Faculty of Natural and Mathematical Sciences
Department of
Engineering/Informatics

King's College London,
Strand Campus, London,
United Kingdom

7CCSMPRJ
Individual Project Submission 2020/21

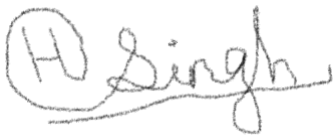
Name: Harshvardhan Singh
Student Number: 20094816
Degree Programme: MSc Computational Finance
Project Title: Using Sentiment to Predict Financial Time Series
Supervisor: McBurney, Peter
Word count: 15000

RELEASE OF PROJECT

Following the submission of your project, the Department would like to make it publicly available via the library electronic resources. You will retain copyright of the project.

☒ I **agree** to the release of my project

☐ I **do not** agree to the release of my project

Signature: 

Date: 1st September 2021

Department of Informatics
King's College London
United Kingdom

7CCSMPRJ MSc Project

USING SENTIMENT TO PREDICT FINANCIAL TIME SERIES

Name: Harshvardhan Singh
Student Number: 20094816
Degree Programme: MSc Computational Finance

Supervisor's Name: McBurney, Peter

This dissertation is submitted for the degree of MSc in Computational Finance

ACKNOWLEDGEMENT

I would like to thank my supervisor, Prof. Peter McBurney, for his immense guidance for this dissertation. I would also like to acknowledge his step-by-step advice on approaching the dissertation problem, starting from collecting data to executing each task using the effective time management methodology of Gantt Chart that helped me on completing this dissertation smoothly and effectively. I would like to thank him for his assistance in giving constructive feedback and conducting various sessions throughout this master dissertation tenure. I would like to thank my parents for believing in me and guiding me throughout my life. Without their support, I could not have reached where I am currently in. Thanks to my friends who helped me throughout the curriculum. I am highly obliged to all the professors in King's College London to give me immense knowledge through their beautiful teaching techniques.

ABSTRACT

Numerous types of research have been done on several aspects of stock market prediction by using sentiment analysis that has given rise to this unique technique to gauge the market's sentiment and thereby predict the market. Such application has attracted many brilliant minds in this lucrative field, such as financial markets, to conduct further research on predicting the financial time series using sentiment analysis. In this project, I have performed financial time series prediction for Apple stock price data using sentiment analysis and tried to compare the forecasted stock price with the actual stock price data through data visualization and Root Mean Squared Error Estimation and further tuned some hyperparameters. The headline dataset is collected over three years from Jan-2014 to Dec-2016 at Computer Science & Engineering Department, Indian Institute of Technology, Roorkee [1], [2]. A lexical approach, VADER, has been used to calculate the sentiment class and sentiment score. The sentiment class is then divided into negative, neutral, and positive, and the sentiment score ranges from -1 to +1. Other information contained in the data is the time stamp, news headline text and news headline URL. Then, the financial time series prediction has been made using LSTM RNN Model and stock sentiment data. The stock data on which this experiment has been done is Apple stock data.

NOMENCLATURE

VADER.	Valence Aware Dictionary for Sentiment Reasoning
LSTM	Long short-term memory
ARIMA	Autoregressive Integrated Moving Average
ANN	Artificial Neural Network
BP	Back-Propagation
RMSE	Root Mean Squared Error
LDA	Latent Dirichlet Allocation
PMI	Pointwise Mutual Information
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
K-NN	K-Nearest Neighbour
RAM	Random Access Memory
API	Application Programming Interface
AAPL	Apple Inc.
LSTM RNN	Long short-term memory Recurrent Neural Network
RNN LSTM	Recurrent Neural Network Long Short-Term Memory

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

The Introduction section talks about the problem statement, aims and objectives, structure of the report, and background theory for sentiment analysis. The motivation for adopting the skeleton structure of the introduction is appropriated from this article [15].

1.1 Problem Statement

Many asset managers, hedge fund managers, Big Financial Institutions running their big trading desks have realised that news sentiment plays an enormous role in their decision-making process. According to their market sentiments, discretionary portfolio managers and traders constantly perform rigorous research and analysis to incorporate the stocks and indices in their portfolios. As we know, it is very cumbersome for these discretionary portfolio managers to quantify the financial instrument's news sentiments manually. The portfolio managers and most discretionary traders face challenges forecasting the financial instruments' manually when they have a gazillion amount of news headlines data popping up on their Bloomberg Terminal's screen, or they have collected those news headlines data. They are in the constant research of forecasting the financial time series data with the help of quantification of the news sentiment data to forecast the stock prices early and thereby manage their portfolio risk early. Nevertheless, quantifying these humongous news sentiment data is not so easy if they perform their research manually. They need to adapt state of the art sentiment analysis techniques to thrive in this cutthroat competition in financial institutions.

Various research has been conducted to prove the effect of news sentiment on the stock price[16]. Many research has been conducted to search for the correlation between news sentiment and stock price. Considering this research, I have built a sentiment analysis system that tries to make use of the news sentiment data and make a forecast of the target financial stock data which will assist the discretionary traders and portfolio managers in forecasting the financial time series ahead of time by using the state of the art technique of sentiment analysis and thereby make the decision ahead of time and mitigate their risk on anything (such position sizing, rebalancing, etc.)

Two types of datasets are used here:

- Historical News headline Data (later filtered for Apple Inc. News): This is used as a source dataset.
- Historical Apple Stock Price Price Data: This is used as a source and a target dataset.

Using the above two datasets, the prediction of financial time series has been conducted using sentiment analysis techniques. Before the prediction, various rounds of data pre-processing have been done. They start by storing the data from the source into the database and then take them into the perfect data structures, thereby calculating the sentiment score using the VADER sentiment library. Feature selection and feature engineering have been performed to produce a high-quality dataset for our model. The chosen model that we have used is the RNN LSTM model architecture. It is one of the model architectures used in the deep learning field. After

feeding the source dataset and target dataset into the model, a financial forecast has been made. Furthermore, a visual comparison between the forecasted time series data and the actual stock price time-series data is made. The software will also walk you through the performance and analysis of the prediction model and help the end-user, such as quantitative researchers, to perform their custom analysis on top of this software.

1.2 Aims and Objectives

Quantitative Researchers, Hedge Fund Managers have done many types of research on dealing with the noises in the financial time series data. However, there is an effect of news sentiment hidden in these financial stock price data among these noises, and many researchers agree with this. They are trying hard to use this correlation between stocks and news sentiment for their trading benefits. There is still much more to explore around this news sentiment data for financial time series forecasting.

This report gives discretionary traders and financial researchers a tool to use the news sentiment headline as their source and thereby help them forecast the financial time series data. The tool will help the end-user see the target financial time series (the apple stock data is used). The resultant data visualization will also give you comparative visualization of the forecasted time series data along with the actual stock price data. This study demonstrates how the stock sentiment data is correlated with its prices. This study will demonstrate the RNN LSTM deep learning model's performance when applied to the individual stock data (Apple stock data) rather than indices in this report. In this study, I have used the combination of previous stock price data and the sentiment data of the last day to forecast the Apple stock price. After dissecting this data into the training and test datasets, the training data is then fed to the prediction model for training purposes. And then, the model is tested against the test dataset and further improved through hyperparameter tuning.

Before modelling the time series data, the tool tries to ensure that the stock sentiment data and stock price data are combined perfectly, which will help the prediction model predict stock price more accurately. This has been achieved through rigorous data pre-processing, which includes calculating the polarity score of the news headline data, gauging the correlation between the source data set (which is our sentiment data) and the target data set (which is our financial time series data), feature scaling and data preparation for feeding the data into the model.

1.3 Report Structure

In the subsequent sections, several aspects of the research have been discussed with the help of individual chapters. However, a brief discussion about the sentiment analysis is discussed in terms of financial forecasting. In Chapter 2, the background theories supporting this research have been discussed with the literature reviews. Chapter 3 recalls the objective in a more detailed way. It explains the design and how it achieves the project aim. Chapter 4 presents and justifies the methodology used to deal with the problem and describes the implementation procedures. The background theory presented in chapter two has been recalled supporting the proposed implementation. Chapter 5 summarises the results obtained from the proposed design and methodology. The procedures to obtain the results have been described. Analysis and evaluation have been performed. Comparisons of the results have been made. It also justifies that the project aims and objectives have been successfully achieved.

Furthermore, finally, Chapter 7 sums and findings this research, describing how project aims have been achieved and address the research questions. This also talks about the contributions and results that have been achieved. It also briefly talks about the future direction of this research.

1.4 Background Theory: Sentiment Analysis

Many traders are using sentiment analysis for trading in a systematic fashion. Sentiment describes the way people feel about security or asset. Blogs, discussion forums, news articles, and Twitter are all ways to express these ideas. Sentiment analysis quantifies this text-based information by assigning a positive or negative score to it. The process of buying or selling securities with the help of sentiment scores is called sentiment trading. Though this is only a theoretical concept, we have seen how sentiments affect the market [3].

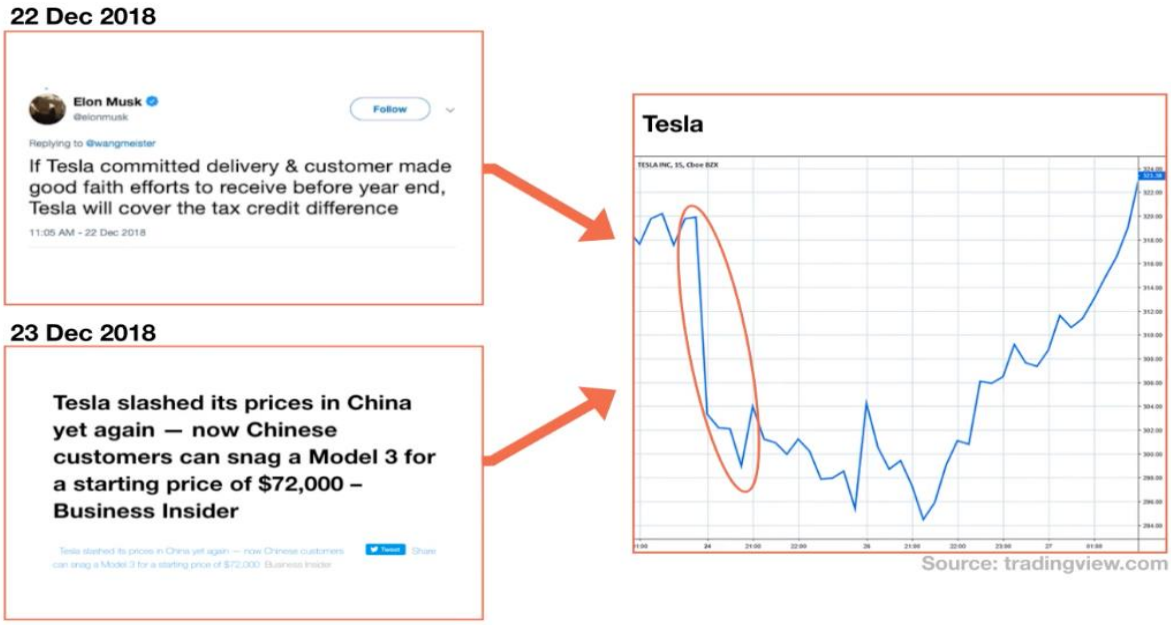


Figure 1.1: Elon Musk Tweet
(Source: [3])

Elon Musk, for instance, tweeted about Tesla. As implied in the tweet, negative sentiments were surrounding it. Tesla was failing to deliver on time. In another news article, it was revealed that Tesla was slashing its price in China. The Tesla stock subsequently traded 6% lower the following day [3].

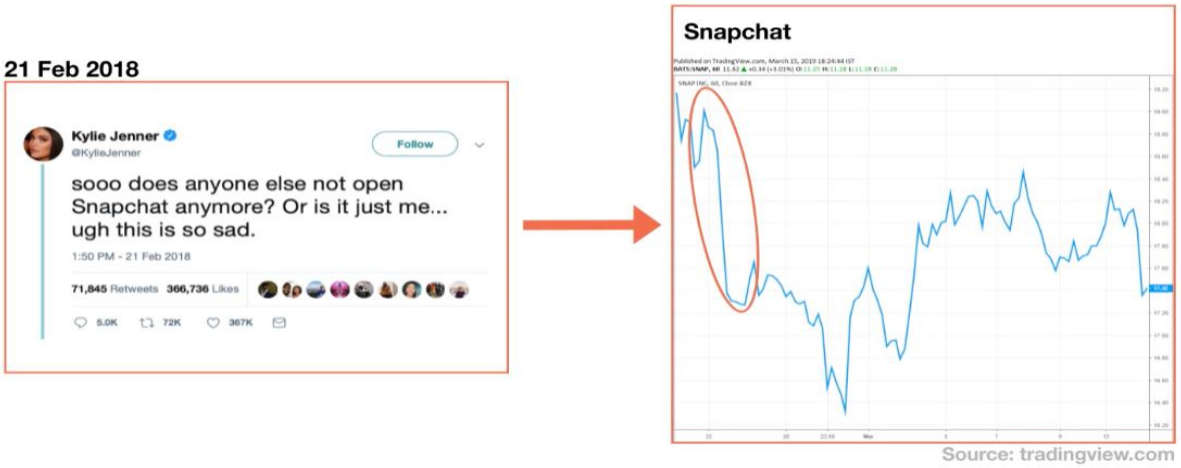


Figure 1.2: Kylie Jenner Tweet about Snapchat
(Source: [3])

Another example relates to when Kylie Jenner, an American reality star, announced that she had discontinued using Snapchat, leading to an approximately 6% decline in the stock price. In addition, sentiment has an impact on the market in numerous other ways [3].

Despite the fact that sentiments seem to be an obvious indicator of market moves, it is difficult to predict. It is difficult for machines to interpret human sentiments or emotions. To accomplish this, we use Natural Language Processing. This field of study examines the interaction between human language and computers. Using it, machines are able to understand sentiments expressed on an even more human level [3].

Information flow occurs in various forms. It can occur as a piece of news; it can come out as pre-news. Sometimes, many rumours lurk around and last, but not least, we have a vast, gigantic amount of sentiment data coming out of various social media sites. In this study, we primarily focus on the news headlines, a combination of company-specific news, product news, and so on [4, p. 4]. There has also been a study on the effect of positive and negative sentiment on the stock price. One such study from [5] has come up that the negative sentiments have a significant effect on the market prices as compared to the positive sentiments [4, p. 9], [5]. We can find evidence of this by looking at the above graphs, which depict the rapid falling of stock prices and then gradually increasing.

The financial market dynamics combine all the judgements made by the portfolio managers and the traders. As we know, the stock prices are greatly influenced by the investor's expectations about the company's growth. The greater the company's growth potential, the greater the number of buyers in the market for that company stock. News plays a massive role in maintaining the investor's expectation about the future growth potential of the stock. For instance, the constant flow of the good news will keep up the market price to grow more as it will attract more buyers, whereas, if the news flow is stagnated, it will leave a wrong sentiment among the investors about the stock, and then that stock might not perform well. So from this, we learn that the news flow can carry a massive role in controlling the supply and demand sentiment among the investors. There are some dangers concerning the news sources. We have witnessed that many blogs and extensive social media sites carry much noise and less news, affecting our sentiment analysis model. So, to tackle this issue, we are strictly working on only the news headlines dataset rather than the tweets from the large social media sites [4, p. 111].

Now that the news headlines are collected, where each news headline incorporates negative and positive sentiments, we would like to go deep down in our analysis and calculate to what extent this news headline depicts a positive or a negative sentence? We would want to quantify the intensity of these news sentiments into a somewhat numerical value that is more readable to our models. This numerical value is known as sentiment score. The main challenge is to classify these sentiment data into positive, negative and neutral categories [4, p. 10].

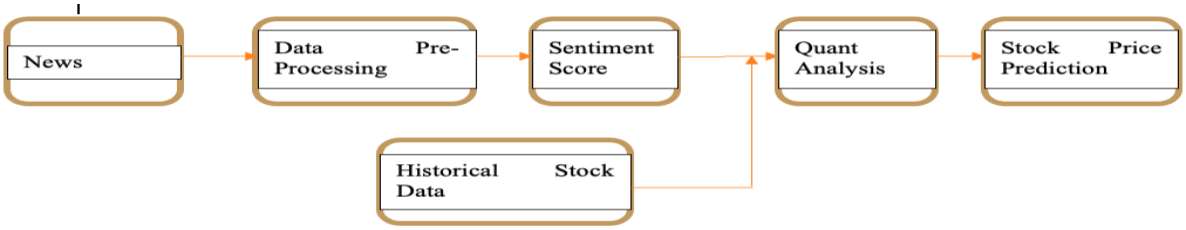


Figure 1.3: Quant Model Architecture using Sentiment Analysis
(Source: [4, p. 18])

In the above-given figure, the design of quantitative model architecture for predicting the stock financial time series data with the help of sentiment analysis is being illustrated. In this architecture, the news headline data are pre-processed at the initial stage to make it more suitable for calculating the sentiment score. After that, we combine these calculated sentiment scores with the historical stock data into the quantitative model, further predicting the financial time series data [4, p. 18].

Inside this sentiment analysis architecture, the sentiment classification and calculation of the polarity score of each news headline data are performed concerning the financial instruments. This classification incorporates positive sentiment probability, which tells the probability that the news sentiment is positive. Its value lies from 0 to 1. Negative sentiment probability, which suggests the probability of negative news sentiment. Neutral sentiment probability, which indicates the probability of news sentiment is neutral. The sum of all these probabilities is equal to 1 [4, p. 111].

Let us deep-dive into the application of sentiment analysis in the stock market. Starting with [6], which has collected approximately five years of financial news data and tried to quantify the relationship between this data and the stock market. They found a positive correlation between the financial news of a specific stock and its financial trading parameters such as price and volume, especially the volume. They found a significant change in the volume of a traded stock after the news of that stock has been released in the market. In short, they found that if there is more news lurking in the market for a specific stock, then in the next period, that stock possesses a large volume of trades. This uncovers that the number of times the stocks have been in the news is directly proportional to the large volume of stock traded in the next consecutive period.

Many academia and researchers have shown their deep interest in the prediction of the stock market. They went against the efficient market hypothesis and dedicated their research to uncover patterns from the noise hidden in the stock market. However, that did not stop them from realising that news is also a great source of information for predicting the stock market. Many types of research say that the stock market does not possess the random walk theory, and to some extent, they do follow a pattern, and the catalyst for those patterns are indeed the tweets and news from social media sites and microblogs. Many investigations have given rise to the statement that the stock market is highly correlated with the tweets, news, blogs and other

information that we see in microblogs. We know that the stock market is a function of greed and fear.

Furthermore, we also know that these greed and fears come from human behaviour, which can be seen in tweets and blogs. This behaviour leads to the decisions made by the traders and portfolio managers, where these decisions further lead to changes in the stock prices. This eventually suggests that tweets and blogs are equally crucial as news are. This fact was put forward by this paper [7], which tried to predict the financial time series of the Dow Jones Industrial Average with the help of vast chunks of tweeter data. They used two sentiment tools, and one tool was supposed to estimate positive and negative sentiments. In contrast, the other tool calculated the sentiment by classifying it into different dimensions of sentiments. And then, the resulting sentiment data that came out of these two tools are compared with the Dow Jones Industrial Average financial time series data. This experiment experienced that the sentiment data generated by the second tool with the help of dimension classification of sentiments helped them predict the stock price significantly.

There is much research done in the field of sentiment analysis applied explicitly to the equities. They are discussed separately as follows:

[56, Ch. 9] talks about predicting the stock return time series data with the help of text mining operations. These operations were applied to texts that are not structured. These texts are the earnings report of the company. This operation will assist the investors in keeping track of the company's long term growth potential by keeping an eye on their balance sheets. The study talks about the relationship between the company's report and its stock returns time series data. The study tries to find out that the more complex the company's report is, the weaker will be its future performance. The term complex refers to the degree of manipulation that has been done while preparing the company's report. The study tries to find out the negative correlation between the degree of complexity of the report and the stock returns data. The study tries to prove that the company's complex financial balance sheet or statements tend to perform poorly in the future.

[56, Ch. 10] investigates the sentiment and investors decisions. It primarily talks about the extreme changes observed in the stock price regime after the announcement of the earnings, and the changes in the accruals also seem to be at a greater extent due to this announcement. It focuses on the fact that the investors make their trading decision more actively when there is an announcement. This paper also recommends the future direction, which further suggests recognising the best techniques to filter and accumulate news sentiment, which can help us to create an index for the stock market sentiment, which helps us to grasp the relationship between stock market sentiment and investors decision with the news sentiment data.

[4, Ch. 5] investigates the impact that news events have on market sentiment. Until now, we have experienced that the stock price change occurs in two ways: first due to the immediate effect of stock news, and second is due to the sentiment or the behaviour of the market participants. An exhaustive amount of research done by investors and other market participants on the news-based trading strategy has added an invaluable improvement in better yielding trading strategies and predicting the financial time series data. This paper also depicts the improvement in trading strategies with the proper implementation of news-sentiment datasets. It has also shown the performance enhancement in the market-neutral trading strategy that has been achieved due to the sophisticated use of sentiment data.

It is due to all the studies that have been mentioned above led us to ask a fundamental research question of predicting the financial time series data using the sentiment analysis, and this research paper aims to strictly implement and comprehensively answer the research question that is mentioned above.

CHAPTER 2: LITERATURE REVIEW

As discussed in the introduction section, this research predicts the financial time series using sentiment analysis. As part of this chapter, we will review the main preliminaries and details related to the subject. The literature review will explore existing literature works that the researchers already do, discuss a broader sentiment effect on the stock market, and dig deeper into forecasting the stock price using the sentiment and stock data. The motivation for adopting the skeleton of the literature review is appropriated from this article [13].

2.1 Effect of Sentiment Data on Stock Market

Tetlock [8] argues about the impact of the media on the financial market. It reviews and analyses the relationship between the media and the financial market. This paper studies various data sources such as newspapers, the internet, social network posts to evaluate the critical information in the financial market. These kinds of data have a great potential to uncover the mystery of the high volatility that frequently occurs in the stock prices and trading volumes. He commented that the news sentiments such as federal reserve policies, housing market sentiment data, recent oil supply shock, exchange rate movements, technological progress, or news about the war, all these events have a great potential to affect the stock market. He performed an experiment and calculated the polarity of the news article column. He counted the frequency of the positive words and negative words respectively in that news column and the polarity score for that particular news column. He found that these polarity scores can translate objectives in the market. He observed that negative words were closely related to the market performance, and the positive words had less relationship with the market. However, the causality relationship between these words and the market was not clear. In order to interpret this, he started to observe whether the markets were responding appropriately to the events that were causing these words. He observed that the magnitude of the effect of the sentiment on the market was about 7%. He commented that the pessimist journalists and optimist journalist media contributed to the stock market movement by using negative and positive words in their news articles. So, through his research, he concluded that the media elevates the investors' mood, which further affects the stock market [14].

Similarly, Hafez P. [9] talks about the role of news events on market sentiment. This study describes how the events in various companies help in the preparation of market sentiment indexes. He demonstrates how using the event novelty score (which tells about the recentness of the news story in the 24 hours) can significantly enhance predicting some index returns. He also demonstrated that the strategies based on market-level sentiment outperform the momentum-based strategies by approximately four times. Likewise, he also added that the strategies based on market-level sentiments yield double-digit returns 80% of the time. He also demonstrated that the industry-level sentiment index could help achieve profit by taking a market neutral position by buying top-performing stocks and selling underperforming stocks.

In 2010, Engelberg [10], and other economists, in their research study, show that the news plays a massive role for short-sellers, as the trading position of a short seller is profitable if they analyse the news sentiment efficiently. The combination of short sales data and news data showed that the negative relationship between short and futures returns is doubled on days when there is good news and four times as large on days when there is bad news. Another question that this paper tries to achieve is whether short-sellers are better news anticipaters than other traders. However, they found that short sellers make their trades simultaneously as other traders trades. They also found that the short-sellers tend to make more profits on the news days as the transaction cost tends to be lower during these news events, suggesting the relationship between stock prices during the news events.

We can deduce there is a strong relationship between what Tetlock [8] uncovers regarding the fact that the negative sentiments were closely related to the market performance with the Engelberg [10], which analyse that the negative correlation between short sales and future returns is four times larger on days when there is bad news.

Mitra [11] and his team commented that the volatility of the stock portfolio could be better calculated using the market sentiment information than traditional option implied volatility estimates. They extended their research and combined both news sentiment and implied volatility data to enhance the market volatility estimates. This approach was taken because the classic factor model does not estimate the volatility correctly as soon as the market sentiment changes. For instance, they tried to capture the volatility using the plain and straightforward GARCH model during the announcement period; by doing so, they observed that the GARCH model predicted the volatility quite the opposite of reality. It predicted the low volatility during the announcement events, predicting the high volatility after the news announcement. In 2010, Zhang [12], with his fellow researchers, used the data consisting of company-specific news frequency, polarity scores, which originated through the NLP sentiment system, to forecast the company's return on its trading volume. By realising the effectiveness of the news sentiment data, they further extended their research and build the market neutral trading strategy based on news sentiment. This paper found the compelling interconnection between the media and the market sentiment, which agrees with Tetlock's research [8].

2.2 Predicting Stock Trend Using Financial Time Series Data

Many researchers have dedicated their research to predicting stock prices by constantly applying cutting edge quantitative tools. Tiwari [17], in their research, predicts the stock price using the combination of the neural network model and the ARIMA model. However, instead of choosing a plain ARIMA model, they made an auto ARIMA model that helped them automate the calibration of the p, q, and d values and choose the best value for their forecasting experiment. They used almost decades of Nifty 50 Index price. They fed it into models like Regression, Neural Network models, Time Series Models, compared the forecasting result from each of them, and then decided the appropriate model for their experiment. They observed that the feed-forward neural network model predicted prices better as compared to other models. Despite introducing Twitter sentiment analysis as a model in their experiment, the prediction model did not incorporate the Twitter sentiment analysis model. They could have coupled Twitter sentiment analysis with the feed-forward neural network model and enhanced their prediction model.

Ariyo [18], in their study, attempts to predict the stock price using an autoregressive integrated moving average (ARIMA) model for a short time frame. The experiment demonstrated that the ARIMA (2, 1, 0) was considered the most appropriate Nokia stock index price prediction model. The accuracy was illustrated by calculating the mean squared error and comparing the model value and the actual stock index price. On the other hand, ARIMA (1, 0, 1) seemed the perfect model to forecast Zenith Bank stock index price. The paper thereby provides evidence of the ARIMA model as a practical choice for forecasting short time frame price time series.

Furthermore, Wichaidit [19] affirms that using the ARIMA model with the cross-correlation method produces improved price prediction of stock price trends in terms of short term movement. The motivation of the cross-correlation method is that it incorporates another highly correlated stock along with the target stock that we are trying to forecast, and this hybrid model is named as CARIMA model, which is an acronym given by them for the Cross-Correlation ARIMA model. The CARIMA model prediction closely correlated with the actual price in the experiment compared to the ARIMA model. They added that despite having a marginally higher mean absolute error than the ARIMA model, the CARIMA model provided a more acceptable prediction than the ARIMA model.

Due to the non-linear behaviour of the stock market prices, several researchers tried to adapt neural network models instead of traditional statistical models to analyse the stock market prices. Abhishek [20], with his fellow researchers, proposed a stock market forecasting model which used an artificial neural network architecture to forecast future prices based on past prices. The model used was a multi-layered feed-forward model, and it was trained using the back-propagation algorithm. They chose the ANN architecture due to its capability to hypothesise the undiscovered datasets such as stock price data, which is also known for its non-linear behaviour. They have used batched mode technique rather than the incremental mode technique to train the neural network model and thereby updates the weights and biases. In terms of the learning algorithm, they used an unsupervised cluster-based algorithm to follow the similar clusters observed in the target variable, in this case, the stock price. By using a supervised learning algorithm, they have iteratively updated the weights and biases. For updation of weights and biases, they used the trainlm and traindx network training function. The experiment was done on Microsoft stock price data. Trainlm function performed better than Traindx in terms of mean squared error estimation. The paper concludes with the future recommendation for experimenting with various artificial neural network architecture and training functions.

Furthermore, in their research, Qin [22] made a comparison between the ARIMA model and the neural network model in terms of stock price prediction accuracy and demonstrated that both of them have great potential to forecast the stock prices in terms of short term time horizon. The paper also acknowledged some of the imperfections of methodology, such as adopting only single-step forecasting even though they could have performed the multi-step forecasting. They further demonstrated that the prediction accuracy between the two methods is almost similar. However, In 2018, Du [23] proposed that the prediction accuracy of the ARIMA model and BP neural network model is not similar and commented that the BP neural network model indeed performed better than the ARIMA model. He also added that if they combine the two models for the prediction rather than using them individually, the performance of the forecast seems to be even better. They found that the prediction accuracy of the ARIMA-BP combined model had the root mean squared error of 18.02%. In comparison, the RMSE of the other two individual models was 89.21% and 65.73%, respectively, which is far more than the combined

model. This suggests that the prediction accuracy of the combined ARIMA-BP model is far better than the individual ARIMA and Back-Propagation (BP) Neural Network model.

Concerning the ARIMA model, Artificial Neural Network Architecture, and their comparison discussed above, Wijaya [21], in their research paper, further compared the accuracy value of the prediction method through the ARIMA and Artificial Neural Network methods. The experiment was done on Indonesian stock data. The paper commented on a couple of deficiencies in using the ARIMA model. The first is that the ARIMA model acts according to the assumptions on autocorrelation, volatility variation, and error term normality. If there is a breaking of any of the dataset's characteristics mentioned above, it might affect the prediction accuracy. On the other hand, the relatively new architecture of the artificial neural network model has an advantage over the assumptions mentioned above. First, ANN behaves like a human brain, which helps it capture the datasets' non-linear behaviour, such as stock price. It learns from the data and accommodates itself accordingly. The only weakness of ANN is that it needs a large dataset. The experiment adds that the prediction accuracy of the ANN model is better than the ARIMA model.

According to the above-discussed kinds of literature, it has been observed that the stock price prediction models based on neural network architecture beat the traditional statistical time series analysis technique such as the ARIMA model in terms of prediction accuracy. It is also observed that the combined use of different models gives a more accurate result than the usage of individual models.

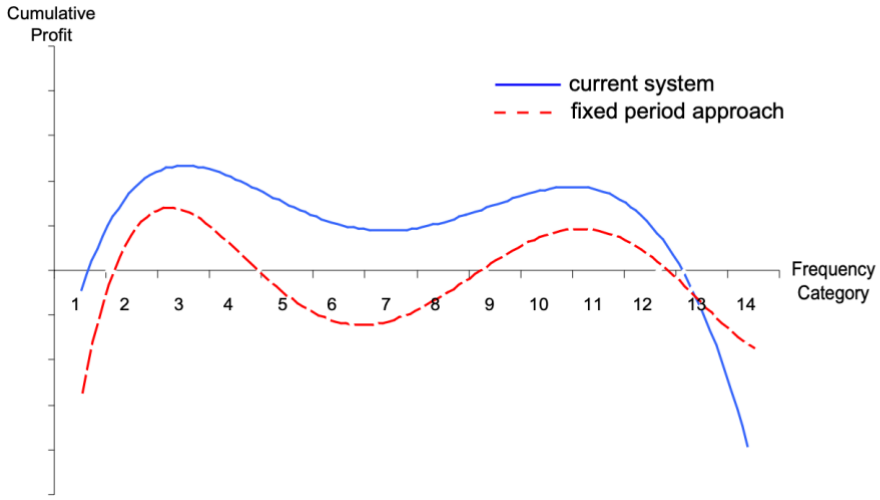
2.3 Predicting Stock Trend Using News Sentiment Data

This section of the literature review comments on the literature that has been taken place on sentiment analysis and text processing techniques for stock trend prediction.

Falinouss [15] used the text-mining technique to analyse the market's behaviour and predict the trend. The paper examines the effect of the news article on the stock market using various kinds of text mining tools. The primary purpose of this research paper is to develop a solution on how to gauge the stock market's movement using news sentiments and how cutting-edge data mining and text mining approaches can help them achieve this. The data category is divided into two categories: stock price data and the second is financial news data. Statistical techniques linear and non-linear modelling, unsupervised learning algorithm such as clustering algorithm, and supervised learning algorithm such as classification algorithm is used in this research project. TF-IDF is used for document representation for the n-dimension matrix of features. Due to the large size of the feature matrix, the random projections technique is used for dimensionality reduction. SVM classifier is used to output results such as stock price rise and drop from the news release data given as an input. The prediction model has achieved the 83% of accuracy. The paper also suggests the future direction of the research, such as applying different machine learning techniques and then comparing the accuracy of each machine learning model, extending the research to develop the trading strategies and applying the machine learning models with the automobile company-specific news sentiment data.

In 2002, Fung, Yu, and Lam [24], in their research study, attempted to forecast the stock price differences affected by the news articles. Event-driven predictions for stocks was made by making the use of news articles. This news based prediction system is dissected into two phases: the training phase and the operational phase. In the training phase, the stock data and the news article data are collected; each trend is identified and labelled. On the other hand, for

news articles, the feature extraction technique is applied. After that, the coordination of the stock data with its news sentiment data is made. Then the prediction model is constructed using the calculated weights. On the other hand, the real-time news article is recognised in the operational phase, making alignment with the stock data. If the alignment is successful, the prediction trend is performed; otherwise, the news is discarded. The stock trend is detected using a t-test piecewise linear segmentation algorithm, whereas the trend labelling is performed with the incremental k-mean clustering algorithm. In the end, the market simulation of buying and selling trading strategy was carried out to calculate the model's profitability. The paper demonstrated the comparison between the fixed period alignment approach and the current news sentiment stock prediction system.



Category	Frequency Range
1	26-110
2	116-193
3	195-219
4	221-274
5	282-337
6	339-425
7	427-494
8	547-618
9	628-696
10	749-891
11	903-2030
12	2127-2733
13	2773-3565
14	3688-8091

Fig 2.1: Cumulative Profit vs Frequency Category
(Source: Fung, Yu, and Lam [24])

The above figure compares the news arrival frequency category with the cumulative profit made by the news sentiment stock prediction system. The paper suggests that more news frequencies resulted in more profit than fixed period interval news arrival by referring to the above graph. However, the fixed period approach performed better when the frequency of the news increased drastically. This is because more news may contain more noise data, leading the current system to perform poorly [15]. In order to tackle this noisy label data, they could have used the generalisation of the Expectation-Maximisation algorithm to reduce the error rate in text classification. Ramakrishnan proposed this Bayesian model with his fellow researchers [25].

Tirea [26], in their research paper, talks about their decision support system that tries to uncover the critical information hidden inside the textual information that is in news articles and financial reports and thereby conducts a trend prediction based on the correlation between this textual information with the stock price. The system then takes the incoming fresh news article and then decide whether the freshly arrived news will affect its stock price movement or not. To deal with the complex architecture of this text mining system, the ontology-based data mining approach is adapted to track the interrelationship between different types of data and to make it more accessible to the model. The architecture of this system is segregated into three subsystems: the portfolio optimisation system, the multi-agent trading system to carry out the process of analysing the stock trend direction, and the text mining system to extract the vital information from the incoming unstructured textual news article. The text mining system is subdivided into classification agents, text mining agents, and analysis agents. The classification

agents classify the news articles based on the predefined categories that are in the ontology system. The text-mining agents extract the relevant information using various kinds of text mining and text summarization techniques. The analysis agents then combine classification agents and text mining agents and analyse the correlation between the textual information found in news articles with the stock price movement. The research paper put forward a future direction on making the system that can detect the reliability of the news sources and then calculate its reliability score; then, traders can decide which news sources they should trust to make their trading decision.

In 2020, Keswani [27] proposed the LDA based model to extract the top words from the financial news articles that are extremely useful in predicting stock price movement and even detecting market crash events. In this paper, LDA uses two kinds of datasets and the first is the historical data of the S&P 500, and the second is news articles about S&P 500 news. By combining these two datasets, the prediction of the market movement and the speculation about the market crash events are made. The paper proposes a future direction regarding using these LDA-based models to combine machine learning techniques to predict market crash events. Yun [28] proposed stock price prediction using the Korean natural language processing technique based on the newspaper article title. The skip-grams technique was adopted to extract the features from the noun words from the Korean sentence, and then these features are passed into the CNN model to predict the stock price movement. The research paper makes us think that with the help of some NLP techniques, the stock prices can be predicted by using the natural language processing system based on the local languages.

Wu [29], with fellow researchers, proposed the stock price prediction by using the sentiment analysis technique with the technical analysis. The pointwise mutual information (PMI) method was used to detect the seed words. After that, the intensity of the sentiments for each news article was calculated. The sentiment intensity denotes the total strength of the stock information present in the stock news document. Then by combining these sentiment intensity scores with the technical analysis trading information, a prediction was made for the future stock price. They demonstrated three models in their research paper. First is SATA_all that blends the sentiment analysis features with the technical analysis features. Next is SATA_seed that incorporate the seed words in the technical analysis trading information. Moreover, the third one is T.A. which is a plain and simple technical analysis approach. After the experiment, the RMSE of SATA_all and SATA_seed was about 35.64 and 35.84, respectively. The RMSE of T.A. was about 51.51, which suggests that the prediction accuracy improves if we incorporate the sentiment data with any prediction model. The paper also proposes the future direction to find the relationship between the sentiment patterns and distribution of the words.

By reviewing the above pieces of literature, we can see the usefulness of the text-mining algorithms in terms of feature extraction from the unstructured data, which can help the model predict the price trend with improved accuracy. This also allows us to research the causality effect of textual information on the stock price movements.

2.4 Predicting Stock Trend Using Time Series and News Sentiment Data

This section of the literature review focuses on the machine learning techniques applied to the news sentiment data and numerical stock price data to predict the stock's future price.

In 2019, Mohan and Sameeta [30], with their other researchers, in their research study, gathered a large number of time series data and the news article data related to that time-series dataset and applied a deep learning algorithm to predict the future stock price. They commented that using large datasets helped them to improve their model accuracy. The datasets used in this research included S&P 500 daily stock price data and around two hundred thousand news article data of the stocks in the S&P 500 index. The system proposed in this research paper uses today's news sentiment data to predict today's stock's closing price with the help of a historical time series dataset of the stock. In their experiment, they divided the dataset into three parts. They allocated 90% of the dataset for the training purpose, 5% of the data were allocated for the cross-validation process, and another 5% of the data was selected for the model testing purpose. ARIMA, Facebook Prophet, and RNN LSTM models were used as a prediction model. After the experiment, they found that the RNN LSTM had better prediction accuracy than the ARIMA and Facebook Prophet models. They also commented that the models did not provide accurate predictions on the high variations of stock prices and kept this as future research work.

While the above literature used RNN LSTM, this research paper [31] used the CNN LSTM prediction model for stock price prediction. The model used the combined input datasets of numerical price dataset and textual news sentiment dataset. This research paper uses an event embedding technique that helps in the feature extraction process and helps advance the stock price prediction process based on event-driven information. The experiment was carried out on three datasets that are provided by the three different renowned data sources. The architecture of the system deals with the news headline and historical price data separately. The news headline data goes through multiple stages of the data pre-processing before it is fed into the prediction model. The pre-processing data stages involved open information extraction framework, event representation and event embedding process. In the same way, the historical price data and the technical indicator information are combined and normalized before it was sent to the prediction model. The system performance was calculated based on accuracy and annualized return that was calculated from trading simulation. After the experiment, the efficacy of the event embedding method was checked. It was observed that model prediction accuracy was enhanced by adding the event embedding and numerical data into our prediction model.

Vargas [32], with his other researchers, conducted a study on the intraday price pattern using a deep learning model using both CNN and RNN architecture. They have applied word embedding and sentence embedding techniques for feature extraction purposes. The sentence embedding technique proved to be better than the word embedding technique as it also dealt with the low denseness of the word vectors. The compound input of technical indicator information and news sentiment data was fed into the Recurrent Convolutional Neural Network model (RCNN) model to predict the intraday behaviour of the time series data. The paper also proposed future work on developing the new embedding techniques to represent the words more robustly [13].

Ni, Wang, and Chen [35] studied the stock market forecasting potential from the combination of tweets embedding data and historical prices. They have adapted a deep learning model architecture for analysis purposes—this research paper pinned point to the fact that tweets have a cross-relationship mechanism. The tweets, comments and reply to the tweets make a structural relationship with each other. This research paper uses this relationship of type of tweets and the deep learning model to forecast the financial time series data. This research is motivated by the DRNews model. This model gives you the semantic information as well as

the cross-relationship textual information of the news events. The research uses the Twitter node algorithm, which computes the cross-relationship between different tweets. The BERT model determines the emotions from the tweet texts. The textual meaning information, the emotions from the tweets, and inter-textual tweet information are combined and stored in the vector-based data structure provided by the Node2vec framework. The essential elements from the tweets were extracted by calculating the scores of each element of the tweets using the term frequency-inverse document frequency algorithm and taking out only those elements with significant scores. The network-like structure of Twitter nodes is created. The elements of the tweets with significant scores are then combined with the tweet replies, and the emotional bias is fed into this network-like structure of the Twitter nodes to output the tweet embedding. The Node2Vec framework was used to encode the tweets and thereby create a tweet representation. The Node2Vec framework uses the combination of depth-first search and breadth-first search crawling algorithms. The BERT model is then combined with this Node2Vec framework to calculate the tweets embeddings. The output from the tweeter node algorithm is combined with stock time-series datasets and fed into the deep learning model to predict the financial time series data for a short timeframe.

The financial market is entirely surrounded by some kind of financial news and online investor sentiments. However, among these qualitative data, there is much unauthorized news lurking around the financial market. For instance, Twitter plays a vital role in terms of providing stock-based tweet sentiment information, but among these tweets, there is more chance of rumours being spreading. In order to tackle this challenge, Hu [36], with other fellow researchers, came up with this unique idea of Hybrid Attention Network-based architecture to forecast the financial time series dataset. This architecture is given an input of the sequence of the latest linked news. This architecture solves the data integrity problem by adopting the principle of sequential content dependency and diverse influence. A customised algorithm is made to optimize the learning process of the model. Sequential Context Dependency is a principle that entails that the news that closely resembles the non-informative news article can be analysed as a consolidated context. This consolidated context can further be used to predict the stock trend. The diverse influence principle talks about the investors making a rational decision about the stock direction based on the diverse news sentiments. Effective and efficient learning entails that the investors first focus on the critical news sentiments that can move the market and thereby gauge the stock direction and then focus on the useless information that somewhat affects the market. They calculated the performance of the proposed framework by creating the market trading simulation-like scenario. They performed an annual backtesting procedure. The simulation first assigns the sentiment score to each of the stocks that are currently in the portfolio. The top-k stocks are then chosen for the next trading day according to their scores. Equal capital allocation is given to each top-k stock based on their opening price of the next trading day. Transaction cost is kept at 0.3%. The average return of the holding trade is then calculated, and then the performance is then evaluated using the average return as a metric. The paper also proposes a future work on combining technical analysis information with news sentiment data to predict the stock trend.

Using the psychological drive of the investors, this paper [37] tried to predict the stock market with the help of the deep learning model. Since stock prices are heteroskedastic in nature, to tackle this heteroskedasticity and non-linearity property, the stock prices were fed into the time series smoothing algorithm, which helped them enhance the prediction model's performance. To insert the investors' psychological drive into the model, they collected an abundance of investors comments about the stock market, and then the sentiment class calculation was performed using the sentiment analysis technique. This sentiment class, specifically the

sentiment index, was basically the binary classification of the stock market's upward or downward movements. This sentiment index, combined with the stock price time series dataset, was fed into the forecasting model. Empirical modal decomposition, which is a smoothing algorithm, is used to remove the heteroskedasticity and trend from the financial time series and convert this complex price time series into the stationary and straightforward price time series to get the enhanced forecasting result. Since there is much information in the market, an attention-based LSTM is used to perform the prediction. This attention-based LSTM model is segregated into three layers: the input layer, the LSTM layer, and the attention layer. The input layer takes care of the sentiment index data and the historical time series data. The LSTM layer consists of the number of LSTM neural network cells. These cells entail the number of days of historical data used to predict the next-day price. The number of cells taken into consideration was 30. The attention layer accumulates the assigned probability that was given to each information at each iteration of time steps into the LSTM layer. The main objective of the attention-based layer was to give more attention to the information that leads to the binary classification of the sentiments. The experiment was started with the simple LSTM model, and then the sentiment indexes were incorporated into the LSTM model to see the effectiveness of the sentiment data on the prediction accuracy. After that, the attention-based layer was incorporated into the model, and at the end, the final model was further optimised with the EDM time series smoothing algorithm to transform the complex time series sequence into the simple stationary sequence. The prediction accuracy was seemed to be improved with each additional modification of the model.

Li and Pan [38] used a blended ensemble deep learning technique to predict the financial time series data using stock prices and textual news datasets. This blended deep learning model consists of two levels. The first level is a combined architecture of GRU and LSTM. GRU is an acronym for Gated Recurrent Units network LSTM is an acronym for Long Short-Term Memory network. The second level consists of FCNN, an acronym for Fully Connected Neural Network, used to enhance forecast performance. The news data was taken from four significant financial news service providers. The polarity scores of WSJ news, CNBC news, Fortune news, and Reuters' news combined with the stock price data was taken into consideration as an input for the prediction model. This data is divided into a training dataset, a cross-validation dataset, and the test dataset. The training dataset was fed into sub-model one, and the cross-validation dataset was fed into sub-model two. The test dataset was used for performance testing purposes. The new training dataset was prepared from these two sub-models results and then fed into the fully connected neural network model for the final prediction. After the experiment, the result showed that the blended ensemble method improved the prediction accuracy by 2X compared to the simple deep learning model. The paper finally proposes to use a more complex data sequence into the more complex ensemble methods to predict the financial time series dataset.

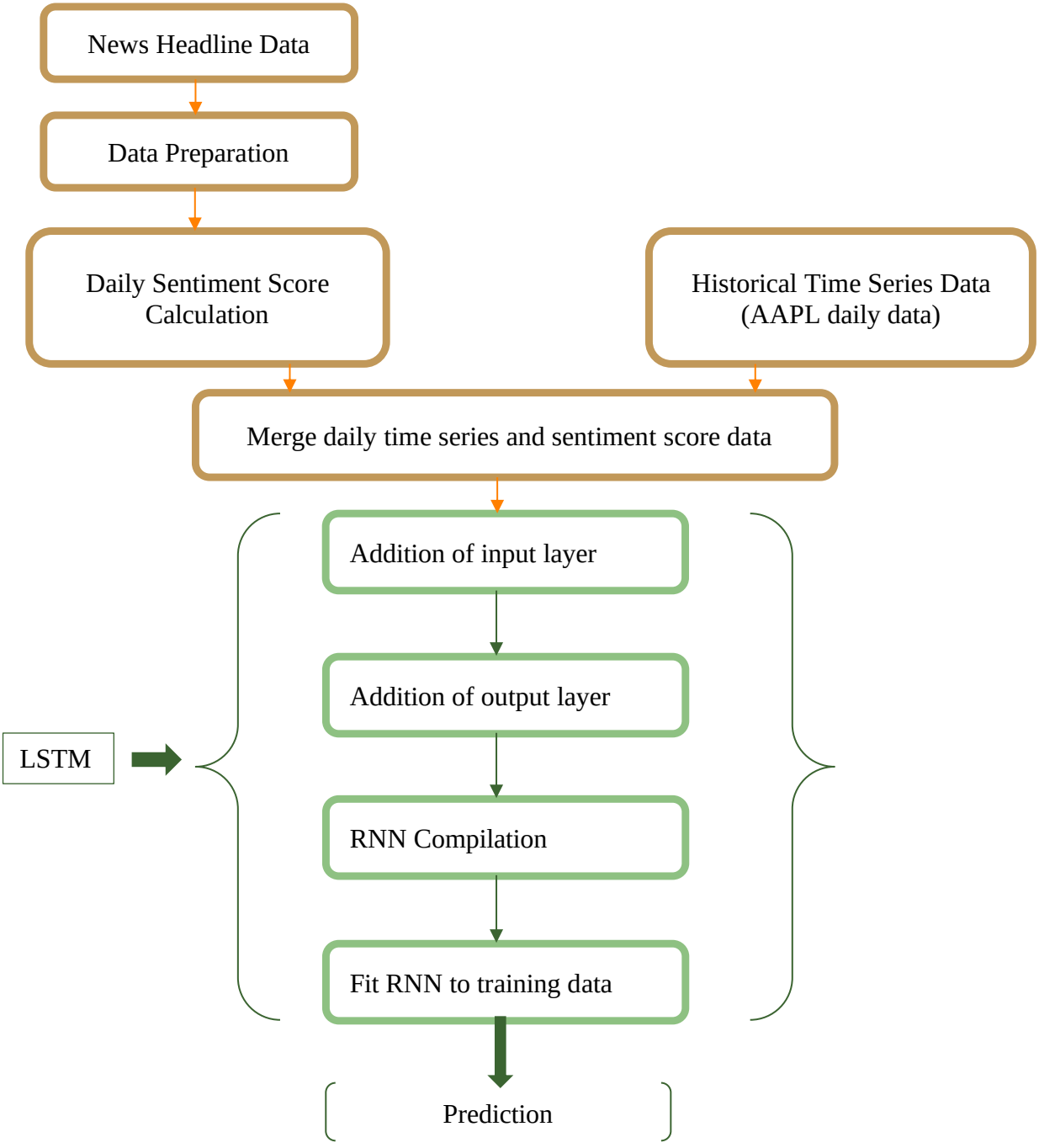
Khedr [39] used data mining techniques such as the K-NN algorithm and sentiment analysis to predict the stock market trend. The architecture of this system is dissected into two segments: the numeric data analysis component and the sentiment analysis component. The numeric data analysis component discretises the numeric value and then assigns the labels to each data instance. In the discretization process, the "positive", "negative", and "equal" attribute value is created. "Positive" attribute value is given when the current closing price is greater than the previous closing price. "Negative" attribute value is given when the current closing price is less than the previous closing price, and "Equal" attribute value is given when the current closing price and the previous closing price are equal. After that, the label of "rise" and "fall" is given to each data instance based on comparing their closing prices. In the sentiment analysis component, text processing is performed on the news data. This process includes tokenization,

data standardization, stop word removal, stemming, abbreviations processing, and token filtration. After that N-gram technique is used for feature extraction purposes. TF-IDF is used to calculate the frequency of the terms in the news corpus and then extract the more important terms. And then, the Naïve Bayesian classifier is used to classify the positive news and negative news sentiment. Finally, the results from the numeric data analysis component and sentiment analysis component are combined and fed into the K-NN algorithm to perform the prediction.

This section discussed the literature review, which used the RNN LSTM model, then came another literature survey using the CNN LSTM model. Then comes another literature that combined the RNN and CNN architecture and improved the prediction model accuracy. Then we discussed the literature that combined the ensemble methods with the deep learning method to improve the prediction further. This proves that the deep learning model is a pioneering technique for understanding the complex sequence of financial time series data.

CHAPTER 3: OBJECTIVES, SPECIFICATIONS AND DESIGN

System Architecture Design (Fig 3.1):



3.1 Research Objective and System Design

As our main objective for this research is to predict the financial time series using sentiment analysis. The architecture adopted in this research is dissected into two parts: Sentiment Analysis and the LSTM prediction model. The sentiment analysis consists of calculating the daily sentiment score from the news headline data. The LSTM model involves financial time series prediction with the extra input of news sentiment score data. The motivation to adopt the LSTM model was taken from this research paper [30].

News Headline Data: News headline data is collected from this site [33]. The dataset contains about 12.5 million news headlines from different domain web pages such as sports, entertainment, business, and finance and from 35 various news sources. This headline dataset is collected over three years from Jan-2014 to Dec-2016 at Computer Science & Engineering Department, Indian Institute of Technology, Roorkee. These resources are subject to a CC-BY 2.5 IN license [1], [2].

Historical Stock Data: This dataset contains the historical apple stock daily time series data. The data is taken from Yahoo Finance [34]. This is daily data from 2014 to 2016.

Data Preparation: The dataset collected from this site [33] is a Mega SQL table. These consist of news headlines, the start timestamp, the end timestamp, the URL, the source data. There are three different Mega SQL files. The first SQL file consists of news headlines from 2014, the second consists of news headlines from 2015, and the third SQL file consists of news headlines from 2016. The python script is used to split up this Mega SQL file into small SQL files and then execute each insert statement within these small SQL files. The split is performed in order to make the data dumping task easy. Moreover, another python script tries to dump all the data inside the small SQL files into my local MySQL database. MySQL database is used for storage purposes.

Daily Sentiment Score Calculation: This step involves the daily sentiment score calculation of each news headline data. A pre-trained VADER Sentiment Analyzer is used for the sentiment score calculation. Before the sentiment score calculation, the daily sentiment trading time is calculated. The goal is to ignore news headlines during the trading hour, and these news headlines will also be not examined for sentiment score calculation.

LSTM Model Prediction: The daily sentiment score and the historical daily stock time series data are then fed into the RNN LSTM model, and the forecasting of the time series data is made. The input layer is added into the neural network model using the training dataset of stock price, and after that, the output layer is added to the neural network. Since this research has used a Recurrent Neural Network model as architecture, we compile the RNN compilation is performed. After the RNN compilation, this RNN-LSTM model has fitted to the training dataset of time series data and the news sentiment data. Then the financial time series prediction is made. The accuracy of the prediction model is calculated using the root mean square error estimate. After looking at the error estimation value, the decision is made whether the model has made the accurate prediction or not.

CHAPTER 4: METHODOLOGY AND IMPLEMENTATION

The research methodology and implementation section extensively describe the methods and implementations taken to solve the problem. It also describes the steps that have been taken in order to implement the methodology. In this section, along with the methodology and implementation, we will also discuss the strengths and limitations of the methodology that has been adopted in this research study.

4.1 Data Collection

Historical News Headline Data: As discussed in the above section, News headline data is collected from this google site [33]. The dataset contains about 12.5 million news headlines from various web pages such as sports, entertainment, business, finance, and various news sources. This headline dataset is collected over three years from Jan-2014 to Dec-2016 at Computer Science & Engineering Department, Indian Institute of Technology, Roorkee. These resources are subject to a CC-BY 2.5 IN license [1], [2]. This is secondary data that were already gathered by researchers Mazumder, Bishnoi, and Patel subjected to this research study [1] while researching the usefulness of the news headline data in the applications dealing with time-aware news analytics. They [1] gathered these news headline data using their custom-made news crawler. The news crawler wriggles throughout the pre-determined list of 87 news websites and collects the news headline data from these websites at every 30 minutes time interval. The news headlines datasets from this site [33] are stored along with their timestamps as an SQL insert statement within the SQL file. Three separate SQL files contain the news headline data in a Mega table from 2014, 1015, and 2016 respectively. The Mega table has six columns: id for the headline, column for news headline data, start timestamp and end timestamp column for each news headline, URL, and source id.

Historical Stock Price Data: This dataset contains the historical apple stock daily time series data. The data is taken from Yahoo Finance [34]. This is daily data from the year 2014 to 2016. Yahoo Finance also works as a search engine. It provides the historical data for almost all the companies listed on all the exchanges in this world. Many researchers and students have used it as it provides historical data for free. It also provides API functionality. You can extract the historical data for multiple stocks optimally through a python programming language, but since this research only requires single stock historical data, this research's API functionality is not used; instead, manual download is performed. The stock that is considered in this project is an Apple Inc. company. The historical data consists of OHLCV data. The OHLCV is an acronym for Open, High, Low, Adj. Close and Volume data that are recorded concerning the multiple timeframes. A daily timeframe is chosen for this research study. This research only uses the Adj. Close price column for forecasting.

4.2 Data Preparation, Cleaning and Pre-processing

The dataset collected from this site [33] is a Mega SQL table, as we discussed above. These consist of news headlines, the start timestamp, the end timestamp, the URL, the source data. There are three different Mega SQL files. The first SQL file consists of news headlines from 2014, the second consists of news headlines from 2015, and the third SQL file consists of news headlines from 2016. The python script is used to split up this Mega SQL file into small SQL files and then execute each insert statement within these small SQL files. The split is performed in order to make the data dumping task easy. The built-in open () function and the built-in operation system O.S. module was used for file management purposes.

Furthermore, another python script tries to dump all the data inside the small SQL files into my local MySQL database. MySQL database is used for storage purposes. This python script uses the Mysql Connector library and the built-in O.S. module. 'mysql.connector.connect()' function is used to establish a connection with the local MySQL database server. This script iteratively read all the split SQL files and executes all the insert statements inside the split SQL file into my local MySQL database. At this stage, I have all the news headline datasets in three separate database tables (separate table for 2014, 2015, and 2016 data) inside my local MySQL database.

Using the Pandas data analysis framework and sqlalchemy library, the news headline data and other necessary columns were transferred from the MySQL database into the Pandas data frame data structure. Three separate Pandas data frames were used to store the news headline dataset from 2014, 2015, and 2016. After that, all the above three Pandas data frame was merged into one big master data frame. At this stage, the news headline dataset from the years 2014, 2015, and 2016 were successfully gathered and stored inside this master data frame for further data analysis and pre-processing data procedures. The total number of rows after merging all three datasets was around 12.5 million rows.

Since we focused on single Apple Inc. stock, we filtered the above dataset to extract all the rows that contained the news headlines that only talked about Apple Inc. company. In order to perform the data filtration for Apple Inc. stock, the data frame was divided into chunks of sub-dataframes, each of 10000 rows. This was done to basically optimize the data filtration process, as iterating through the 12.5 million rows did not seem to be optimal. While iterating through each chunk, the removal of missing values was performed, and the transformation of each news headline into lowercase was performed to ease the further filtration process. While iterating through the chunks, we filtered all the rows that either contained 'aapl' or 'apple' as a substring to filter out all the rows about the Apple Inc. company. This filtration further reduced our data to around 80251 number of rows. The duplicate news headlines were found when the data were inspected. Then the rows contained the duplicate news headline. After that, we filtered out the necessary columns from the data frame, such as the news headlines and timestamps of each headline for further feature engineering purposes.

Pandas data analysis library for data pre-processing: Due to the nature of this research project which involved a tremendous amount of data processing, cleaning, and pre-processing, the famous Pandas data analysis library was used. One of the robust features of the pandas is that it saves them time and let one focus more on the data analysis process, as many of the data processing and handling part is already implemented inside the library, so achieving more functionality with less code. It also works well with other financial APIs as well as other tools

and data analysis libraries. Pandas help you with their massive data frame data structures to better represent the sizeable two-dimensional dataset. It helps in performing the SQL-like data analysis process. Because it stores the data in the data frame, which further resides in the computer memory, and because the computer memory (RAM) is known for its fast read-write procedures, it makes pandas a tremendous tool to perform read-write and data munging procedures optimally. Since this research project involved a time series dataset, there was no doubt to choose pandas for analysing these time-series datasets [40].

4.3 Using State of the Art Technique: VADER

There are different approaches to calculate the sentiment score. Based on the different types of research, the sentiment score calculation approach is categorized into two parts: The lexical-based approach and the Machine learning-based (such as BERT Model) approach. Various studies have talked about the machine learning approach and lexical based approach. One such study is this paper [41], which talks about a survey regarding machine learning and a lexical based approach to categorize the sentiments. [42] discusses the approach that we should adopt for sentiment calculation. It explains that the approach selection is determined by the number of aspects, for instance, data availability and in which domain we are looking? For example, the stock market, healthcare sector or insurance sector and so on. In the machine learning approach, the machine learning classifier models predict the sentiment score from the text data. First, the machine learning model is selected, then the feature extraction is performed on the input textual data, and then these features are fed into the machine learning model in the training phase. After the model is the training phase. After the model is trained using the textual training datasets, the new text data is then fed into this model to predict the polarity score. Various algorithms are used to perform the sentiment analysis, such as Naïve Bayes, Linear Regression, Support Vector Machines, Deep Learning. Naïve Bayes algorithm allocates the probability of positive and negative sentiments for the text. Linear regression compares the words and the corpus with their corresponding polarities and then learns from it. Then it predicts the polarity of the new textual data that is fed into it as testing data. In the deep learning model, the text data is fed into the neural network training model, and then the sentiment prediction is made by this model, and then the model is tested against the testing datasets [43].

As a training dataset, we need the texts such as tweets and news headlines as a feature and their corresponding compound score as a label to feed into the training model. However, the compound scores are not readily available for any tweet's datasets or the news headlines datasets to train the model. This lexical based approach tackles this problem. In the lexical based approach, a dictionary is created that consists of thousands of lexicons and each lexicon is allocated a sentiment score. After that, the compound score is calculated with the help of this dictionary of lexicons [42].

One such method is called VaderSentiment, which is a python class inside the VADER python package. It contains the dictionary of the lexicons. For instance, this paper [44] talks about the VADER approach's performance compared to other lexical-based approaches and found that VADER calculated the compound score with 99% accuracy, which is better than the other methods. Additional customized lexicons can also be added to the lexicon dictionary [42]. Due to this motivation, I ignored the machine learning-based (such as the BERT model) approach. In this research project, I adopted a lexical-based approach, VADER, to calculate the sentiment class and score.

[42] discusses VADER sentiment score calculation through the following example. The output from the VaderSentiment class consists of four types of sentiment score: negative score, neutral score, positive score, and the compound score. The first three scores are basically normalized in the range [0, 1], and the compound score is calculated with the help of all lexicons in the text and normalized in the range [-1, 1]. In the process of these scores' calculation, the VADER first get the scores of all the words in the text from the lexicon dictionary mapping and then add all the scores of the negative, positive, and neutral sentiments, and after summing them all, it normalizes the total score in the range [-1, 1]. The formula looks like this:

$$Positive\ Score = Positive\ Sum + Number\ of\ words\ with\ positive\ scores$$
$$Negative\ Score = Negative\ Sum + Number\ of\ words\ with\ negative\ scores$$
$$Neutral\ Score = Neutral\ Sum + Number\ of\ words\ with\ neutral\ scores$$

$$Positive\ (Normalized) = \frac{Positive\ Score}{Total\ Score}$$
$$Negative\ (Normalized) = \frac{Negative\ Score}{Total\ Score}$$
$$Neutral\ (Normalized) = \frac{Neutral\ Score}{Total\ Score}$$

When it comes to compound score calculation, the sum of all the scores of the text gets passed into this kind of function to get the normalized score between [-1, 1][42]:

$$y = \frac{x}{\sqrt{x^2 + \mathrm{k}}}$$

Another advantage of using VADER is that it also gives more emphasis on the intensity of the text. It analyses the sentiment intensity using capitalisation, Intensifiers, Negative lexical features, Contrastive Conjunctions, and Punctuation. Capitalization entails emphasizing the sentiment, which in consequence updates the compound score. Intensifier entails the adverbs used in text such as "marginally", "particularly", and "increasingly", and so on. The negative lexical feature emphasizes the negative words used in the text, which inverts the compound score of the text. Contractive conjunctions make the compound score shift accordingly due to the use of words such as "but", "however", "at least", and so on. Punctuation, like multiple exclamation marks, question marks, also affects the compound score. These advantages led me to use the VADER for this research project [42].

4.4 Feature Engineering for Sentiment Data

This section talks about the feature engineering steps taken to calculate appropriate daily trading timestamps for each news headline and daily sentiment scores.

4.4.1 Calculating daily trading timestamp for each news headlines

In this part of the research methodology, we have prepared the trading timestamp of all the AAPL (Apple Inc.) stock news headline data. We tried to collect all the news headlines between yesterday's closing trading time and tomorrow's opening trading time. While doing this, we ignored all the news headlines during the trading hours for the sentiment calculation. This was achieved by iterating through the news headlines and creating the appropriate market open time for each headline. We calculated the four timestamps through each iteration. For each day, the

calculation of today's opening time, today's closing time, yesterday's closing time, and tomorrow's opening time. After calculating the above four timestamps, the conditional check was performed. If the news headline falls between yesterday's market close time and today's market open time, this trading time column will have today's market open timestamp. In the same way, if the news headline falls between today's market close time and tomorrow's market open time, then this trading time column will have tomorrow's market open timestamp [45].

4.4.2 Calculating Daily Sentiment Score.

In this section, the average value of the sentiment data on a given day was calculated. This average value represents the aggregated mean sentiment score for that day. The calculation of the average sentiment score was done by taking the mean of all the sentiment scores for that particular day. The aggregation of all the news sentiment data after 4:00 pm and 9:30 am was made, and then the average value was calculated accordingly [45].

4.5 Prediction Model Selection

This section recalls some of the machine learning and deep learning model literature reviews reviewed in the above literature review section to help back up why a particular model and architecture has been used in this research project.

In the above literature review, we discussed how Tiwari [17] used the neural network model in their research for stock price prediction. They used the models such as Regression, Time Series Model, Neural Network Model and compared the results of each model. After the comparison, they concluded that the feed-forward neural network model predicted the prices better than the other models.

To tackle the non-linear property of the stock price, we reviewed that Abhishek [20] presented a stock market forecasting model based on the artificial neural network with multi-layered feed-forward architecture to forecast the financial time series data. They used this model for its capability to postulate unseen data such as stock price data. We also reviewed that Qin [22] used a neural network model for their stock price prediction research and found them as a potential architecture in terms of short-term stock price prediction. Moreover, we reviewed that Du [23] used the back-propagation neural network model and the ARIMA model and observed that the BP neural network model had better accuracy in terms of stock price prediction than the ARIMA model. Later they also found that the combination of BP Neural Network and ARIMA model achieved greater prediction accuracy than using them individually.

Later we also recapitulated the research paper of Wijaya [21], which compared the accuracy of the ARIMA and artificial neural network model. The paper also mentions the drawbacks of using the ARIMA model, such as stationarity, autocorrelation, normality of the error term and how artificial neural network overcome those hurdles.

By this time, we have realized that the performance of the deep learning model is better than the other models. Then we tried to review the literature, which also incorporated the news sentiment data and the stock price data as an input to the deep learning model for the stock price prediction. For this, we started reviewing this research paper [30] that applied a deep learning model to predict the stock price data using news sentiment data. They applied a couple of different machine learning models such as ARIMA, Facebook Prophet, and the RNN LSTM model for the prediction. After testing these models, they found that the RNN LSTM model

had better prediction accuracy than other models. On the other hand, we reviewed this research paper [31] which used the CNN LSTM prediction model for stock price prediction. The input for these models was the stock price data and textual news sentiment data. The model prediction accuracy seemed to be enhanced with event embedding and numerical data into their model.

Furthermore, reviewed this paper, Vargas [32], which conducted a price time series pattern recognition study using a deep learning model which used the combination of RNN, and CNN architecture known as the RCNN model. Besides that, this research paper [35] that reviewed in the literature review adapted a deep learning model architecture for stock market prediction with the input of tweets embedding data and historical prices. Another research that we reviewed [37] predicted the stock market price using an attention-based deep learning model with stock prices and investors' sentiment data as an input.

Reviewing the above research papers enabled this research project to adopt a deep learning model with Long Short-term Memory (LSTM) cells under Recurrent Neural Network (RNN) architecture using AAPL news sentiment data and AAPL stock price data as an input.

4.6 LSTM RNN Prediction Model

This section starts with the description and introduction of the Long Short-Term Memory (LSTM) and Recurrent Neural Network (RNN) Architecture concept.

RNN is used to exhibit past sequence behaviour of inputs where preceding outputs are used as input. It is one of the types of ANN (Artificial Neural Network) architecture that uses time series data as an input. The sequence modelling problem can be designed using one-to-many, many-to-one, and many-to-many methods.

RNN (Fixed length memory model): The model predicts the next value from the past sequences of values. In terms of finance, it can be framed as, given a sequence of historical stock prices and the current day's news sentiment data, we want to predict the next day's stock price value. In the stock price prediction case study, since the stock prices have a specific pattern, we need large sequences of historical time series data rather than most recent data to gauge the pattern accurately. So, the larger the window size of the past data, the better the model can gauge the seasonal pattern and thereby better predict the stock price value [46].

RNN (Infinite Memory): This model aims to look at much earlier datasets to predict the next value. Such a model consists of 'Infinite Memory' of sequences. The model requires the maintenance of the model's state at various timestamps and to propagate those states as we move forward the timestamps. This neural network architecture can maintain the information of all the previous timestamps that it has traversed. The architecture consists of inputs, hidden layers, and neurons. In this, the input gets fed into the activation function, and the activation gets further fed into their neurons and following activation function as a hidden layer. Each of them produces its respective outputs. These hidden layers of memory transfer make the recurrent neural network model maintain the model's state [46].

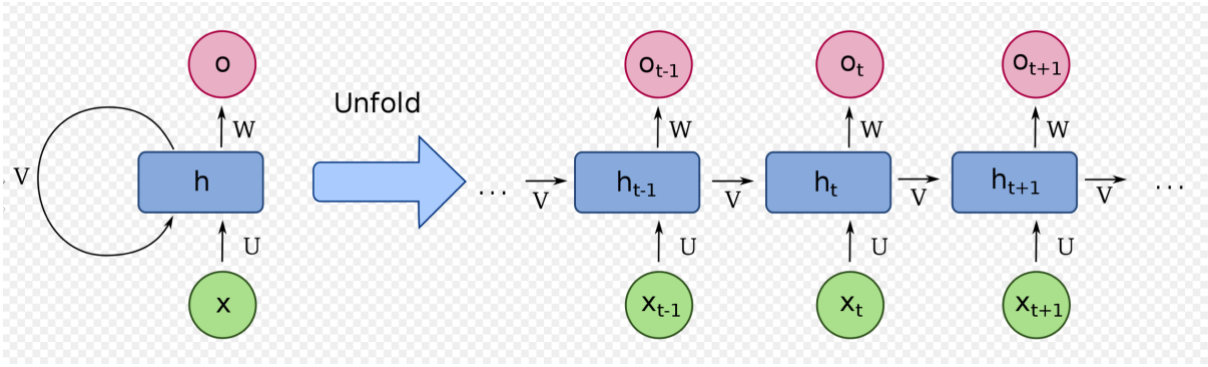


Fig 4.1: Unfolded Recurrent Neural Network Architecture
(Source: Wikipedia [47])

RNN Model (Weight Sharing): In a particular layer, we have several neurons that accept X = [x1, x2, x3….] as an input feature with their respective weights W = [w1, w2, w3…], which outputs the activation value. There are several types of weights associated with this architecture. There is a weight for the input feature, weights for activation value, and a weight associated with the previous activation value shared with the following hidden layers. Same weights are shared across this recurrent architecture. Overall, each input in this architecture carries all the memory of the previous inputs. In this way, the weights are shared across different unrolling of the same neural networks [46].

4.7 Performing the Prediction Using News Sentiment and Stock Price Data

For the model prediction, we have segregated the combined stock price and news sentiment dataset into three parts: training set, cross-validation set, and test set. The dataset is sorted according to the DateTime index. The prediction model predicts the future stock price with the combination of sequential input of news sentiment data of the current day and stock price data of the past 40 days using Recurrent Neural Network architecture with LSTM cells. We have used news sentiment score as an Apple Inc. stock news sentiment data and adjacent closing price of Apple Inc. stock price data along with the DateTime index timestamp.

In order to build a robust model, we have segregated the dataset into three parts: training set, validation set, and testing set.

The training dataset is mainly used to train the model. In this phase, the weights in our neural network model get amended during the back-propagation stage. The aim is to fit the model and generalize the model to perform well on the unseen dataset. The datasets from the validation set and testing set should be abstained to be used under the training phase to avoid the overfitting of the model. A large portion of the dataset is reserved for training purposes. We have allocated 64% of the total dataset for the training set. Most of the learning of the model from the data is done through a training dataset [48, 49].

The validation dataset, as the name suggests, is used for validation purposes. The model learned through the training dataset gets validated on these validation datasets to avoid overfitting. The validation dataset is like a hybrid dataset on which you neither train your model nor perform the final testing of the model. It is just used for validating the performance of the model learned through the training dataset. We can calculate the error function of the model learned through the training set to validate if the model has learned correctly through the validation set. This

dataset is entirely independent of the training dataset. We have allocated 16% of the dataset for the validation set. Although we have allocated some portions from the training datasets for the validation set, we have only used this dataset for validating and not for the training purpose [48, 49].

The test dataset is an entirely distinct dataset, separately from the training dataset and the validation dataset. The aim is to perform the model accuracy measure using the test dataset to see how well the model can generalize on the unseen dataset. The test data is mainly used to measure the final model accuracy. There is minimum overfitting if a model fits right to both the training as well as the testing dataset. So basically, it is used for impartial assessment of the prediction model. The size of the test dataset is usually tiny. We have allocated 20% of the dataset for the testing set [48, 49].

Feature Scaling: Since the model in this research project uses recurrent neural network architecture, it uses a gradient descent-based algorithm for optimization purposes. This demands that the data be scaled. Here we have used MinMaxScaler class from the scikit-learn python package. This scales the data in the range of [0, 1]. This is also called the normalization scaling technique. The MinMaxScaler formula looks like this [50, 51]:

$$X' = \frac{X - Xmin}{Xmax - Xmin}$$

After feature scaling, we prepared our dataset of the stock price of window size 62 and the news sentiment dataset of the last day. After that, we converted the above-prepared dataset into a three-dimensional tensor to handle batch gradient descent. Then we used a sequential model. A sequential model is a pile of layers. The layers are the input layer, hidden layer, and output layer. The sequential model is created by using an 'add' function [52]. We have added an LSTM cell with specific numbers of units. This is the LSTM as an input layer. The activation function used for this LSTM cell is the hyperbolic tangent function. This activation function is usually applied to neural networks. Its value ranges from -1 to 1. The equation of tanh:

$$\tanh = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

The hyperbolic tangent function is a scaled version of the sigmoid function, making it have the same property as a sigmoid function. However, the hyperbolic tangent function is usually considered to be better than the sigmoid function. Since the stock is considered to have a non-linear property, the hyperbolic tangent function (tanh) is non-linear, making it a perfect activation function [53, 54].

Then we set the output layer and train the model by fitting the model with the training dataset using the model.fit(). Along with that, we also pass the validation dataset to rectify our model. After training the model under the recurrent neural network architecture, we test our model using the test dataset. Through this process, we get our predicted stock prices. However, these predicted stock prices are in the scaled format. In order to scale this predicted data back to its original values, we used the inverse_transform() function. This function was applied to the MinMaxScaler algorithm that we used earlier. After rescaling back to the original form, we finally got the predicted stock prices.

CHAPTER 5: RESULTS, ANALYSIS AND EVALUATION

In this section, the summary of the results obtained from the proposed design and methodology has been made along with the procedure to obtain those results. A comparison between different models is made.

5.1 Analysis of the Financial Time Series and News Sentiment Data

After combining the apple stock news sentiment score data with the Apple stock price data, we analyzed the number of positive sentiment scores and negative sentiment scores. Also, we analyzed the trend of the stock price. The below graph shows the closing price of the Apple stock price from 2014 to 2016. The y-axis represents the apple stock price value, and the x-axis denotes the dates.

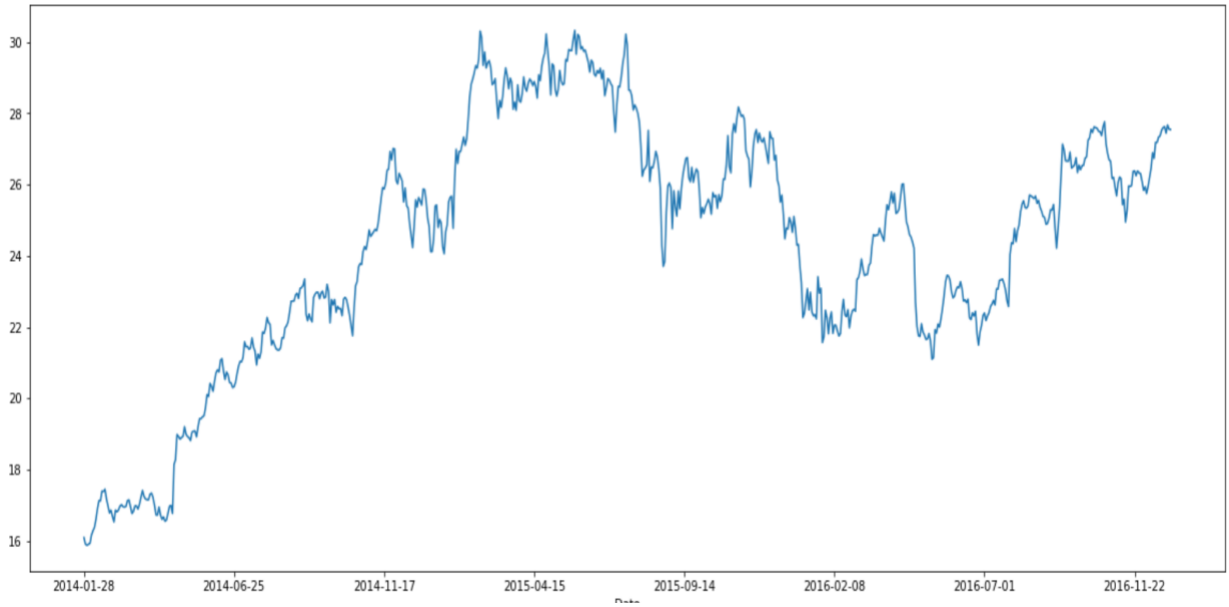


Fig 5.1: Apple Inc. stock Adj. Closing Price

According to the analysis, the positive sentiment count (450) is higher than the negative sentiment count (266). We can see how this influences the Apple stock price, which we can see is steadily increasing. However, this does not increment sharply. Basically, It is a series of positive trends (from 2014-01-28 to 2014-11-17), neutral trends (from 2014-11-17 to 2015-07), decreasing trends (from 2015-07 to 2016-05), and again positive trends (from 2016-05 to 2016-11). Most of the time, the combination of positive and negative trends can be seen. Having a quick look at the line graph and probability distribution will confirm this price trend.

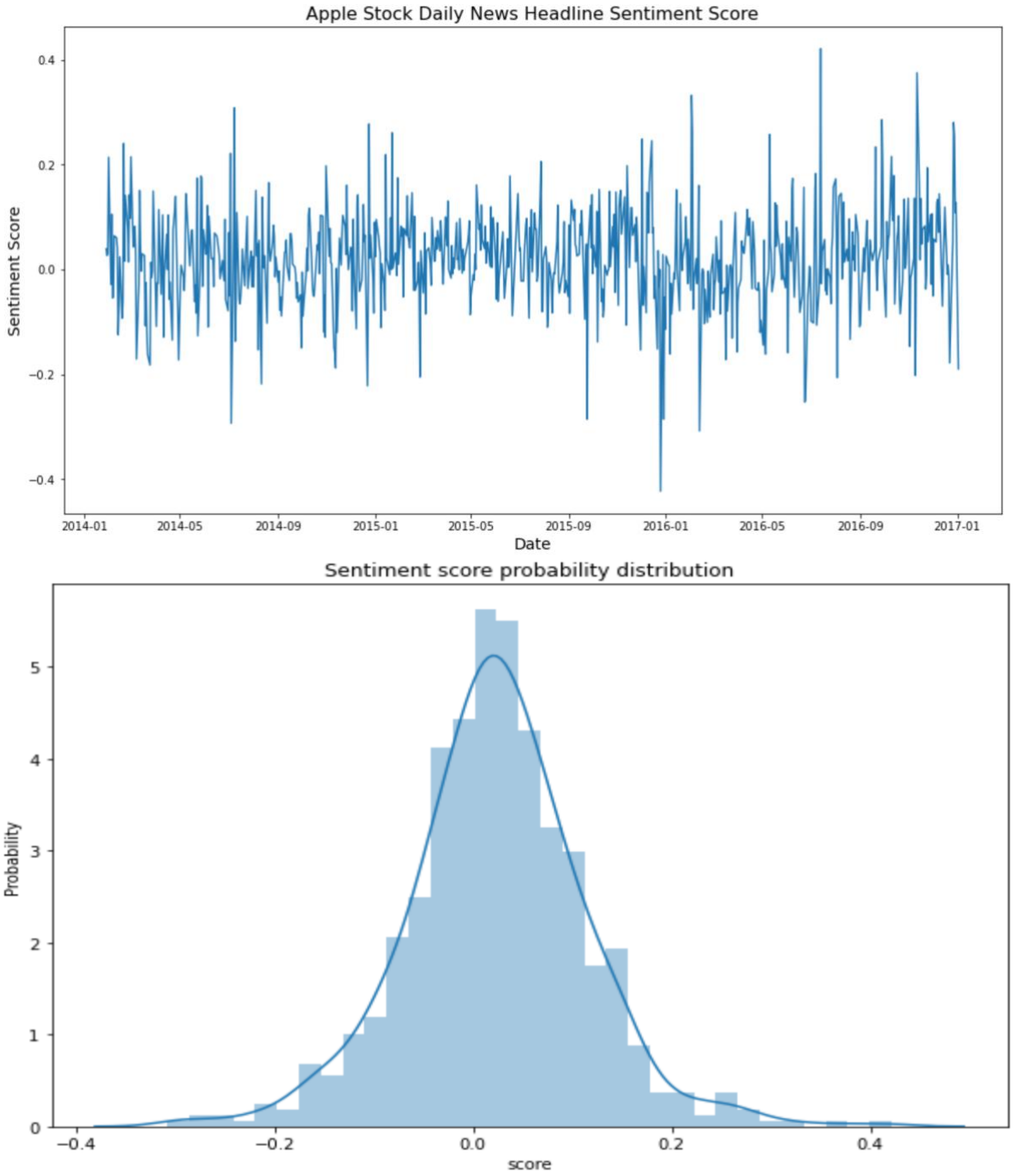
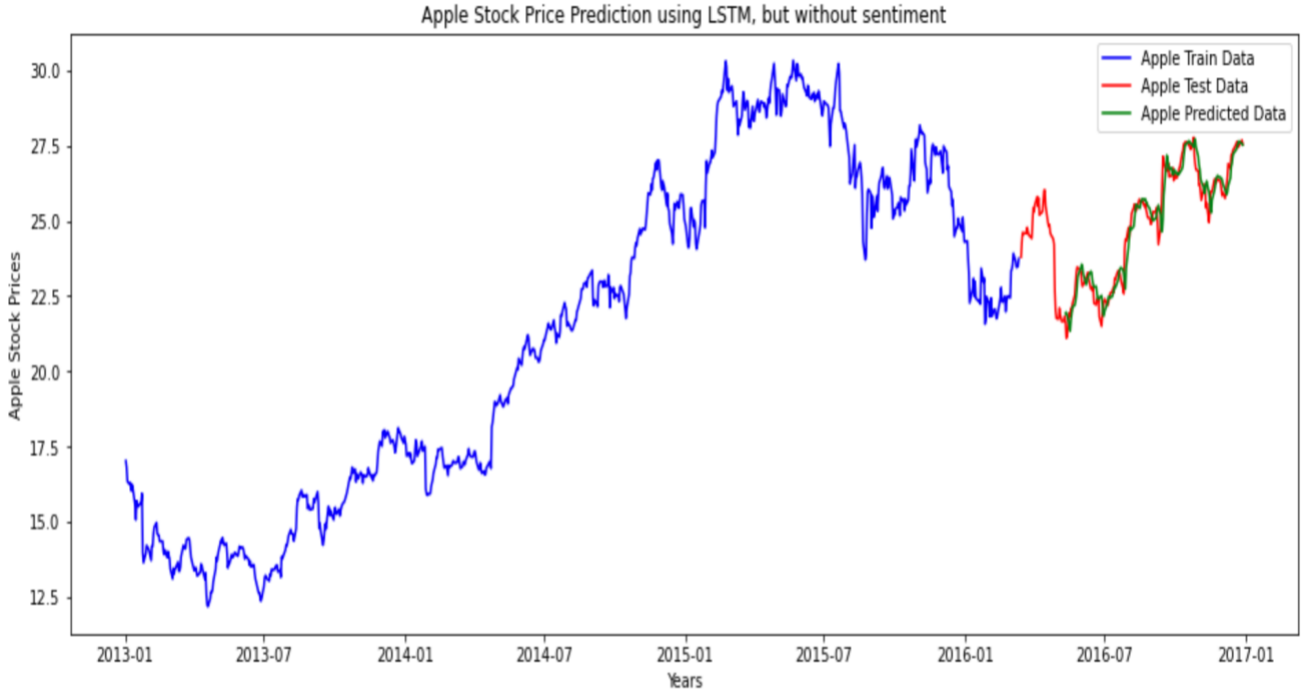


Fig 5.2: Daily Sentiment Score and its Probability Distribution

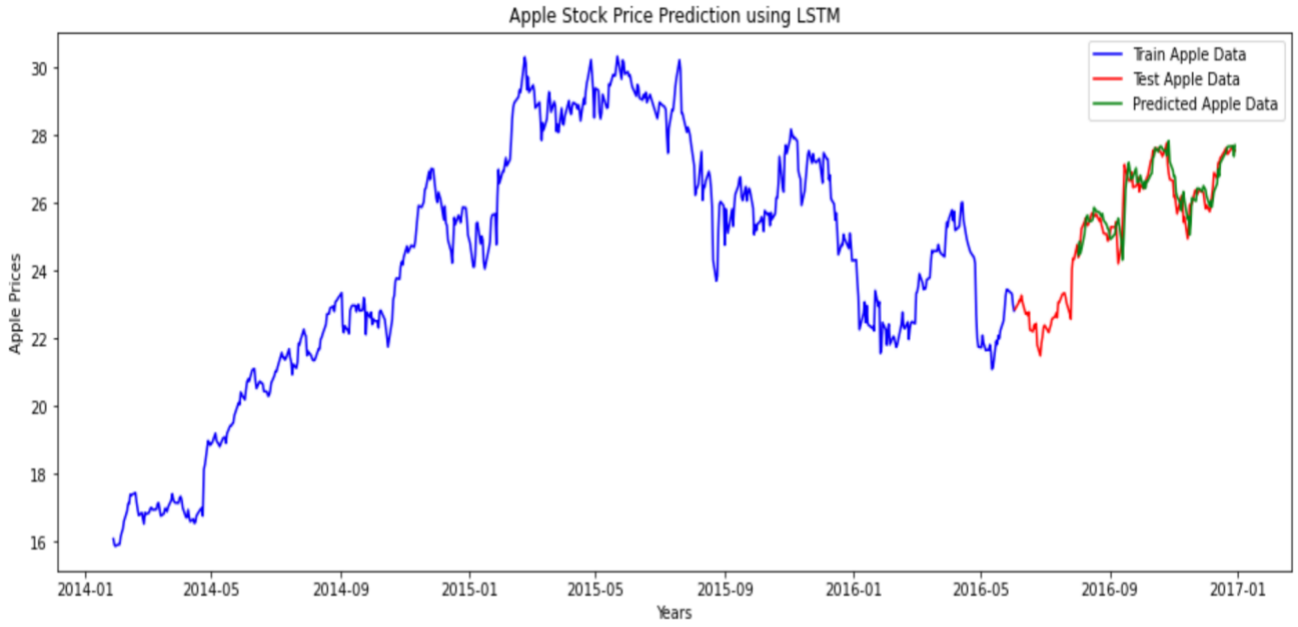
The sentiment score calculated by the VADER correctly captures the sentiment, which is quite a mixture of positive and neutral sentiment but marginally inclined towards the positive side, as we can see it has shifted somewhat to the right. However, overall the sentiment is entirely at an intersection of the positive and neutral sentiment (as illustrated by the above chart).

5.2 Comparison of LSTM without sentiment vs LSTM with sentiment Data.

In this section, we compare the LSTM RNN prediction model using RMSE as a performance metric.



RMSE_LSTM = 0.48



RMSE_LSTM = 0.43

Fig 5.3: Comparison of LSTM RNN Prediction model without vs with sentiment data.

As we can see, the first chart depicts the Apple stock price prediction using only the LSTM RNN prediction model. On the other hand, the following chart below represents the Apple stock price using the combination of the Apple Inc. stock news sentiment score and the Apple stock price data into the LSTM RNN prediction model. From a glance, it appears that the RMSE has reduced to 0.43 from 0.48. Sentiment from News has helped LSTM RNN to improve prediction even further. This model combines the Apple stock price data from the last 40 days with Apple Inc. News Sentiment Compound Scores which was calculated using VADER for the last day to predict the Apple stock price for the next day.

We have used Root Mean Square Error (RMSE) as a performance metric. RMSE is also called root mean square deviation. The method is used to calculate the spread between the actual observed value and the predicted value. It is used for estimating the model accuracy. RMSE is always a positive number due to the nature of its formula (square root of the mean of the spread) [55]. It is generally assumed that a value between 0 to 0.5 is acceptable. The lower the value of the RMSE, the better the accuracy of the model. Since the RMSE value is 0.43 and 0.48, it denotes that the model accuracy is acceptable.

	LSTM RNN (without sentiment data)	**LSTM RNN (with sentiment data)**
RMSE	0.48	0.43

Table 5.1 RMSE value comparison for LSTM RNN (with vs without sentiment data)

5.3 Hyperparameter tuning using Grid Search Technique

Number of neurons	**Rate of Learning**	**Number of Epochs**	**Number of batches**
115	0.01	44	8
120	0.03	46	12
125	0.06	48	16
130	0.09	50	32

Table 5.2 Table for Hyperparameter Values

In the process of hyperparameter tuning, I tried to find the best possible value for the hyperparameter that yields the better prediction of the financial time series data. The Hyperparameter that I chose for the LSTM RNN prediction model were units, learning rate, number of epochs, and number of batches. After the hyperparameter optimization, we found that the above green-colored highlighed hyperparameter values slightly improved the prediction accuracy to 0.422099 from 0.43.

The aim of the project has been met. We successfully predicted the Apple stock price data using the Apple Inc. stock news sentiment data and demonstrated the result through data visualization and Root Mean Square Error estimate (for prediction accuracy). Moreover, we observed that the ability of the model prediction got enhanced when we introduced the news sentiment data into our model. Then Hyperparameter tuning further helped us to slighlty improve the prediction accuracy and to better tune the model.

CHAPTER 6: LEGAL, SOCIAL, ETHICAL AND PROFESSIONAL ISSUES

The data used in this research project is freely accessible in the public domain. The dataset that has been used in this project contains data regarding news headlines that have already been collected and made publicly available through different web sources of about 35 news sources. The resources are subjected to CC-BY 2.5 IN licence. Proper references have been provided in the references section with regards to the data collection. A proper source of the data has been mentioned in the references section.

CHAPTER 7: CONCLUSION

This chapter provides the vital points and briefly discuss our results in this research project. The chapter provides information about the steps undertaken to solve this research paper's research question as a final thought and concludes by providing information about future work and development.

7.1 Final Thought

In this research paper, we compared the prediction accuracy of the LSTM RNN model without the sentiment data and the LSTM RNN model with the sentiment data. We used Root Mean Square Error (RMSE) as a performance metric. The RMSE of the LSTM RNN model without the sentiment data was found to be 0.48, while the RMSE of the LSTM RNN model with the sentiment data was found to be 0.43. This showed that the prediction accuracy of our model got enhanced after introducing the news sentiment data into our model, which further proves that the news does affect the stock market prices. We can leverage this to make our stock price prediction model more accurate. We also analysed the news sentiment data and the financial time series data and found correlations between them. Further we also slightly improved the model with hyperparameter tuning.

After reading a number of literature reviews in the literature review section that utilized the combination of recurrent neural network and convolutional neural network architecture, RNN LSTM model, and another literature review that utilized the convolutional neural network with LSTM model, another literature that utilized both architecture for improved prediction performance, it was convincing to use deep learning model as a prediction model. Further reading through the literature proved that using ensemble learning methods combined with deep learning improved the prediction further. As a result, I felt that deep learning was a pioneering technique for explaining financial time series data. Additionally, Recurrent Neural Network Architecture attempts to predict the next value by looking at much earlier datasets. An infinite series of sequences makes up such a model. Maintaining the model's state at different timestamps and propagating those states continuously along the timeline requires the model's state to be maintained. Using this neural network architecture, it is possible to maintain the information of all previous timestamps. We incorporated this architecture for stock price prediction because this makes it an ideal architecture for stock market prediction.

We adopted a lexical-based approach (VADER) instead of a machine learning-based approach (the BERT Model) for sentiment score calculation. Due to the lack of availability of the sentiment score data, I rejected the BERT model and chose to adopt the lexical based using the VADER sentiment python library.

7.2 Future Work and Development

Apart from predicting the stock price data, since we have calculated our sentiment score from a lexical based approach, we can even now try to predict the sentiment scores with the available sentiment data using various machine learning models. We can apply the XGBoost tree-based model for the sentiment score prediction. We can also use neural networks for sentiment score prediction and then use those models for trading strategies to trade various kinds of financial instruments.

We can further use these models to analyse Twitter data sentiment analysis, as tweets closely resemble the news headline data. We can also combine various statistical learning with neural network algorithms for stock price prediction using news sentiment data. In terms of the calculation of sentiment score, we can use various other tools such as TextBlob, Nltk, and various other sentiment analysis tools. Regarding financial instruments, we can further apply sentiment analysis to predict various kinds of commodities since commodities prices are highly correlated with news events and investor sentiment. A commodity like oil prices is most highly correlated with news events.

As this paper [55] suggests, we can also apply the combination of machine learning and sentiment analysis to forecast the financial risks. As many researchers have done many kinds of research of financial risk forecast using traditional machine learning and statistical learning techniques, we can also enhance the financial risk forecasting performance by introducing sentiment analysis into our model, in the same way, that we used the sentiment analysis for our financial price forecasting in this research project. Financial volatility plays a crucial role in various asset pricing models; one example is options pricing. It makes it very critical to forecasting the financial risks. This paper tries to forecast financial volatility using financial news information with the artificial neural network model and support vector machine model. It also tries to find the relationship between the volatility sentiments and the financial instrument price volatility. The system takes the financial news volumes to form the Google Finance and financial trading volumes data from Yahoo Finance. Two approaches have been adopted: the GARCH model with artificial neural network approach and the GARCH model with support vector machine. The financial news volumes are fed into the GARCH-Artificial Neural Network model. The financial trading volumes data are fed into the GARCH-Support Vector Machine model for training these models. After the training of these two models, forecasting of the financial volatility is made. To capture the impact of online news information on the financial volatility value, sentiment polarity and intensity are calculated for each news.

Furthermore, we can also make a portfolio construction application based on the overall sentiment scores of all the financial assets in our portfolio and then maintain the top performers and losers assets in our portfolio, thereby taking a long and short position on these financial assets.

REFERENCES

[1] S. Mazumder, B. Bishnoi and D. Patel, "News Headlines: What They Can Tell Us?," in *Proceedings of the 6th IBM Collaborative Academia Research Exchange Conference (I-CARE) on I-CARE 2014 - I-CARE 2014*, New York, New York, USA, 2014.

[2] K. Gupta, V. Mittal, B. Bishnoi, S. Maheshwari and D. Patel, "AcT: Accuracy-aware crawling techniques for cloud-crawler," in *World Wide Web*, 2016.

[3] Quantra, "Sentiment Analysis for Trading," Quantra, QuantInsti, 21 August 2019. [Online]. Available: https://youtu.be/eleYknWuJXw. [Accessed 10 July 2021].

[4] G. Mitra and L. Mitra, The Handbook of News Analytics in Finance, Chichester, West Sussex, UK: John Wiley & Sons, Ltd., 2011.

[5] P. C. TETLOCK, "Giving Content to Investor Sentiment: The Role of Media in the Stock Market," *The Journal of Finance*, vol. 62, no. 3, pp. 1139-1168, 6 2007.

[6] M. Alanyali, H. S. Moat and T. Preis, "Quantifying the Relationship Between Financial News and the Stock Market," *Scientific Reports*, vol. 3, no. 1, p. 3578, 20 12 2013.

[7] J. Bollen, H. Mao and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science*, vol. 2, no. 1, pp. 1-8, 3 2011.

[8] P. C. Tetlock, "The Role of Media in Finance," in *Handbook of Media Economics*, vol. 1, New York, NY: Elsevier, 2015.

[9] P. A. Hafez, "How news events impact market sentiment," RavenPack International S.L., 2010.

[10] J. Engerlberg , A. V. Reed and M. C. Ringgenberg, "How are Shorts Informed? Short Sellers, News, and Information Processing," *SSRN Electronic Journal*, 17 March 2010.

[11] L. R. Mitra, G. Mitra and D. di Bartolomeo, "Equity Portfolio Risk (Volatility) Estimation Using Market Information and Sentiment," *SSRN Electronic Journal*, 2008.

[12] Zhang, W. &. S. and S. , "Trading Strategies to Exploit News Sentiment," 2010.

[13] S. Usmani and J. A. Shamsi, "News sensitive stock market prediction: literature review and suggestions," *PeerJ Computer Science*, vol. 7, p. e490, 4 5 2021.

[14] P. Tetlock, "Paul Tetlock: The Role of Media in the Stock," Columbia Business School, 11 April 2016. [Online]. Available: https://youtu.be/_3HMTgmNw3s. [Accessed 28 July 2021].

[15] P. Falinouss, "Stock trend prediction using news articles: a text mining approach," 2007.

[16] Pearce, Douglas K. and V. Vance Roley, "Stock Prices and Economic News," *The Journal of Business*, vol. 58, no. 1, pp. 49-67, 18 7 2021.

[17] S. Tiwari, A. Bharadwaj and S. Gupta, "Stock price prediction using data analytics," in *2017 International Conference on Advances in Computing, Communication and Control (ICAC3)*, 2017.

[18] A. A. Ariyo, A. O. Adewumi and C. K. Ayo, "Stock Price Prediction Using the ARIMA Model," in *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, 2014.

[19] S. Wichaidit and S. Kittitornkun, “Predicting SET50 stock prices using CARIMA (Cross Correlation ARIMA),” in *2015 International Computer Science and Engineering Conference (ICSEC)*, 2015.

[20] K. Abhishek, A. Khairwa, T. Pratap and S. Prakash, “A stock market prediction model using Artificial Neural Network,” in *2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12)*, 2012.

[21] Y. B. Wijaya, S. Kom and T. A. Napitupulu, “Stock Price Prediction: Comparison of Arima and Artificial Neural Network Methods - An Indonesia Stock's Case,” in *2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies*, 2010.

[22] J. Qin, Z. Tao, S. Huang and G. Gupta, “Stock Price Forecast Based on ARIMA Model and B.P. Neural Network Model,” in *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, 2021.

[23] Y. Du, “Application and analysis of forecasting stock price index based on combination of ARIMA model and B.P. neural network,” in *2018 Chinese Control And Decision Conference (CCDC)*, 2018.

[24] G. P. C. Fung, J. X. Yu and W. Lam, “News Sensitive Stock Trend Prediction,” in *Advances in Knowledge Discovery and Data Mining*, vol. 2336, Berlin, Heidelberg: Springer, 2002, pp. 481-493.

[25] G. Ramakrishnan, K. P. Chitrapura, R. Krishnapuram and P. Bhattacharyya, “A model for handling approximate, noisy or incomplete labeling in text classification,” in *Proceedings of the 22nd international conference on Machine learning - ICML '05*, New York, New York, USA.

[26] M. Tirea and V. Negru, “Text Mining News System - Quantifying Certain Phenomena Effect on the Stock Market Behavior,” in *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 2015.

[27] K. Keswani, I. Das, B. Shrivastava, A. Gupta and R. Katarya, “LDA Based model for mining textual features from financial news articles,” in *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*.

[28] H. Yun, G. Sim and J. Seok, “Stock Prices Prediction using the Title of Newspaper Articles with Korean Natural Language Processing,” in *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2019.

[29] J. L. Wu, C. C. Su, L. C. Yu and . P. C. Chang, “Stock price prediction using combinational features from sentimental analysis of stock news and technical analysis of trading information.,” in *International proceedings of economics development and research*, 2012.

[30] S. Mohan, S. Mullapudi, S. Sammeta, P. Vijayvergia and D. C. Anastasiu, “Stock Price Prediction Using News Sentiment Analysis,” in *2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService)*, 2019.

[31] P. Oncharoen and P. Vateekul, “Deep Learning for Stock Market Prediction Using Event Embedding and Technical Indicators,” in *2018 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA)*, 2018.

[32] M. R. Vargas, B. S. L. P. de Lima and A. G. Evsukoff, “Deep learning for stock market prediction from financial news articles,” in *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, 2017.

[33] D. Patel, J. Barua, B. Bishnoi and K. Gupta, “News Headline Collection,” [Online]. Available: https://sites.google.com/view/headlinedataset/home. [Accessed June 2021].

[34] “Yahoo Finance,” Yahoo Finance, [Online]. Available: https://finance.yahoo.com/quote/AAPL/history?p=AAPL. [Accessed June 2021].

[35] H. Ni, S. Wang and P. Cheng, “A hybrid approach for stock trend prediction based on tweets embedding and historical prices,” *World Wide Web,* vol. 24, no. 3, pp. 849-868, 22 5 2021.

[36] Z. Hu, W. Liu, J. Bian, X. Liu and T.-Y. Liu, “Listening to Chaotic Whispers,” in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, New York, NY, USA, 2018.

[37] Z. Jin, Y. Yang and Y. Liu, “Stock closing price prediction based on sentiment analysis and LSTM,” *Neural Computing and Applications,* vol. 32, no. 13, pp. 9713-9729, 19 7 2020.

[38] Y. Li and Y. Pan, “A novel ensemble deep learning model for stock prediction based on stock prices and news,” 2020.

[39] A. E. Khedr, S.E.Salama and N. Yaseen, “Predicting Stock Market Behavior using Data Mining Technique and News Sentiment Analysis,” *International Journal of Intelligent Systems and Applications,* vol. 9, no. 7, pp. 22-30, 8 7 2017.

[40] P. Asirinaidu, “Pandas for Data Analysis and their Benefits,” Medium, 29 June 2017. [Online]. Available: https://medium.com/@asirinaidu1990/pandas-for-data-analysis-and-their-benefits-891ec3e998ae. [Accessed 5 August 2021].

[41] H. Zhang, W. Gan and B. Jiang, “Machine Learning and Lexicon Based Methods for Sentiment Classification: A Survey,” in *2014 11th Web Information System and Application Conference*, 2014.

[42] “Trading Strategies with News and Tweets: Approaches to Calculate the Sentiment Score, Sentiment Score Using VADER,” Quantra, [Online]. Available: https://quantra.quantinsti.com/course/trading-twitter-sentiment-analysis. [Accessed 15 August 2021].

[43] R. Wolff, “Sentiment Analysis & Machine Learning,” MonkeyLearn, 20 April 2020. [Online]. Available: https://monkeylearn.com/blog/sentiment-analysis-machine-learning/. [Accessed 15 August 2021].

[44] C. Hutto and E. Gilbert, “Vader: A parsimonious rule-based model for sentiment analysis of social media text.,” in *Proceedings of the International AAAI Conference on Web and Social Media.,* 2014.

[45] D. T. Benzschawel, “Natural Language Processing in Trading: Sentiment Score and Strategy Logic,” Quantra, [Online]. Available: https://quantra.quantinsti.com/course/natural-language-processing-trading. [Accessed 16 August 2021].

[46] AI Sciences, “Deep Learning: Recurrent Neural Networks with Python (Chapter 4: RNN Architecture),” Packt, [Online]. Available: https://www.packtpub.com/product/deep- learning-recurrent-neural-networks-with-python-video/9781801079167. [Accessed 17 August 2021].

[47] “Recurrent Neural Network,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Recurrent_neural_network. [Accessed 17 August 2021].

[48] “Training, validation, and test sets,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Recurrent_neural_network. [Accessed 17 August 2021].

[49] "The Train, Validation, Test Split and Why You Need It," roboflow, [Online]. Available: https://blog.roboflow.com/train-test-split/. [Accessed 18 August 2021].

[50] "Min Max Scaler," Scikit Learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html. [Accessed 18 August 2021].

[51] "Feature Scaling For Machine Learning: Understanding the Difference Between Normalization and Standardization," Analytics Vidya, [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/. [Accessed 18 August 2021].

[52] "What is meant by sequential model in Keras," Stack Overflow, [Online]. Available: https://stackoverflow.com/questions/57751417/what-is-meant-by-sequential-model-in-keras. [Accessed 19 August 2021].

[53] "Understanding Activation Functions In Neural Networks," Medium, [Online]. Available: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0. [Accessed 19 August 2021].

[54] "Tanh Activation," Papers with code, [Online]. Available: https://paperswithcode.com. [Accessed 19 August 2021].

[55] Wu D.D., Olson D.L., " Financial Risk Forecast Using Machine Learning and Sentiment Analysis," in Enterprise Risk Management in Finance, Palgrave Macmillan, London, 2015.

[56] G. Mitra and Yu, Xiang, Handbook of Sentiment Analysis in Finance, London, OptiRisk Systems Ltd., May 2016.

APPENDICES: Source Code Listings

Code Listing 1: Data Preparation.ipynb

1.1 Split the mega SQL file into chunks of small SQL files.

```
# importing an os module
counter = 0
# Do it for NewsHeadlines-2014.sql, NewsHeadlines-2015.sql, NewsHeadlines-
2016.sql

with open(r"/Users/Harsh/Desktop/NewsHeadlines-2016.sql", encoding='utf8')
as f:
    for l in f:
        if l.startswith("INSERT INTO `Megatable`"):
            counter = counter + 1

        # Do it for 2014, 2015, 2016

        if
os.path.exists(os.path.join(r"/Users/harsh/Desktop/2016",str(counter) +
".sql")):
            # appending if the file already exists
            obj_write = 'a'
        else:
            # create a new file if not already present in the path
mentioned
            obj_write = 'w'

        # Do it for 2014, 2015, 2016

        with open(os.path.join(r"/Users/harsh/Desktop/2016",str(counter) +
".sql"), obj_write, encoding='utf8') as fw:
            fw.write(l)
```

1.2 Dumping the data (in the form of an INSERT statement) that are inside the SQL files into the MySQL database

```
# importing the python os module
# importing MySql Connector driver

# Inserting the 2014 year news headline data from SQL files into the first
database table (sentimentone)

directory_path = r'/Users/harsh/Desktop/2014'
number_of_files = len(os.listdir(directory_path))
```

```
db_conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Strats@1996",
    database="sentimentone"
)


for itr in range(1, number_of_files + 1):
    try:
        # reading the '.sql' file
        with open(os.path.join(directory_path, str(itr) + ".sql")) as sf:
            sql_statement_in_string_format = sf.read()

        curr_pointer_to_database = db_conn.cursor()
        curr_pointer_to_database.execute(sql_statement_in_string_format)
        curr_pointer_to_database.close()
        db_conn.commit()

    except Exception as e:
        print(e)

# Inserting the 2015 year news headline data from the SQL files into the
second database table (sentimenttwo)

directory_path = r'/Users/harsh/Desktop/2015'
number_of_files = len(os.listdir(directory_path))


db_conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Strats@1996",
    database="sentimenttwo"
)


for itr in range(1, number_of_files + 1):
    try:
        # reading the '.sql' file
        with open(os.path.join(directory_path, str(itr) + ".sql")) as sf:
            sql_statement_in_string_format = sf.read()

        curr_pointer_to_database = db_conn.cursor()
        curr_pointer_to_database.execute(sql_statement_in_string_format)
        curr_pointer_to_database.close()
        db_conn.commit()

    except Exception as e:
        print(e)

# Inserting the 2016 year news headline data from the SQL files into the
third database table (sentimentthree)


directory_path = r'/Users/harsh/Desktop/2016'
number_of_files = len(os.listdir(directory_path))
```

```
db_conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Strats@1996",
    database="sentimentthree"
)


for itr in range(1, number_of_files + 1):
    try:
        # reading the '.sql' file
        with open(os.path.join(directory_path, str(itr) + ".sql")) as sf:
            sql_statement_in_string_format = sf.read()

        curr_pointer_to_database = db_conn.cursor()
        curr_pointer_to_database.execute(sql_statement_in_string_format)
        curr_pointer_to_database.close()
        db_conn.commit()

    except Exception as e:
        print(e)
```

1.3 Importing the data from each three database table into combined pandas data frame data structure

```
# importing python object relational mapper: SQLAlchemy
# importing python data analysis library: Pandas

engine =
create_engine("mysql+pymysql://{user}:{pw}@localhost/{db}".format(user="roo
t",
pw="Strats@1996",
db="sentimentone"))

database_con = engine.connect()
df_2014 = pd.read_sql("select * from sentimentone.Megatable", database_con)

engine_2 =
create_engine("mysql+pymysql://{user}:{pw}@localhost/{db}".format(user="roo
t",
pw="Strats@1996",
db="sentimenttwo"))

database_con_2 = engine_2.connect()
df_2015 = pd.read_sql("select * from sentimenttwo.Megatable",
database_con_2)

engine_3 =
create_engine("mysql+pymysql://{user}:{pw}@localhost/{db}".format(user="roo
t",
```

```
pw="Strats@1996",
db="sentimentthree"))

database_con_3 = engine_3.connect()
df_2016 = pd.read_sql("select * from sentimentthree.Megatable",
database_con_3)

# Merging the data into one combined dataset
frames = [df_2014, df_2015, df_2016]
result = pd.concat(frames)

result.rename(columns={'newsHeadline': 'news_headline'}, inplace=True)
result.to_csv("news_headline_sentiment_data_combined.csv", index=False)
```

1.4 Filtering the News Headlines that are related to Apple Inc. company

```
# reading the news headline data that was created above
chunks_of_dataframes =
pd.read_csv('news_headline_sentiment_data_combined.csv',
                                    chunksize=10000)

# Intializing the list, which will contain the list of
# dataframes that have news headlines related to apple stock
list_of_apple_news_headlines_dataframes = []

for chunk in chunks_of_dataframes:
    # checking the missing values and then dropping those values
    chunk.dropna(inplace=True)
    # converting entire string into lowercase
    chunk.news_headline = chunk.news_headline.str.lower()

    # extracting those news headlines that talks about apple stock
    apple_news = chunk.loc[(chunk.news_headline.str.contains('aapl')) |
                  (chunk.news_headline.str.contains('apple'))]
    # adding the above fitered dataframe into the list
    list_of_apple_news_headlines_dataframes.append(apple_news)



# concatinating all the list of dataframes, now
# 'list_of_apple_news_headlines_dataframes' is itself a dataframe
list_of_apple_news_headlines_dataframes =
pd.concat(list_of_apple_news_headlines_dataframes)


# calculating the count of the news headlines which has
count_of_apple_stock_news = {
    'number of news headlines regarding apple stock':
len(list_of_apple_news_headlines_dataframes)
}

# checking the number of news headlines that are related to apple stock
count_of_apple_stock_news
```

```
list_of_apple_news_headlines_dataframes.rename(columns={'newsHeadline':
'news_headline'}, inplace=True)

# Writes the dataframe into csv file.
list_of_apple_news_headlines_dataframes.to_csv('news_headline_sentiment_dat
a_combined_aapl.csv')
```

Code Listing 2: Sentiment Score Calculation.ipynb

2.1 Calculating Sentiment Score of each News Headlines

```
# importing python data analysis library: Pandas

# There were around 80000 rows in aapl_news_headline data.
# Then I deleted the duplicated rows by news_headline column
# Then filtered the dataframe and only took the necessary columns such as
['news_headline', 'time_stamp', 'URL'].
# Earlier 'time_stamp' column was called 'start_time_stamp'


# using the vaderSentiment
# importing lexicon based sentiment analysis tool: vaderSentiment

analyzer = SentimentIntensityAnalyzer()
new_words = {
    "bear" : -2.0,
    "bull" : 2.0,
}
analyzer.lexicon.update(new_words)
analyzer.lexicon["bull"]

# Now let's get to the business and calculate sentiment class
# and sentiment score for our news headline data
def sentimenClassCalculation(negative, positive):

    if negative > positive:
        return -1
    elif positive > negative:
        return 1
    else:
        return 0

my_apple_dataset = pd.read_csv("aapl_news_headline.csv")
# Calculate sentiment_score
my_apple_dataset["sentiment_score"] =
my_apple_dataset["news_headline"].apply(lambda news:
analyzer.polarity_scores(news)['compound'])
my_apple_dataset.head()

#Calculate sentiment_class

my_apple_dataset["sentiment_class"] = 0
```

```
for i in range(0, len(my_apple_dataset)):

    score = analyzer.polarity_scores(my_apple_dataset["news_headline"][i])
    my_apple_dataset["sentiment_class"][i] =
sentimenClassCalculation(score['neg'], score['pos'])

my_apple_dataset.to_csv("aapl_news_headline.csv", index=False)
```

Code Listing 3: Daily Sentiment Score.ipynb

3.1 Reading the prepared apple news headline data

```
# data analysis library: pandas
# importing python data analysis library: Pandas

# Reading the prepared apple news headline data
apple_headlines_data = pd.read_csv('aapl_news_headline.csv',
parse_dates=True)

apple_headlines_data.head()

# filtering the necessary columns
apple = apple_headlines_data[['time_stamp', 'news_headline',
'sentiment_class', 'sentiment_score']]
apple.tail()

# checking the type of time_stamp
type(apple['time_stamp'].iloc[0])

# changing the type of time_stamp column from string format to datetime
format
apple['time_stamp'] = pd.to_datetime(apple['time_stamp'], utc=True)
type(apple['time_stamp'].iloc[0])

# sorting the apple sentiment dataframe according to the time_stamp
apple.sort_values(by='time_stamp', inplace=True)
apple.head()
```

3.2 Preparation of daily opening trading timestamp for news headlines

```
'''
The code contains the preparation of daily opening trading timestamp for
news headlines.
This code is adapted from a Medium post from Flavio Valerio Roncagliolo on
29/07/2021.

Medium Post here:
https://medium.com/analytics-vidhya/sentiment-analysis-with-vader-and-
algorithmic-trading-d7bef3c29c4b

'''
```

```
# importing datetime and BDay for calculating business day

apple["trading_time"] = pd.Series([])
for i in range(0, len(apple)):
    # calculating today's opening date, today's closing date, yesterday's
closing time,
    # and tomorrows's opening date for the news headline
    todays_opening_date =
pd.to_datetime(apple["time_stamp"].iloc[i]).floor('d').replace(hour=9,minut
e=30) - BDay(0)
    todays_closing_date =
pd.to_datetime(apple["time_stamp"].iloc[i]).floor('d').replace(hour=16,minu
te=0) - BDay(0)
    yesterdays_closing_date = (todays_opening_date -
BDay()).replace(hour=16,minute=0)
    tomorrows_opening_date = (todays_closing_date +
BDay()).replace(hour=9,minute=30)

    # The trading time column will include today's market open timestamp if
the headline falls between
    # yesterday's closing time and today's market open time.
    if
((pd.to_datetime(apple["time_stamp"].iloc[i])>=yesterdays_closing_date) &
(pd.to_datetime(apple["time_stamp"].iloc[i])<todays_opening_date)):
        apple.at[i, "trading_time"] = todays_opening_date

    # if the news headline falls between today's market close time and
tomorrow's market open time,
    # the time in this trading column will be tomorrow's open time.
    elif
((pd.to_datetime(apple["time_stamp"].iloc[i])>=todays_closing_date) &
(pd.to_datetime(apple["time_stamp"].iloc[i])<tomorrows_opening_date)):
        apple.at[i, "trading_time"] = tomorrows_opening_date

    else:
        pass

apple.tail()
```

3.3 Calculate Daily Sentiment Score

```
apple_daily_sentiment =
apple.groupby('trading_time').sentiment_score.agg('mean').to_frame('score')

apple_daily_sentiment.head()

# importing python data visualization library: Matplotlib
plt.figure(figsize=(15, 8))
plt.plot(apple_daily_sentiment)
plt.title('Apple Stock Daily News Headline Sentiment Score', fontsize=16)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Sentiment Score', fontsize=14)
plt.show()

# Because of all the above operations, the indices are disturbed, so
resetting the index
```

```
apple_daily_sentiment.reset_index(inplace=True)

# creating date column
apple_daily_sentiment['Date'] =
pd.to_datetime(apple_daily_sentiment['trading_time']).dt.floor('d')

# we don't need time zone information, so removing it
apple_daily_sentiment['Date'] = apple_daily_sentiment['Date'].apply(lambda
d: d.replace(tzinfo=None))

# setting date as an index
apple_daily_sentiment.set_index("Date", inplace=True)
apple_daily_sentiment.head()
```

3.4 Store Daily Sentiment Score in CSV file

```
apple_daily_sentiment.to_csv('apple_daily_sentiment.csv')
```

Code Listing 4: Merging Apple Stock Prices Data and Sentiment Data.ipynb

4.1 Read Apple Stock Prices Data

```
# importing python data analysis library: Pandas

# Read the daily 'AAPl' stock prices, that is taken from Yahoo finance
aapl_stock_data = pd.read_csv("stock_data_aapl_2014_2016.csv", index_col=0)
aapl_stock_data = aapl_stock_data.sort_index()
# index is changed to datetime
aapl_stock_data.index = pd.to_datetime(aapl_stock_data.index)

aapl_stock_data.head()
```

4.2 Read Apple Stock News Headline Sentiment Score

```
# Read sentiment data that we prepared earlier
apple_daily_sentiment_score = pd.read_csv('apple_daily_sentiment.csv',
index_col=0)

# index is changed to datetime
apple_daily_sentiment_score.index =
pd.to_datetime(apple_daily_sentiment_score.index)

apple_daily_sentiment_score.head()
```

4.3 Merging Stock Price and Sentiment data

```
# Join the two dataframes: aapl_stock_data and apple_daily_sentiment_score
```

```
prices_and_sentiment_score = pd.concat([aapl_stock_data,
apple_daily_sentiment_score], axis=1)

prices_and_sentiment_score.dropna(inplace=True)

prices_and_sentiment_score.head()

prices_and_sentiment_score.to_csv("prices_and_sentiment_score.csv")
```

Code Listing 5: Apple Stock Price Prediction Without Sentiment Data.ipynb

5.1 Data Pre-preprocessing

```
# importing python data analysis library: Pandas
# importing statistical data visualization library: Seaborn
# importing scientific computing package: Numpy
# importing python data visualization library: Matplotlib

apple = pd.read_csv("AAPL-2.csv")[["Date", "Adj Close"]]

# checking the shape of the dataset
apple.shape

'''
The code contains the demonstration of splitting the dataset into training,
validation, test dataset, and model prediction. This code is adapted from a
medium post from Shagun Kala on 31/07/2021.

Medium Post here:
https://medium.com/@kala.shagun/stock-market-prediction-using-news-
sentiments-f9101e5ee1f4

'''

# Training ratio = 80%
# Testing ratio = 20%
train_apple, test_apple = apple[0:int(len(apple)*0.8)],
apple[int(len(apple)*0.8):]

train_apple = train_apple.set_index('Date', drop= False)
test_apple = test_apple.set_index('Date', drop= False)

train_apple.shape, test_apple.shape

#Changing datatype of date column from string to datetime

# importing python datetime library
train_apple['Date'] = pd.to_datetime(train_apple['Date'])
test_apple['Date'] = pd.to_datetime(test_apple['Date'])

train_apple['Date'] = train_apple['Date'].dt.date
test_apple['Date'] = test_apple['Date'].dt.date
```

```
plt.figure(figsize=(15,8))
plt.title('Apple Sock')
plt.xlabel('Years')
plt.ylabel('Adj Close')
plt.plot(train_apple['Date'], train_apple['Adj Close'], 'blue',
label='Training Data')
plt.plot(test_apple['Date'], test_apple['Adj Close'], 'red', label='Testing
Data')
plt.legend()

#train_apple.to_csv('train.csv')
#test_apple.to_csv('test.csv')

#Splitting the dataset into training, validation, and testing dataset
#Train ratio = 64 %
#Validation ratio = 16%
#Test ratio = 20%


X_train_apple, X_validate_apple, _ = np.split(apple, [int(0.64 *
len(apple)), int(0.8 * len(apple))])

# Setting Date column as an index, this will help to perform data
visualization with x-axis as a Date column

X_train_apple = X_train_apple.set_index('Date', drop= False)
X_validate_apple = X_validate_apple.set_index('Date', drop= False)



print(len(X_train_apple), len(X_validate_apple), len(test_apple))

#Changing datatype of date column from string to datetime
# importing python datetime module
X_train_apple['Date'] = pd.to_datetime(X_train_apple['Date'])
X_validate_apple['Date'] = pd.to_datetime(X_validate_apple['Date'])

X_train_apple['Date'] = X_train_apple['Date'].dt.date
X_validate_apple['Date'] = X_validate_apple['Date'].dt.date

plt.figure(figsize=(15,8))
plt.title('Apple Stock Train-CV-Test Split')
plt.xlabel('Years')
plt.ylabel('Adj Close')
plt.plot(X_train_apple['Date'], X_train_apple['Adj Close'], 'blue',
label='Apple Training Data')
plt.plot(X_validate_apple['Date'], X_validate_apple['Adj Close'], 'red',
label='Apple CV Data')
plt.plot(test_apple['Date'], test_apple['Adj Close'], 'green', label='Apple
Test Data')
plt.legend()
```

5.2 Feature Scaling and Data Preparation

```
# Feature Scaling: We need data to be scaled as a gradient descent based
optimization algorithm requires it.
# importing MinMaxScaler function from data analysis library: Scikit-learn

# Feature scaling for apple stock closing price
stock_price_scaling_object = MinMaxScaler()

closing_price_train = X_train_apple['Adj Close'].values.reshape(-1, 1)
closing_price_validate = X_validate_apple['Adj Close'].values.reshape(-1,
1)
closing_price_test = test_apple['Adj Close'].values.reshape(-1, 1)

minmax_scaled_training_stock_price_data =
stock_price_scaling_object.fit_transform(closing_price_train)
minmax_scaled_validation_stock_price_data =
stock_price_scaling_object.transform(closing_price_validate)
minmax_scaled_test_stock_price_data =
stock_price_scaling_object.transform(closing_price_test)

# Preparing the dataset for training the model

# This is like a regression task, so we need two datasets, the input (X)
and the output(Y)


# The sequential relationship is maintained in the datasets,
# because we are currently exploiting the sequential model.
def data_preparation(stockPrice, sequenceLength):

    # X is acting like an input feature vectors (stock price data of window
size of 40).
    # Y is acting like a vector of corresponding target (last day stock
price) values.
    X, Y = [], []
    for i in range(len(stockPrice)-sequenceLength-1):

        # iteratively extracting the look back sequence length of stock
price data. Here sequenceLength = 40
        stock_price = stockPrice[i:(i+sequenceLength), 0]

        # X is a combination of the stock price of the previous 40 days
        X.append(stock_price)

        # Y is the stock price of the last day. This next value will be
treated as the target
        Y.append(stockPrice[i + sequenceLength, 0])

    return np.array(X), np.array(Y)


train_input_featureX, train_targetY =
data_preparation(minmax_scaled_training_stock_price_data, 40)

validation_input_featureX, validation_targetY =
data_preparation(minmax_scaled_validation_stock_price_data, 40)
```

```
test_input_featureX, test_targetY =
data_preparation(minmax_scaled_test_stock_price_data, 40)

# Here, For example, train_input_featureX has 602 sequences, and each
sequence has a total of 40 time-steps.
# After this, we will transform these matrices into different batches by
reshaping them so that
# they get accepted by our recurrent neural network.

train_input_featureX.shape, validation_input_featureX.shape,
test_input_featureX.shape

#Converting the above-prepared datasets into a 3-dimensional tensor to
handle the batch gradient descent.

train_input_featureX, train_targetY = np.array(train_input_featureX),
np.array(train_targetY)
train_input_featureX = np.reshape(train_input_featureX,
                                  (train_input_featureX.shape[0],
train_input_featureX.shape[1], 1))

validation_input_featureX, validation_targetY =
np.array(validation_input_featureX), np.array(validation_targetY)
validation_input_featureX = np.reshape(validation_input_featureX,
                                       (validation_input_featureX.shape[0],
validation_input_featureX.shape[1], 1))

test_input_featureX, test_targetY = np.array(test_input_featureX),
np.array(test_targetY)
test_input_featureX = np.reshape(test_input_featureX,
                                 (test_input_featureX.shape[0],
test_input_featureX.shape[1], 1))

train_input_featureX.shape, validation_input_featureX.shape,
test_input_featureX.shape
```

5.3 Price Prediction With LSTM RNN Model

```
'''
The code contains the demonstration of the model prediction. This code is
adapted from a medium post from Shagun Kala on 31/07/2021.

Medium Post here:
https://medium.com/@kala.shagun/stock-market-prediction-using-news-
sentiments-f9101e5ee1f4

'''

# Importing Sequential model from Keras API of Tensorflow 2
# Importing Dense layer from Keras API of Tensorflow 2
# Importing LSTM layer from Keras API of Tensorflow 2
# Importing Tensorflow library
# Importing Keras API of Tensorflow 2
# Importing backend function from Keras API of Tensorflow 2
```

```
# Defining the sequential model
model = Sequential()

# Adding LSTM layer with 128 units, hyperbolic tangent as an activation
function,
# and input shape in case of train_input_featureX will be (100, 1)
model.add(LSTM(units=128, activation='tanh',

kernel_initializer=tf.keras.initializers.glorot_uniform(seed=26),
               input_shape = (train_input_featureX.shape[1], 1),
               unroll = True))

# After setting the recurrent layer, now adding the output layer
model.add(Dense(1, name="output_layer"))


#Defining our loss function
def root_mean_squared_error(y_true, y_pred):

    error = y_pred - y_true
    squared_error = backend.square(error)
    mean_squared_error = backend.mean(squared_error)
    square_root_of_mean_squared_error = backend.sqrt(mean_squared_error)
    return square_root_of_mean_squared_error

# RNN model compiling
model.compile(optimizer = keras.optimizers.Adam(learning_rate=0.01),
              loss = root_mean_squared_error)



# Fitting the RNN model to the Training set
model.fit(train_input_featureX, train_targetY,
          epochs = 50,
          batch_size = 16,
          validation_data = (validation_input_featureX,
validation_targetY))

#Predicting on test data

predicted_stock_price = model.predict(test_input_featureX)
predicted_stock_price =
stock_price_scaling_object.inverse_transform(predicted_stock_price)

# Visualising our prediction model results

# importing python data visualization library: Matplotlib
# importing statistical data visualization library: Seaborn
# importing sqrt function from math module
# importing mean_squared_error function from data analysis library: Scikit-
learn

plt.figure(figsize=(16, 6))
plt.plot(train_apple['Date'], train_apple['Adj Close'], color = 'blue',
label = 'Train Apple Data')
plt.plot(test_apple['Date'], test_apple['Adj Close'].values, color = 'red',
label = 'Test Apple Data')
```

```
plt.plot(test_apple.iloc[41:]['Date'], predicted_stock_price, color =
'green', label = 'Predicted Apple Data')
plt.title('Apple Stock Price Prediction using LSTM')
plt.xlabel('Years')
plt.ylabel('Apple Prices')
plt.legend()
plt.show()



RMSE_LSTM = sqrt(mean_squared_error(test_apple.iloc[41:]['Adj
Close'].values, predicted_stock_price))

print(f"RMSE_LSTM = {round(RMSE_LSTM,2)}")
```

Code Listing 6: Apple Stock Price Prediction With Sentiment Data.ipynb

6.1 Apple Stock Price and Sentiment Data Analysis

```
# importing python data analysis library: Pandas
# importing statistical data visualization library: Seaborn
# importing scientific computing package: Numpy
# importing python data visualization library: Matplotlib

df = pd.read_csv("prices_and_sentiment_score.csv", index_col = 0)
df.head()

#Analyze how many negative and positive sentiments we have and how our
apple stock is trending.

positive_sentiment_count = len(df[df['score'] > 0])
negative_sentiment_count = len(df[df["score"] < 0])

print("positive_sentiment_count : {} ".format(positive_sentiment_count))
print("negative_sentiment_count : {} ".format(negative_sentiment_count))

# Plot our apple prices
df["Adj Close"].plot(figsize=(20,8))


'''
The code contains the demonstration of the probability distribution of
sentiment score.
This was adapted from a medium post from Shagun Kala on 31/07/2021.
Medium Post here: https://medium.com/@kala.shagun/stock-market-prediction-
using-news-sentiments-f9101e5ee1f4

'''

# Now let's test the proability distribution of the sentiment score that
was calculated by VADER
plt.figure(figsize=(10, 7))
sns.distplot(df['score'])
plt.ylabel('Probability')
plt.title('Sentiment score probability distribution')
```

6.2 Data Pre-Processing

```
apple = pd.read_csv("prices_and_sentiment_score.csv")[["Date", "score",
"Adj Close"]]
apple.head()

print(len(apple))

'''
The code contains the demonstration of splitting the dataset into training,
validation and test dataset.
This idea for split-ratio is adapted from a medium post from Shagun Kala on
31/07/2021.

Medium Post here:
https://medium.com/@kala.shagun/stock-market-prediction-using-news-
sentiments-f9101e5ee1f4

'''

#Splitting the dataset into training, validation, and testing dataset
#Train ratio = 64 %
#Validation ratio = 16%
#Test ratio = 20%

train_apple = apple[0:int(len(apple)*0.8)]
X_train_apple, X_validate_apple, test_apple = np.split(apple, [int(0.64 *
len(apple)), int(0.8 * len(apple))])

# Setting Date column as an index, this will help to perform data
visualization with x-axis as a Date column
train_apple = train_apple.set_index('Date', drop= False)
X_train_apple = X_train_apple.set_index('Date', drop= False)
X_validate_apple = X_validate_apple.set_index('Date', drop= False)
test_apple = test_apple.set_index('Date', drop= False)


print(len(X_train_apple), len(X_validate_apple), len(test_apple))
```

6.3 Feature Scaling and Data Preparation

```
'''
The code contains Feature scaling.
This was adapted from a medium post from Shagun Kala on 31/07/2021.

Medium Post here:
https://medium.com/@kala.shagun/stock-market-prediction-using-news-
sentiments-dc4c24c976f7

'''

# Feature Scaling: We need data to be scaled as a gradient descent based
optimization algorithm requires it.
```

```
# importing MinMaxScaler function from data analysis library: Scikit-learn

# Feature scaling for apple stock closing price
stock_price_scaling_object = MinMaxScaler()

closing_price_train = X_train_apple['Adj Close'].values.reshape(-1, 1)
closing_price_validate = X_validate_apple['Adj Close'].values.reshape(-1,
1)
closing_price_test = test_apple['Adj Close'].values.reshape(-1, 1)

minmax_scaled_training_stock_price_data =
stock_price_scaling_object.fit_transform(closing_price_train)
minmax_scaled_validation_stock_price_data =
stock_price_scaling_object.transform(closing_price_validate)
minmax_scaled_test_stock_price_data =
stock_price_scaling_object.transform(closing_price_test)

# Feature scaling for apple stock sentiment score
sentiment_score_scaling_object = MinMaxScaler()

sentiment_score_train = X_train_apple['score'].values.reshape(-1, 1)
sentiment_score_validate = X_validate_apple['score'].values.reshape(-1, 1)
sentiment_score_test = test_apple['score'].values.reshape(-1, 1)

minmax_scaled_training_sentiment_score_data =
sentiment_score_scaling_object.fit_transform(sentiment_score_train)
minmax_scaled_validation_sentiment_score_data =
sentiment_score_scaling_object.transform(sentiment_score_validate)
minmax_scaled_test_sentiment_score_data =
sentiment_score_scaling_object.transform(sentiment_score_test)

# Preparing the dataset for training the model

# This is like a regression task, so we need two datasets, the input (X)
and the output(Y)


# The sequential relationship is maintained in the datasets because we are
currently exploiting the sequential model.
def data_preparation(stockPrice, sentimentScore, sequenceLength):

    # X is acting like an input feature vectors (combination of stock price
+ sentiment score of last day).
    # Y is acting like a vector of corresponding target (last day stock
price) values.
    X, Y = [], []
    for i in range(len(stockPrice)-sequenceLength-1):

        # iteratively extracting the look back sequence length of stock
price data. Here sequenceLength = 40
        stock_price = stockPrice[i:(i+sequenceLength), 0]

        # iteratively extracting the sentment score of last day
        sentiment_score = sentimentScore[i+sequenceLength-1]

        # X is a combination of the stock price of the previous 40 days and
the sentiment score of the last day
```

```
        X.append(np.append(stock_price,sentiment_score))

        # Y is the stock price of the last day. This next value will be
treated as the target
        Y.append(stockPrice[i + sequenceLength, 0])

    return np.array(X), np.array(Y)


train_input_featureX, train_targetY =
data_preparation(minmax_scaled_training_stock_price_data,

minmax_scaled_training_sentiment_score_data,
                                                    40)

validation_input_featureX, validation_targetY =
data_preparation(minmax_scaled_validation_stock_price_data,

minmax_scaled_validation_sentiment_score_data,
                                                              40)

test_input_featureX, test_targetY =
data_preparation(minmax_scaled_test_stock_price_data,

minmax_scaled_test_sentiment_score_data,
                                                  40)

# Here, For example, train_input_featureX has 423 sequences, and each
sequence has a total of 41 time-steps.
# After this, we will transform these matrices into different batches by
reshaping them so that
# they get accepted by our recurrent neural network.

train_input_featureX.shape, validation_input_featureX.shape,
test_input_featureX.shape

#Converting the above-prepared datasets into a 3-dimensional tensor to
handle the batch gradient descent.

train_input_featureX, train_targetY = np.array(train_input_featureX),
np.array(train_targetY)
train_input_featureX = np.reshape(train_input_featureX,
                                  (train_input_featureX.shape[0],
train_input_featureX.shape[1], 1))

validation_input_featureX, validation_targetY =
np.array(validation_input_featureX), np.array(validation_targetY)
validation_input_featureX = np.reshape(validation_input_featureX,
                                       (validation_input_featureX.shape[0],
validation_input_featureX.shape[1], 1))

test_input_featureX, test_targetY = np.array(test_input_featureX),
np.array(test_targetY)
test_input_featureX = np.reshape(test_input_featureX,
                                 (test_input_featureX.shape[0],
test_input_featureX.shape[1], 1))
```

```
train_input_featureX.shape, validation_input_featureX.shape,
test_input_featureX.shape
```

6.4 Apple Stock Price Prediction With Sentiment Data Using LSTM RNN Model

```
'''
The code contains RNN modelling using the keras library and data
visualization of the dataset.
This was adapted from a medium post from Shagun Kala on 31/07/2021.

Medium Post here:
https://medium.com/@kala.shagun/stock-market-prediction-using-news-
sentiments-dc4c24c976f7

'''

# Importing Sequential model from Keras API of Tensorflow 2
# Importing Dense layer from Keras API of Tensorflow 2
# Importing LSTM layer from Keras API of Tensorflow 2
# Importing Tensorflow library
# Importing Keras API of Tensorflow 2
# Importing backend function from Keras API of Tensorflow 2


# Defining the sequential model
model = Sequential()

# Adding LSTM layer with 128 units, hyperbolic tangent as an activation
function,
# and input shape in case of train_input_featureX will be (100, 1)
model.add(LSTM(units=128, activation='tanh',
        kernel_initializer=tf.keras.initializers.glorot_uniform(seed=26),
        input_shape = (train_input_featureX.shape[1], 1),
        unroll = True))

# After setting the recurrent layer, now adding the output layer
model.add(Dense(1, name="output_layer"))


#Defining our loss function
def root_mean_squared_error(y_true, y_pred):

    error = y_pred - y_true
    squared_error = backend.square(error)
    mean_squared_error = backend.mean(squared_error)
    square_root_of_mean_squared_error = backend.sqrt(mean_squared_error)
    return square_root_of_mean_squared_error

# RNN model compiling
model.compile(optimizer = keras.optimizers.Adam(learning_rate=0.01),
              loss = root_mean_squared_error)



# Fitting the RNN model to the Training set
```

```
model.fit(train_input_featureX, train_targetY,
          epochs = 50,
          batch_size = 16,
          validation_data = (validation_input_featureX,
validation_targetY))

#Predicting on test data

predicted_stock_price = model.predict(test_input_featureX)
predicted_stock_price =
stock_price_scaling_object.inverse_transform(predicted_stock_price)

#Changing datatype of date column from string to datetime

import datetime as dt
train_apple['Date'] = pd.to_datetime(train_apple['Date'])
test_apple['Date'] = pd.to_datetime(test_apple['Date'])

train_apple['Date'] = train_apple['Date'].dt.date
test_apple['Date'] = test_apple['Date'].dt.date

# Visualising our prediction model results

# importing python data visualization library: Matplotlib
# importing statistical data visualization library: Seaborn
# importing sqrt function from math module
# importing mean_squared_error function from data analysis library: Scikit-
learn

plt.figure(figsize=(16, 6))
plt.plot(train_apple['Date'], train_apple['Adj Close'], color = 'blue',
label = 'Train Apple Data')
plt.plot(test_apple['Date'], test_apple['Adj Close'].values, color = 'red',
label = 'Test Apple Data')
plt.plot(test_apple.iloc[41:]['Date'], predicted_stock_price, color =
'green', label = 'Predicted Apple Data')
plt.title('Apple Stock Price Prediction using LSTM')
plt.xlabel('Years')
plt.ylabel('Apple Prices')
plt.legend()
plt.show()



RMSE_LSTM = sqrt(mean_squared_error(test_apple.iloc[41:]['Adj
Close'].values, predicted_stock_price))

print(f"RMSE_LSTM = {round(RMSE_LSTM,2)}")
```

6.5 Hyperparameter Tuning using Grid Search Technique

```
# importing python data analysis library: Pandas
# importing statistical data visualization library: Seaborn
# importing scientific computing package: Numpy
# importing python data visualization library: Matplotlib
```

```
apple = pd.read_csv("prices_and_sentiment_score.csv")[["Date", "score",
"Adj Close"]]
apple.head()

#Splitting the dataset into training, validation, and testing dataset
#Train ratio = 64 %
#Validation ratio = 16%
#Test ratio = 20%

train_apple = apple[0:int(len(apple)*0.8)]
X_train_apple, X_validate_apple, test_apple = np.split(apple, [int(0.64 *
len(apple)), int(0.8 * len(apple))])

# Setting Date column as an index, this will help to perform data
visualization with x-axis as a Date column
train_apple = train_apple.set_index('Date', drop= False)
X_train_apple = X_train_apple.set_index('Date', drop= False)
X_validate_apple = X_validate_apple.set_index('Date', drop= False)
test_apple = test_apple.set_index('Date', drop= False)


print(len(X_train_apple), len(X_validate_apple), len(test_apple))

# Feature Scaling: We need data to be scaled as gradient descent based
optimization algorithm requires it.
from sklearn.preprocessing import MinMaxScaler

# Feature scaling for apple stock closing price
stock_price_scaling_object = MinMaxScaler()

closing_price_train = X_train_apple['Adj Close'].values.reshape(-1, 1)
closing_price_validate = X_validate_apple['Adj Close'].values.reshape(-1,
1)
closing_price_test = test_apple['Adj Close'].values.reshape(-1, 1)

minmax_scaled_training_stock_price_data =
stock_price_scaling_object.fit_transform(closing_price_train)
minmax_scaled_validation_stock_price_data =
stock_price_scaling_object.transform(closing_price_validate)
minmax_scaled_test_stock_price_data =
stock_price_scaling_object.transform(closing_price_test)

# Feature scaling for apple stock sentiment score
sentiment_score_scaling_object = MinMaxScaler()

sentiment_score_train = X_train_apple['score'].values.reshape(-1, 1)
sentiment_score_validate = X_validate_apple['score'].values.reshape(-1, 1)
sentiment_score_test = test_apple['score'].values.reshape(-1, 1)

minmax_scaled_training_sentiment_score_data =
sentiment_score_scaling_object.fit_transform(sentiment_score_train)
minmax_scaled_validation_sentiment_score_data =
sentiment_score_scaling_object.transform(sentiment_score_validate)
minmax_scaled_test_sentiment_score_data =
sentiment_score_scaling_object.transform(sentiment_score_test)

# Preparing the dataset for training the model
```

```
# This is like a regression task, so we need two datasets the input (X) and
the output(Y)


# The sequential relationship is maintained in the datasets, because we are
currently exploiting the sequential model.
def data_preparation(stockPrice, sentimentScore, sequenceLength):

    # X is acting like an input feature vectors (combination of stock price
+ sentiment score of last day).
    # Y is acting like a vector of corresponding target (last day stock
price) values.
    X, Y = [], []
    for i in range(len(stockPrice)-sequenceLength-1):

        # iteratively extracting the look back sequence length of stock
price data. Here sequenceLength = 40
        stock_price = stockPrice[i:(i+sequenceLength), 0]

        # iteratively extracting the sentment score of last day
        sentiment_score = sentimentScore[i+sequenceLength-1]

        # X is a combination of the stock price of previous 40 days and
sentiment score of the last day
        X.append(np.append(stock_price,sentiment_score))

        # Y is the stock price of the last day. This next value will be
treated as the target
        Y.append(stockPrice[i + sequenceLength, 0])

    return np.array(X), np.array(Y)


train_input_featureX, train_targetY =
data_preparation(minmax_scaled_training_stock_price_data,

minmax_scaled_training_sentiment_score_data,
                                                        40)

validation_input_featureX, validation_targetY =
data_preparation(minmax_scaled_validation_stock_price_data,

minmax_scaled_validation_sentiment_score_data,
                                                                    40)

test_input_featureX, test_targetY =
data_preparation(minmax_scaled_test_stock_price_data,

minmax_scaled_test_sentiment_score_data,
                                                      40)


# Here, For example, train_input_featureX has 423 sequences and each
sequence have total of 41 time-steps.
# After this, we will transform these matrices into different batches by
reshaping them so that
```

```
# they get accepted by our recurrent neural network.

train_input_featureX.shape, validation_input_featureX.shape,
test_input_featureX.shape

#Converting the above prepared datasets into 3-dimentional tensor to handle
the batch gradient descent.

train_input_featureX, train_targetY = np.array(train_input_featureX),
np.array(train_targetY)
train_input_featureX = np.reshape(train_input_featureX,
                                  (train_input_featureX.shape[0],
train_input_featureX.shape[1], 1))

validation_input_featureX, validation_targetY =
np.array(validation_input_featureX), np.array(validation_targetY)
validation_input_featureX = np.reshape(validation_input_featureX,
                                       (validation_input_featureX.shape[0],
validation_input_featureX.shape[1], 1))

test_input_featureX, test_targetY = np.array(test_input_featureX),
np.array(test_targetY)
test_input_featureX = np.reshape(test_input_featureX,
                                 (test_input_featureX.shape[0],
test_input_featureX.shape[1], 1))

train_input_featureX.shape, validation_input_featureX.shape,
test_input_featureX.shape

# Importing Sequential model from Keras API of Tensorflow 2
# Importing Dense layer from Keras API of Tensorflow 2
# Importing LSTM layer from Keras API of Tensorflow 2
# Importing Tensorflow library
# Importing Keras API of Tensorflow 2
# Importing backend function from Keras API of Tensorflow 2
# importing sqrt function from math module
# importing mean_squared_error function from data analysis library: Scikit-
learn

'''
LSTM RNN Hyperparameter Tuning.
This was adapted from a Kaggle post on 05/08/2021.

Kaggle Post here:
https://www.kaggle.com/kamyarazar/stock-price-prediction-lstm-
hyperparameter-tuning

'''

# importing product function from python Itertools module
def LSTM_RNN_Hyperparameter_Optimization(hyper_parameter_value_matrix,
                                         train_input_featureX,
                                         train_targetY,
                                         validation_input_featureX,
                                         validation_targetY,
                                         test_input_featureX):
```

```
    # Unpacking all the hyperparamters arrays
    number_of_units_list, rate_of_learning_list, number_of_epochs_list,
number_of_batches_list = hyper_parameter_value_matrix


    # Gernerating all possible combinations of hyperparameter values using
itertool.product() function
    all_combinations_of_hyperparameter_values =
list(product(number_of_units_list,

rate_of_learning_list,

number_of_epochs_list,

number_of_batches_list))
    result = []

    for i in range(0, len(all_combinations_of_hyperparameter_values)):



print("*****************************START*********************************
****")

        # tuple unpacking of hyperparameter values
        number_of_units, rate_of_learning, number_of_epochs,
number_of_batches = all_combinations_of_hyperparameter_values[i]


        # Defining the sequential model
        model = Sequential()

        # Adding LSTM layer with 'number_of_units' units, hyperbolic
tangent as an activation function,
        # and input shape in case of train_input_featureX will be (100, 1)
        model.add(LSTM(units=number_of_units, activation='tanh',

kernel_initializer=tf.keras.initializers.glorot_uniform(seed=26),
                       input_shape = (train_input_featureX.shape[1], 1),
                       unroll = True))

        # After setting the recurrent layer, now adding the output layer
        model.add(Dense(1, name="output_layer"))


        #Defining our loss function
        def root_mean_squared_error(y_true, y_pred):

            error = y_pred - y_true
            squared_error = backend.square(error)
            mean_squared_error = backend.mean(squared_error)
            square_root_of_mean_squared_error =
backend.sqrt(mean_squared_error)
            return square_root_of_mean_squared_error

        # RNN model compiling
```

```
        model.compile(optimizer =
keras.optimizers.Adam(learning_rate=rate_of_learning),
                      loss = root_mean_squared_error)



        # Fitting the RNN model to the Training set
        model.fit(train_input_featureX, train_targetY,
                  epochs = number_of_epochs,
                  batch_size = number_of_batches,
                  validation_data = (validation_input_featureX,
validation_targetY))


        #Predicting on test data
        predicted_stock_price = model.predict(test_input_featureX)
        predicted_stock_price =
stock_price_scaling_object.inverse_transform(predicted_stock_price)

        # Root mean square error
        RMSE_LSTM = sqrt(mean_squared_error(test_apple.iloc[41:]['Adj
Close'].values, predicted_stock_price))

        result.append(list((number_of_units, rate_of_learning,
                            number_of_epochs, number_of_batches,
                            RMSE_LSTM)))

print("*****************************END*************************************
**")


    return result




# Two dimensional matrix (Hyper Parameter value matrix)

# [
#     [number of units],
#     [rate of learning],
#     [number of epochs],
#     [number of batches]
# ]

hyper_parameter_value_matrix = [

    [115, 120, 125, 130],
    [0.01, 0.03, 0.06, 0.09],
    [44, 46, 48, 50],
    [8, 12, 16, 32]
]

hyperparameter_tuning =
LSTM_RNN_Hyperparameter_Optimization(hyper_parameter_value_matrix,
```

```
train_input_featureX,
                                                                train_targetY,

validation_input_featureX,

validation_targetY,

test_input_featureX)

hyperparameter_tuning = pd.DataFrame(hyperparameter_tuning)
hyperparameter_tuning = hyperparameter_tuning.sort_values(by=[4],
ascending=True)

# Top 5 hyperparameters values
hyperparameter_tuning.head()
```