Predicting Stock Market Movement Using Naïve Bayes Model for Sentiment Analysis.

Sheila Mbadi
Carnegie Mellon University
2 PUBLICATIONS **0** CITATIONS

Jomo Kenyatta University of Agriculture and Technology (JKUAT)

Department of Computing

ICS2406: Computer Systems Project

REF: JKU/2/83/022

Predicting Stock Market Movement Using Naïve Bayes Model for Sentiment Analysis.

Author:
Name: Mbadi Atieno Sheila Reg. No: CS281-1556/2014

Submission Date: ____________________________ Sign: ________________________

Course: BSc. Computer Science

Supervisor 1:
Name: Prof. Waweru Mwangi Sign: _______________Date: ________________

Supervisor 2:
Name: Dr. Wekesa Chemwa Sign: _______________Date: ________________

Declaration

This research project is my original work and has not been presented for a degree in any other University

Mbadi Atieno Sheila

CS281-1556/2014

…………………. …………………

Signature Date

Acknowledgement

First and foremost, I would like to thank the Lord for giving me the strength and wisdom to finish this project.

My supervisors, Prof. Waweru Mwangi and Dr. Wekesa Chemwa. The guidance and assistance they provided throughout the project has made the realization of this project possible.

I would also like to thank Mrs. Harriet Ratemo for the never ending support as the Computer Systems project coordinator. Her diligence in directing this project process made the journey smooth.

My family and friends are much appreciated for the financial and emotional assistance they accorded me through the research. Their contribution to the success of this project is beyond measure and demands an acknowledgement.

Abstract

Stock market prediction is a complex task as markets are quite hard to understand.

The efficient market hypothesis suggests that stock prices are a function of information and rational expectations, and that newly revealed information about a company's prospects is almost immediately reflected in the current stock price. This would imply that all publicly known information about a company, would affect the movement of stock prices. One source of public information is the social media platform Twitter.

The aim of this project is to compare the accuracy of two stock market prediction models. The first one using only stock data such as open, close, high, low and adjusted close prices as features and the other one using all the aforementioned features plus the public sentiment about the company from Twitter.

Table of Figures

Table of Contents

List of Abbreviations

LSTM – Long Short Term Memory

API – Application Programming Interface

CSV – Comma Separated Values

HDF5 - Hierarchical Data Format 5

H5 – HDF5 files

AMZN – Amazon Stock Ticker

AAPL – Apple Stock Ticker

CHAPTER 1: INTRODUCTION

1.1 Background of Study

Social networks like Facebook and twitter have changed the way people communicate. People use such outlets to express their views and opinions on various topics. These information is beneficial to data analysts, businesses and other institutions that mine various opinions from users as feedback which they use to get insight on a product or service offered.

Sentiment analysis is used to extract such remarks of users and then gives them a polarity of positive, negative or neutral. The neutral case is usually ignored as it normally holds no weight in a study.

Sentiment analysis can be defined as using Natural Language Processing (NLP), statistics, or machine learning methods to extract, identify, or otherwise characterize the opinion content of a text unit. (Introduction to Sentiment Analysis, 2017)

The stock also called capital stock, of a corporation is constituted of the equity stock of its owners. A single share of the stock represents fractional ownership of the corporation in proportion to the total number of shares. (Stock, 2017)

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit for an investor. Thus there is need to come up with an accurate means of predicting stock market trend. (Stock market prediction, 2017)

Naïve Bayes classifier is a fast and simple machine learning classifier that can be used in the sentiment classification of text. However it is not the most efficient classifier thus there is need to improve it so as to yield high classification accuracy.

The efficient market hypothesis suggests that stock prices are a function of information and rational expectations, and that newly revealed information about a company's prospects is almost immediately reflected in the current stock price. This would imply that all publicly known information about a company, which obviously includes its price history, would already be reflected in the current price of the stock. Accordingly, changes in the stock price reflect release of new information and changes in the market generally.

The efficient market hypothesis therefore suggests that using public sentiments to predict stock market movement as well as a stock's price history will yield more accurate predictions.

1.2 Problem Statement

Using only price history to predict stock market movement does not yield the most accurate predictions. There is need to come up with a new model of predicting stock prices.

The Efficient Market Hypothesis (EMH) states that stock market prices are largely driven by new information; thus, it is prudent to incorporate public sentiments when coming up with a prediction model for stock market prices. (Burton, 2003).

Thus there is need to come up with a model that incorporates public sentiments when prediction stock market movement.

1.3 Objectives

1.3.1 Main Objective

1. To develop a hybrid stock market prediction model.

1.3.2 Specific Objectives

1. To study the enhancement of the Naïve Bayes classifier for sentiment classification of tweets and use it.
2. To study the best techniques for stock prediction and implement one.
3. To implement a hybrid stock prediction model and calculate its accuracy.

1.3.3 Research Questions

1. How can the Naïve Bayes classifier be enhanced to improve the classification accuracy of tweets?
2. What is the best technique for stock prediction?
3. Is it possible to implement a hybrid stock prediction model and check its accuracy?

1.4 Justification

Stock market prediction is complex as markets are quite hard to understand. There is need for a stock prediction model that will provide predictions with a high degree of accuracy so as to yield significant profit by any investor.

A lot of research has been conducted on the topic of stock prediction but very few actually take public sentiments into consideration. From Bollen, Mao, & Zeng (2010) research work, the incorporation of sentiment analysis in predicting stock prices results in predictions with a higher degree of accuracy when compared to those that only use price history. Therefore it is prudent to come up with a stock prediction model that takes public sentiment into account.

1.5 Schedule

Activity	**Estimated Start and End Date**	**Deliverables**
Project Identification and Proposal Writing	28th September - 6th October 2017	Project Proposal
Project Research	10th October - 12th January 2018	Literature Review Document and Methodology
Project Analysis and Design	7th May - 4th June 2018	Analysis and Design Document
System Implementation	11th June - 16th July 2018	Working Prototype
System Testing and Debugging	17nd July- 27th July 2018	Final Prototype
Project Submission	10th August 2018	Complete System with Documentation

Figure 1. 1 Table of Schedule

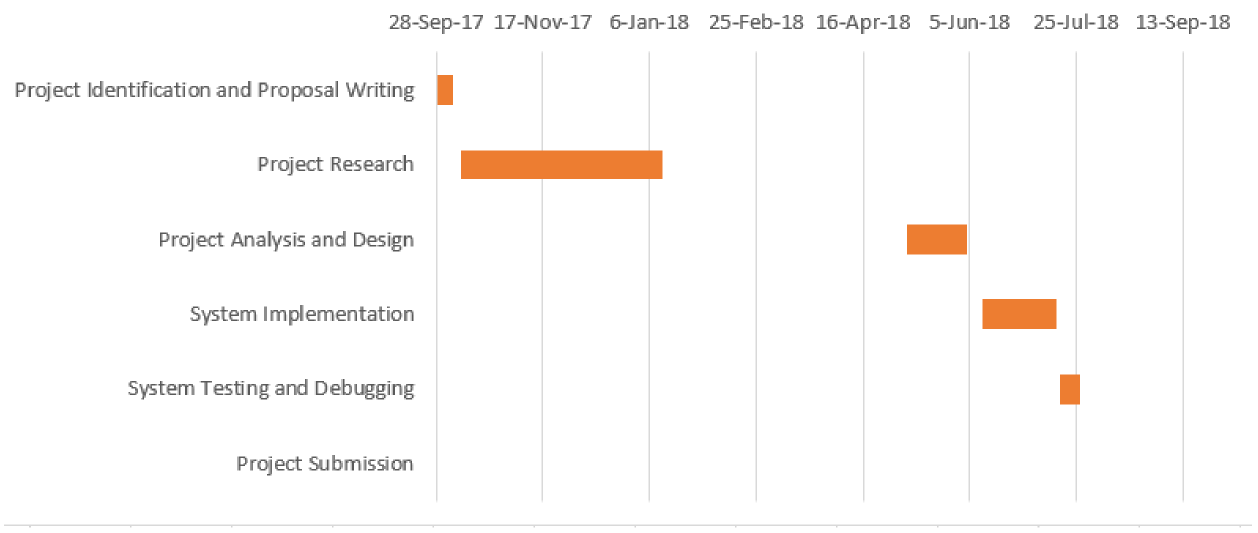


Figure 1. 2 Gantt Chart

1.6 Budget

Resource	**Cost**	**Justification**
Printing and Binding	Kshs 1500	Need for printing and binding of documents
Airtime and Bundle Purchase	Kshs 3500	Researching on the internet and contacting supervisor when there is need

Flash Disk	Kshs 2000	Data transfer when printing and backing up of data
Total	**Kshs 7000**	

Figure 1. 3 Budget table

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This section covers research on sentiment analysis, stock prediction and the Naive Bayes algorithm. Steps in sentiment analysis is covered as well as the techniques used in stock prediction. Previous work done on the area of study is also be studied.

2.2 Sentiment Analysis

2.2.1 Introduction

Sentiment Analysis is an area of study within Natural Language Processing that is concerned with identifying the mood or opinion of subjective elements within a text towards an entity (Bhadane, Dalal & Doshic, 2015). The entity can represent individuals, events, topics or a product/service offered.

It is becoming a popular area of research and social media analysis, especially around user reviews and tweets. It is a special case of text mining generally focused on identifying opinion polarity, and while it’s often not very accurate, it can still be useful (Text Classification for Sentiment Analysis – Naive Bayes Classifier, 2010).

The tasks involved in sentiment analysis include finding opinions, identify the sentiments they express, and then classifying their polarity.

2.2.2 Sentiment Classification Techniques

There are two main sentiment classification techniques, these are lexical-based approach and machine learning approach. These two have subgroups which are shown in Figure 1 (Medhat, Hassan, & Korashy, 2014).

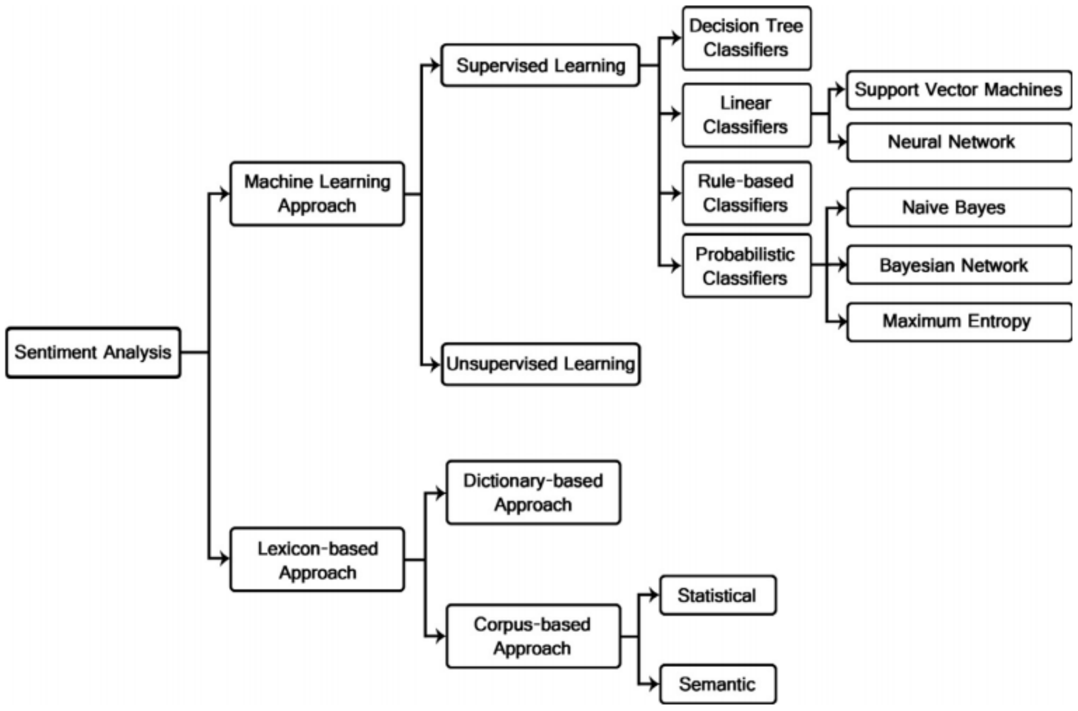


Figure 2. 1 Sentiment Classification Techniques

Source: Medhat et al., (2014)

Lexical and machine learning approaches are sentiment classification techniques, they are used to classify a given piece of natural language text according to the opinions expressed in it.

For the dictionary-based lexical approach, a dictionary is prepared to store the polarity values of lexicons. For calculating polarity of a text, polarity score of each word of the text, if present in the dictionary, is added to get an ‘overall polarity score’. If the overall polarity score of a text is positive, then that text is classified as positive, otherwise it is classified as negative. Though this approach seems very basic, variants of this lexical approach have been reported to have considerably high accuracy.

Since the polarity of the text depends on the score given to each lexicon, there has been a large volume of work dedicated to discovering which lexical information is most efficient. A target word along with the word ‘good’, and a second with the target word with the word ‘bad’ was used by Turney (2002) in the AltaVista search engine queries. The polarity of the target word was determined by the search result that returned the most hits. This approach resulted in accuracy of 65%. In another research, Turney & Littman (2003) mapped the semantic association between the target word and each word from the selected set of positive and negative words to a real number. By subtracting a word's association strength to a set of negative words from its association strength to a set of positive words, an accuracy rate of 82% was achieved.

For the machine learning approach, a series of feature vectors are chosen and a collection of tagged corpora is used to prepare a model. The model can then be used to classify an untagged corpus of text. In machine learning approach, the selection of features is crucial to the success rate of the classification. Most commonly, a variety of unigrams, which are, single words from a document, or n-grams, which are, two or more words from a document in sequential order are chosen as feature vectors. Most commonly employed classification techniques of machine learning are Support Vector Machines (SVMs), Naive Bayes and Maximum Entropy algorithm. Other techniques include Neural Networks, Bayesian Networks and Decision Tree Classifiers. The accuracy results for these algorithms greatly depends on the features selected.

2.2.3 Annotation

To check the accuracy of any emotion detecting algorithm, the results need to be compared to a human-labeled text. The process in which humans manually label a text is called annotation. Annotation can be done on multiple levels: word, sentence, paragraph, section, or even the entire document.

Annotation can be based on polarity, emotion and intensity. When annotating on polarity, text is labeled with positive, negative or neutral emotion. Text annotated on emotion is labeled based on some predefined list. The most common lists of emotions used are those suggested by Ekman, Izard and Plutchik (Mulcrone, 2012). Additionally, some studies annotate the text by labeling the intensity of the emotion. Intensity is based on a numeric scale, but there are no standards for this type of annotation. The first two categories are the ones that are applicable for this study, this is because emotions are closely related to the polarity of texts.

In general, studies either use pre-annotated datasets to test an algorithm or undergo a small annotation process. In the latter case, annotators who are qualified to label emotion in text, such as psychologists are used. The text is annotated to the given specification and level of analysis. Sometimes annotators are given an additional word list that consists of words from the original text. These lists help determine which words are attached to a specific emotion and which vary by context. When the annotation process is complete the agreement among the annotators is calculated using a method such as the Kappa Value.

2.3 Steps in Sentiment Analysis

2.3.1 Data Gathering

The data to be studied is collected from the appropriate data source.

2.3.2 Preprocessing

Once the data to be analyzed is collected, the text is split into individual words each word becoming a feature in the feature vector which is stored in a bag-of-words. This breakdown of a sentence into individual words is known as tokenization.

Although many tokenizers are geared towards throwing punctuation marks away, for sentiment analysis a lot of valuable information could be deduced from them. "!" puts extra emphasis on the negative/positive sentiment of the sentence, while "?" can mean uncertainty (no sentiment). ", ', [], () can mean that the words belong together and should be treated as a separate sentence. Same goes for words which are bold, *italic* or <u>underlined</u>. But symbols such as the "@" symbol, and links can be removed as they are regarded as noise generating elements.

Word Normalization should be applied. This is the reduction of each word to its base/stem form (by chopping of the affixes). This is known as stemming or lemmatizing. An example is walking to walk. Capital letters should be normalized to lowercase, unless it occurs in the middle of a sentence; this could indicate the name of a writer, place, brand etc. Words with an apostrophe should also be handled. "George's phone" should obviously be tokenized as "George" and "phone", but I'm, we're, they're should be translated as I am, we are and they are. To make it even more difficult, it can also be used as a quotation mark.

Other preprocessing steps include the elimination of stop words. These are the un-informative words in tweets which include "so", "and", "or" and "the". A stop-word list can be created or searched against a language-specific stop word dictionary. Further, Parts of Speech taggers (PoS) are used to classify words into the English 8-parts of speech. Nouns and pronouns do not contain any sentiment according to (Fang and Zhan, 2015). As such these words are exempted from the classification process.

2.3.3 Feature Extraction and Sentiment Classification

After the text has been segmented into sentences, each sentence has been segmented into words, the words have been tokenized and normalized. We can make a simple bag-of-words model of the text. In this bag-of-words representation you only take individual words into account and give each word a specific subjectivity score. This subjectivity score can be looked up in a sentiment lexicon. If the total score is negative the text will be classified as negative and if it is positive the text will be classified as positive.

The sentiment lexicon can be created using some simple statistics of the training set. To do this the class probability of each word present in the bag-of-words will be determined.

The sentiment lexicon is simple to make, but is less accurate because it does not take the word order of the grammar into account. A simple improvement on using unigrams would be to use bigrams and trigrams. That is, not to split a sentence after words like "not", "no", "very", "just" etc. It is easy to implement but can give significant improvement to the accuracy.

The best words to put in a bag-of-words include salient words that give domain specific information and discriminatory words that help to clear distinctions of polarities (Mulcrone, 2012). The bag-of-words model simply uses a statistical approach to classify polarities.

2.3.4 Train and build model

The above mentioned steps are carried out in the training set. A test set is then used to do classifications and to tell the efficiency of the classifier.

2.4 Application of Machine Learning in Sentiment Analysis

2.4.1 Naïve Bayes Classifier

A Naive Bayes classifier is a simple probabilistic model based on the Bayes rule along with a strong independence assumption.

The Naïve Bayes model involves a simplifying conditional independence assumption. That is, given a class (positive or negative), the words are conditionally independent of each other. Due to this simplifying assumption the model is termed as "naïve". This assumption does not affect the accuracy in text classification by much but makes really fast classification algorithms applicable for the problem.

The maximum likelihood probability of a word belonging to a particular class is given by the equation:

$$\boxed{P(x_i|C) = \frac{Count\ of\ x_i\ in\ texts\ of\ class\ C}{Total\ number\ of\ words\ in\ texts\ of\ class\ C}}$$

The frequency counts of the words can be stored in hash tables during the training phase.

According to the Bayes Rule, the probability of a particular text belonging to some particular class is given by:

$$\boxed{P(c_i|t) = \frac{P(t_i|c_i)\ * P(c_i)}{P(t)}}$$

If the simplifying conditional independence assumption is used, that is, given a class (positive or negative) the words are conditionally independent of each other. The following equation will be used.

$$\boxed{P(c_i|t) = \frac{(\prod P(x_i|c_j))\ * P(c_j)}{P(t)}}$$

Here the x_i's are the individual words of the text. The classifier outputs the class with the maximum posterior probability.

Naive Bayes classifiers are thought to be less accurate than their more sophisticated counterparts like support vector machines and logistic regression classifier. A simple Naive Bayes classifier can be enhanced to match the classification accuracy of these more complicated models for sentiment analysis. The advantages of using Naive Bayes as our classifier are:

- Naive Bayes classifiers due to their conditional independence assumptions are extremely fast to train and can scale over large datasets.
- They are robust to noise and less prone to over-fitting.
- Ease of implementation is also a major advantage of Naive Bayes.

A significantly high accuracy can be achieved by applying the following processes to the simple Naive Bayes classifier:

1. **Bernoulli Naïve Bayes**

Duplicate words are removed from the text as they don't add any additional information, this type of Naïve Bayes algorithm is called Bernoulli Naïve Bayes.

Including just the presence of a word instead of the count has been found to improve performance marginally, when there is a large number of training examples.

The data under study should be distributed according to the multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued variable. Samples are required to be represented as binary-valued feature vectors

The decision rule for Bernoulli naive Bayes is based on

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

In text classification, word occurrence vectors (rather than word count vectors) are be used to train and use this classifier.

2. Laplacian Smoothing

If the classifier encounters a word that has not been seen in the training set, the probability of both the classes would become zero and there won't be anything to compare between. This problem can be solved by Laplacian smoothing

$$\boxed{P(x_i C_j) = \frac{Count\ (x_i)\ +\ k}{(k\ +\ 1)\ *(No\ of\ words\ in\ class\ C_j)}}$$

Usually, k is chosen as 1. This way, there is equal probability for the new word to be in either class. Since Bernoulli Naïve Bayes is used, the total number of words in a class is computed differently. For the purpose of this calculation, each text is reduced to a set of unique words with no duplicates.

3. Negation Handling

Negation handling is one of the factors that contribute significantly to the accuracy of this classifier. A major problem faced during the task of sentiment classification is that of handling negations. Since we are using each word as a feature, the word "good" in the phrase "not good" will be contributing to positive sentiment rather that negative sentiment as the presence of "not" before it is not taken into account.

To solve this problem a simple algorithm for handling negations using state variables and bootstrapping will be devised. This builds on the idea of using an alternate representation of negated forms. This algorithm uses a state variable to store the negation state. It transforms a word followed by a not into “not_” + word. Whenever the negation state variable is set, the words read are treated as “not_” + word. The state variable is reset when a punctuation mark is encountered or when there is double negation.

Since the number of negated forms might not be adequate for correct classifications. It is possible that many words with strong sentiment occur only in their normal forms in the training set. But their negated forms would be of strong polarity.

This problem is addressed by adding negated forms to the opposite class along with normal forms of all the features during the training phase. That is to say if we encounter the word “good” in a positive document during the training phase, we increment the count of “good” in the positive class and also increment the count of “not_good” for the negative class. This is to ensure that the number of “not_” forms are sufficient for classification. This modification will result in a significant improvement in classification accuracy (about 1%) due to bootstrapping of negated forms during training. This form of negation handling can be applied to a variety of text related applications

4. n-grams

Generally, information about sentiment is conveyed by adjectives or more specifically by certain combinations of adjectives with other parts of speech.

This information can be captured by adding features like consecutive pairs of words (bigrams), or even triplets of words (trigrams). Words like "very" or "definitely" don't provide much sentiment information on their own, but phrases like "very bad" or "definitely recommended" increase the probability of a document being negatively or positively biased. By including bigrams and trigrams, we will be able to capture this information about adjectives and adverbs. Bigrams will be used as the 280 characters limit will hinder the use of more n-grams. The counts of the n-grams will be stored in a hash table along with the counts of unigrams.

5. Feature Selection

Feature selection is the process of removing redundant features, while retaining those features that have high disambiguation capabilities.

The use of higher dimensional features like bigrams and trigrams presents a problem, that of the number of features increasing. Most of these features are redundant and noisy in nature. Including them would affect both efficiency and accuracy. A basic filtering step of removing the features/terms which occur only once will be performed. The features can then be further filtered on the basis of mutual information.

2.5 Stock Prediction

2.5.1 Introduction

Stock prediction methodologies fall into three broad categories which can and often do overlap. They are fundamental analysis, technical analysis/charting and technological methods.

Technical analysis is the interpretation of the price action of a company's underlying stock (or any tradable financial instrument). It utilizes various charts and statistical indicators to determine price support/resistance, range and trends. It identifies historically relevant price patterns and behaviors to help forecast potential direction of the stock. This methodology focuses only on the price of the shares, not the operations of the company (Technical Analysis, 2017).

Technical analysis seeks to determine the future price of a stock based solely on the (potential) trends of the past price (a form of time series analysis). Techniques such as exponential moving average (EMA) are employed.

This research work will focus on technical analysis methodology.

2.5.2 Techniques Used in Stock Prediction

2.5.2.1 Time Series Analysis

A time series is an ordered sequence of values of a variable at equally spaced time intervals. It can be taken on any variable that changes over time.

In investing, it is common to use a time series to track the price of a security over time. This can be tracked over the short term, such as the price of a security on the hour over the course of a business day, or the long term, such as the price of a security at close on the last day of every month over the course of five years.

Time series analysis can be useful to see how a given asset, security or economic variable changes over time. It can also be used to examine how the changes associated with the chosen data point compare to shifts in other variables over the same time period. It can therefore be used to determine the relationship between public sentiments of a stock and the movement in price of the mentioned stock.

Time series can be used for quantitative forecasting by using information regarding historical values and associated patterns to predict future activity. Most often, this relates to trend analysis, cyclical fluctuation analysis and issues of seasonality.

Inherent in the collection of data taken over time is some form of random variation. There exist methods for reducing or canceling the effect due to random variation. An often-used technique is "smoothing". This technique, when properly applied, reveals more clearly the underlying trend, seasonal and cyclic components. Therefore, in this research work there will be need for **smoothing functions** that react quickly to changes in the signal, hence the need for **moving averages**.

2.5.2.1.1 Moving Average (MA)

A moving average (MA) is a widely used indicator in technical analysis that helps smooth out price data by filtering out the “noise” from random price fluctuations and form a trend following indicator. They do not predict price direction, but rather define the current direction with a lag. Moving averages lag because they are based on past prices. Despite this lag, moving averages help smooth price action and filter out the noise.

The two basic and commonly used moving averages are the:

i. **Simple moving average (SMA).**
ii. **Exponential moving average (EMA)**.

2.5.2.1.1.1 Simple Moving Average

A Simple Moving Average is formed by computing the average price of a security over a specific number of periods. Most moving averages are based on closing prices. A 5-day simple moving average is the five-day sum of closing prices divided by five. As its name implies, a moving average is an average that moves. Old data is dropped as new data comes available. This causes the average to move along the time scale.

2.5.2.1.1.2 Exponential Moving averages (EMA)

An exponential moving average (EMA) is a type of moving average that is similar to a simple moving average, except that more weight is given to the latest data. It's also known as the exponentially weighted moving average. This type of moving average reacts faster to recent price changes than a simple moving average.

It is often used where latency is critical, such as in real time financial analysis. In this average, the weights decrease exponentially. Each sample is valued some percent smaller than the next most recent sample. With this constraint the moving average can be calculated very efficiently.

The formula is:

$$avg_t = (alpha * sample_t) + ((1 – alpha) * avg_{t-1})$$

Where alpha is a constant that describes how the simple weights decrease over time. For example if each sample was to be weighted at 80% of the value of the previous sample, you would set alpha = 0.2.

Each new sample needs to be average with the value of the previous average. So computation is very fast. In theory all previous samples contribute to the current average, but their contribution becomes exponentially smaller over time.

This is a very powerful technique, and probably the best in getting a moving average that responds quickly to new samples, has good smoothing properties and is fast to compute.

It is therefore beneficial to apply the Exponential Moving Average in our Time Series Analysis to get better results in predicting the direction of the stock market movement.

2.5.2.1.2 Long Short Term Memory (LSTM) Neural Network

Long Short-Term Memory models are extremely powerful time series models. They can predict an arbitrary number of steps into the future. An LSTM module (or cell) has 5 essential components which allows it to model both long-term and short-term data.

Cell state (c_t): This represents the internal memory of the cell which stores both short term memory and long term memories

Hidden state (h_t): This is output state information calculated with respect to current input, previous hidden state and current cell input which are eventually use to predict the future stock market prices. Additionally, the hidden state can decide to only retrieve the short or long-term or both types of memory stored in the cell state to make the next prediction.

Input gate (i_t): Decides how much information from current input flows to the cell state

Forget gate (f_t): Decides how much information from the current input and the previous cell state flows into the current cell state

Output gate (o_t): Decides how much information from the current cell state flows into the hidden state, so that if needed LSTM can only pick the long-term memories or short-term memories and long-term memories.

A cell is pictured below.

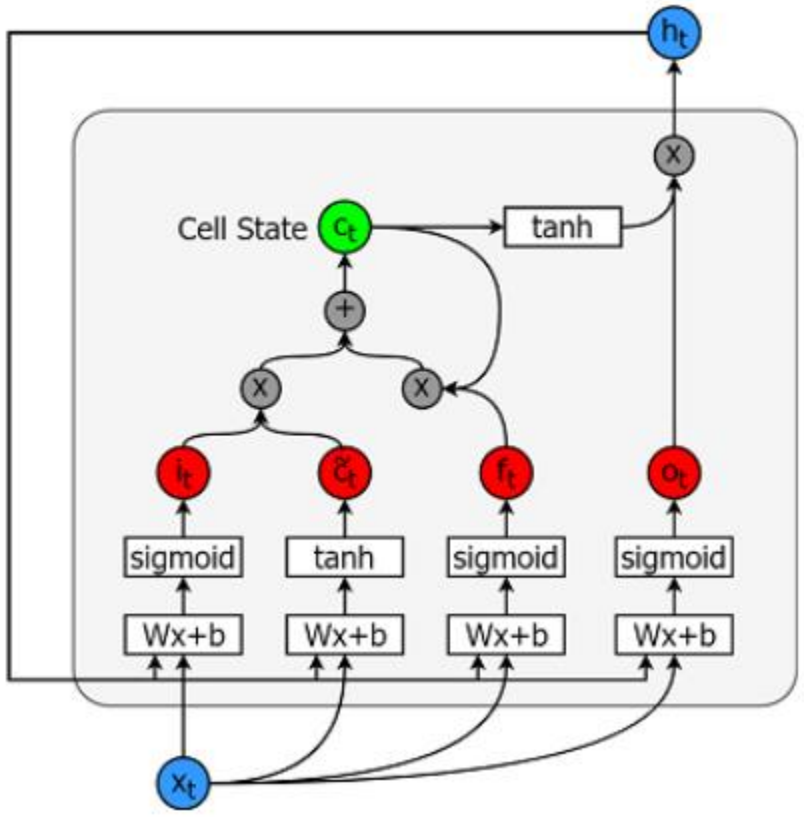


Figure 2. 2 Long Short Term Memory Cell

Source: Thushan Ganegedara, (2018). LSTM Neural Network

2.6 Previous Work

2.6.1 Twitter Sentiment Analysis to Predict the Stock Market Movement

Earlier work done by Bollen, Mao, & Zeng, (2010) shows how collective mood on Twitter (aggregate of all positive and negative tweets) is reflected in the Dow Jones Industrial Average (DJIA) index movements. They investigated whether public sentiment, as expressed in large-scale collections of daily Twitter posts, could be used to predict the stock market.

They used two tools to measure variations in the public mood from tweets submitted to the Twitter service from February 28, 2008 to December 19, 2008. The first tool, OpinionFinder, analyzed the text content of tweets submitted on a given day to provide a positive vs. negative daily time series of public mood. The second tool, Google-Profile of Mood States (GPOMS), similarly analyzed the text content of tweets to generate a six-dimensional daily time series of public mood to provide a more detailed view of changes in public along a variety of different mood dimensions.

The resulting public mood time series were correlated to the DJIA to assess their ability to predict changes in the DJIA over time. The results indicated that the prediction accuracy of a standard stock market prediction models is significantly improved when certain mood dimensions are included, but not others. In particular variations along the public mood dimensions of Calm and Happiness as measured by GPOMS seem to have had a predictive effect, but not general happiness as measured by the OpinionFinder tool.

The research found an accuracy of 87.6% in predicting the daily up and down changes in the closing values of the DJIA and a reduction of the Mean Average Percentage Error by more than 6%. This therefore shows that there is a strong correlation between twitter data and stock market movement.

2.6.2 Sentiment Classification using an Enhanced Naive Bayes Model.

Narayanan, Arora & Bhatia showed that a simple Naive Bayes classifier can be enhanced to match the classification accuracy of more complicated models for sentiment analysis by choosing the right type of features and removing noise by appropriate feature selection. Naive Bayes classifiers were thought to be less accurate than their more sophisticated counterparts like support vector machines and logistic regression. The enhanced Naïve Bayes classifier was able to achieve an 88.8% accuracy from the 73.77% accuracy when using the original Naive Bayes algorithm with Laplacian smoothing. The improvements from applying the various processes are illustrated in Figure 2.

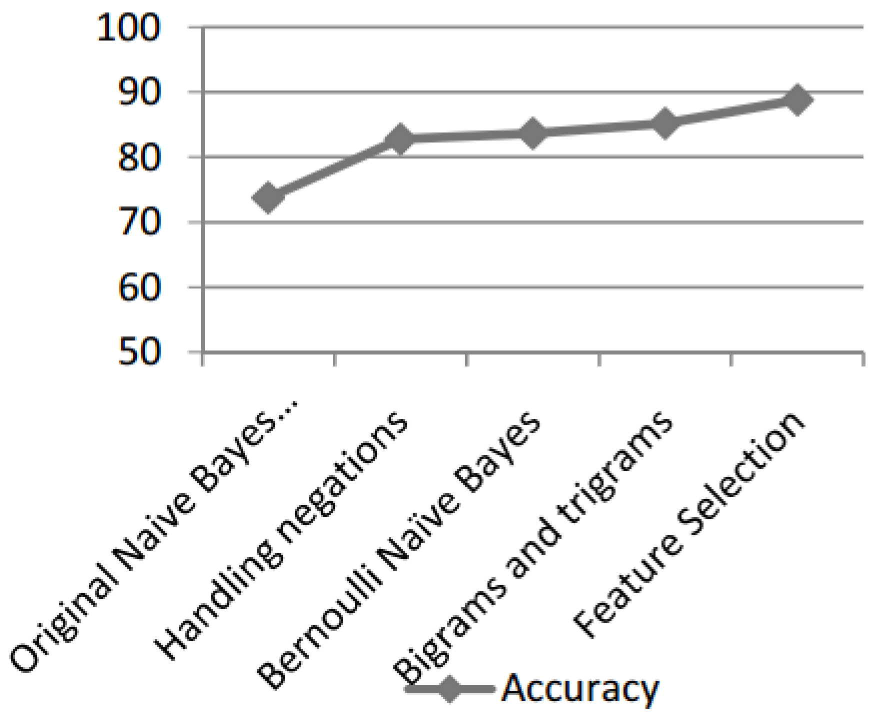


Figure 2. 3 Evolution of classification accuracy.

Source: Narayanan et al., (2007)

2.6.3 Forecasting Stock Market Trend using Exponential Moving Average

From the research, Alexander Decker concluded that the curve of the moving average shows the market/ stock trend. If the curve is in an upward direction, the market heads towards Bull Run. If the moving average is going down the market is in Bear phase and a flat moving average shows consolidation.

He further states that there are 3 steps in calculating the EMA:

1. Calculate the simple moving average for the initial EMA value. An exponential moving average (EMA) has to start somewhere, so a simple moving average is used as the previous period's EMA in the first calculation.
2. Calculate the weighting multiplier.

3. Calculate the exponential moving average for each day between the initial EMA value and today, using the price, the multiplier, and the previous period's EMA value.

The formula below is for a 10-day EMA.

Initial SMA: 10-period sum / 10

Multiplier: (2 / (Time periods + 1)) = (2 / (10 + 1)) = 0.1818 (18.18%)

EMA: {Close – EMA (previous day)} x multiplier + EMA (previous day).

By using this technical analysis tool, he was able to find out the trends in different stocks; whether they are going upwards, downwards or stagnant. This increases the chance of investors to predict the prices more accurately by prediction the trend reversals in advance and hence increased profit in the share markets.

In short, EMA when combined with market behavioral analysis, such as Twitter sentiments about a company, then the probability of it being correct for forecasting market trends will increase.

2.6.4 Predicting Stock Prices Using LSTM

Murtaza Roondiwala, Harshal Patel and Shraddha Varma presented a recurrent neural network (RNN) and Long Short-Term Memory (LSTM) approach to predict stock market indices.

They argued that Recurrent Neural Networks (RNN) had proved one of the most powerful models for processing sequential data. Long Short-Term memory is one of the most successful RNNs architectures. LSTM introduce the memory cell, a unit of computation that replaces traditional artificial neurons in the hidden layer of the network. With these memory cells, networks are able to effectively associate memories and input remote in time, hence suit to grasp the structure of data dynamically over time with high prediction capacity.

For analyzing the efficiency of the system they used the Root Mean Square Error (RMSE). The model was found to have the least error when all the four stock parameters they were analyzing were used (i.e. High, Low, Close and Open) and also when the number of epochs used in training the model were many, in their case they used 500 epochs.

2.7 Conclusion

Opinion mining of people's thoughts on the stock market is very important in the construction of a stock prediction model. When combined with stock prediction techniques such as the use of Long Short Term Memory Neural Networks, the degree of accuracy improves significantly.

2.8 METHODOLOGY

2.8.1 Introduction

This section covers the research design, research methods, data collection tools used to gather information during the research and the development tool used

2.8.2 Research Design

A research design is an arrangement of conditions or collections.

The research design chosen for this study is the Correlation design type and the subtype under consideration is Observational study. The type of observational study that will be looked at is longitudinal study which is a correlational research study that involves repeated observations of the same variables over long periods of time.

The longitudinal study is chosen as the twitter feeds as well as the stock data to be analyzed will involve repeated observations of these variables over a long period of time before conclusions are made.

2.8.3 Research Methodology

2.8.3.1 Research Methods/ Processes/ Techniques/ Procedure

The works reviewed in this research were selected and analyzed based on the following criteria:

1. Steps that can be taken to enhance the Naïve Bayes classifier for it to be used for sentiment analysis classification with a high degree of accuracy.
2. Techniques that can be used for stock market prediction.
3. Studies that show a correlation between stock market movement and the public sentiment on this stocks

2.8.4 Data Collection Tools

The techniques used for data collection are:

1. **The study of existing literature.**

This is so because the research is a confirmatory one. The research tests priori hypotheses that were set by researches who have studied the proposed solution such as Bollen et.al. Previous studies carried out by various researchers formed a rich source of information and was the basis of all other data collection techniques used.

2. **Primary Data collection**

Twitter data and the stock data to be analyzed comes directly from the source. Twitter feed studied are those that users have typed.

3. **Secondary Data collection**

Secondary data in form of an annotated stock market corpus are used in this research to train the sentiment classification model.

2.8.5 Development Tool

2.8.5.1 Process Model

2.8.5.1.1 Waterfall Model

This Software Development Lifecycle Model (SDLC) model is used as the requirements are well documented, clear and fixed. This model is simple and easy to understand and use and it has clearly defined stages and milestones which are key in this project.

The sequential phases in this model are:

1. Requirement Gathering and analysis
2. System Design.
3. Implementation.
4. Integration and Testing.
5. Deployment of system
6. Maintenance

CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

3.1 Systems Analysis

3.1.1 Introduction

This section will cover the system methodology used, the data sources, feasibility study and the system requirements.

3.1.2 Systems Development Methodology

Structured Systems Analysis and Design Methodology (SSADM) will be used.

Structured design methods propose that software design should be tackled in a methodical way. Designing a system involves following the steps of the method and refining the design of a system at increasingly detailed levels.

3.1.3 Data Sources

The sentiment data to be analyzed will come from Twitter, the social media platform. The platform offers an Application Programming Interface that is free and easy to use and it provides a lot of public sentiment about various products and services that are traded in the stock market.

The stock market data will come from Alpha Vantage Inc. It is a leading provider of free APIs for real-time and historical data on stocks among other financial data. The best part is that they provide this data for free and their Application API is well documented making it easy to use.

3.1.4 Challenge in Sentiment Analysis of Twitter Data

Twitter data costs a lot of money, and if it has not been possible to retrieve or set up a system to retrieve Twitter data within 7 days on a topic of interest, then it becomes difficult to obtain the data. This is because using the free Twitter public API ecosystem it is only possible to retrieve Twitter data going back in time 7 days. However, it is possible to obtain this data through other means such as obtaining them from Twitter at a fee, by using a licensed reseller of Twitter data or by using an existing dataset. Historical Twitter data can range from not that expensive, to very expensive depending on both the query and time of retrieval (Ahmet, 2015).

3.1.5 Feasibility Study

This is the analysis of a problem to determine if the conditions are right for it to be solved effectively. The results of the study will determine whether the solution should be implemented.

Four aspects of the project were studied.

3.1.5.1 Technical Feasibility

The technical feasibility checks whether the right technology exists to solve the problem.

The Twitter streaming API supported all the data requirements when it came to obtaining public sentiments. All its shortcomings can be easily solved. One major one was only obtaining data from the past 7 days and this was be mitigated by storing the results in Comma Separated Values (CSV) files for future references.

The financial API from Alpha Vantage Inc. was perfect for our needs and no further modification was needed.

Python Programming Language provided us with its Natural Language Toolkit (NLTK) library which was crucial for natural language processing and also SciKit-Learn library and Keras for scientific analysis of data.

3.1.5.2 Economic Feasibility

Economic feasibility studies the costs and benefits associated with a project.

The project cost was feasible as all the APIs used were free and the software to implement the project was open source. The benefits that come with implementing this project far outweighs any cost that was incurred in terms of internet fees and money for printing the final documentation.

3.1.5.3 Schedule Feasibility

This checks to see if the solution can be implemented in the stipulated time.

The project was implemented in the stipulated time. The setbacks in terms of the lecturers strike meant making some minor adjustments in our schedule. But all the objectives were still achieved.

3.1.5.4 Operational Feasibility

The operational feasibility seeks to assess if the solution will work and if it will be possible to maintain it.

The interface of the solution follows all the standards that have been put in place by User Interface experts to ensure it is user friendly. This means that the intended users will have an easy time in learning to use it thus ensuring no resistance is faced.

The project is also well documented ensuring that future maintenance be it corrective or an enhancement of its features will be easy for whomever will be working on it.

3.1.6 Systems Requirements

3.1.6.1 *Functional Requirements*

The following are the functionalities required from the system:

1. User signup and login: This is done for security purposes and to restrict use of the app to only users that are in the database.
2. Training the naive Bayes classifier for sentiment classification. The naive Bayes classifier offers accurate classification of sentiments as this was key in ensuring the model had a high degree of accuracy.
3. Analyzing stock market data for accurate predictions. Time series and Long Short Term Memory Neural Networks were used to ensure a high degree of accuracy in the stock prediction that was be carried out.
4. Reporting statistics for stock market movement. Graphs will be used as they offer good visualization of results.
5. Historical twitter data, stock data and statistical storage: Twitter data will be filtered and stored for future access in CSV files and also stock data. The trained model will also be stored in h5 files.

3.1.6.2 *Non Functional Requirements*

1. User friendliness: The system is easy to use.
2. Reliability and availability: This two will depend on the Internet availability as tweets and stock data need an Internet connection to be fetched.
3. High response time: The trained classifiers is stored and reused later to ensure that retraining is not needed as the training takes a long time.
4. Interoperability: The system can be used in all major operating systems, which are Microsoft Windows and Linux Operating System platforms.

3.2 Systems Design

3.2.1 Introduction

This section covers the architecture, modules, components and data for the system to satisfy the specified requirements.

It will be divided into physical and logical design.

3.2.2 Logical Design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system.

3.2.2.1 Data Flow Diagram (DFD)

A data flow diagram is a graphical representation of the flow of data through an information system. It shows how information is input to and output from the system, the sources and destinations of that information, and where that information is stored.

In this project data flow diagram will be used to model the flow of data, relationships and storage of messages which will be used in training and retraining of the model.

3.2.2.1.1 Level 0 DFD/ Context Diagram

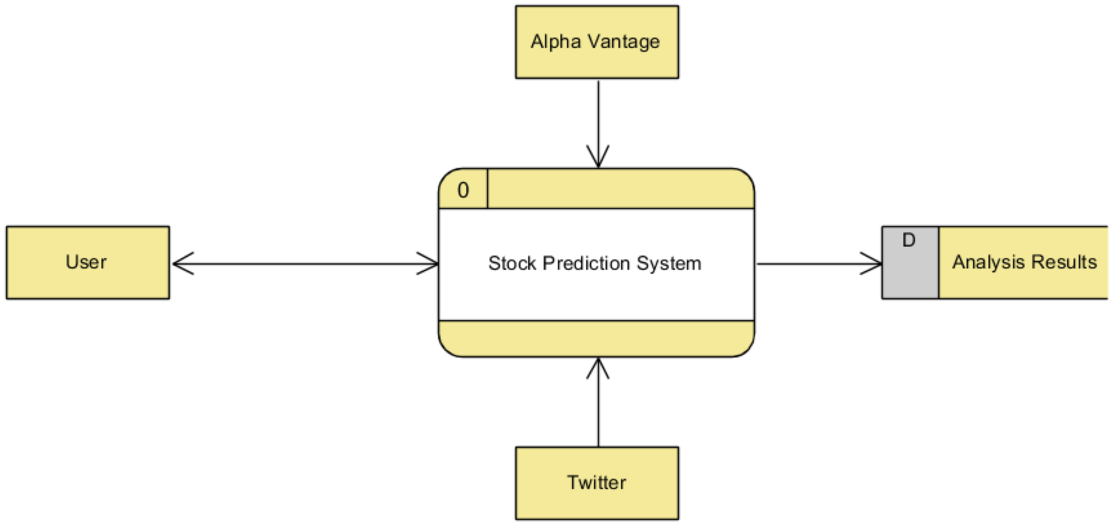


Figure 3. 1 Data flow diagram level 0 showing the relationship between the system user, twitter and alpha vantage systems as well as the storage of analyzed results

3.2.2.1.2 Level 1 DFD

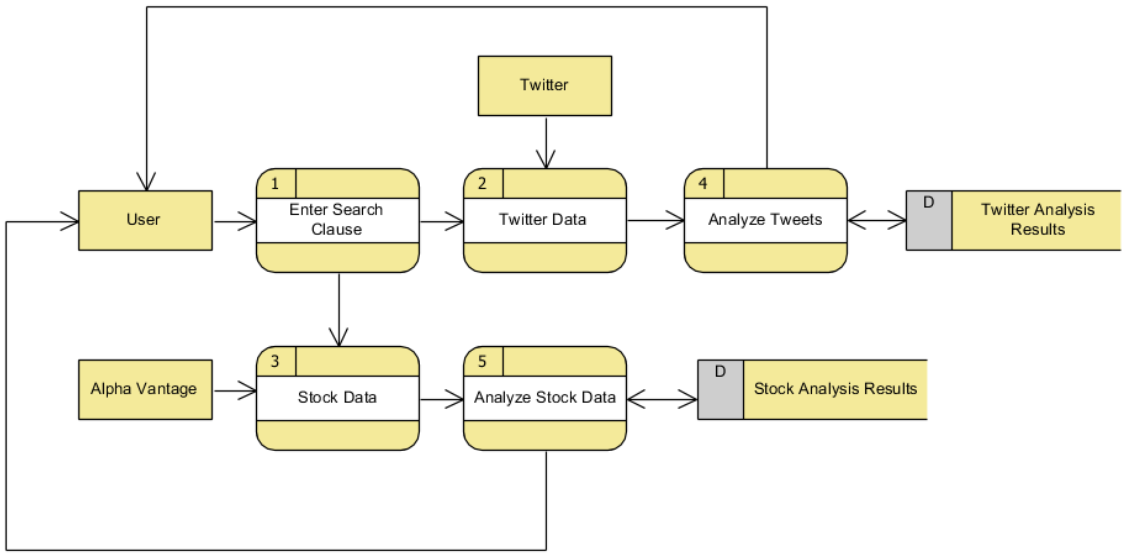


Figure 3. 2 Data flow diagram level 1 showing basic processes of the system and the data flow between the processes

3.2.2.1.3 Level 2 DFD

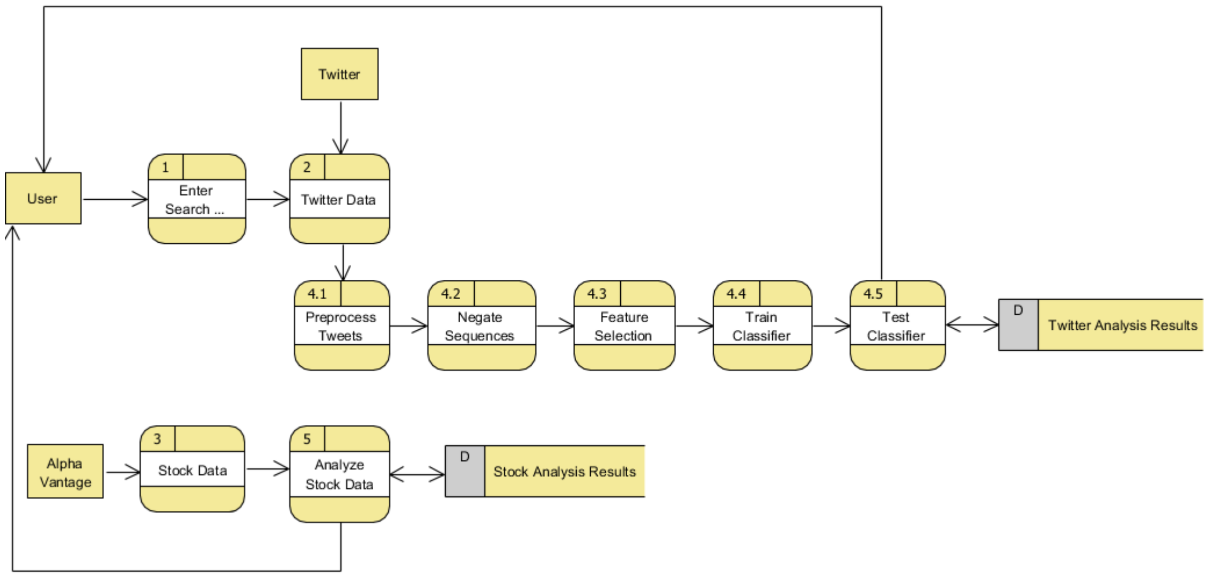


Figure 3. 3 Data flow diagram level 2 showing more detailed look at the sub-processes in process 4

3.2.2.2 Activity Diagram

Activity diagram shows the flow of activities in a system. It describes the sequence from one activity to another, and describes the parallel, branched and concurrent flow of the system.

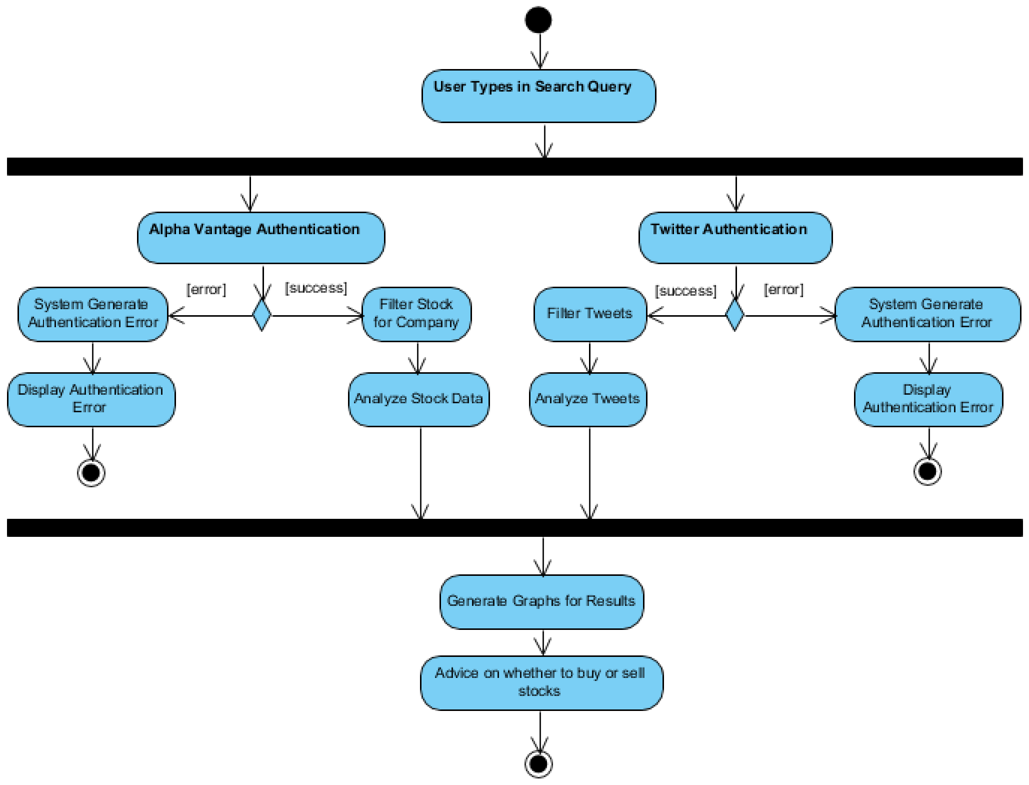

User Types in Search Query
Alpha Vantage Authentication
Twitter Authentication
[error]
[success]
System Generate Authentication Error
Filter Stock for Company
Filter Tweets
[success]
[error]
System Generate Authentication Error
Display Authentication Error
Analyze Stock Data
Analyze Tweets
Display Authentication Error
Generate Graphs for Results
Advice on whether to buy or sell stocks

Figure 3. 4 Activity diagram showing the flow of activities in the system

3.2.2.3 Use Case Diagram

Use case diagram is used to gather system requirements and actors in the system.

Actors define the role played by a user or any other system that interacts with the system being designed. For our case the actors are an investor using the system, twitter and alpha vantage system.

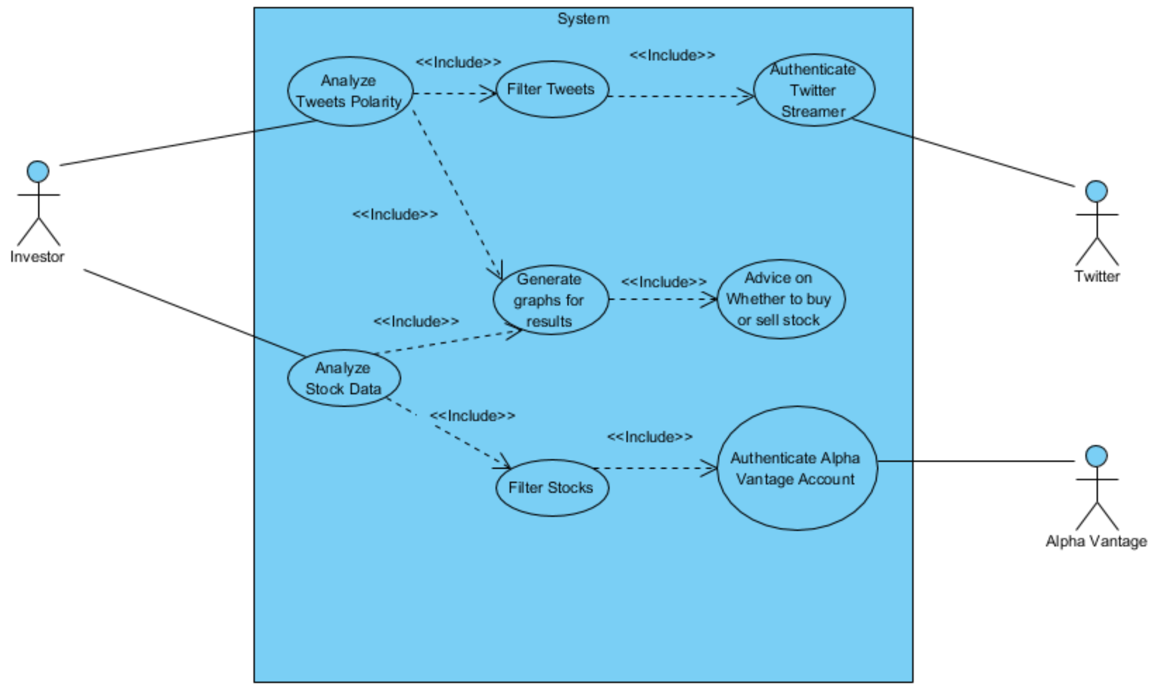
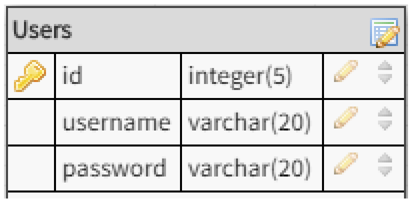


Figure 3. 5 Use case diagram showing the actors of the system and the various use cases

3.2.2.4 Entity Relationship Diagram (ERD)

The application will not need to store a lot of information in the database. Only the users table is need to store a users' details after signing up and the data will also be used to authenticate the user when they log in to the system.

The purpose of the user table is to accord the system with some level of security and also to limit access of the system to only the users saved in the system database.



Users		
id	integer(5)	
username	varchar(20)	
password	varchar(20)	

Figure 3. 6 Login table database design

3.2.3 Physical Design

The physical design relates to the actual input and output processes of the system. This is explained in terms of how data is input into a system, how it is verified/authenticated, how it is processed, and how it is displayed.

3.2.3.1 User Interface Design

User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions.

The UI is the platform through which users will be able to access the system, thus it will be necessary to make it as simple as possible to make it user friendly for all types of users.

A user can stream in the current tweets for the 2 stock tickers being studied, AAPL and AMZN using the Stream Tweets button. A user can also stream stock data from AAPL and AMZN using the Get Stock Data button. Streaming of these tweets will require internet access. When the data is being streamed, the button will turn from green to gray, and on completion, the button will revert back to the original green color.

There will be dropdown lists to choose the company whose data is to be analyzed and later on visualized as a graph. The first dropdown analyzes stock data only and the second one is for the hybrid model that incorporates sentiment polarity about a company as a feature.

The graphs will be displayed using the Matplotlib UI and they will be saved for comparison. The accuracy of the results will be computed using Root Mean Squared Error (RMSE). RMSE will be the reporting statistic used in comparing the 2 models.

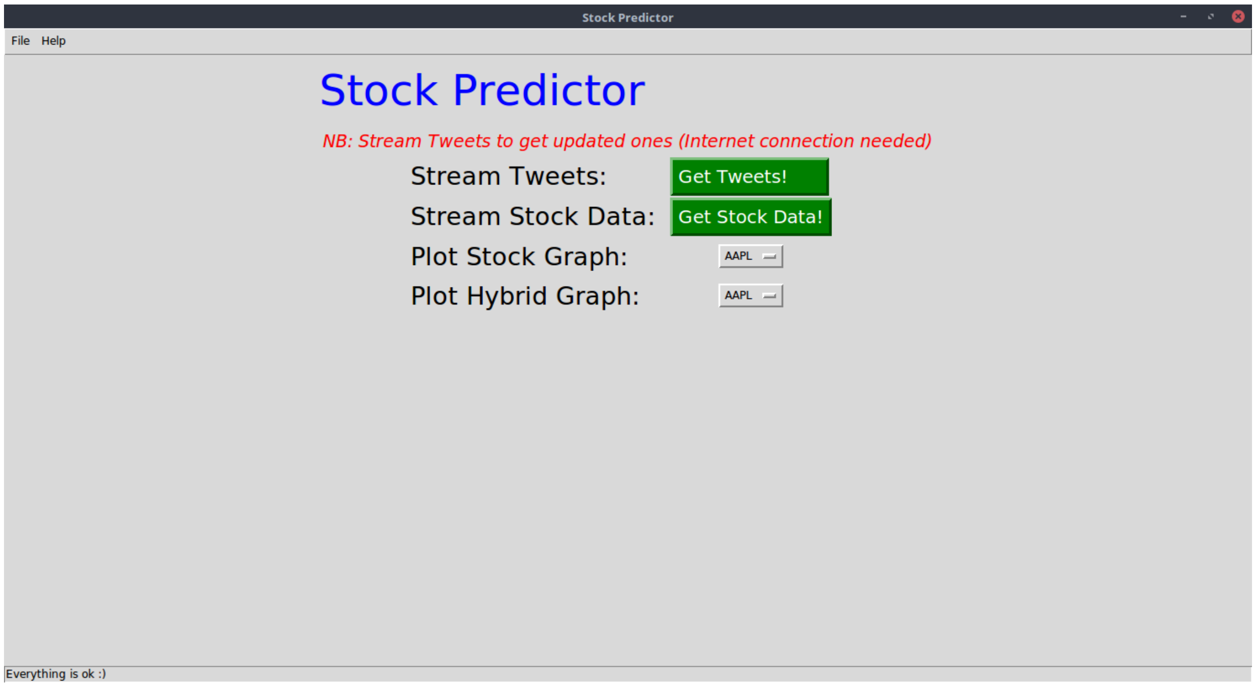


Figure 3. 7 User interface diagram

3.2.4 Systems Architecture

The system architecture is built on 3 key components which play an important role in the success of the system as a whole.

i. The **Interface** where the user interacts with the system
ii. The **Sentiment Classifier Component**. Classifies user sentiments that are later on used as a feature in the stock prediction component.
iii. The **Stock Prediction Component.** Analyzes stock data, help's in showing the performance of the hybrid model and the model that only uses stock data.

CHAPTER 4: CODING AND IMPLEMENTATION

4.1 Introduction

This section elaborates on how the system is set to achieve the set objectives.

4.2 System Implementation

4.2.1 Software Tools Used

1. Linux Mint Operating System: This was the Operating System upon which the development of the system took place.

2. Visual Studio Code: This is the text editor that was used in writing of the python scripts.

4.2.2 Libraries Used

1. Keras Deep Learning Library: Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.
2. Scikit-learn (sklearn): A simple and efficient Python tool for data mining and data analysis Built on NumPy, SciPy, and matplotlib
3. Pandas: An open source library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.
4. NumPy: Fundamental package for scientific computing with Python. It contains among other things a powerful N-dimensional array object
5. Matplotlib: A Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
6. Tweepy: An easy-to-use Python library for accessing the Twitter API.
7. Textblob: A Python library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.
8. Nltk: Natural Language ToolKit is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning and wrappers for industrial-strength NLP libraries.

9. Tkinter: Tkinter is Python's de-facto standard GUI (Graphical User Interface) package

4.2.3 Stock Prediction

The following is the step by step procedure followed during the stock prediction process:

4.2.3.1 Importing Libraries

The following modules were obtained from Keras Deep Learning Library:

1. Sequential: Used for the creation of a linear stack of layers. A Sequential model is created by passing a list of layer instances to the constructor. They are:

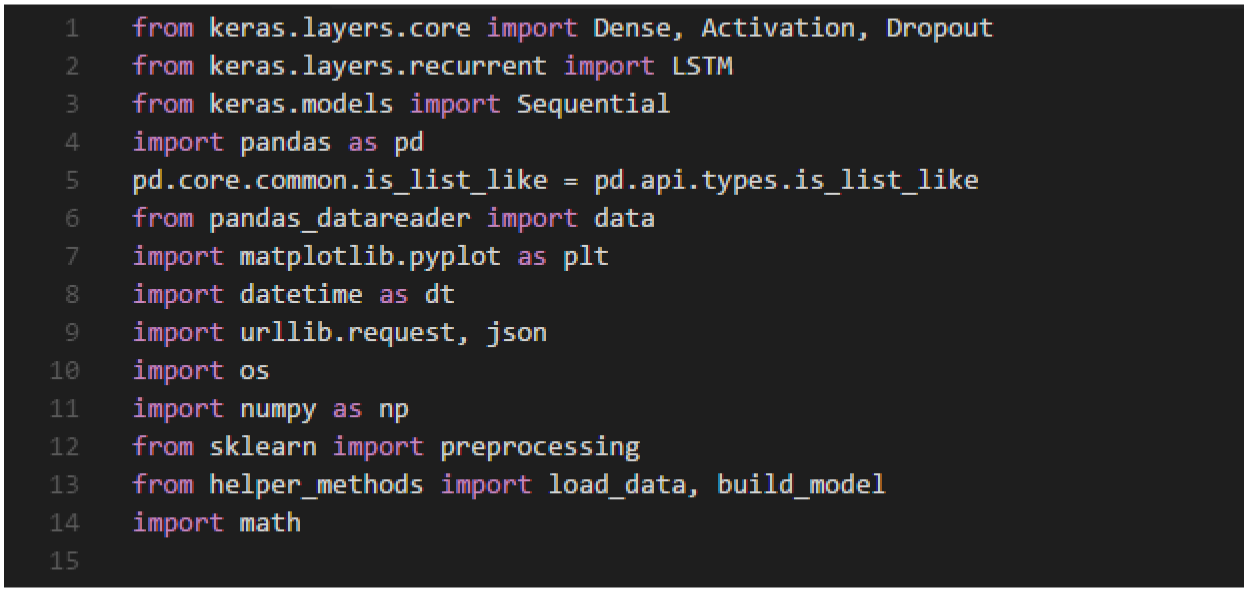
   a) Dense: A layer where each unit or neuron is connected to each neuron in the next layer

   b) Activation: Specifies the activation function to be applied to an output.

2. Dropout: A regularization technique, which aims to reduce the complexity of the model with the goal to prevent overfitting.

3. LSTM: Used to build a Long Short Term Memory network.

4. Preprocessing library from scikit-learn used for feature extraction and normalization.



```
from keras.layers.core import Dense, Activation, Dropout
from keras.layers.recurrent import LSTM
from keras.models import Sequential
import pandas as pd
pd.core.common.is_list_like = pd.api.types.is_list_like
from pandas_datareader import data
import matplotlib.pyplot as plt
import datetime as dt
import urllib.request, json
import os
import numpy as np
from sklearn import preprocessing
from helper_methods import load_data, build_model
import math
```

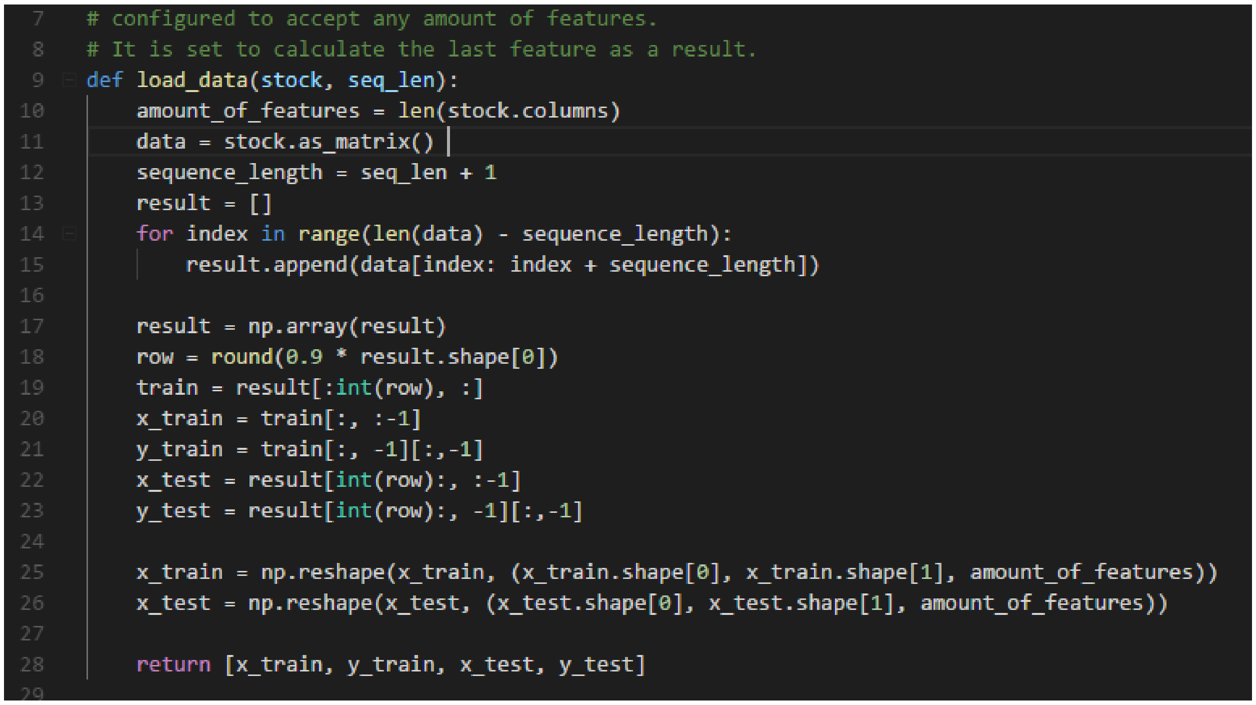
Figure 4. 1 Importing Libraries

4.2.3.2 Loading Data and Preprocessing

The stock data is obtained from AlphaVantage API and the data is queried by a company's stock Ticker. An API key is required for data access to be granted.

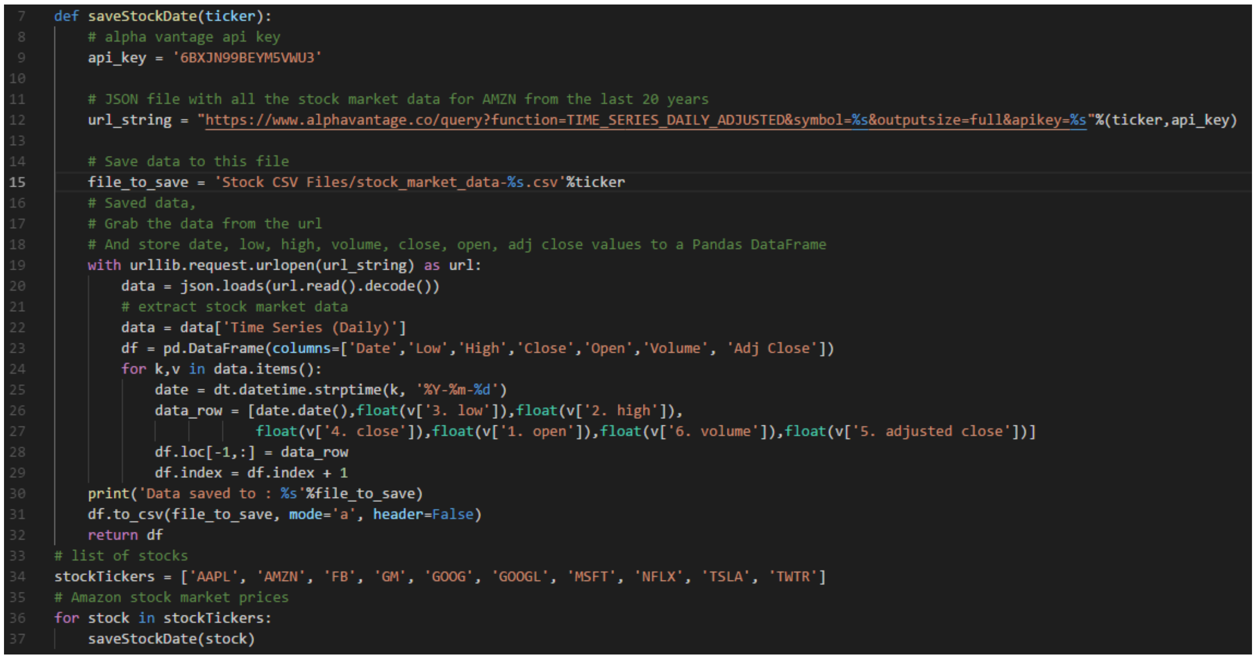
The date, low, high, volume, close, open, adjusted close values from the API are first stored in a Pandas DataFrame. These are the features that are to be analysed. The data is then saved to a csv file, named by the company's stock ticker awaiting further processing.

The data is sorted by date and then normalized by use of the MinMaxScaler a function from scikit learn library. A custom helper function load_data() is used to split the data into training and testing data sets.



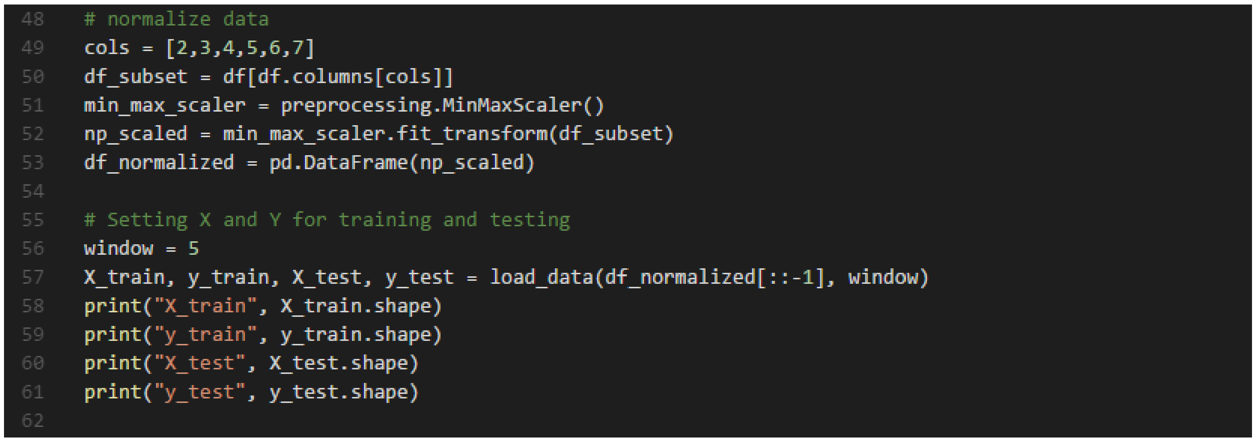
```python
# configured to accept any amount of features.
# It is set to calculate the last feature as a result.
def load_data(stock, seq_len):
    amount_of_features = len(stock.columns)
    data = stock.as_matrix()
    sequence_length = seq_len + 1
    result = []
    for index in range(len(data) - sequence_length):
        result.append(data[index: index + sequence_length])

    result = np.array(result)
    row = round(0.9 * result.shape[0])
    train = result[:int(row), :]
    x_train = train[:, :-1]
    y_train = train[:, -1][:,-1]
    x_test = result[int(row):, :-1]
    y_test = result[int(row):, -1][:,-1]

    x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], amount_of_features))
    x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], amount_of_features))

    return [x_train, y_train, x_test, y_test]
```

Figure 4. 2 Load data helper function



```python
def saveStockDate(ticker):
    # alpha vantage api key
    api_key = '6BXJN99BEYM5VWU3'

    # JSON file with all the stock market data for AMZN from the last 20 years
    url_string = "https://www.alphavantage.co/query?function=TIME_SERIES_DAILY_ADJUSTED&symbol=%s&outputsize=full&apikey=%s"%(ticker,api_key)

    # Save data to this file
    file_to_save = 'Stock CSV Files/stock_market_data-%s.csv'%ticker
    # Saved data,
    # Grab the data from the url
    # And store date, low, high, volume, close, open, adj close values to a Pandas DataFrame
    with urllib.request.urlopen(url_string) as url:
        data = json.loads(url.read().decode())
        # extract stock market data
        data = data['Time Series (Daily)']
        df = pd.DataFrame(columns=['Date','Low','High','Close','Open','Volume', 'Adj Close'])
        for k,v in data.items():
            date = dt.datetime.strptime(k, '%Y-%m-%d')
            data_row = [date.date(),float(v['3. low']),float(v['2. high']),
                        float(v['4. close']),float(v['1. open']),float(v['6. volume']),float(v['5. adjusted close'])]
            df.loc[-1,:] = data_row
            df.index = df.index + 1
    print('Data saved to : %s'%file_to_save)
    df.to_csv(file_to_save, mode='a', header=False)
    return df
# list of stocks
stockTickers = ['AAPL', 'AMZN', 'FB', 'GM', 'GOOG', 'GOOGL', 'MSFT', 'NFLX', 'TSLA', 'TWTR']
# Amazon stock market prices
for stock in stockTickers:
    saveStockDate(stock)
```

Figure 4. 3 Loading data and saving it to csv file



```python
# normalize data
cols = [2,3,4,5,6,7]
df_subset = df[df.columns[cols]]
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(df_subset)
df_normalized = pd.DataFrame(np_scaled)

# Setting X and Y for training and testing
window = 5
X_train, y_train, X_test, y_test = load_data(df_normalized[::-1], window)
print("X_train", X_train.shape)
print("y_train", y_train.shape)
print("X_test", X_test.shape)
print("y_test", y_test.shape)

```

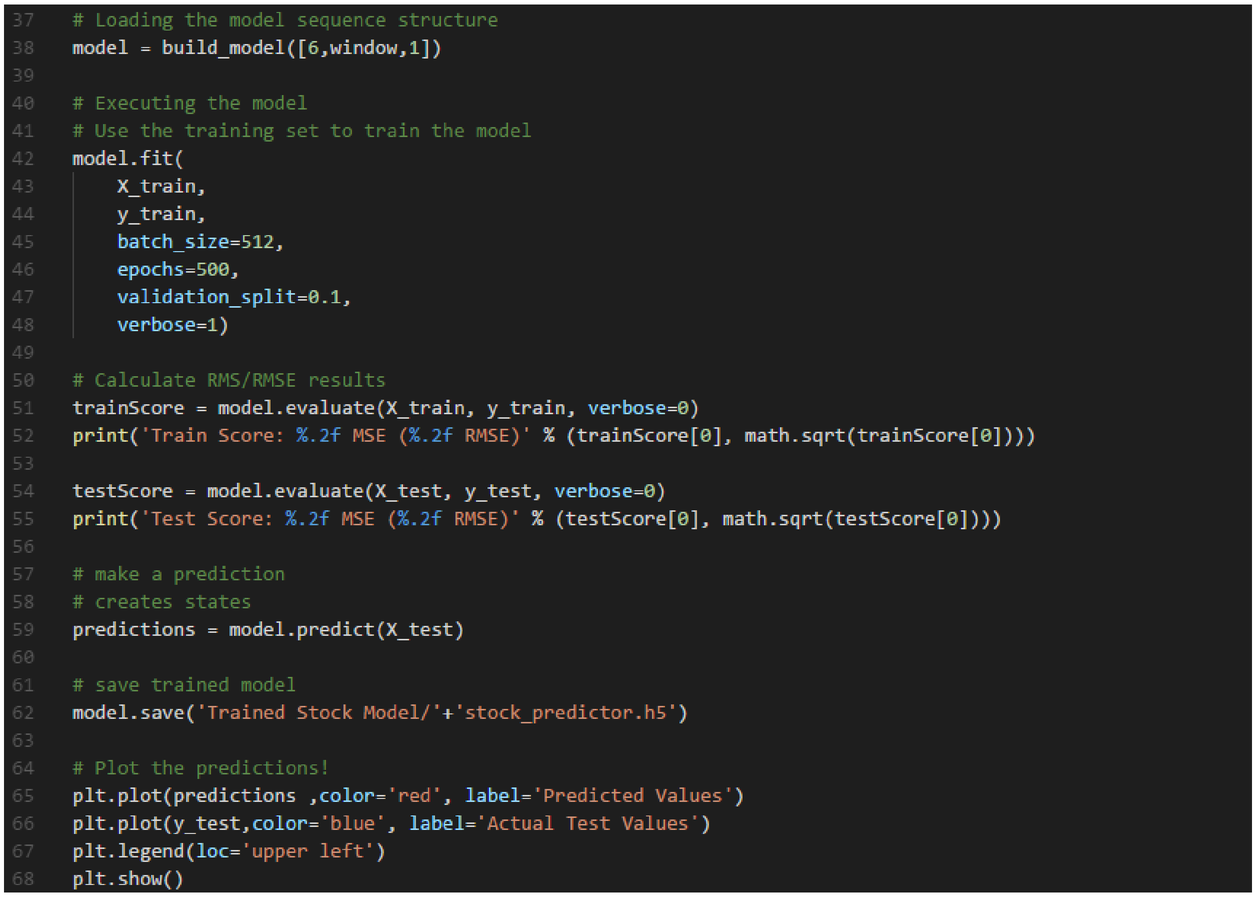
Figure 4. 4 Stock data preprocessing

4.2.3.3 Building the Model, Training and Testing

A Long Short Term Memory model is then built by use of build_model() helper function. The model is then trained by use of the training data sets. The accuracy of the model is then tested by calculating Mean Squared Error and RMSE on both the training and test data sets.

Finally, predictions are made on the test set and a graph is plotted. The graph shows the actual movement of stock closing values against the movement of predicted stock closing values.

The model is then saved for making future predictions.

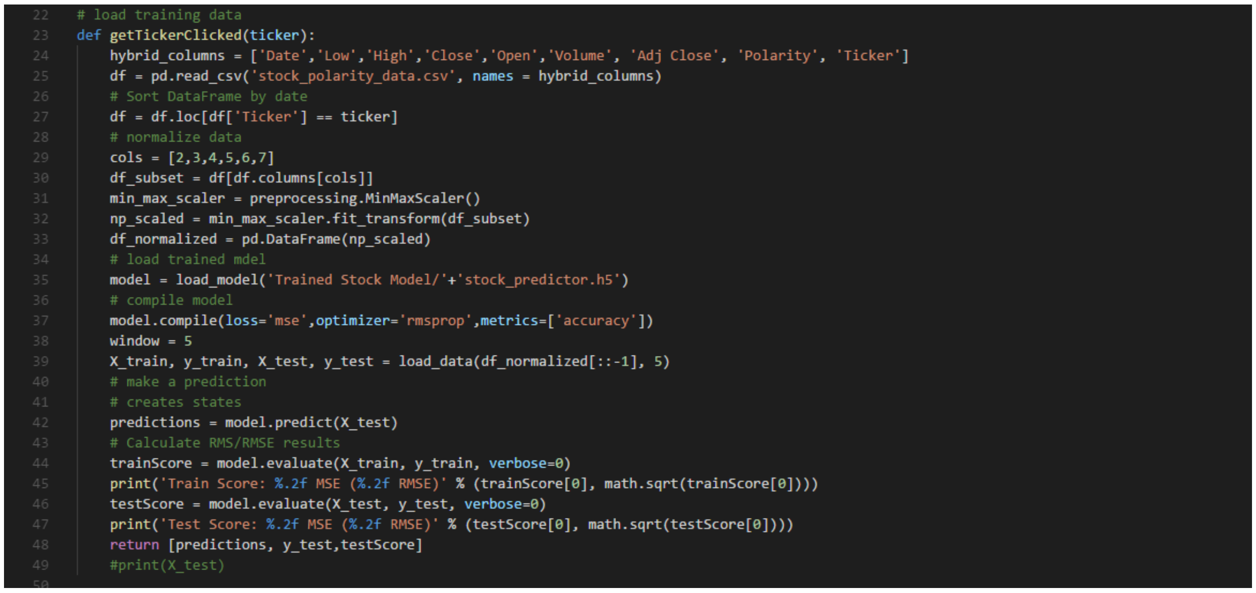


```python
# Loading the model sequence structure
model = build_model([6,window,1])

# Executing the model
# Use the training set to train the model
model.fit(
    X_train,
    y_train,
    batch_size=512,
    epochs=500,
    validation_split=0.1,
    verbose=1)

# Calculate RMS/RMSE results
trainScore = model.evaluate(X_train, y_train, verbose=0)
print('Train Score: %.2f MSE (%.2f RMSE)' % (trainScore[0], math.sqrt(trainScore[0])))

testScore = model.evaluate(X_test, y_test, verbose=0)
print('Test Score: %.2f MSE (%.2f RMSE)' % (testScore[0], math.sqrt(testScore[0])))

# make a prediction
# creates states
predictions = model.predict(X_test)

# save trained model
model.save('Trained Stock Model/'+'stock_predictor.h5')

# Plot the predictions!
plt.plot(predictions ,color='red', label='Predicted Values')
plt.plot(y_test,color='blue', label='Actual Test Values')
plt.legend(loc='upper left')
plt.show()
```

Figure 4. 5 Building the model, training and testing

4.2.3.4 Making Future Predictions

The saved Keras model is loaded using the load function and then trained.



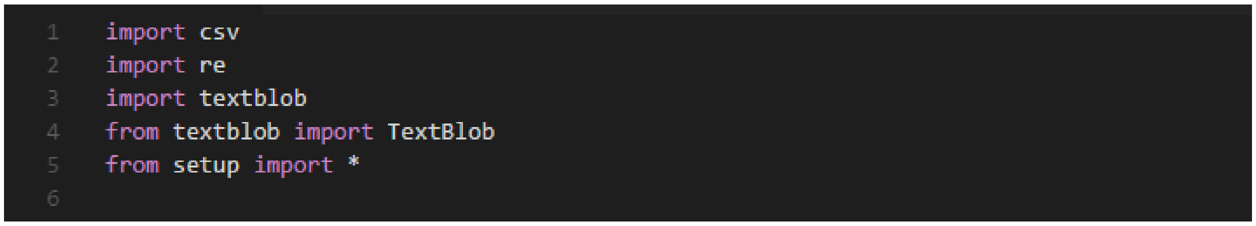
```python
# load training data
def getTickerClicked(ticker):
    hybrid_columns = ['Date','Low','High','Close','Open','Volume', 'Adj Close', 'Polarity', 'Ticker']
    df = pd.read_csv('stock_polarity_data.csv', names = hybrid_columns)
    # Sort DataFrame by date
    df = df.loc[df['Ticker'] == ticker]
    # normalize data
    cols = [2,3,4,5,6,7]
    df_subset = df[df.columns[cols]]
    min_max_scaler = preprocessing.MinMaxScaler()
    np_scaled = min_max_scaler.fit_transform(df_subset)
    df_normalized = pd.DataFrame(np_scaled)
    # load trained mdel
    model = load_model('Trained Stock Model/'+'stock_predictor.h5')
    # compile model
    model.compile(loss='mse',optimizer='rmsprop',metrics=['accuracy'])
    window = 5
    X_train, y_train, X_test, y_test = load_data(df_normalized[::-1], 5)
    # make a prediction
    # creates states
    predictions = model.predict(X_test)
    # Calculate RMS/RMSE results
    trainScore = model.evaluate(X_train, y_train, verbose=0)
    print('Train Score: %.2f MSE (%.2f RMSE)' % (trainScore[0], math.sqrt(trainScore[0])))
    testScore = model.evaluate(X_test, y_test, verbose=0)
    print('Test Score: %.2f MSE (%.2f RMSE)' % (testScore[0], math.sqrt(testScore[0])))
    return [predictions, y_test,testScore]
    #print(X_test)
```

Figure 4. 6 Making Future Predictions

4.2.4 Sentiment Analysis

4.2.4.1 Importing Libraries

The following is the code snippet of the libraries used.



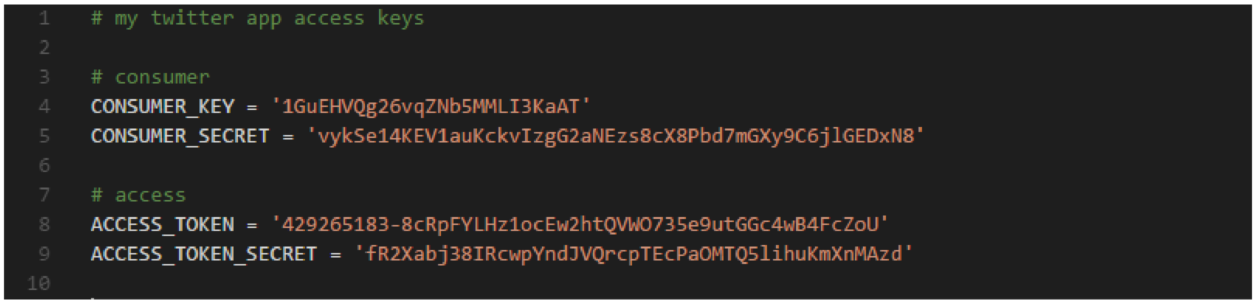
```python
import csv
import re
import textblob
from textblob import TextBlob
from setup import *

```

Figure 4. 7 Importing Libraries.

4.2.4.2 Twitter API Setup and Tweets Filtering

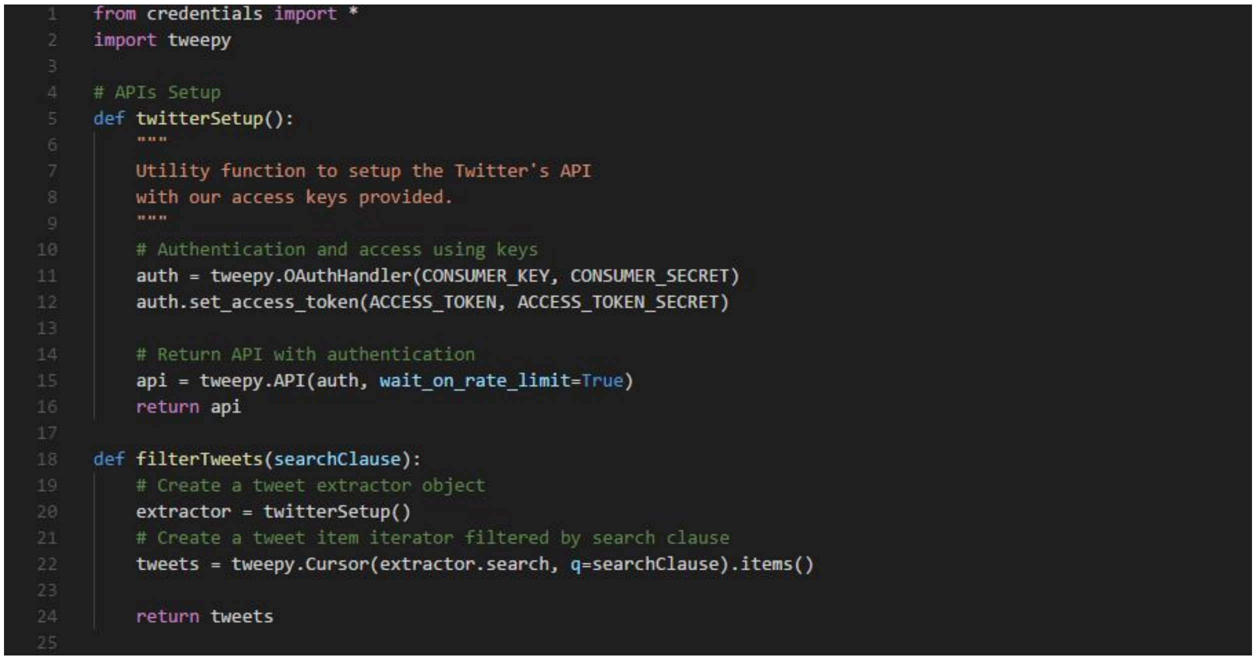
Tweepy library is used to access the twitter API due to the simplicity it provides.

The tweets are then filtered by a company's stock ticker.



```python
# my twitter app access keys

# consumer
CONSUMER_KEY = '1GuEHVQg26vqZNb5MMLI3KaAT'
CONSUMER_SECRET = 'vykSe14KEV1auKckvIzgG2aNEzs8cX8Pbd7mGXy9C6jlGEDxN8'

# access
ACCESS_TOKEN = '429265183-8cRpFYLHz1ocEw2htQVWO735e9utGGc4wB4FcZoU'
ACCESS_TOKEN_SECRET = 'fR2Xabj38IRcwpYndJVQrcpTEcPaOMTQ5lihuKmXnMAzd'

```

Figure 4. 8 Credentials file



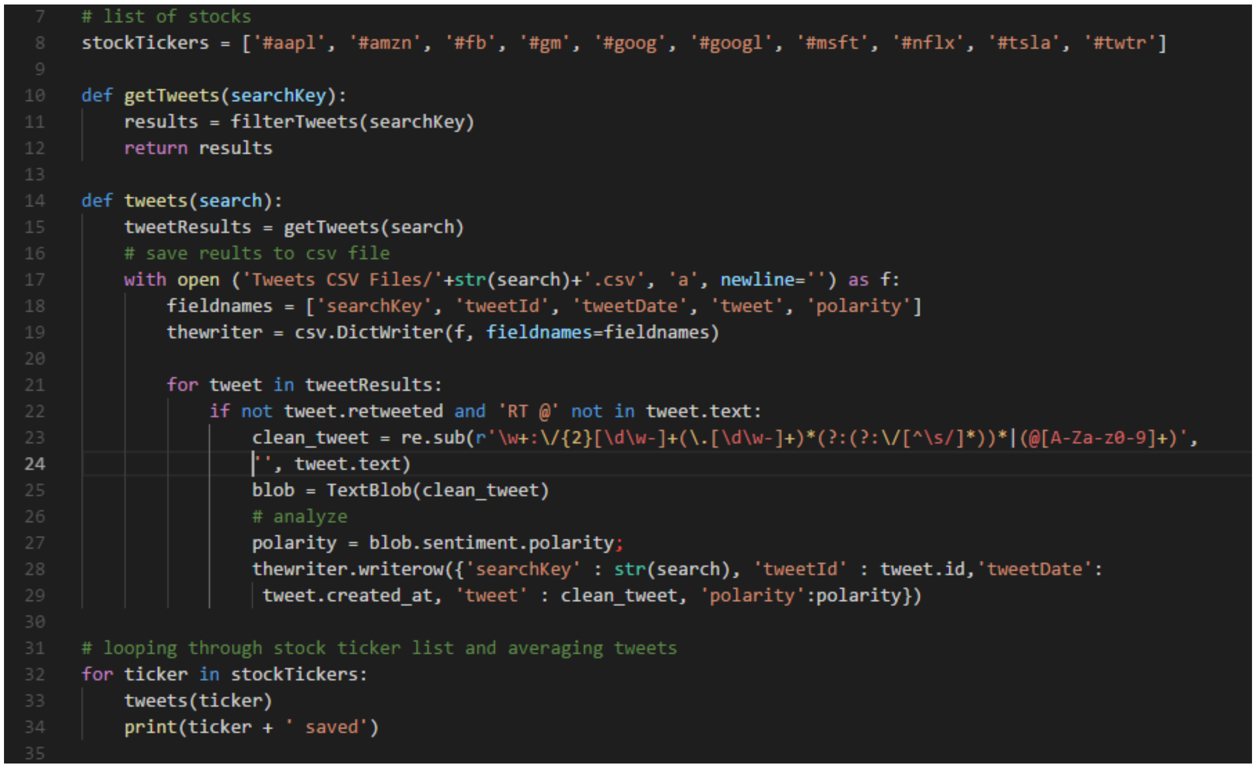
```python
from credentials import *
import tweepy

# APIs Setup
def twitterSetup():
    """
    Utility function to setup the Twitter's API
    with our access keys provided.
    """
    # Authentication and access using keys
    auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
    auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)

    # Return API with authentication
    api = tweepy.API(auth, wait_on_rate_limit=True)
    return api

def filterTweets(searchClause):
    # Create a tweet extractor object
    extractor = twitterSetup()
    # Create a tweet item iterator filtered by search clause
    tweets = tweepy.Cursor(extractor.search, q=searchClause).items()

    return tweets

```

Figure 4. 9 Twitter setup and API filtering

4.2.4.3 Tweet Polarity

Tweets are first preprocessed to remove links and @ symbols as they don't add any value to tweets.

TextBlob library is used to assign polarities to tweets. It has been implemented to use Bernoulli Naive Bayes Model for sentiment analysis classification. It also handles negations thus its performance is better than just using Naïve Bayes Model. The annotated corpus used is from the Natural Language ToolKit (NLTK).

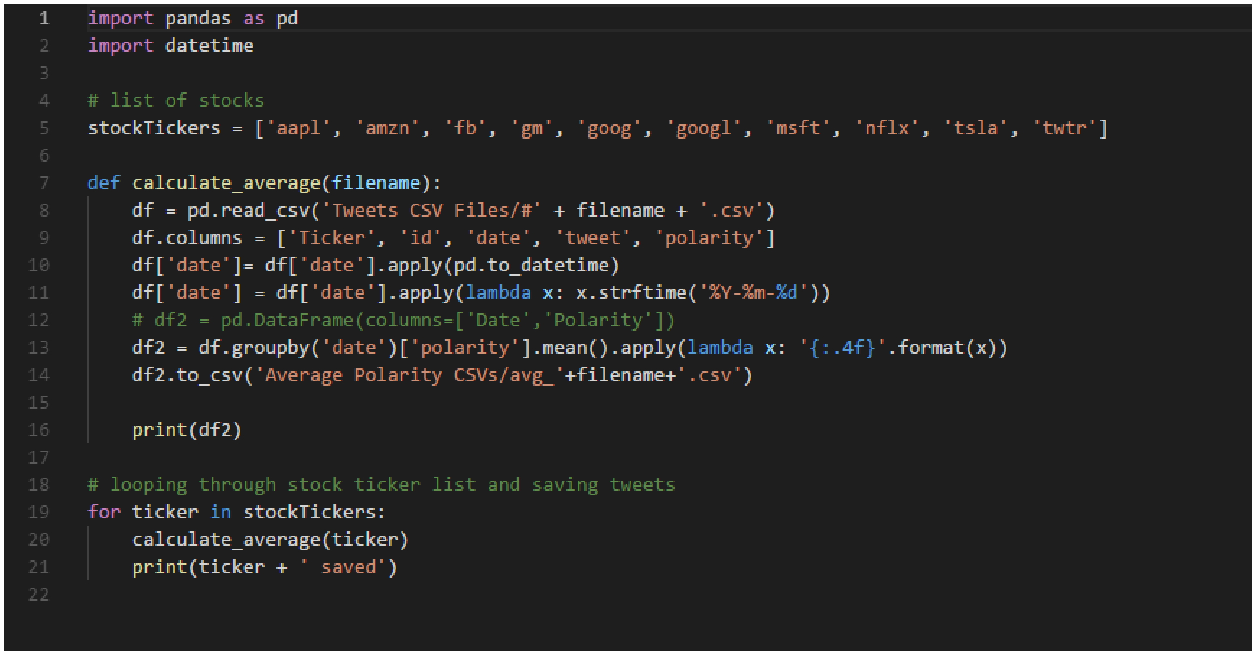


```python
# list of stocks
stockTickers = ['#aapl', '#amzn', '#fb', '#gm', '#goog', '#googl', '#msft', '#nflx', '#tsla', '#twtr']

def getTweets(searchKey):
    results = filterTweets(searchKey)
    return results

def tweets(search):
    tweetResults = getTweets(search)
    # save reults to csv file
    with open ('Tweets CSV Files/'+str(search)+'.csv', 'a', newline='') as f:
        fieldnames = ['searchKey', 'tweetId', 'tweetDate', 'tweet', 'polarity']
        thewriter = csv.DictWriter(f, fieldnames=fieldnames)

        for tweet in tweetResults:
            if not tweet.retweeted and 'RT @' not in tweet.text:
                clean_tweet = re.sub(r'\w+:\/{2}[\d\w-]+(\.[\d\w-]+)*(?:(?:\/[^\s/]*))*|(@[A-Za-z0-9]+)',
                '', tweet.text)
                blob = TextBlob(clean_tweet)
                # analyze
                polarity = blob.sentiment.polarity;
                thewriter.writerow({'searchKey' : str(search), 'tweetId' : tweet.id,'tweetDate':
                 tweet.created_at, 'tweet' : clean_tweet, 'polarity':polarity})

# looping through stock ticker list and averaging tweets
for ticker in stockTickers:
    tweets(ticker)
    print(ticker + ' saved')
```

Figure 4. 10 Tweet polarity classifier

4.2.4.4 Calculating Average Daily Polarity

The average daily polarity is calculated to enable us know the general polarity of the day.



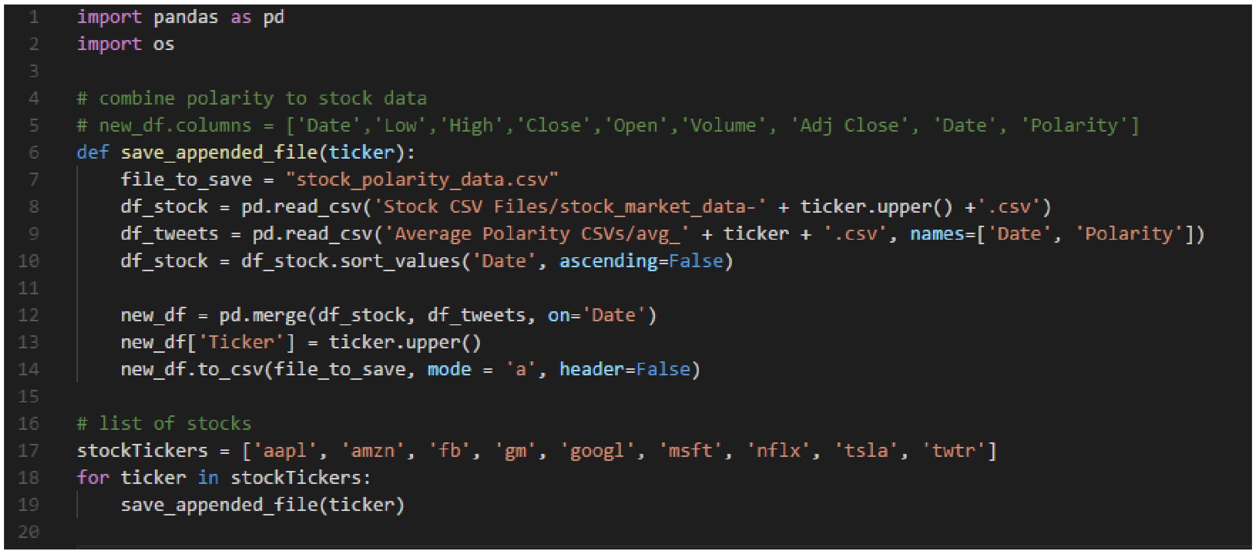
```python
import pandas as pd
import datetime

# list of stocks
stockTickers = ['aapl', 'amzn', 'fb', 'gm', 'goog', 'googl', 'msft', 'nflx', 'tsla', 'twtr']

def calculate_average(filename):
    df = pd.read_csv('Tweets CSV Files/#' + filename + '.csv')
    df.columns = ['Ticker', 'id', 'date', 'tweet', 'polarity']
    df['date']= df['date'].apply(pd.to_datetime)
    df['date'] = df['date'].apply(lambda x: x.strftime('%Y-%m-%d'))
    # df2 = pd.DataFrame(columns=['Date','Polarity'])
    df2 = df.groupby('date')['polarity'].mean().apply(lambda x: '{:.4f}'.format(x))
    df2.to_csv('Average Polarity CSVs/avg_'+filename+'.csv')

    print(df2)

# looping through stock ticker list and saving tweets
for ticker in stockTickers:
    calculate_average(ticker)
    print(ticker + ' saved')
```

Figure 4. 11 Calculating average daily polarity

4.2.5 Stock Prediction Incorporating Sentiments

The polarity value is added as a feature to the already existing feature vector.

The sentiment data is appended to the csv files of stock. They are appended based on Date and also based on a company's ticker. This ensures the sentiment data is of the same date as the stock data and that it belongs to the given stock company.

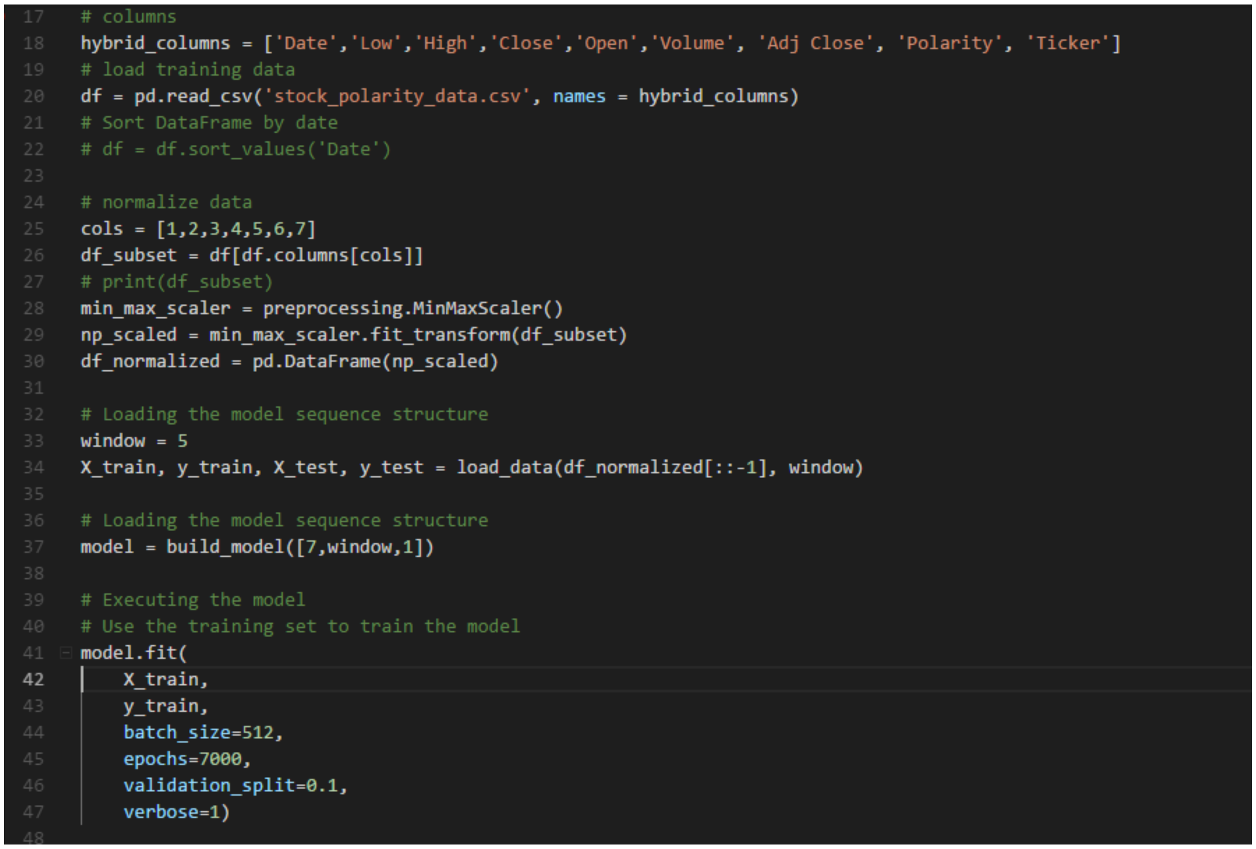


```python
import pandas as pd
import os

# combine polarity to stock data
# new_df.columns = ['Date','Low','High','Close','Open','Volume', 'Adj Close', 'Date', 'Polarity']
def save_appended_file(ticker):
    file_to_save = "stock_polarity_data.csv"
    df_stock = pd.read_csv('Stock CSV Files/stock_market_data-' + ticker.upper() +'.csv')
    df_tweets = pd.read_csv('Average Polarity CSVs/avg_' + ticker + '.csv', names=['Date', 'Polarity'])
    df_stock = df_stock.sort_values('Date', ascending=False)

    new_df = pd.merge(df_stock, df_tweets, on='Date')
    new_df['Ticker'] = ticker.upper()
    new_df.to_csv(file_to_save, mode = 'a', header=False)

# list of stocks
stockTickers = ['aapl', 'amzn', 'fb', 'gm', 'googl', 'msft', 'nflx', 'tsla', 'twtr']
for ticker in stockTickers:
    save_appended_file(ticker)

```

Figure 4. 12 Appending polarity to stock data

The hybrid model is also trained using the Long Short Term Memory (LSTM) neural network. The only difference is the addition of the sentiment polarity data related to a company for that particular day.

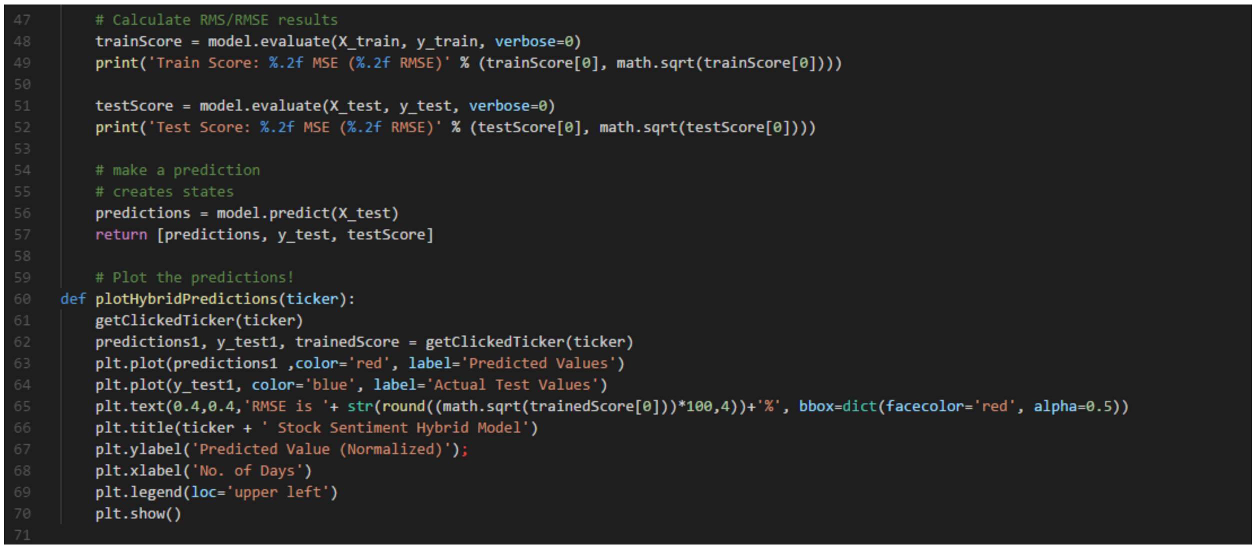
The training is done on the same neural network, the difference being only the feature vectors used. This is the case so that only the addition of twitter sentiment can be studied.



```python
# columns
hybrid_columns = ['Date','Low','High','Close','Open','Volume', 'Adj Close', 'Polarity', 'Ticker']
# load training data
df = pd.read_csv('stock_polarity_data.csv', names = hybrid_columns)
# Sort DataFrame by date
# df = df.sort_values('Date')

# normalize data
cols = [1,2,3,4,5,6,7]
df_subset = df[df.columns[cols]]
# print(df_subset)
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(df_subset)
df_normalized = pd.DataFrame(np_scaled)

# Loading the model sequence structure
window = 5
X_train, y_train, X_test, y_test = load_data(df_normalized[::-1], window)

# Loading the model sequence structure
model = build_model([7,window,1])

# Executing the model
# Use the training set to train the model
model.fit(
    X_train,
    y_train,
    batch_size=512,
    epochs=7000,
    validation_split=0.1,
    verbose=1)

```

Figure 4. 13 Hybrid Stock Prediction using Sentiments (a)

The Root Mean Squared Error (RMSE) is calculated to get the score of the training set as well as the test set. The results are then saved to a model so as to speed up future predictions by not having to retrain the model.



```python
    # Calculate RMS/RMSE results
    trainScore = model.evaluate(X_train, y_train, verbose=0)
    print('Train Score: %.2f MSE (%.2f RMSE)' % (trainScore[0], math.sqrt(trainScore[0])))

    testScore = model.evaluate(X_test, y_test, verbose=0)
    print('Test Score: %.2f MSE (%.2f RMSE)' % (testScore[0], math.sqrt(testScore[0])))

    # make a prediction
    # creates states
    predictions = model.predict(X_test)
    return [predictions, y_test, testScore]

    # Plot the predictions!
def plotHybridPredictions(ticker):
    getClickedTicker(ticker)
    predictions1, y_test1, trainedScore = getClickedTicker(ticker)
    plt.plot(predictions1 ,color='red', label='Predicted Values')
    plt.plot(y_test1, color='blue', label='Actual Test Values')
    plt.text(0.4,0.4,'RMSE is '+ str(round((math.sqrt(trainedScore[0]))*100,4))+'%', bbox=dict(facecolor='red', alpha=0.5))
    plt.title(ticker + ' Stock Sentiment Hybrid Model')
    plt.ylabel('Predicted Value (Normalized)');
    plt.xlabel('No. of Days')
    plt.legend(loc='upper left')
    plt.show()

```

Figure 4. 14 Hybrid Stock Prediction using Sentiments (b)

4.2.6 User Interface Implementation

Tkinter library is used to make a desktop application with the following UI.

4.2.6.1 User signup and login

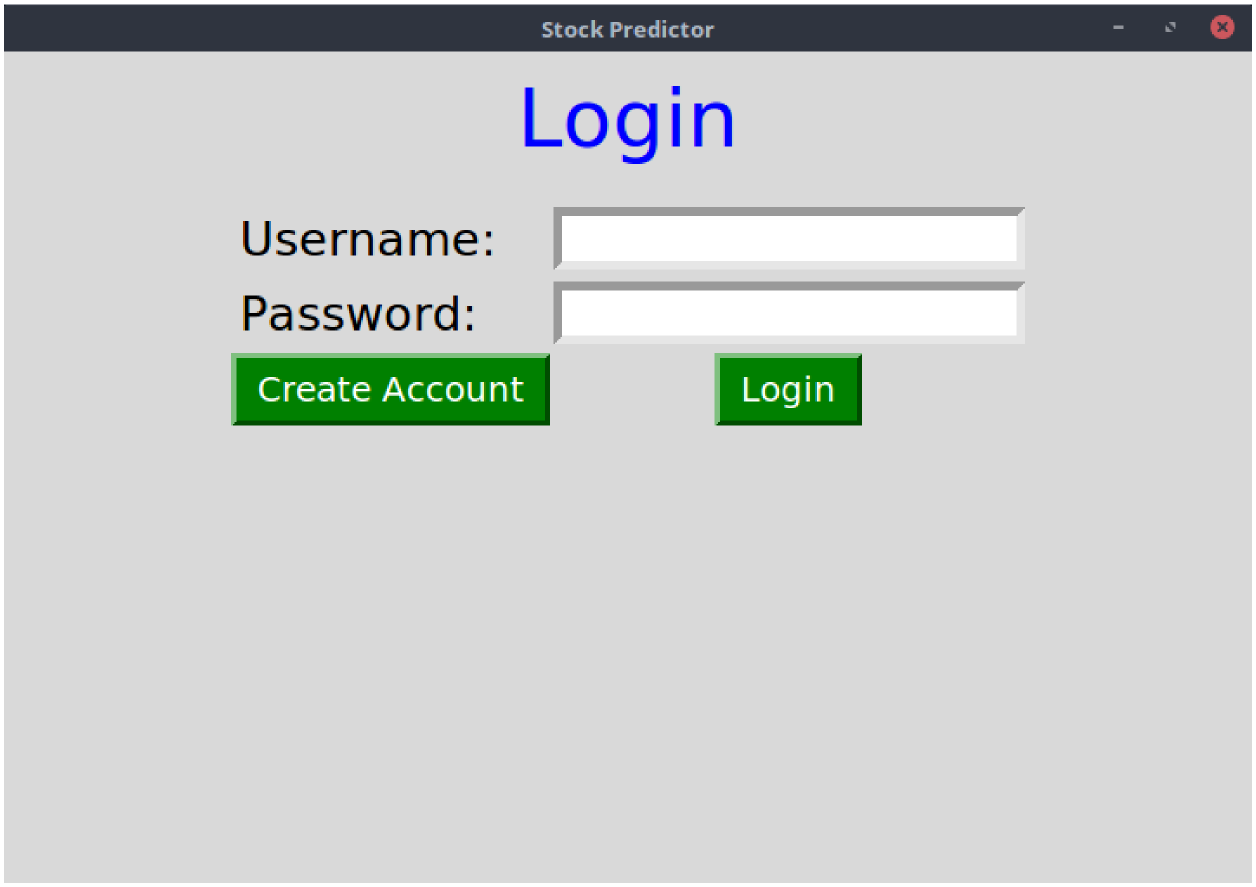
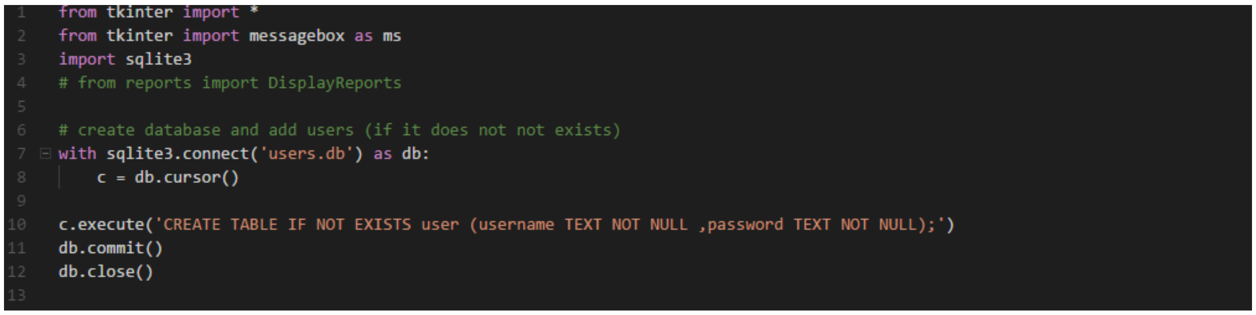


Figure 4. 15 User Login interface

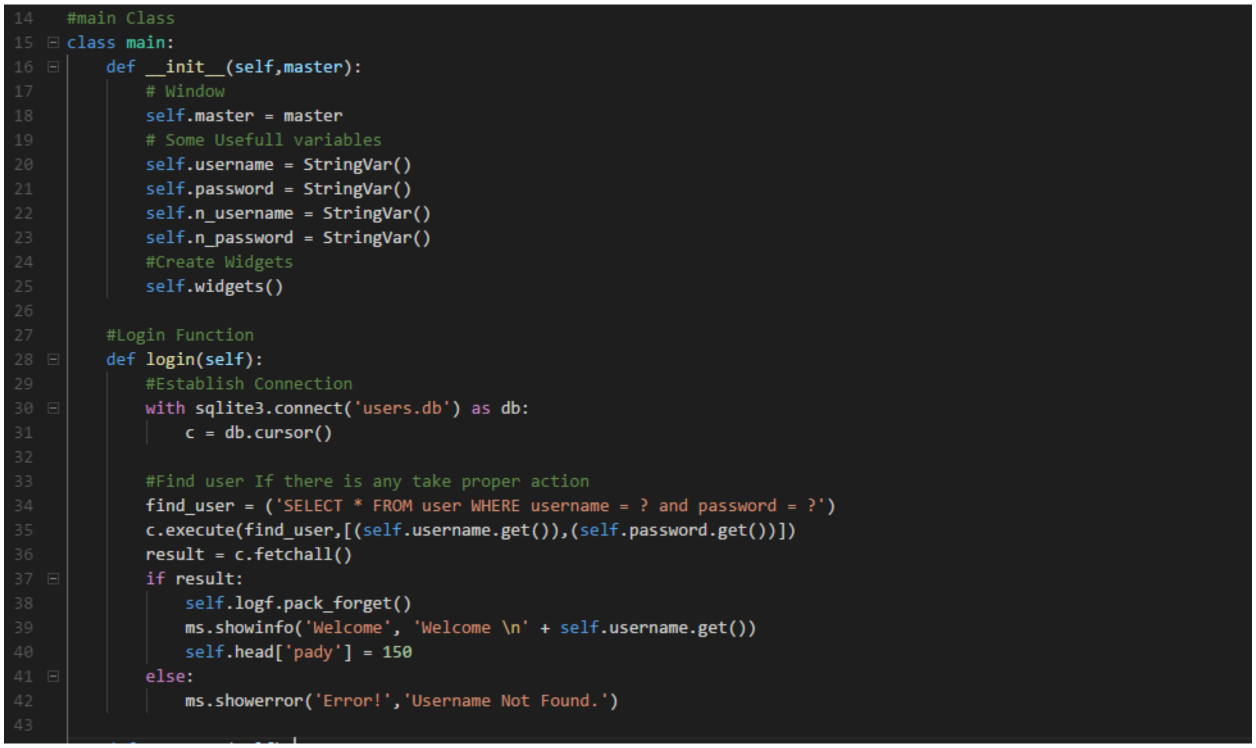
A new user signs up before being given access to the app and an existing user can log in to the system. After successful login the user is redirected to the stock prediction system.

SQLite database is used due to the simplicity it provides and also because we only have one table thus there are no complex relationships that would necessitate the use of a more sophisticated database.



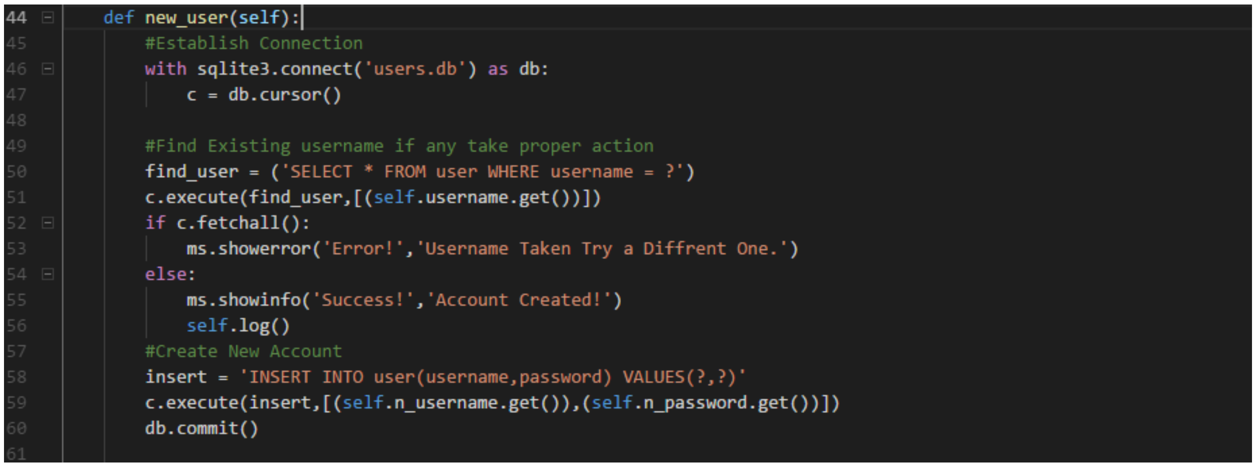
```python
from tkinter import *
from tkinter import messagebox as ms
import sqlite3
# from reports import DisplayReports

# create database and add users (if it does not not exists)
with sqlite3.connect('users.db') as db:
    c = db.cursor()

c.execute('CREATE TABLE IF NOT EXISTS user (username TEXT NOT NULL ,password TEXT NOT NULL);')
db.commit()
db.close()
```

Figure 4. 16 Creation of a new database if it does not exist



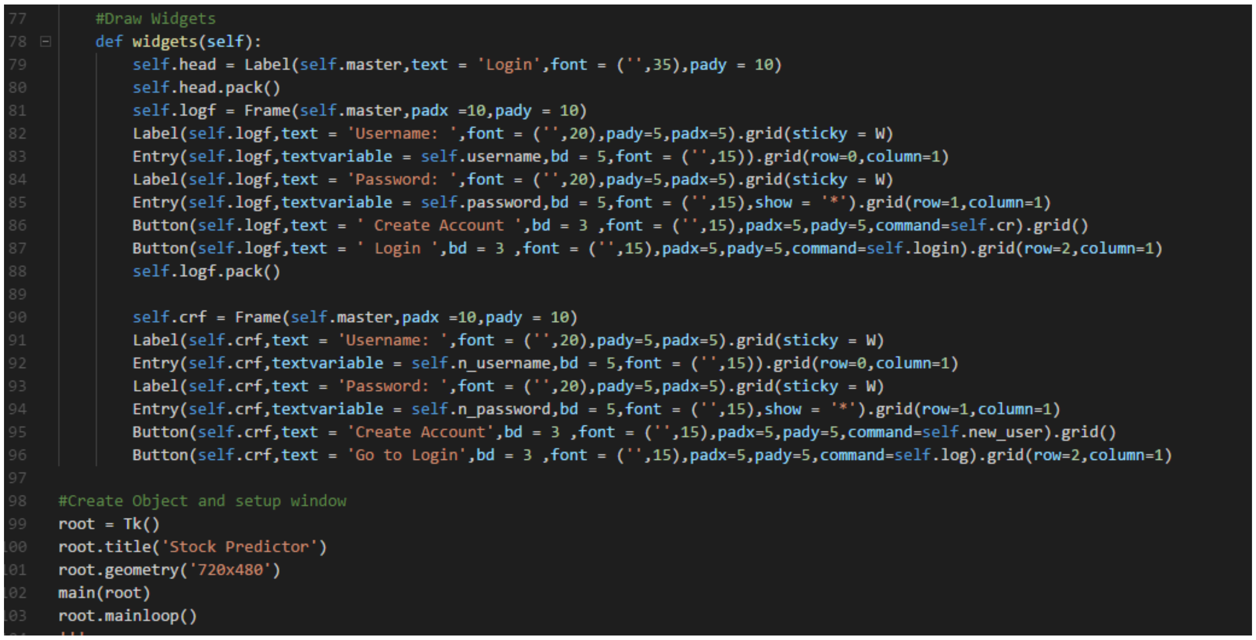
```python
#main Class
class main:
    def __init__(self,master):
        # Window
        self.master = master
        # Some Usefull variables
        self.username = StringVar()
        self.password = StringVar()
        self.n_username = StringVar()
        self.n_password = StringVar()
        #Create Widgets
        self.widgets()

    #Login Function
    def login(self):
        #Establish Connection
        with sqlite3.connect('users.db') as db:
            c = db.cursor()

        #Find user If there is any take proper action
        find_user = ('SELECT * FROM user WHERE username = ? and password = ?')
        c.execute(find_user,[(self.username.get()),(self.password.get())])
        result = c.fetchall()
        if result:
            self.logf.pack_forget()
            ms.showinfo('Welcome', 'Welcome \n' + self.username.get())
            self.head['pady'] = 150
        else:
            ms.showerror('Error!','Username Not Found.')

```

Figure 4. 17 User Interface class main to display login and signup interface



```python
    def new_user(self):
        #Establish Connection
        with sqlite3.connect('users.db') as db:
            c = db.cursor()

        #Find Existing username if any take proper action
        find_user = ('SELECT * FROM user WHERE username = ?')
        c.execute(find_user,[(self.username.get())])
        if c.fetchall():
            ms.showerror('Error!','Username Taken Try a Diffrent One.')
        else:
            ms.showinfo('Success!','Account Created!')
            self.log()
        #Create New Account
        insert = 'INSERT INTO user(username,password) VALUES(?,?)'
        c.execute(insert,[(self.n_username.get()),(self.n_password.get())])
        db.commit()
```

Figure 4. 18 Adding new user to database



```python
    #Draw Widgets
    def widgets(self):
        self.head = Label(self.master,text = 'Login',font = ('',35),pady = 10)
        self.head.pack()
        self.logf = Frame(self.master,padx =10,pady = 10)
        Label(self.logf,text = 'Username: ',font = ('',20),pady=5,padx=5).grid(sticky = W)
        Entry(self.logf,textvariable = self.username,bd = 5,font = ('',15)).grid(row=0,column=1)
        Label(self.logf,text = 'Password: ',font = ('',20),pady=5,padx=5).grid(sticky = W)
        Entry(self.logf,textvariable = self.password,bd = 5,font = ('',15),show = '*').grid(row=1,column=1)
        Button(self.logf,text = ' Create Account ',bd = 3 ,font = ('',15),padx=5,pady=5,command=self.cr).grid()
        Button(self.logf,text = ' Login ',bd = 3 ,font = ('',15),padx=5,pady=5,command=self.login).grid(row=2,column=1)
        self.logf.pack()

        self.crf = Frame(self.master,padx =10,pady = 10)
        Label(self.crf,text = 'Username: ',font = ('',20),pady=5,padx=5).grid(sticky = W)
        Entry(self.crf,textvariable = self.n_username,bd = 5,font = ('',15)).grid(row=0,column=1)
        Label(self.crf,text = 'Password: ',font = ('',20),pady=5,padx=5).grid(sticky = W)
        Entry(self.crf,textvariable = self.n_password,bd = 5,font = ('',15),show = '*').grid(row=1,column=1)
        Button(self.crf,text = 'Create Account',bd = 3 ,font = ('',15),padx=5,pady=5,command=self.new_user).grid()
        Button(self.crf,text = 'Go to Login',bd = 3 ,font = ('',15),padx=5,pady=5,command=self.log).grid(row=2,column=1)

#Create Object and setup window
root = Tk()
root.title('Stock Predictor')
root.geometry('720x480')
main(root)
root.mainloop()
```

Figure 4. 19 Login and signup widgets

4.2.6.2 Stock Prediction

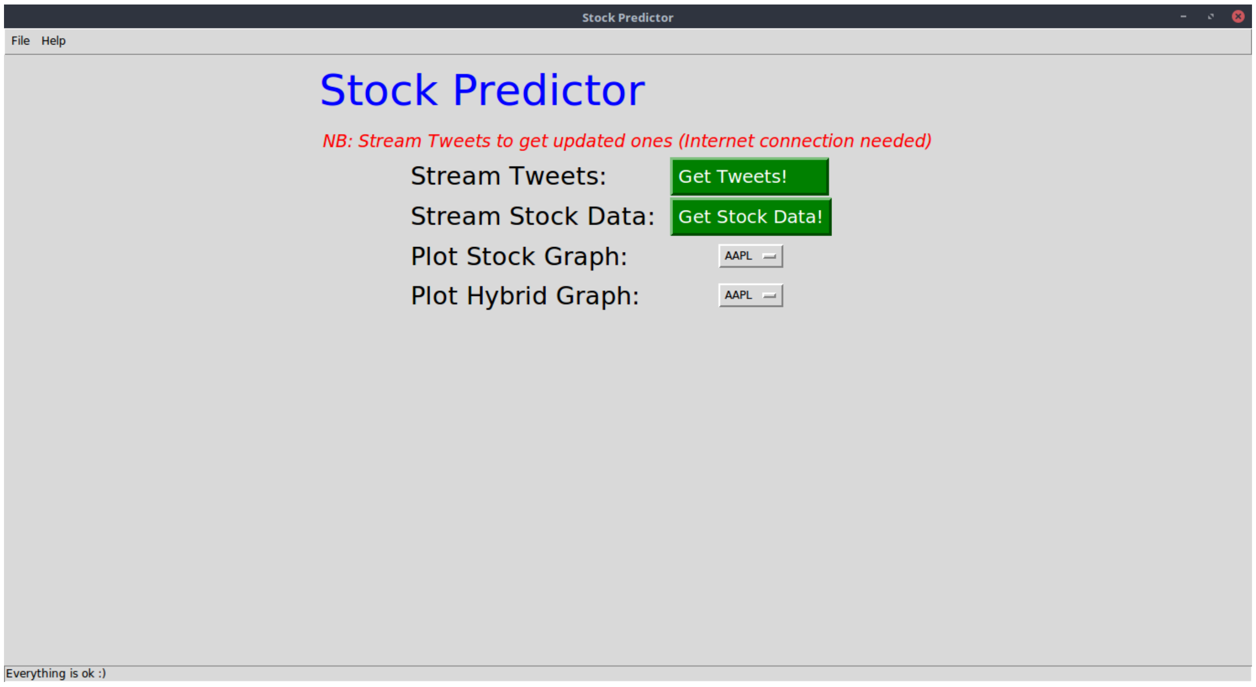
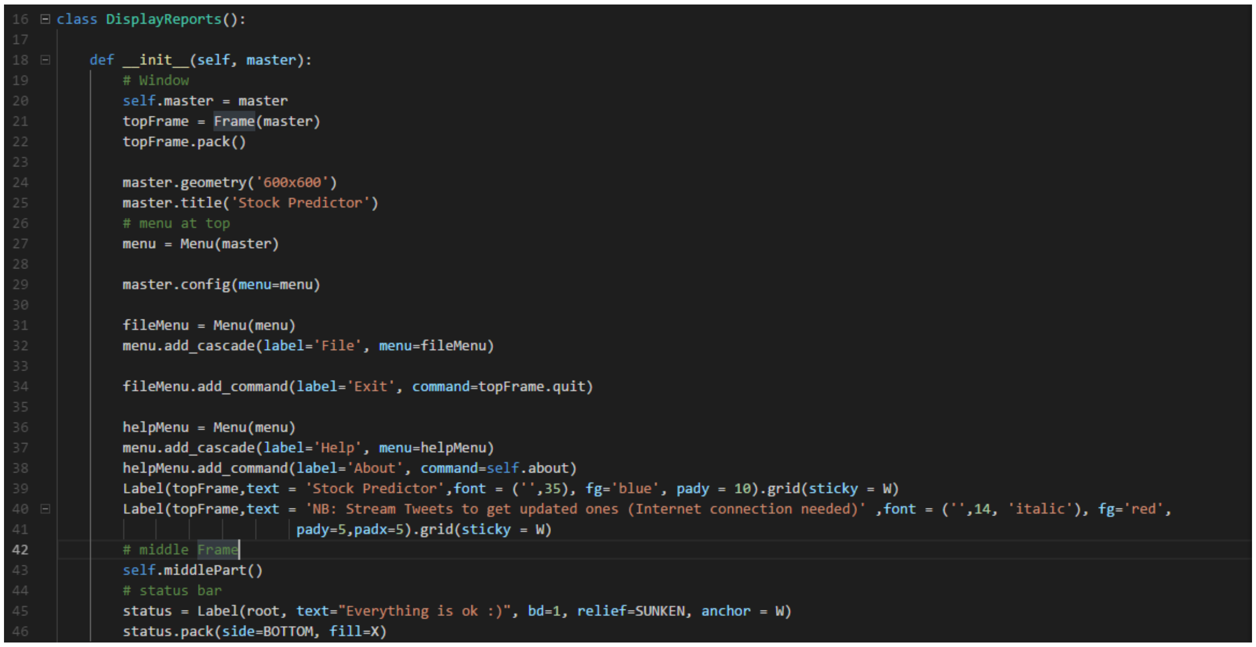


Figure 4. 20 Stock Predictor Main Interface

There is a menu at the top that will be used to exit the application as well as have an about section that will inform the user on the version of the system that they are currently running.

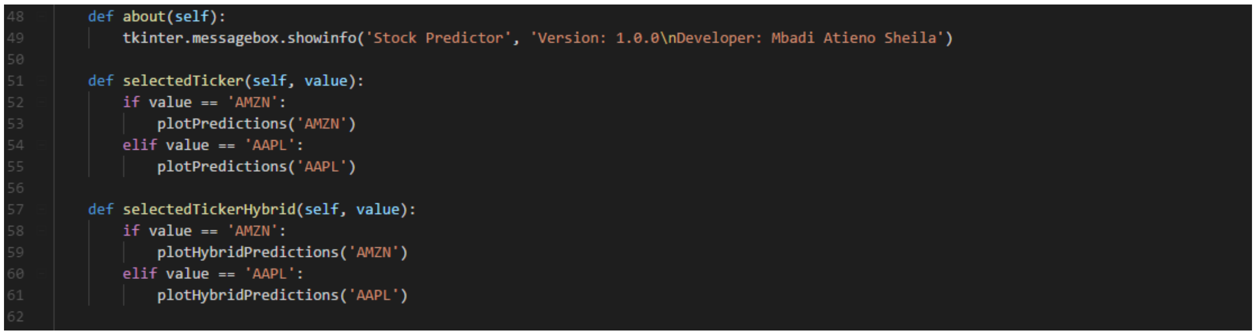
There will be 2 buttons. One that is used to update the twitter data in the CSV files by invoking the twitter data API. The other one will be used to update the stock data in the csv files by updated data from the alpha vantage API.

There will be 2 dropdown lists containing a company's ticker, one will be used. They will be used to dynamically plot graphs based on the available data. The first dropdown list plots stock data that don't incorporate sentiments and the other one plots a graph of the company's stock data with sentiment polarity added as a feature. The graphs will be displayed in an interactive window which is based on Matplotlib library. The reason for this is that it adds a range of options for the user. One major one is the ability to save the graph as an image so that it can be viewed at a later date.



```
class DisplayReports():

    def __init__(self, master):
        # Window
        self.master = master
        topFrame = Frame(master)
        topFrame.pack()

        master.geometry('600x600')
        master.title('Stock Predictor')
        # menu at top
        menu = Menu(master)

        master.config(menu=menu)

        fileMenu = Menu(menu)
        menu.add_cascade(label='File', menu=fileMenu)

        fileMenu.add_command(label='Exit', command=topFrame.quit)

        helpMenu = Menu(menu)
        menu.add_cascade(label='Help', menu=helpMenu)
        helpMenu.add_command(label='About', command=self.about)
        Label(topFrame,text = 'Stock Predictor',font = ('',35), fg='blue', pady = 10).grid(sticky = W)
        Label(topFrame,text = 'NB: Stream Tweets to get updated ones (Internet connection needed)' ,font = ('',14, 'italic'), fg='red',
                          pady=5,padx=5).grid(sticky = W)
        # middle Frame
        self.middlePart()
        # status bar
        status = Label(root, text="Everything is ok :)", bd=1, relief=SUNKEN, anchor = W)
        status.pack(side=BOTTOM, fill=X)
```

Figure 4. 21 User Interface Implementation (a)



```
    def about(self):
        tkinter.messagebox.showinfo('Stock Predictor', 'Version: 1.0.0\nDeveloper: Mbadi Atieno Sheila')

    def selectedTicker(self, value):
        if value == 'AMZN':
            plotPredictions('AMZN')
        elif value == 'AAPL':
            plotPredictions('AAPL')

    def selectedTickerHybrid(self, value):
        if value == 'AMZN':
            plotHybridPredictions('AMZN')
        elif value == 'AAPL':
            plotHybridPredictions('AAPL')
```

Figure 4. 22 User Interface Implementation (b)

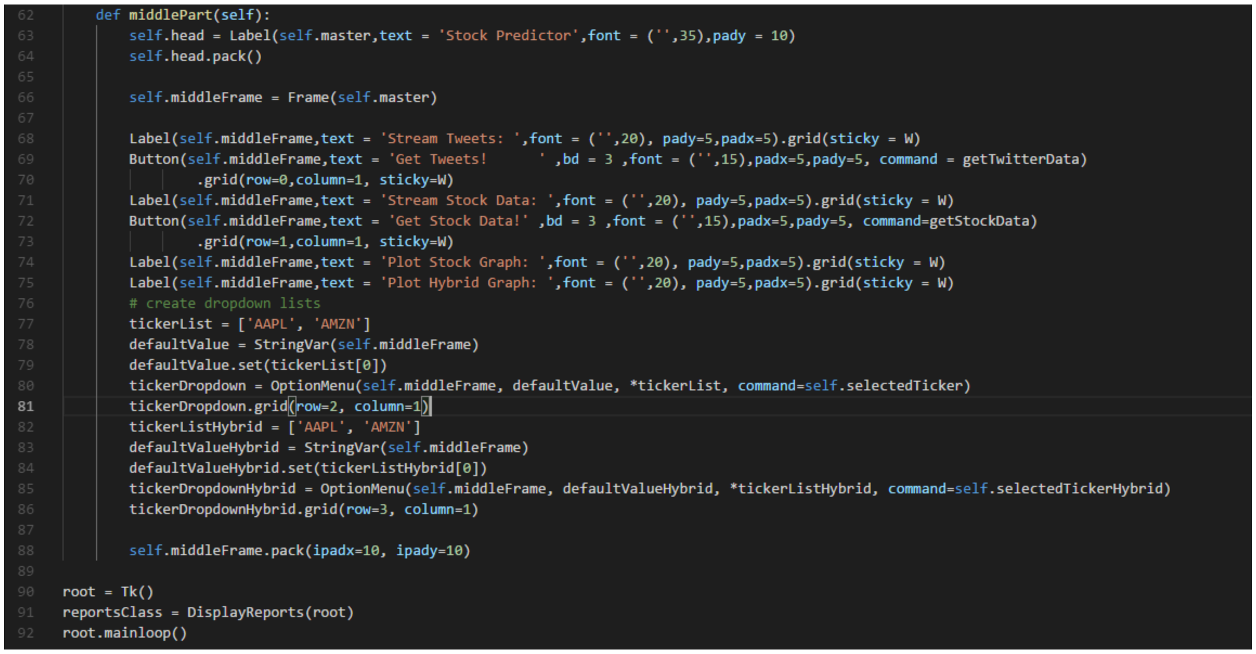

```python
    def middlePart(self):
        self.head = Label(self.master,text = 'Stock Predictor',font = ('',35),pady = 10)
        self.head.pack()

        self.middleFrame = Frame(self.master)

        Label(self.middleFrame,text = 'Stream Tweets: ',font = ('',20), pady=5,padx=5).grid(sticky = W)
        Button(self.middleFrame,text = 'Get Tweets!      ' ,bd = 3 ,font = ('',15),padx=5,pady=5, command = getTwitterData)
                .grid(row=0,column=1, sticky=W)
        Label(self.middleFrame,text = 'Stream Stock Data: ',font = ('',20), pady=5,padx=5).grid(sticky = W)
        Button(self.middleFrame,text = 'Get Stock Data!' ,bd = 3 ,font = ('',15),padx=5,pady=5, command=getStockData)
                .grid(row=1,column=1, sticky=W)
        Label(self.middleFrame,text = 'Plot Stock Graph: ',font = ('',20), pady=5,padx=5).grid(sticky = W)
        Label(self.middleFrame,text = 'Plot Hybrid Graph: ',font = ('',20), pady=5,padx=5).grid(sticky = W)
        # create dropdown lists
        tickerList = ['AAPL', 'AMZN']
        defaultValue = StringVar(self.middleFrame)
        defaultValue.set(tickerList[0])
        tickerDropdown = OptionMenu(self.middleFrame, defaultValue, *tickerList, command=self.selectedTicker)
        tickerDropdown.grid(row=2, column=1)
        tickerListHybrid = ['AAPL', 'AMZN']
        defaultValueHybrid = StringVar(self.middleFrame)
        defaultValueHybrid.set(tickerListHybrid[0])
        tickerDropdownHybrid = OptionMenu(self.middleFrame, defaultValueHybrid, *tickerListHybrid, command=self.selectedTickerHybrid)
        tickerDropdownHybrid.grid(row=3, column=1)

        self.middleFrame.pack(ipadx=10, ipady=10)

root = Tk()
reportsClass = DisplayReports(root)
root.mainloop()
```

Figure 4. 23 User Interface Implementation (c) showing the creation and layout of views

CHAPTER 5: TESTING

5.1 Introduction

This chapter elaborates on the tests carried out to measure the system's achievement of the objectives.

5.2 Objectives of Testing

1. To demonstrate the working functionality of the system.
2. To assess the usability of the system.
3. To assess the system's accuracy.

5.3 Testing Scope

The following are the tests that the system was subjected to:

5.3.1 Functional Testing

Functional testing is a way of checking software to ensure that it has all the required functionality that's specified within its functional requirements.

The system is supposed to accepts a user's query that is in form of a stock ticker, runs natural language processing on the input to classify its polarity, combine this sentiment data to stock data. Train and test a model based on this new data and compare performance of the model without sentiment data and that which has sentiment data added.

The system achieves all this objectives.

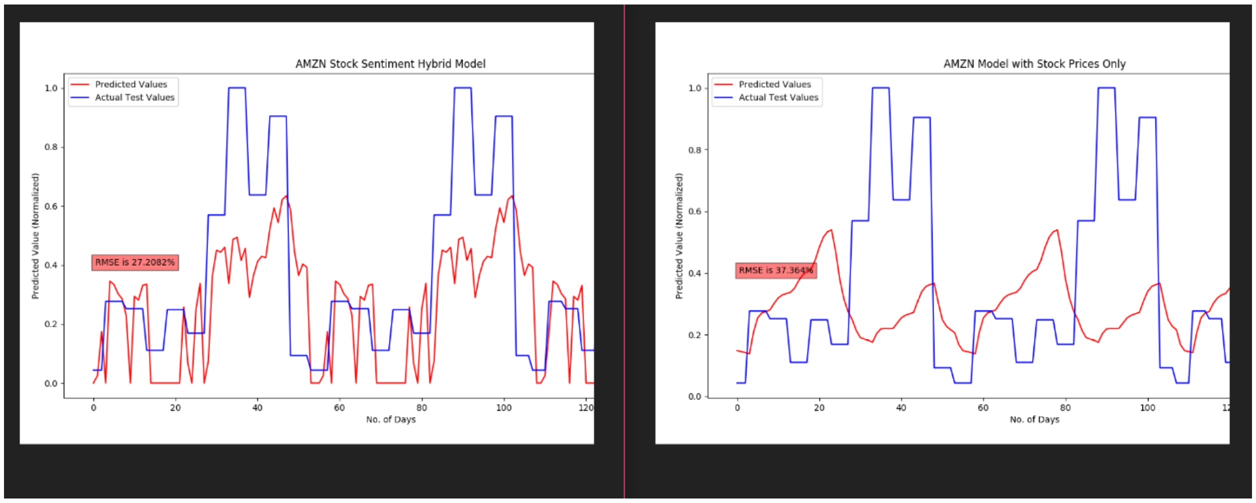


Figure 5. 1 Functional Testing

5.3.1.1 Performance Testing
This test is used determine how a system performs in terms of responsiveness and stability under a particular workload.

In terms of speed, the software performs well as the trained model is saved as a Hierarchical Data Format 5 (hdf5) file using Keras, this prevented retraining of the model before using it and this saved a significant amount of time.

In terms of stability, the system is tested under data of different sizes and it performed well in all instances.

5.3.1.2 User Interface Testing
This entails testing the system User Interface to ensure it meets its specifications.

The UI is simple to use as it is only composed of buttons and graphs. The buttons are used to navigate through the various data, based on stock ticker.

The interface is therefore user friendly.

5.3.1.3 Usability Testing
Usability testing is a way to see how easy it is to use a software by testing it with real users.

Navigating through the various display pages is easy as the navigation is only done through button clicks.

5.3.1.4 Portability Testing
Portability testing is the process of determining the degree of ease or difficulty to which a software component or application can be effectively and efficiently transferred from one hardware, software or other operational or usage environment to another.

This software can run perfectly on Windows Operating System environment despite the fact that it was built on a Linux machine.

5.3.2 Significance of Testing
System testing was vital in the following ways:

1. The different scope of tests enabled us to monitor the quality of the system in different aspects from start to end of development.

2. It enabled us to know that the system provides reliable output with a consistent relationship to input.

CHAPTER 6: RESULTS AND DISCUSSION

6.1 Introduction

This section presents the findings and outcomes of our research and discusses them. Input to output mapping is also discussed.

6.2 Results

6.2.1 Input-Output Mapping

The system accepts a search clause by clicking a button.

Only two stock tickers were used as they had the most Twitter Data, which are stocks for Apple Inc. and Amazon.

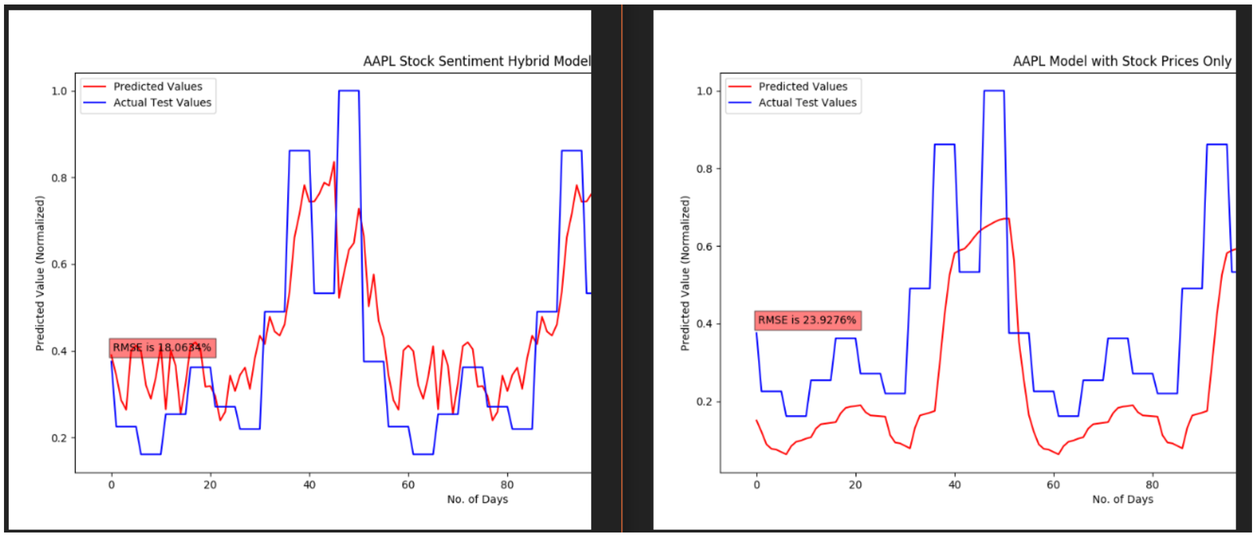


Figure 6. 1 Comparison of Apple Inc. Prediction

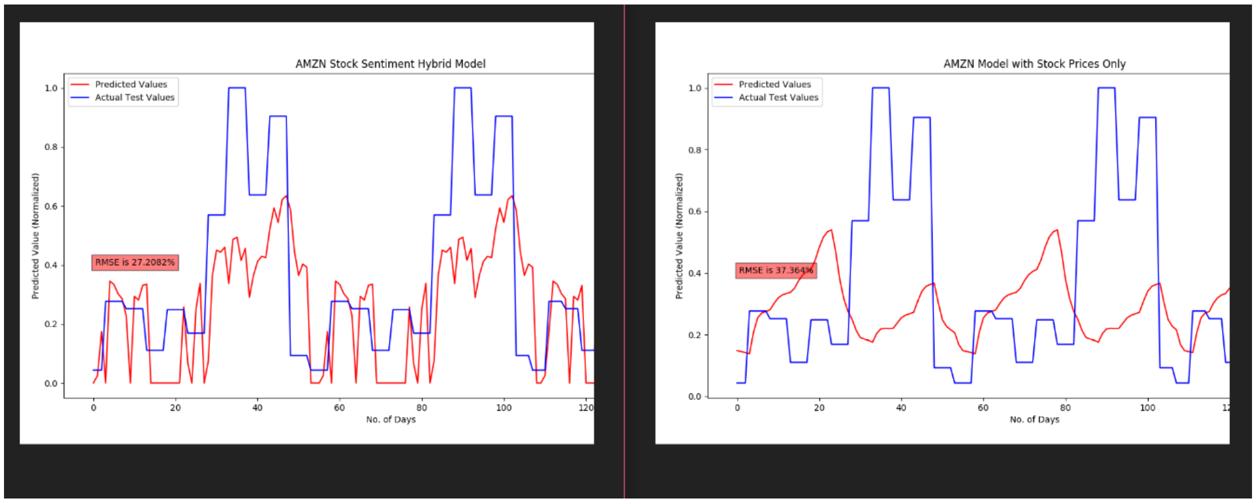


Figure 6. 2 Comparison of Amazon Stock Prediction

Daily tweet polarity is incorporated as a feature to the Long Short Term Memory (LSTM) Neural Network. Two models are used, one incorporating Tweets polarity and another without. The results are displayed in a time series graph and the results are compared.

6.2.2 Comparison of the two Models

The model that incorporated tweet sentiment polarity performed better than the one that had only stock data. In other words, the hybrid model had the best performance.

6.3 Discussion

Bollen et.al suggested that incorporating public sentiment had a positive impact on the movement of stock market prices. My research also shows that incorporating tweets makes the predictions more accurate.

However the twitter data to be analyzed wasn't enough so the performance of the model wasn't at its best. An increase in data will most likely lead to more accurate predictions.

CHAPTER 7: CHALLENGES, RECOMMENDATIONS AND CONCLUSIONS

7.1 Challenges

The realization of this project was successful. The hybrid model that incorporated tweets performed better but the accuracy could be improved.

Lack of funds made it very hard to get historical twitter data thus I was limited to working with the most recent tweets. Having more tweets would have increased the accuracy of the hybrid model.

7.2 Future Work

An annotated dataset for stock data should be created. This will significantly improve the classification accuracy of tweet polarity.

7.3 Recommendations

This research should be localized to Kenya to aid in the improvement of our economy.

7.4 Conclusion

Currently, there is need of an accurate model that can be used for accurate predictions of stock market movement. Our hybrid model performed well on this task, but further enhancement to tweet polarity classification can significantly improve these results.

Achievement of our Objectives

1. To study the enhancement of the Naïve Bayes classifier for sentiment classification of tweets and use it.
   Achievement: The work of Narayanan, Arora & Bhatia was studied. TextBlob, a Python library that incorporates all their suggestions was used for sentiment analysis classification.
2. To study the best techniques for stock prediction and implement one.
   Achievement: Time Series Analysis was studied and this led us to the study of Long Short Term Memory (LSTM) Neural Networks which have very good results in stock prediction. A model was then built using LSTM Neural Networks and its results were impressive.

3. To implement a hybrid stock prediction model and calculate its accuracy.
   Achievement: A hybrid stock prediction model that incorporated tweet polarity was implemented and its accuracy was computed using Root Mean Squared Error (RMSE).

   All objectives were achieved, and the hybrid model was an enhancement to the model that does not incorporate sentiments.

Reference

1. *Introduction to Sentiment Analysis*. (2017). [EBook] p.5. Retrieved from: https://lct-master.org/files/MullenSentimentCourseSlides.pdf.
2. *Stock market prediction*. (2017) Retrieved from https://en.wikipedia.org/wiki/Stock_market_prediction.
3. *Stock.* (2017) Retrieved from https://en.wikipedia.org/wiki/Stock.
4. *Technical Analysis* (2017) Retrieved from https://www.investorsunderground.com/technical-analysis/
5. Burton G. Malkie (2003). The efficient market hypothesis and its critics. *Journal of Economic Perspectives*, *17*(1), 59–82.
6. Johan Bollen, Huina Mao, and Xiao-Jun Zeng (2010). Twitter mood predicts the stock market. *1*(1), 1–8.
7. Murtaza Roondiwala, Harshal Patel and Shraddha Varma (2015). Predicting Stock Prices Using LSTM. *International Journal of Science and Research (IJSR), 6(4),* 1754-1756.
8. Bhadane, C., Dalal, H., & Doshic H., (2015). Sentiment analysis: Measuring opinions. *International Conference on Advanced Computing Technologies and Applications (ICACTA-2015)*. 808-814. Retrieved from http://www.sciencedirect.com/science/article/pii/S1877050915003956
9. *Text Classification for Sentiment Analysis – Naive Bayes Classifier*. (2010). Retrieved from https://streamhacker.com/2010/05/10/text-classification-sentiment-analysis-naive-bayes-classifier/
10. Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal, 5,* 1093-1113
11. Turney, P. (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. *Proceedings of ACL*. 417-424.
12. Turney, P.D., Littman. (2003). M.L.Measuring Praise and Criticism: Inference of Semantic Orientation from Association. *ACM Transactions on Information Systems*. 315-346.
13. Ahmet, T. (2015). *Text classification and sentiment analysis.* Retrieved from http://ataspinar.com/2015/11/16/text-classification-and-sentiment-analysis/

14. Ahmet, T. (2016). *Sentiment analysis with bag-of-words.* Retrieved from http://ataspinar.com/2016/01/21/sentiment-analysis-with-bag-of-words/
15. Mulcrone, K. (2012). Detecting Emotion in Text. University of Minnesota.
16. Fang, X. & Zhan, J. (2015). Sentiment analysis using product review data.
17. Narayanan, V., Arora, I., & Bhatia A. (2007). Fast and accurate sentiment classification using enhanced Naive Bayes model.
18. Naïve Bayes (2017). Retrieved from http://scikit-learn.org/stable/modules/naive_bayes.html
19. Time Series (2017). Retrieved from https://www.investopedia.com/terms/t/timeseries.asp#ixzz55mTuzkK8
20. Moving Average (MA), (2017). Retrieved from. https://www.investopedia.com/terms/m/movingaverage.asp#ixzz55mXdcgu6
21. Moving Averages- Simple and Exponential, (2017). Retrieved from. http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:moving_averages
22. Exponential Moving Averages For Irregular Time Series, (2013) https://oroboro.com/irregular-ema/
23. Thushan Ganegedara, (2018). *LSTM Neural Network.* Retrieved from https://www.datacamp.com/community/tutorials/lstm-python-stock-market