number of tweets you want to scrape, I scraped ten thousand tweets. When scraping I discover that what will determine if you can scrape data from the start year is the total number of tweets you specify, for example, scraping ten thousand tweets got me only a month of tweets, but when I tried a hundred thousand tweet limit I got more than a year tweets. The API took some time before scraping the ten thousand tweets, the more tweets you scape the more time it takes to scrape. I made sure i convert the scraped tweets into a pandas dataframe.

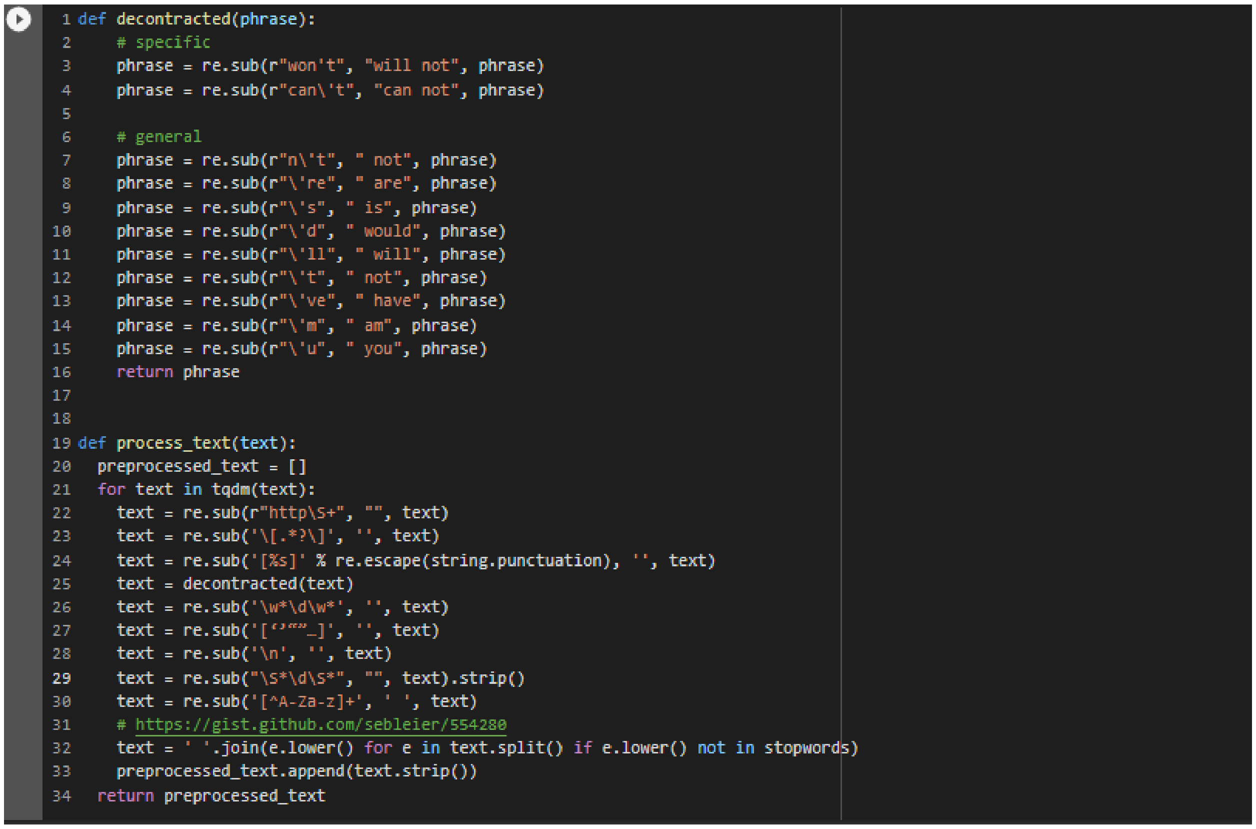


```
today = date.today()
date_today = today.strftime("%Y-%m-%d")

query = "(from:$GOOG) until:{} since:2015-01-01".format(date_today)
tweets = []
limit = 10000


for tweet in sntwitter.TwitterSearchScraper(query).get_items():

    # print(vars(tweet))
    # break
    if len(tweets) == limit:
        break
    else:
        tweets.append([tweet.date, tweet.username, tweet.content])

df = pd.DataFrame(tweets, columns=['Date', 'User', 'Tweet'])
print(df)

# to save to csv
#df.to_csv('/content/drive/MyDrive/Eric/tesla.csv')
```

Figure 12. Scraping Data with snscrape.

3.3.2 Data Cleaning

After scraping the data, I created functions that remove stopwords, this is because stopwords to not provide clarification/meaning to our predicted class. I then converted shortened words or phrases to their normal form to provide more clarification/meaning to the sentiment, remove punctuation marks, and convert them to small letters. The data is now ready to be used for sentiment analysis.



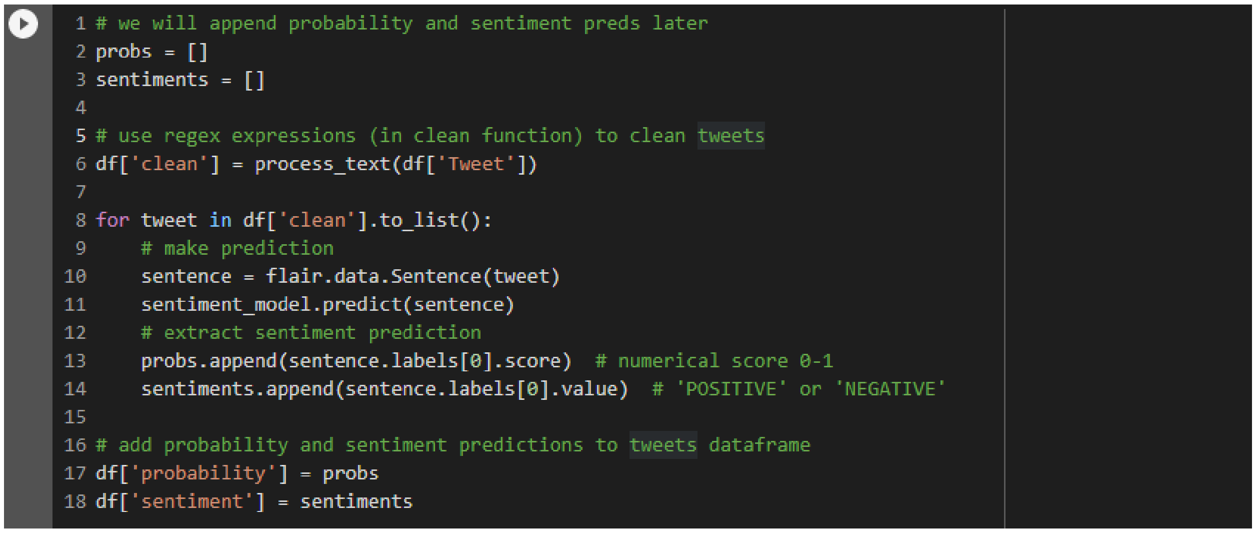
```python
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    phrase = re.sub(r"\'u", " you", phrase)
    return phrase


def process_text(text):
  preprocessed_text = []
  for text in tqdm(text):
    text = re.sub(r"http\S+", "", text)
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = decontracted(text)
    text = re.sub('\w*\d\w*', '', text)
    text = re.sub('[‘’“”_]', '', text)
    text = re.sub('\n', '', text)
    text = re.sub("\S*\d\S*", "", text).strip()
    text = re.sub('[^A-Za-z]+', ' ', text)
    # https://gist.github.com/sebleier/554280
    text = ' '.join(e.lower() for e in text.split() if e.lower() not in stopwords)
    preprocessed_text.append(text.strip())
  return preprocessed_text
```

Figure 13. Data Preprocessing Functions.

3.3.3 Train Sentiment Analysis Model

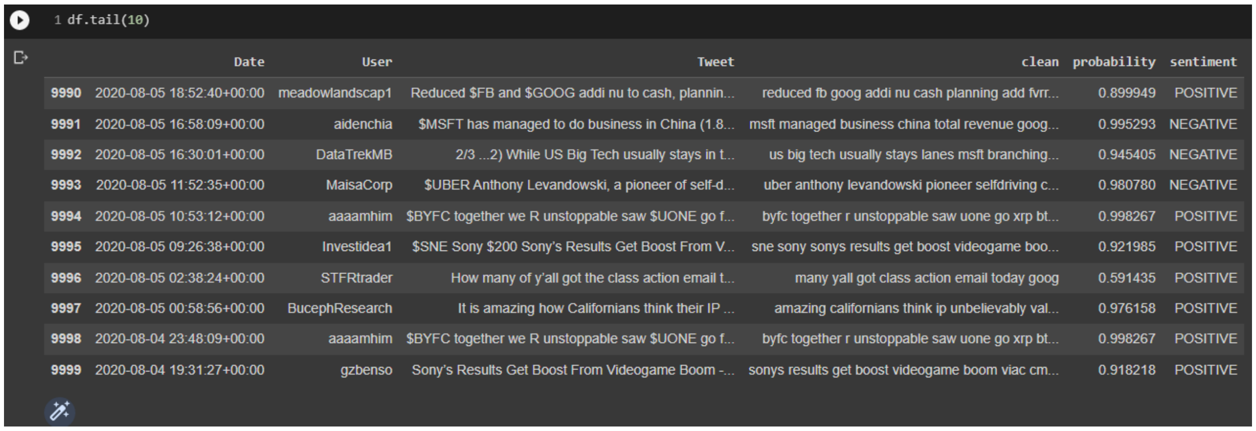
To predict the sentiment of the tweets I used a pre-trained sentiment model called flair, it is an NLP framework built on top of PyTorch. It uses word embeddings to represent a word or a text as vectors. Flair allows you to combine word embeddings like BERT and Elmo embeddings, flair also has its own embedding. It can be used for Text Classification, Name Entity Recognition, Part of Speech Tagging, etc. For this study, I am using the flair pre-trained model for text classification. I want to classify if a tweet about a particular stock has a positive sentiment or negative sentiment. Positive sentiment over a period may prove that the market is bullish and a negative sentiment may prove that the market is bearish. If the number of positive sentiments is far higher than the negative sentiment and our machine learning and LSTM models are all showing that the market will rise, it is a good indication that one can buy that particular stock.

The flair predicts function returns the probability score of the sentiment and the sentiment either positive or negative.



```python
# we will append probability and sentiment preds later
probs = []
sentiments = []

# use regex expressions (in clean function) to clean tweets
df['clean'] = process_text(df['Tweet'])

for tweet in df['clean'].to_list():
    # make prediction
    sentence = flair.data.Sentence(tweet)
    sentiment_model.predict(sentence)
    # extract sentiment prediction
    probs.append(sentence.labels[0].score)  # numerical score 0-1
    sentiments.append(sentence.labels[0].value)  # 'POSITIVE' or 'NEGATIVE'

# add probability and sentiment predictions to tweets dataframe
df['probability'] = probs
df['sentiment'] = sentiments
```

Figure 14. Prediction with flair pre-trained model.



```python
df.tail(10)
```

	Date	User	Tweet	clean	probability	sentiment
9990	2020-08-05 18:52:40+00:00	meadowlandscap1	Reduced $FB and $GOOG addi nu to cash, plannin...	reduced fb goog addi nu cash planning add fvrr...	0.899949	POSITIVE
9991	2020-08-05 16:58:09+00:00	aidenchia	$MSFT has managed to do business in China (1.8...	msft managed business china total revenue goog...	0.995293	NEGATIVE
9992	2020-08-05 16:30:01+00:00	DataTrekMB	2/3 ...2) While US Big Tech usually stays in t...	us big tech usually stays lanes msft branching...	0.945405	NEGATIVE
9993	2020-08-05 11:52:35+00:00	MaisaCorp	$UBER Anthony Levandowski, a pioneer of self-d...	uber anthony levandowski pioneer selfdriving c...	0.980780	NEGATIVE
9994	2020-08-05 10:53:12+00:00	aaaamhim	$BYFC together we R unstoppable saw $UONE go f...	byfc together r unstoppable saw uone go xrp bt...	0.998267	POSITIVE
9995	2020-08-05 09:26:38+00:00	Investidea1	$SNE Sony $200 Sony's Results Get Boost From V...	sne sony sonys results get boost videogame boo...	0.921985	POSITIVE
9996	2020-08-05 02:38:24+00:00	STFRtrader	How many of y'all got the class action email t...	many yall got class action email today goog	0.591435	POSITIVE
9997	2020-08-05 00:58:56+00:00	BucephResearch	It is amazing how Californians think their IP ...	amazing californians think ip unbelievably val...	0.976158	POSITIVE
9998	2020-08-04 23:48:09+00:00	aaaamhim	$BYFC together we R unstoppable saw $UONE go f...	byfc together r unstoppable saw uone go xrp bt...	0.998267	POSITIVE
9999	2020-08-04 19:31:27+00:00	gzbenso	Sony's Results Get Boost From Videogame Boom -...	sonys results get boost videogame boom viac cm...	0.918218	POSITIVE

Figure 15. Result of the Flair Prediction.

CHAPTER FOUR

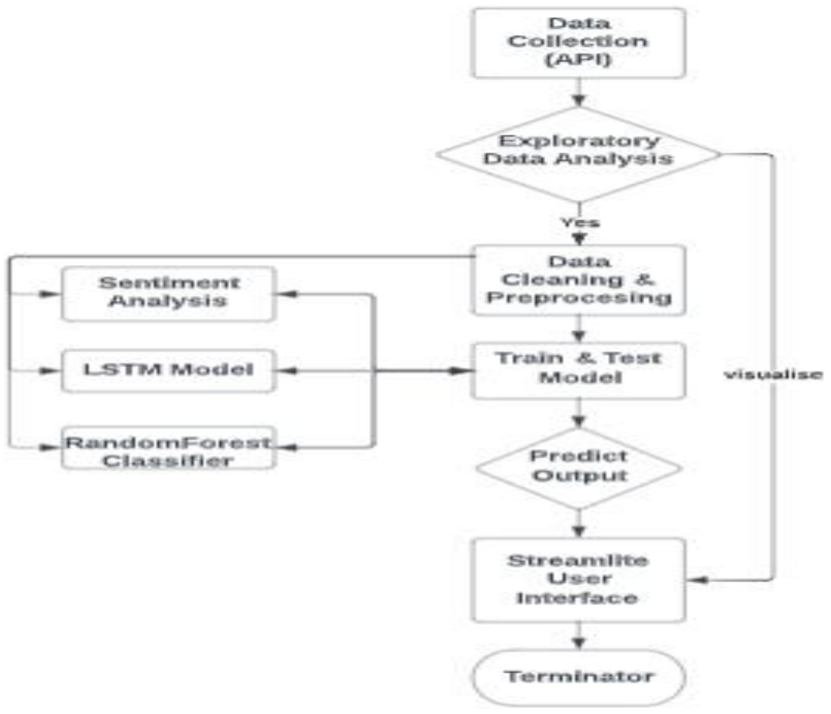
4.1 Front End Development Life Cycle

A waterfall development cycle also popularly known as linear sequetial life cycle was implemented in the development of front-end user interface. Which means each phase in the development cycle will be completed before the next stage can commence. The implication is that there is no overlapping in each phase. The reason for adopting the waterfall development cycle is to optimize the storage capacity of my user-interface(Streamlite) because my front-end user inter-face is running on free-version which has limited storage capacity.

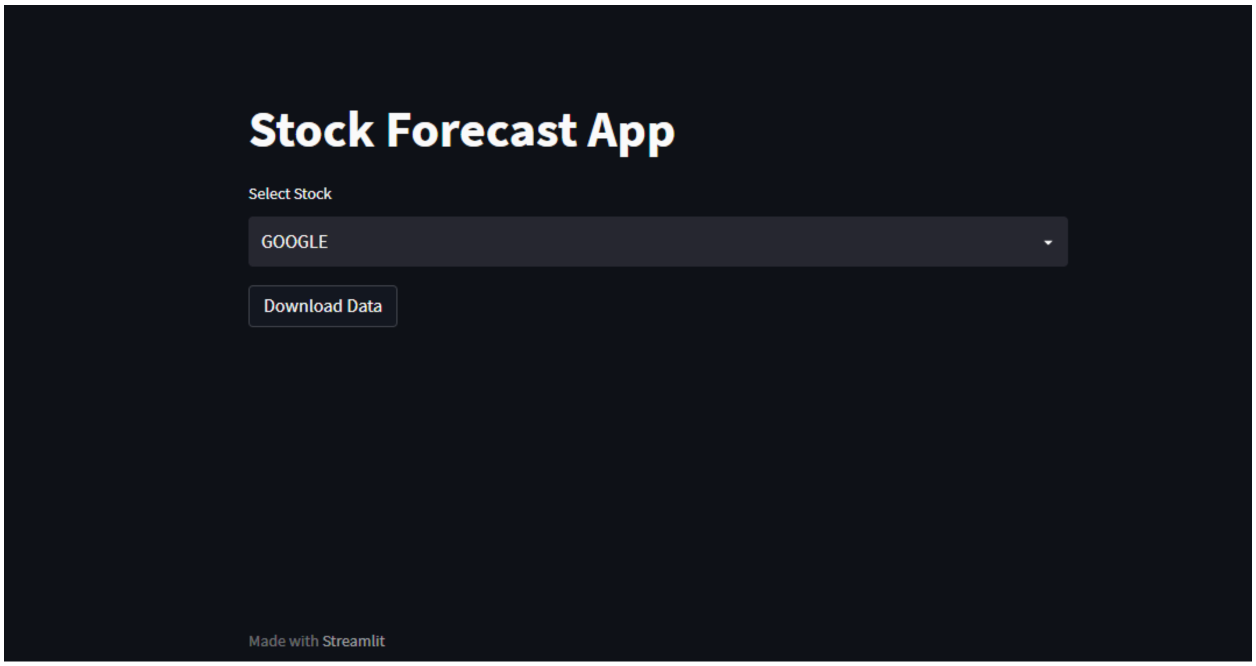
I adopted StreamLite as my front-end user interface because it is light weight and has a faster processing speed as compared to other web applications.

4.2 System Design Flow chat.

The system was design to carry out its first task of data collection with API and then conduct an exploratory data analysis to be rendered(visualize) on stream lite user interface, thereafter a data cleaning & preprocessing exercise done differently because the data will have to be preprocessed in 3 format because I am implementing 3 different models. Same was applied to the training and testing phase.

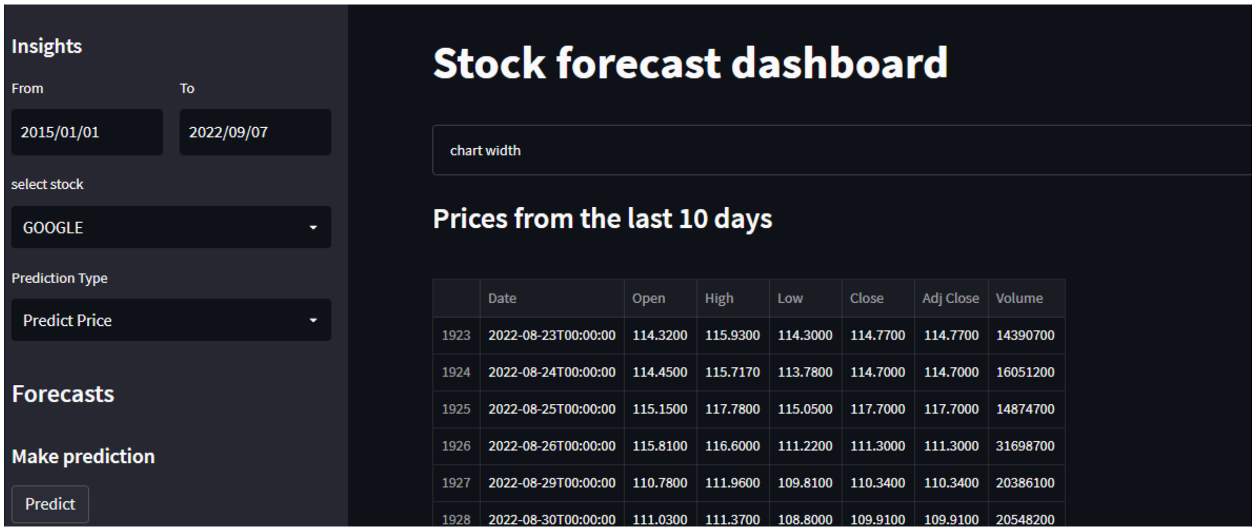


Home Page of User Interface.


Stock Forecast App

Select Stock

GOOGLE



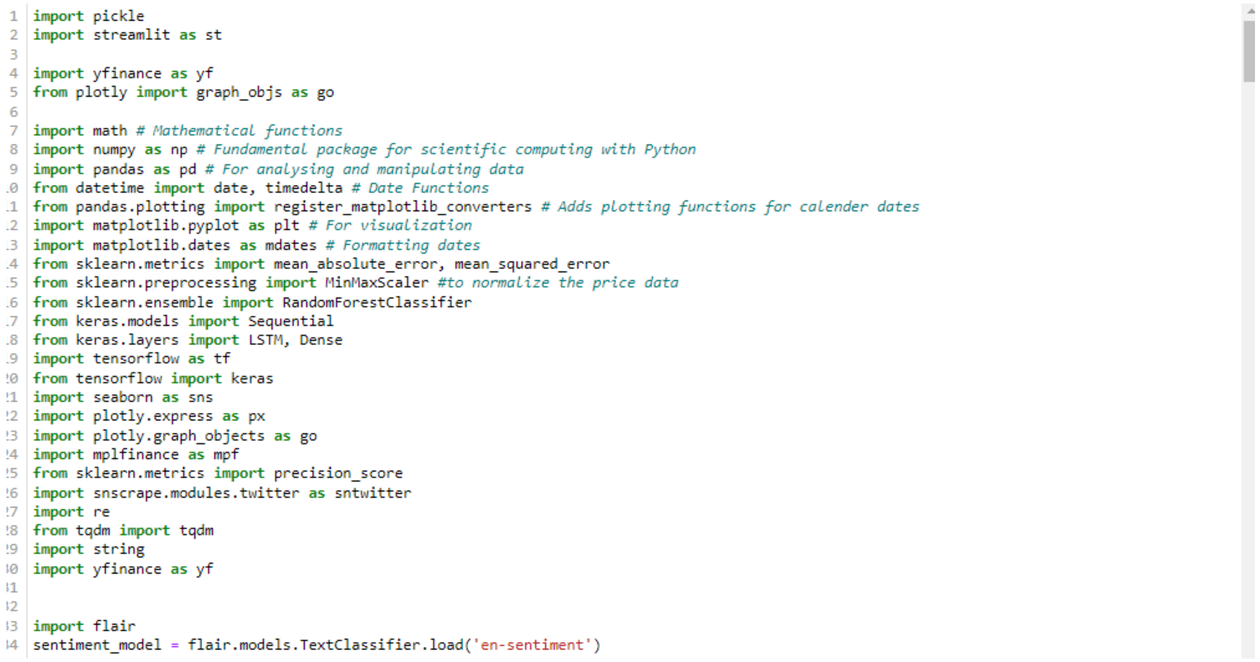
Stock forecast dashboard

Prices from the last 10 days

	Date	Open	High	Low	Close	Adj Close	Volume
1923	2022-08-23T00:00:00	114.3200	115.9300	114.3000	114.7700	114.7700	14390700
1924	2022-08-24T00:00:00	114.4500	115.7170	113.7800	114.7000	114.7000	16051200
1925	2022-08-25T00:00:00	115.1500	117.7800	115.0500	117.7000	117.7000	14874700
1926	2022-08-26T00:00:00	115.8100	116.6000	111.2200	111.3000	111.3000	31698700
1927	2022-08-29T00:00:00	110.7800	111.9600	109.8100	110.3400	110.3400	20386100
1928	2022-08-30T00:00:00	111.0300	111.3700	108.8000	109.9100	109.9100	20548200

Please see below steps adopted in creating the Graphic User Interface.

- **Step 1 - Importation of Libraries**



```python
import pickle
import streamlit as st

import yfinance as yf
from plotly import graph_objs as go

import math # Mathematical functions
import numpy as np # Fundamental package for scientific computing with Python
import pandas as pd # For analysing and manipulating data
from datetime import date, timedelta # Date Functions
from pandas.plotting import register_matplotlib_converters # Adds plotting functions for calender dates
import matplotlib.pyplot as plt # For visualization
import matplotlib.dates as mdates # Formatting dates
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing import MinMaxScaler #to normalize the price data
from sklearn.ensemble import RandomForestClassifier
from keras.models import Sequential
from keras.layers import LSTM, Dense
import tensorflow as tf
from tensorflow import keras
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import mplfinance as mpf
from sklearn.metrics import precision_score
import snscrape.modules.twitter as sntwitter
import re
from tqdm import tqdm
import string
import yfinance as yf


import flair
sentiment_model = flair.models.TextClassifier.load('en-sentiment')
```

4.3 Front-End Development Tools

Pickle – This package was imported because it was used to save the model after training and testing. For the model to be implemented the picked model/file will have to be reference on the front-end development.

Streamlite – Streamlite was import as 'st' which represent its alias, this package is the user interface adopted.

Numpy & Pandas – This library was imported for scientific computation and data analysis.

Yfinance – yfinance is the yahoo finance package on python. This package was imported on the front-end user interface because it is the data collection point for price prediction and market trend being used by LSTM and Random Forest Classifier.

Plotly – This is a python library imported user-interface to visualize my result/prediction.

Datetime – The datetime module was imported because the task is a timeseries problem hence it is required on the front-end user interface.

Sklearn.metrics – This module was imported to measure the performance of our model

Sklearn.preprocessing – This module was imported to preprocess our data for model implementation and preprocess the data for users to understand on the front-end. Normalization of the price data

RandomForest Classifier – This package was imported being the model to be implemented for market trend.

Keras & TensorFlow – Both packages were installed on the front end user interface. They are both deep learning module for LSTM implementation on the front-end

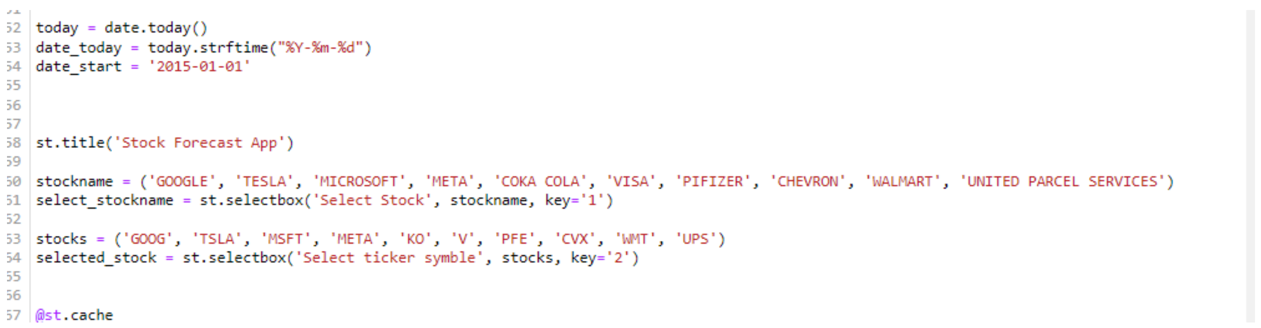
Seaborn – This package was installed to make high visualization of result.

Snscrape.module – This is a python module for scraping tweet on twitter and its being imported on the front-end development area

Flair -This is a Natural Language Processing library for text classification. This is being used on for my sentiment analysis.

4.4 System Design Implementation

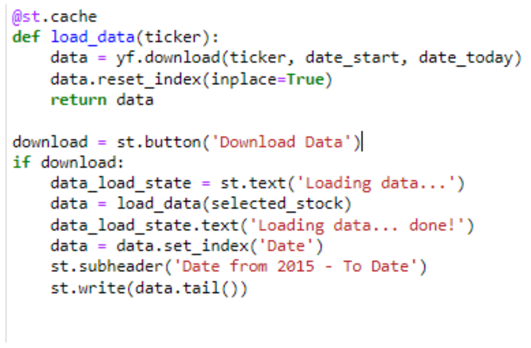
- Design/Define Data Collection Parameters & Web Application Title



```
today = date.today()
date_today = today.strftime("%Y-%m-%d")
date_start = '2015-01-01'



st.title('Stock Forecast App')

stockname = ('GOOGLE', 'TESLA', 'MICROSOFT', 'META', 'COKA COLA', 'VISA', 'PIFIZER', 'CHEVRON', 'WALMART', 'UNITED PARCEL SERVICES')
select_stockname = st.selectbox('Select Stock', stockname, key='1')

stocks = ('GOOG', 'TSLA', 'MSFT', 'META', 'KO', 'V', 'PFE', 'CVX', 'WMT', 'UPS')
selected_stock = st.selectbox('Select ticker symble', stocks, key='2')


@st.cache
```

One of the purpose of this study is to create a machine learning prediction systme for 10 global brands. The data collection parameters were set on the front end development area to specify the list of company's , symbols for the representation of each company's stock and the start date for historical data collection. The function st.title was used to name the title of the web application as 'Stock Forecast App'

- Design Progress Bar for User Request



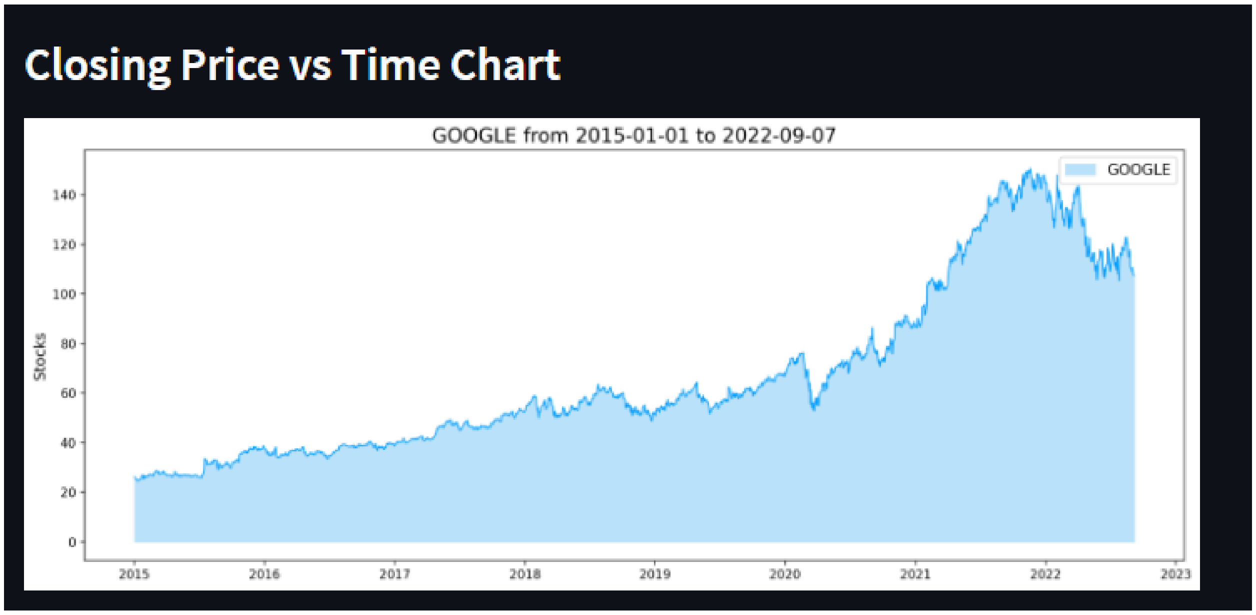
```
@st.cache
def load_data(ticker):
    data = yf.download(ticker, date_start, date_today)
    data.reset_index(inplace=True)
    return data

download = st.button('Download Data')
if download:
    data_load_state = st.text('Loading data...')
    data = load_data(selected_stock)
    data_load_state.text('Loading data... done!')
    data = data.set_index('Date')
    st.subheader('Date from 2015 - To Date')
    st.write(data.tail())
```

Based on the the st.cache was used to memorize the equations and check for the ticker, start date and current date at the point of data fetching while the st.button was used as the widget for downloading data. The function data_load_state = st.text('load data…') was created to show the progress of the user request while fetching information regarding the stock of a specific company while the function data_load_state.text('loading data… done') was used to send a message to the user that the request to fetch information has been complemented. While the function st.subheader('Date from 2015- To Date') was use to inform the user the available data on the dashboard.

- Visualize Exploratory Data Analysis for Users

An exploratory data analysis graph was displayed for users to see the relationship between time and close price for a selected stock. This was made possible sung the below code. St.subheader('Closing Price vs Time Chart') was used to name our graph. Showing the relationship between close price and time.

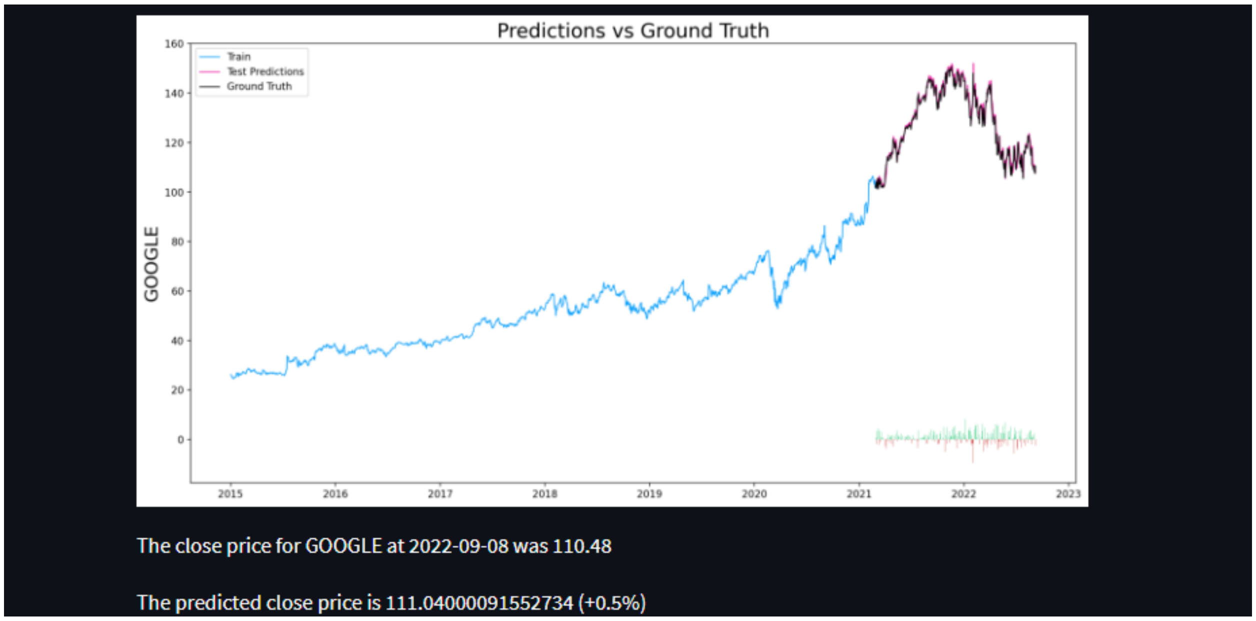
```python
st.subheader('Closing Price vs Time Chart')
def plot_raw_data():
    register_matplotlib_converters()
    years = mdates.YearLocator()
    fig, ax1 = plt.subplots(figsize=(16, 6))
    ax1.xaxis.set_major_locator(years)
    x = data.index
    y = data['Close']
    ax1.fill_between(x, 0, y, color='#b9e1fa')
    ax1.legend([select_stockname], fontsize=12)
    plt.title(select_stockname + ' from '+ date_start + ' to ' + date_today, fontsize=16)
    plt.plot(y, color='#039dfc', label=select_stockname, linewidth=1.0)
    plt.ylabel('Stocks', fontsize=12)
    st.pyplot(fig)

plot_raw_data()
```


Closing Price vs Time Chart

4.5 Result of Study

4.5.1 Result - Price Prediction

The main purpose of this study is to use artificial intelligence algorithm to predict stock market price for 10 global companies. The result for price prediction was done using graph presenting predicted price and ground truth for the stock. Please see below.

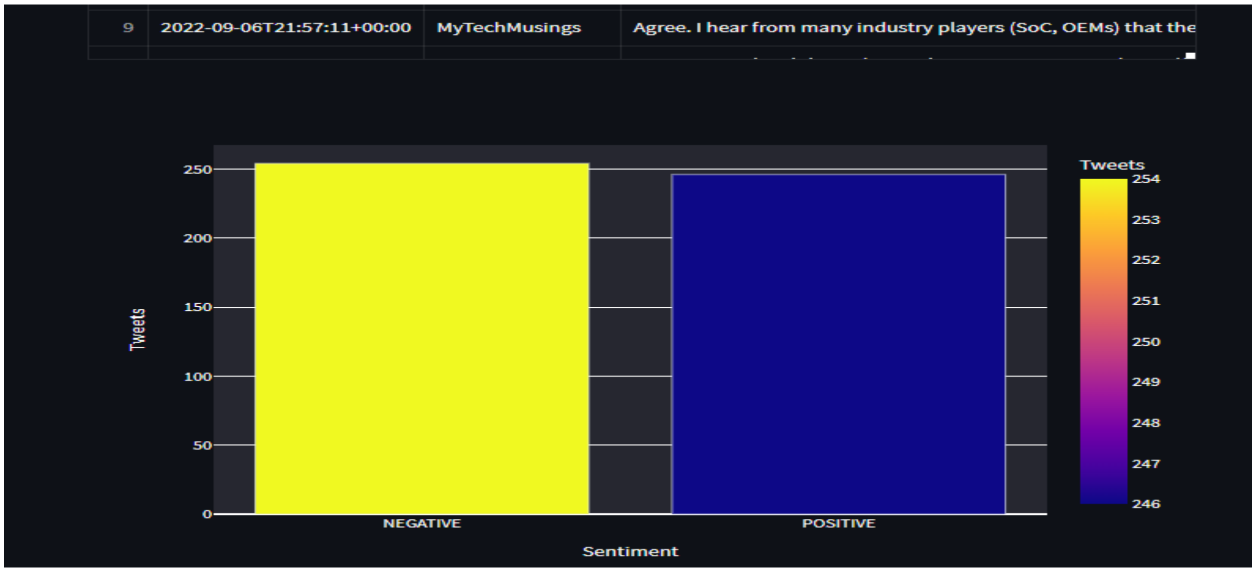


The close price for GOOGLE at 2022-09-08 was 110.48

The predicted close price is 111.04000091552734 (+0.5%)

4.5.2 Result - Sentiment Analysis.

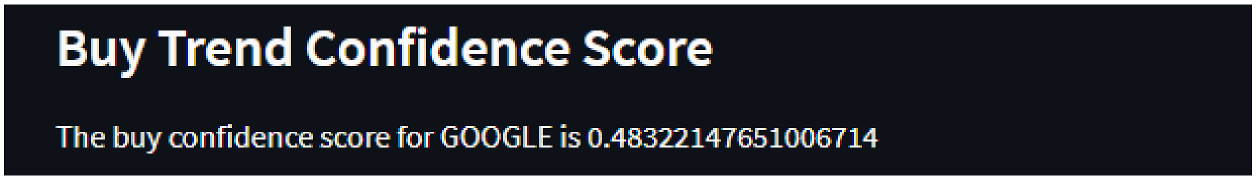
Another purpose of this study is to use artificial intelligence algorithms to advise users if to buy or sell stock using sentiment. The system use sentiment analysis outcome to advise users if to buy or not by scraping real-time tweet on twitter, classify the tweet to either be positive or negative and display it using a bar chat to measure number of positive and negative tweet to advise users/traders the general sentiment concerning the stock of a particular company. The result of sentiment analysis is displayed using a bar chat. Please see below.



9	2022-09-06T21:57:11+00:00	MyTechMusings	Agree. I hear from many industry players (SoC, OEMs) that the

4.5.3 Result - Random Forest.

Another outcome of this study is to advise customers if to buy or not using a confidence score. The confidence score technique is used to let users know how confident the system prediction and this result is was displayed in the form of a score between 0 to 1. Please see below.



Buy Trend Confidence Score

The buy confidence score for GOOGLE is 0.48322147651006714

CHAPTER FIVE

5.1 Challenges & Limitation.

1 Storage Capacity Challenges - The front-end application(StreamLite) has limited storage capacity to process users request. This is because the application is running on API from Yfinance, real-time sentiment from twitter from year 2015 till date. Collection of this data on a single click require a large memory capacity with a premium streamlite version whereas my application was built on a free version of Streamlite. To make it run, users will have to close the application windows and restart when there is an error on storage capacity or upgrade Streamlite to paid version for the purpose to access more storage capacity

2 1-day Prediction Window – The model cannot assist users in making long term prediction into the future for example prediction 30, 60, 90 days predictions into the future this because the model was designed to predict stock market prices and confidence score to buy or not for 1 day in the future for 1 day. i.e tomorrow.

3 Access to Twitter data – The application is running on a 3rd party API which was used to scrap tweet from Twitter. I tried to access the twitter API, I was only granted access to 7 days tweet from twitter, I had to use a 3rd party package in python called snscrape to access twitter data. However, the challenge with snscrape is that it takes a long time to download large amount of data from twitter compared to the direct twitter API

4 Source of Sentiment – Another challenge or limitation associated with this study is that the data source or scraping of sentiment is only limited to 1 application which is twitter.

5 External Factors associated with Target Variables– There are others factors that influences the stock performance of companies such as financial performance from quarterly to annual performance that were not considered as a factor. However, the purpose of sentiment model on the system was to capture these factors(what do people say) but the limitation with using sentiment is that not all external factors associated with stock performance can be captured by sentiment analysis.

5.2 Conclusion

In conclusion, this study shows you can combine three different analyses in other to make an informative decision on whether to trade a particular stock or not using artificial intelligence algorithm. I want to believe that there are improvements that can be made to better the result of these studies. An example will be to use several data sources of data to obtain sentiment rather than 1(twitter). This is to check that sentiment outcome is consistent regardless of the sentiment or data source. I will advise future researchers not to deploy 3rd party APIs which can be very slow. But to use the direct API from the data source like facebook, twitter etc for the purpose of optimizing the turnaround time of users request when the system is fetching data for prediction. Future work can also be extended to included all other variables/indexes that have direct influence on the performance of a stock e.g the historical and current financial performance of the company.

1 unique distinction of this study as compared to other studies in the implementation of artificial intelligence algorithm for stock market prediction is the use of confidence score between 0 to 1 to advise users/traders on how confident the system is for every prediction or advise provided for a particular stock. Previous studies done in stock market prediction are concerned about prediction of price and market trend but the question of how confident the prediction is or advise provided remain unanswered.

References

Alonso, Miquel N., Gilberto Batres-Estrada, and Aymeric Moulin. 2018 "Deep learning in finance: Prediction of stock returns with long short-term memory networks." https://doi.org/10.1002/9781119522225.ch13 pp 251-277

Alessio Staffini 2022 "Stock Price Forecasting by a Deep Convolutional Generative Adversarial Network" https://doi.org/10.3389/frai.2022.837596

Borges, T.A and Neves. R.F (2020) 'Applied Soft Computing' Journal 90 106187.https://doi.org/10.1016/j.asoc.2020.106187 P2.

Dos Santos Pinheiro, Leonardo, and Mark Dras. 2017 "Stock market prediction with deep learning: A character-based neural language model for event-based trading." Proceedings of the Australasian Language Technology Association Workshop 2017

Erkam Guresen, Gulgun Kayakutlu, Tugrul U. Daim, 2011 "Using artificial neural network models in stock market index prediction, Expert Systems with Applications", Volume 38, Issue 8, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2011.02.068 Pp 10389-10397,

Fama, Eugene F. 1965 "The Behavior of Stock-Market Prices." The Journal of Business 38, no. 1. http://www.jstor.org/stable/2350752. Pp 34-105

Gilberto Batres-Estrada 2015 "Deep Learning for Multivariate Financial Time Series" https://docplayer.net/20203888-Deep-learning-for-multivariate-financial-time-series-gilberto-batres-estrada.html

Ian J. Goodfellow. 2014 "Generative Adversarial Network" Departement d'informatique et de recherche op ´ erationnelle ´ Universite de Montr ´ eal ´ Montreal, QC H3C 3J7.

Mikhail Moshov 2022Decision trees for regular factorial languages, Elsevier Inc. https://doi.org/10.1016/j.array.2022.100203 p3

Mehdi Khashei & Zahra Hajirahimi (2018): "A comparative study of series Arima/mlp hybrid models for stock price forecasting" Communications in Statistics - Simulation and Computation, DOI: 10.1080/03610918.2018.1458138

Nan Jin , Zhao Wu , Hefei Wang, 2020. A hybrid model integrating deep learning with investor sentiment analysis for stock price prediction. P5

Person John, 2013, Mastering the Stock Market : High Probability Market Timing and Stock Selection Tools, John Wiley & Sons, Incorporated, 2012. P59

Ping-Feng Pai, Chih-Sheng Lin. 2005. “A hybrid ARIMA and support vector machines model in stock price forecasting”, Omega, Volume 33, Issue 6, ISSN 0305-0483. https://doi.org/10.1016/j.omega.2004.07.024 pp 497 - 505

RI Rasel and Nasimul 2016 “Hasan Financial instability analysis using ANN and feature selection technique: Application to stock market price prediction” 10.1109/ICISET.2016.7856515

Shah, A., Gor, M., Sagar, M. 2022. A stock market trading framework based on deep learning architectures. Multimed Tools Appl 81,. https://doi.org/10.1007/s11042-022-12328-x pp 14153–14171

Tao, M., Gao, S., Mao, D., Huang, H., 2022, Knowledge graph and deep learning combined with a stock price prediction network focusing on related stocks and mutation points, https://doi.org/10.1016/j.jksuci.2022.05.014, p2

Versamopoulos S., Berteis k., Almudelver G., 2018. Designing neural network based decoders for surface codes.

Xuzhi Deng, Aoshuang Ye, Jiashi Zhong , Dong Xu, Wangwang Ya, Zhaofang Song, Zitong Zhang, Jin Guo, Tao Wana, Yifan Tian, Hongguang Pan, Zhijing Zhang, Hui Wang, Chen Wu, Jiajia Shao, Xiaoyi Chen, 2022, Bagging–XGBoost algorithm based extreme weather identification and short-term load forecasting model, https://doi.org/10.1016/j.egyr.2022.06.072 p8664

Xingyu Zhou, Zhisong Pan, Guyu Hu, Siqi Tang, Cheng Zhao, 2018 "Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets", Mathematical Problems in Engineering, vol. 2018, Article ID 4907423,. https://doi.org/10.1155/2018/4907423