PREDICTION OF FINANCIAL COMMODITIES USING AI

The *Kaavish* Report
presented to the academic faculty

by

Talal Zahid

in partial fulfillment of the requirements for
Bachelor of Science
Computer Science
Dhanani School of Science and Engineering

Habib University

Spring 2019

PREDICTION OF FINANCIAL COMMODITIES USING AI

This *Kaavish* project was advised by:

Dr. Musabbir Majeed
Faculty of Computer Science
Habib University

Approved by the Faculty of Computer Science on June 13, 2019.

An investment in knowledge pays the best interest

Benjamin Franklin

ACKNOWLEDGEMENTS

First and foremost, I have to thank my research supervisor, Dr.Musabbir Majeed. Without his assistance and weekly assigned meeting, this paper would have never been accomplished. I would like to thank you for instilling in me the habit of getting work done on time, a habit which I am still working on.

I would also like to show gratitude to the Kaavish committee, including Dr.Shahid Hussain, Dr Saleha Raza, Dr Umair Azfar and other members for their constant support and preparedness to tackle any challenges that came about during my final year.

TABLE OF CONTENTS

Acknowledgments **v**

List of Tables **viii**

List of Figures **ix**

1 Introduction and Background **1**
1.1 Related Works 4

2 Design Details **6**
2.1 Data Design 6
2.1.1 Financial News Headlines 6
2.1.2 Stock Market Data 7
2.2 Data Pre-Processing 8
2.2.1 Stock Market Data Pre-Processing 8
2.2.2 Financial News Headlines Data Pre-Processing 8
2.3 Economic Indicators 10
2.3.1 Relative Strength Index (RSI) 10
2.3.2 Moving average convergence divergence (MACD) 11
2.3.3 Bollinger Band 11
2.4 Evaluation Metrics 13
2.4.1 Binary Classification Metrics 13
2.4.2 Regression Metrics 15

3 Model Details **17**
3.1 Model M1: Binary Classification using Multinomial Naive Bayes . . 17
3.1.1 Multinomial Naive Bayes calculation 17
3.1.2 Results 19
3.2 Model M2: Binary Stock Classifier using Word2vec and MLP-Neural Network 19
3.2.1 Word2vec 19
3.2.2 Sentiment Analyzer 21
3.2.3 Stock Price, Headlines and Sentiment Co-relation 22
3.2.4 Data-set split 22
3.2.5 Neural Network Classifier 23
3.3 Model M3: Stock Price prediction using VADER Lexicon and LSTM Neural Network 25
3.3.1 Vader Sentiment Analysis 26
3.3.2 Pearson correlation between stock prices and sentiment score 27
3.3.3 LSTM neural network model for Vader scores 28

3.4 Model M4: LSTM NN Regression Model using Word2vec 32
3.4.1 Model M4-A: Binary sentiment classifier using Word2vec embedding from Model M2 and LSTM NN 32
3.4.2 Model M4-B: LSTM NN for stock prediction using regression 34

4 Discussion **39**
4.1 Binary classification results 39
4.2 Regression results 41
4.3 Regression to Binary classification results 42

5 Conclusion and Future Directions **44**

A Source Code **47**

References **49**

LIST OF TABLES

3.1 Results from the MNB classifier. 19
3.2 Results for Model M2. 24
3.3 Vader Sentiment Polarity Scores 26
3.4 Directional results for Model M3. 32
3.5 Binary classification results for Model M4-A. 34
3.6 Optimum parameter values from the tuning set. 36
3.7 Binary classification after conversion using method on page 31. . . . 38

4.1 Results for binary directional classification. 39
4.2 Results for stock price using regression 41
4.3 Results for binary directional classification after conversion from regression results using the method on page 31. 42

LIST OF FIGURES

1.1 Network architecture for the language model. In each step the output of the LSTM layer predicts the probability distribution of the next character. 5
1.2 Details of the technical indicators used in [7] is summarized. 5

2.1 Working of the News Data Scrapper. 7
2.2 Sample text Pre-Processing 9
2.3 Apple Inc stock history from Sep 2013 - May 2014. Blue lines above and below the main signal are the upper and lower bands. The dotted line is the middle band. 12

3.1 Sample output from the MNB Sentiment Analyzer. 18
3.2 The 200 dimension vector for the word *good* in the vector space. . . 20
3.3 Visualization of a 2D vector space, with dimensionality reduced. . . 20
3.4 Word-Cloud plot where size of word represents tf-idf score of each word. 22
3.5 Neural Network architecture for model M2. 23
3.6 Accuracy graph of model M2. 25
3.7 Vader sentiment scores 27
3.8 Visualization of sentiment scores with respect to stock price. Stock data from Google Series A stock from Jan 2010 - Jan 2019. 27
3.9 Training Data-set sample. t is the current day, while t-1 is the previous day. 29
3.10 LSTM architecture for Model M3. 29
3.11 M3 result: Real vs Predicted price for Google stock from Jan - 2017 to Jan 2019. 31
3.12 LSTM NN architecture for sentiment classifier. 33
3.13 LSTM architecture for stock price predictor. 35
3.14 LSTM architecture for stock price predictor. 36
3.15 Selecting best performing Batch size based on RMSE. 37
3.16 Selecting best performing N based on RMSE. 37
3.17 Prediction over actual stock price graph. 38

EXECUTIVE SUMMARY

Stock market prediction has been an area of interest for a few decades now and much recently, there have been a lot of new research in the field of neural networks and natural language processing (NLP), that has resulted in satisfactory results. Sentiment analysis is one such sub-field of NLP that has made possible the extraction of emotions expressed in a body of text. These sentiments have been shown to co-relate with the change in stock prices. Previously twitter has been used to show a positive co-relation between the positive and negative sentiments gathered from a collection of tweets, and directional stock movements. In this paper, I expand upon these findings as I develop multiple predictive models that incorporates these technologies to predict not only the directional changes in the stock market but also the stock prices. For this purpose I use New York Times article headlines for the top 5 IT corporations listed on the S&P 500 index, and use word2vec for encoding these headlines into vectors. These vectors are fed into a LSTM NN, along with momentum-based economic indicators to supplement the predictive model. This has resulted in a MAPE score of 1.15% for the regression based model and 65% direction accuracy, hence, successfully showing a co-relation between stock prices and public sentiment.

CHAPTER 1
INTRODUCTION AND BACKGROUND

Prediction of stock prices has been a hot topic for a number of decades. Both, researchers and investors are equally interested in it. Especially since the introduction of fast and efficient processing capabilities, machine learning technologies have opened up new possibilities to process and analyze huge chunks of data and extract meaningful information that was hidden before. Stock price prediction has too enjoyed from this phenomenon and a lot of studies have been conducted, with research ongoing.

Early studies to predict the stock market were primarily based on two main philosophies: Technical and Fundamental approach [1]. The fundamentalist approach is that a companys relative data such as its earnings and ratios are sufficient to predict the movement of its stock. The second is the Technical approach, according to which historical prices and volumes of stock are analyzed to get the predicted stock price. These studies that are based only on a companys historical stock price and other economic indicators, have produced unsatisfactory results due to the volatility of the stock market [2].

Many such early attempts to forecast the stock market fell prey to two main concepts: the Efficient Market Hypothesis (EMH) and the Random Walk theory [2]. The EMH suggests that the main factors driving the stock market are the current events and the news that circulates around those events in the form of news headlines and articles. Due to the random nature of such events, they cannot be assigned a probability and hence, follow a Random Walk Pattern according to which they can only be predicted with 50% accuracy at best [3].

The EMH has since, been debunked as numerous studies have proven that the stock market does not exactly follow a Random Walk [4] [5] [6]. One recent study on the Turkish stock market predicted the stocks listed on the BIST100 index

with up to 58% accuracy [7]. For this purpose, they used economic indicators for the stock in focus along with data from a set of other related commodities such as currencies and stocks.

More recently, research in sentiment analysis suggests that the stock movement is not random and the early movements of stock price, those that soon follow an event, can be predicted through the sentiment extracted through textual analysis of information on the internet [4] [5]. A great deal of interest and research is being put in this field (a subfield of NLP Natural Language Processing). This data which is extracted from quarterly reports, breaking news stories or public messages, reflects the sentiments of concerned entities towards the stock in question and these sentiments can significantly influence the stock movement of that company.

Most existing research on financial-textual sentiment analysis depends on recognizing a predefined set of keywords and relating those keywords to the overall sentiment indicated by the text containing that keyword. Such techniques have shown promising but inconsistent ability to predict the future stock movements [4]. With the advent of social media and the growth of online presence of major firms, textual data has become abundant. In the form of news articles alone, many media groups publish daily and intra-daily articles related to particular company or sector. These articles carry certain sentiments regarding major topics being discussed related to that particular company and hence, drive the overall public opinion.

Psychological studies have found that sentiments driven through emotions, apart from the factual knowledge, plays a key role in the cognitive process of reaching a decision [8]. Further evidence in the behavioral finance studies verify that emotions do affect finance related decisions. It is, along these lines, reasonable to conclude that sentiments driven though news play an equally important role in stock movements as other factors.

Several studies have had varying success in predicting stock market through sentiments extracted through social media and news outlets. Bollen, Mao and

Zeng [9], reported up to 80% accuracy in predicting the daily up/down stock movement on DJIA using public sentiments extracted through user tweets (a text message on the social media website called Twitter. Limited to 140 characters). Another study by Robert P. Schumaker and Hsinchun Chen [1] used a Support Vector Machine tailored for predictive analysis of financial news data represented using Bag of Words and Noun Phrases. They achieved up to 57% directional accuracy on the daily stock price movements on the S&P 500 index.

In this paper, I contribute to the field of sentiment analysis as I use this technique to predict the stock market movements and prices of Google's Series A stock using different machine learning models. The textual data will consist of financial news headlines extracted from the New York Times from Jan 2010 to March 2019. These articles will be used to extract the sentiment (positive or negative) which will be fed into a machine learning model as a feature. Another set of features for my prediction model come from a selected set of economic indicators that are presented in [3]. The aim of this research is to reinforce the sentiments from the sentiment analysis on financial news headlines, with the economic indicators, to account for the market volatility. Using these techniques, I have predicted the directional stock movements, as well as the forecast prices for the next day.

1.1 Related Works

There are numerous studies related to stock market prediction using sentiment analysis which I have referred to in my paper, however, the most prominent one is by Pinheiro & Dras [6]. It is an automated trading system based on Efficient Markets Hypothesis (EMH), that, given the release of news information about a company, predicts changes in stock prices. Their system is trained to predict both, changes in the stock price of the company mentioned in the news article and the points of the corresponding stock exchange index (S&P 500) on which that company is listed. Their model consists of two LSTM NNs. The first builds a representation for the input, a character level language model. The second is the recurrent neural network used for the prediction. It is a classifier that takes the input from the first LSTM and predicts whether the price will rise or fall in the chosen time-frame. They found that the prediction accuracy was better for the following days price movement compared with the following week, which were in turn better than the following year.

What I have incorporated from this study is the custom character embedding model that they used to train their LSTM NN with the news articles (Shown in figure 1.1). In my research, I have taken inspiration from their character model by implementing a word embedding model for my textual data which consists of financial news headlines and trained it on daily stock movements, essentially creating a sentiment classifier. For the purposes of prediction of stock movement, I followed a similar setup to [10] [4], in which they concatenated all news articles regarding a particular category on each day and predicted if the closing price on the day *t+1* would increase when compared with the closing price on day *t*.

Furthermore, the idea to reinforce textual analysis with economic indicators came from the study [6]. They concluded that their word embedding model could benefit from more complex architectures that would supplement the feature set sent to their Recurrent Neural Network for training. For this purpose, I reviewed

[7] which made use of 25 technical indicators to form a feature set of 30 inputs (Shown in figure 1.2). However, due to the limitation of resources and the scope of my research, I carefully selected 3 of the indicators which had the best correlation with the stock movement.

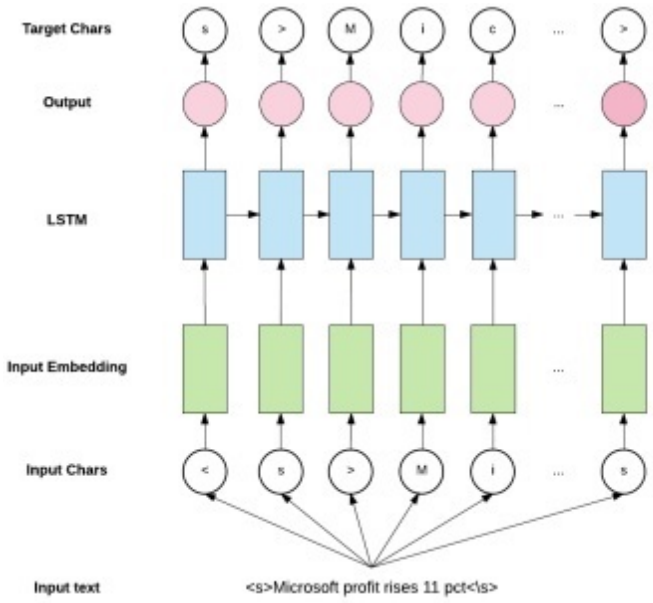

Figure 1.1: Network architecture for the language model. In each step the output of the LSTM layer predicts the probability distribution of the next character.

Indicator	Explanation	Indicator	Explanation
OP	Open price	MOM(x)	Momentum measures change in stock price over last x days
HI	High price	MACD(x,y)	x days moving average convergence and divergence
LO	Low price	TEMA(x)	Triple exponential moving average
CL	Close price	PPO(x,y)	Percentage price oscillator
ROC(x)	Rate of change	CCI(x)	Commodity channel index
ROCP(x)	Rate of change pct.	WILLR(x)	Larry William's R%
K%(x)	Stochastic K%	RSI(x)	Relative strength index
D% (x)	D% is the moving average of K%	ULTOSC(x,y,z)	Ultimate oscillator
BIAS(x)	x-days bias	ATR(x)	Average true range
MA(x)	x-days moving average	MEDPRICE(x)	Median price
EMA(x)	x-days exponential moving average	MIDPRICE(x)	Medium price
signL(x,y)	A signal line is also known as a trigger line	HH(x)	Highest price
		LL(x)	Lowest price

Figure 1.2: Details of the technical indicators used in [7] is summarized.

CHAPTER 2
DESIGN DETAILS

This section contains details about the data design that is the prerequisite to the machine learning models described in section 3. More specifically it includes details regarding data collection, data pre-processing and the evaluation metrics.

2.1 Data Design

The two types of data that was collected in this research are the financial news headlines and the stock prices.

2.1.1 Financial News Headlines

To collect the financial news from the web, two sources were used. The first is the *New York Times* and the second is the *News Api*. Both of these sources have limited free access using their API's. So to work around that problem, I wrote a custom script for automatically scrapping the new articles using automated calls to the API, whenever the limit for each call is reached. The working of the News Data Scrapper is summarized in the figure 2.1 below.

Using the New York Times API, a total of 6189 financial news headlines on the top 5 IT Corporations (Google, Microsoft, Apple Inc, Amazon, Facebook) were extracted. The dates of these headlines range from Jan - 2010 to March - 2019. The API allows for the search result to be filtered according to the chosen time period and the keyword (that must be contained in the headline). These headlines are used to train/test the sentiment analyzer described in the section 3.

Due to lack of retrievable data from the New York Times API, News Api was used for further data collection. A total of 10000 headlines were selected using the keyword 'Google'. These headlines are used to train/test the stock predictor, also described in section 3.

2.1.2 Stock Market Data

Stock prices were extracted using the Alpha Vantage - Stock Time Series API. Using this API, real-time and historical global stock data was retrieved for the 5 selected stocks (Google, Microsoft, Apple Inc, Amazon, Facebook) ranging from Jan - 2010 to March - 2019. The temporal resolutions used is daily. For the daily stock time series, the 'daily adjusted closing price' is selected.

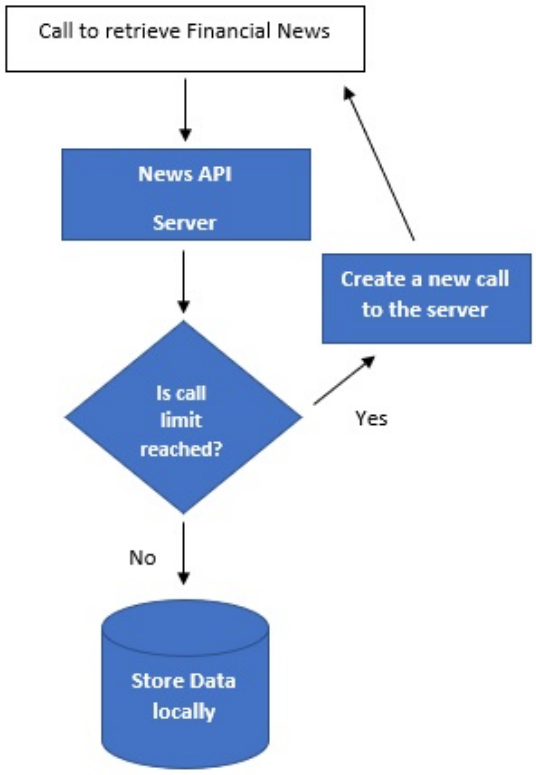

Figure 2.1: Working of the News Data Scrapper.

2.2 Data Pre-Processing

Before the data is fed into the the Sentiment Analyzer and the Stock Predictor modules, it is normalized as described below.

2.2.1 Stock Market Data Pre-Processing

The Stock Data that is collected is often incomplete. This is understandable as the stock market is often closed on weekends and/or due to public holidays. As news articles that are released on these days, must be co-related to a stock price for that day, this price is approximated using a simple technique presented by Goel [11].

$$\mathbf{s} = \frac{x + y}{2} \tag{1}$$

Where s is the first missing value for the day $t+1$, x is the stock price on some day t and y is the next stock price available on the day $t+n$ with n stock prices missing in between. As the stock prices generally follow a concave function, the rest of the missing values can be found in a similar manner.

2.2.2 Financial News Headlines Data Pre-Processing

The headlines extracted using the News York Times API and the News API, must be normalized before they can be used as input for the machine learning models. This task of text processing was achieved using python's NLTK (Natural Language ToolKit), using the following steps.

1. *Tokenization:* Headlines are broken up into individual string of characters based on white-space.

2. *Removal of non-alphabets:* This includes numbers, punctuation marks or any other non-alphabet character.

3. *Removal of non-words:* Sequence of alphabetic characters that do not represent a word or a name such as URLs, hash-tags, email-address etc, are removed.

4. *Lemmatization:* Removes word inflections to get the root word. For e.g the word *driving* and *driver* will be reduced to *drive*.

5. *Case normalization:* All the words are converted to lower case.

Raw Headlines	Processed Headlines
Facebook Says Cambridge Analytica Harvested Data of Up to 87 Million Users!	['facebook', 'say', 'cambridg', 'analytica', 'harvest', 'data', 'of', 'up', 'to', 'million', 'user']
Twitter, Facebook, Slack: Using Every Tool to Hear What Readers Think.	['twitter', 'facebook', 'slack', 'use', 'everi', 'tool', 'to', 'hear', 'what', 'reader', 'think']
Mark Zuckerberg Can Still Fix This Mess.	['mark', 'zuckerberg', 'can', 'still', 'fix', 'thi', 'mess']

Figure 2.2: Sample text Pre-Processing

2.3 Economic Indicators

Economic Indicators are used as a part of the technical analysis of the stock market. They are widely used to predict future stock prices and trends within them [7]. The only input that is required to derive them are the historical stock prices. The 3 economic indicators that I have used in my research are listed below.

2.3.1 Relative Strength Index (RSI)

RSI index is used to contrast the stock price increase with the losses and gauges if the market for that stock is bearish or bullish (whether the stock is over or undersold). The indicator outputs a value in between 0 to 100. Typically, if the RSI value is over 70, it means that the stock is being overbought. As the demand is increasing, the stock price goes up and it becomes reasonable to sell the stock. On the other hand, if the RSI values goes below 30, it demonstrates a purchase action. This threshold value is subject to change, based on the stock in question [12]. Present below is the equation used to calculate RSI. I have taken n to be 14 days.

$$\mathbf{RSI} = 100 - \frac{100}{1 + RS} \tag{2}$$

$$\mathbf{RS} = \frac{\text{Average of all up moves in the last n time-steps}}{\text{Average of all down moves in the last n time-steps}} \tag{2.1}$$

2.3.2 Moving average convergence divergence (MACD)

MACD is an economic indicator used in technical analysis of stock prices. It measures the strength of a stock over a certain period of time. EMA (Exponential Moving Average) is used in conjunction with MACD. The formula for MACD is defined as.

$$\textbf{MACDLine} = \text{12DAY EMA - 26DAY EMA} \tag{3.1}$$

$$\textbf{SignalLine} = \text{9DAY EMA of MACDLine} \tag{3.2}$$

$$\textbf{EMA}_t = (P * \alpha) + (\text{EMA}_{t-1} * (1 - \alpha)) \tag{3.3}$$

$$\alpha = \frac{2}{1 + N} \tag{3.4}$$

Where t is the current time-step, N is the number of time steps and P is the current price. Whenever the MACD Line goes below the Signal Line, its an indication to buy the stock and when the MACD Line goes above the Signal Line, it indicates to sell the stock [13].

2.3.3 Bollinger Band

Bollinger Band is a volatility based indicator used for technical analysis of stocks.

$$\textbf{MiddleBand} = \text{20Days Simple Moving Average (SMA)} \tag{4.1}$$

$$\textbf{UpperBand} = \text{MiddleBand + ((20Days SD) of StockPrice * 2)} \tag{4.2}$$

$$\textbf{LowerBand} = \text{MiddleBand - ((20Days SD) of StockPrice * 2)} \tag{4.3}$$

Where, SMA is Simple Moving Average and SD is Standard Deviation of a particular time-frame. To analyze the results of Bollinger Bands, we look at the proximity of the stock price in relation to the Upper and Lower Bands. If the stock price is near or upper the Upper Band, the this indicates to sell the stock due to overbuying. On the other hand, stock price being close or lower the Lower Band, indicates buying of stock [12].

Figure 2.3: Apple Inc stock history from Sep 2013 - May 2014. Blue lines above and below the main signal are the upper and lower bands. The dotted line is the middle band.

2.4 Evaluation Metrics

For the purpose of evaluating the neural network model and the predicted outputs, several metrics are used. These can be defined into two categories, binary classification metrics and regression metrics.

2.4.1 Binary Classification Metrics

These are used to evaluate the performance of the binary classifier used for the directional prediction of stock movement.

Binary cross-entropy loss

Cross-entropy loss is a metric used to evaluate performance of a classifier that outputs probability between 0 to 1. The value of this metric, also known as the loss value, increases as the predicted probability diverges from the actual value. Binary cross-entropy loss is a special case in which the number of classifications are equal to 2. It is calculated as follows.

$$\mathbf{BCL} = -(y \log(p) + (1 - y) \log(1 - p)) \tag{5}$$

Where y is the binary indicator 0/1, if class label c is the correct classification for an observation o. p is the predicted probability for an observation o is of class c.

True Positives - TP

These are the correctly predicted positive values. For example, the predicted value is 1 and the actual value is also 1.

True Negatives - TN

These are the correctly predicted negative values. For example, the predicted value is 0 and the actual value is also 0.

False Positives - FP

These are the falsely predicted positive values. For example, the predicted value is 1 but the actual value is 0.

False Negatives - FN

These are the falsely predicted negative values. For example, the predicted value is 0 but the actual value is 1.

Accuracy

Accuracy is the simplest and most intuitive measure of binary prediction models. It is calculated as the ratio of correctly predicted values over total values.

$$\textbf{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{6.1}$$

Precision

Precision is the ratio of correctly predicted positive values over total predicted positive values. High precision value equates to a model with low false positives.

$$\textbf{Precision} = \frac{TP}{TP + FP} \tag{7.1}$$

Recall

Recall is the ratio of correctly predicted positive values over total actual positive values. High recall value equates to a model with high positive prediction rate.

$$\textbf{Recall} = \frac{TP}{TP + FN} \tag{8.1}$$

F1 Score

F1 Score is the weighted average of the two metrics defined above, precision and recall. For the most part, accuracy is the best evaluation metric but, if the class distribution is uneven, and the cost of both the classes are not even, F1 score is a better metric.

$$\textbf{F1 Score} = \frac{2 * Recall * Precision}{Recall + Precision} \tag{9.1}$$

2.4.2 Regression Metrics

These are used to evaluate the performance of the regression neural networks used for the prediction of stock prices.

Mean Square Error - MSE

For regression problems, Mean Square Error (MSE), is selected as the loss function. The equation is provided below.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{10.1}$$

Where $\hat{y}_i$ is the actual value, y_i is the predicted value. Each sample from the batch gives the square error. The mean of this error is taken over all n samples in the batch.

Root Mean Square Error - RMSE

RMSE measures the error between two function mappings. In my regression models, it shows how well the predicted stock prices fit the actual stock prices. The value represents the mean error along the set of data-points. RMSE is calculated as follows.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\Big(y\theta_i - y_i\Big)^2} \tag{11.1}$$

Where $y\theta$ is the observed value, y is the predicted value and n is the number of data points.

Mean Absolute Percentage Error - MAPE

MAPE is a measure of accuracy for regression based models and is used heavily in time-series forecasting. It expresses the error in percentage rather the the actual value like RMSE, so it becomes more suitable as a comparison metric. It is calculated as follows.

$$\text{MAPE} = \frac{100}{n} \sum_{t=1}^{n} \frac{|A_t - F_t|}{A_t} \%, \tag{12.1}$$

where A_t is real value and F_t is the corresponding forecast or prediction and n is the number of data points.

CHAPTER 3

MODEL DETAILS

For the task of stock prediction, I deployed four different machine learning models. These can be categorized into two categories. The first is the binary stock classification, in which I only predict the directional movements of the stocks (stock price for the next day will increase or decrease). Model M1, M2 and M4-A, fall into this category. For the regression models, I used two different models, M3 and M4-B. These are defined into sections below.

3.1 Model M1: Binary Classification using Multinomial Naive Bayes

Multinomial Naive Bayes (MNB) is used as a baseline for the sentiment classification problem. The main principle behind this approach is to calculate the class probabilities assigned to headlines using the joint probability of class and words. In this case, there are 2 classes, positive and negative into which we have to categorize the headlines.

3.1.1 Multinomial Naive Bayes calculation

The goal of the MNB is to calculate the probabilities of a headline belonging to each class. From this outcome, we can select the the class with the highest probability. The training and test data consists of 6189 headlines, from the New York Times Api, containing the top 5 IT corporations listed on the S&P 500 index. The MNB probability for each headline is calculated as below.

First we calculate the probability of each class j from the n classes available. In this case we have 2 classes.

$$\pi_j = \frac{class_j}{\sum_{n=1}^{2} class_n} \tag{13.1}$$

Now for each headline, we calculate the conditional probability of each word i occurring in a class j.

$$P(i|j) = \frac{word_{ij}}{word_j} \tag{13.2}$$

Or if the word i is not in the vocabulary of words V, we add a constant factor α. In the denominator we add the size of V and 1, as we assume that each word has occurred exactly one more time to accommodate the missing word. This is Laplace smoothing.

$$P(i|j)^* = \frac{word_{ij} + \alpha}{word_j + |V| + 1}, \alpha = 0.001 \tag{13.3}$$

The final probability for a headline occurring in class j is given as follows, where f_i is the total frequency of word i and t_i is the inverse document frequency (idf), from equation 3.2. Idf is used because to stop frequently occurring words from dominating the final probability. Log of probabilities is taken to avoid the under-flows.

$$Pr(j) = \log \pi_j + \sum_{i=1}^{|V|} f_i . \log(t_i . P(i|j)) \tag{13.4}$$

For each headline, the positive and negative probabilities are calculated and compared. The class with the greater probability determines the final sentiment of the headline. A sample is shown below.

Headline	P(pos)	P(Neg)	Result
New York's Amazon Deal Is a Bad Bargain	0.34	0.65	Negative
Google Adds the Human Body to Its Search Functions	0.56	0.43	Positive

Figure 3.1: Sample output from the MNB Sentiment Analyzer.

3.1.2 Results

This model was tested on 1200 unlabelled headlines (80:20 split ratio) from the New York Times. The results are compiled below.

Accuracy	Precision	Recall	F1-Score
0.531	0.574	0.504	0.568

Table 3.1: Results from the MNB classifier.

3.2 Model M2: Binary Stock Classifier using Word2vec and MLP-Neural Network

The second model is a binary stock classifier based on sentiment analysis using Word2vec. In order to create the classifier, first a language model has to be defined that take the pre-processed data and converts it into numeric form. This process is called Word Embedding for which I have chosen Word2vec. These word-embeddings will then be used to train a neural network model to predict stock price directional change using the labels we will define.

3.2.1 Word2vec

Word2vec is the embedding I choose for this task. The reason behind this is that it allows us to capture the context in which words appear in a text, all the while, it is a very efficient way of handling data through out the machine learning model. The way it works is that there is a unique vector created for every word in the corpus. Using this, a vector space is created with 200 dimensions, as shown in figure 3.1. The striking feature of word2vec is that vectors of word with similar meaning and context will be closer to each other in the vector space. Figure 3.2 shows the closeness of similar words.

```
      w2v['good']

Out[49]:
array([-1.04697919,  0.79274398,  0.23612066,  0.05131698,  0.0884205 ,
       -0.08569263,  1.45526719,  0.42579028, -0.34688851, -0.08249081,
        0.45873582,  2.36241221,  1.05570769, -1.99480379, -1.80352235,
       -0.15522274, -0.20937157, -0.07138395,  0.62345725,  1.50070465,
       -0.02835625, -0.08164574,  1.28233111,  1.75218308, -1.38450599,
        2.12812686,  0.8680591 ,  0.32937863, -0.72903335, -0.57105792,
        0.53118968, -0.39874691, -1.13426244, -1.43289971, -0.24573426,
        0.33088401, -0.88485849, -1.01581001, -0.62239277, -0.11345106,
        1.33353424, -0.49697381, -0.36537576,  0.76834393,  1.68711364,
       -1.03052866,  0.28044644,  0.41823646, -3.47753811,  0.13425322,
       -0.38362527,  2.05668998, -0.57867765,  1.93026328,  0.03931952,
        0.82015723,  0.11956126,  1.37940645, -0.47558281, -0.34119719,
        0.57716691, -1.48764825, -0.5627619 ,  0.55062455,  1.50399065,
        1.92141354,  0.68401223,  0.65160483, -0.0780897 ,  0.30544546,
        1.10016072, -0.78553891,  0.56758875, -0.1569887 , -0.65370613,
        1.4484303 ,  0.83104122,  1.25601828, -0.69895804,  1.50273371,
       -1.18243968, -0.11410122, -0.78283215,  0.49858055, -1.18107879,
       -0.27116439, -0.08542393, -1.55652511,  0.58338171,  1.57096207,
       -1.8447808 , -0.46724516, -0.38275295, -1.59960091,  0.84984666,
        0.04224969,  1.69833565,  1.54516482,  1.16857958, -1.36557281,
        0.71473658, -0.29552877, -1.55104351,  1.5771817 ,  0.29726478,
        1.40087354,  0.57571775,  0.16996922, -0.00522773,  0.06951592,
        1.81895649,  2.49026823,  0.93306369, -1.06985188, -2.24981689,
        1.37557173, -0.67554522,  1.10980988, -1.41456699,  2.3959322 ,
       -0.50968719,  2.00125957,  2.67398214,  0.1460651 , -1.47851145,
        0.87178862, -1.80060601, -0.53303391, -0.58677369,  0.53175789,
        1.96556151, -0.61592281, -1.42631042, -1.48672009,  1.13366914,
```

Figure 3.2: The 200 dimension vector for the word *good* in the vector space.

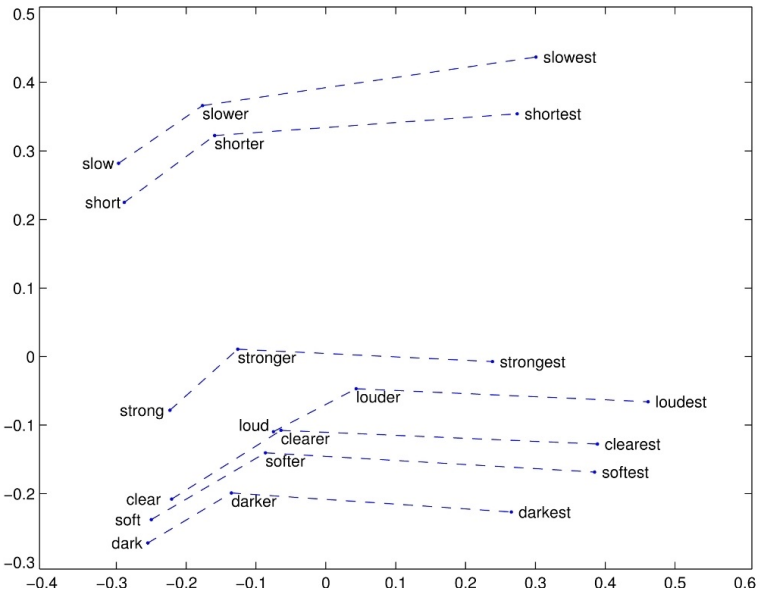

Figure 3.3: Visualization of a 2D vector space, with dimensionality reduced.

3.2.2 Sentiment Analyzer

The word2vec model has converted each word into a unique vector in the vector space. But now we need to represent whole headlines into a vector. For this we need to join all the words that a headline contains. For example, for the headline, *Google fire CEO*, it's vector would consist of combining separate vectors of *Google*, *fire* and *CEO*.

Tf-idf Score

A simple approach would be to sum these three vectors and take the average. However, a more novel approach is to take the weighted average of each word. This weight represents the importance of that word in the data-set. Tf-idf (term frequency-inverse document frequency) score can used to rank such vectors according to their importance. It is calculated as follows.

$$\textbf{tf-idf(word,headline)} = \text{tf(word,headline)} \cdot \text{idf(word)} \tag{14.1}$$

$$\textbf{tf(word,headline)} = \text{frequency of word in headline} \tag{14.2}$$

$$\textbf{idf(word)} = \log\ (1 + \frac{1 + \text{nh}}{1 + \text{df(headline,word)}}) \tag{14.3}$$

$$\textbf{df(headline,word)} = \text{number of headlines containing word} \tag{14.4}$$

$$\textbf{nh} = \text{number of headlines} \tag{14.5}$$

The tf-idf of a *word* in a *headline* is proportional to the number of times *word* occurs in the *headline* but is also offset by the frequency of the *word* in the collection of the headlines in the data-set. This takes cares of words that occur too frequently, such as stop words. Hence we only get the relevant words from the headline that carry the most weight.

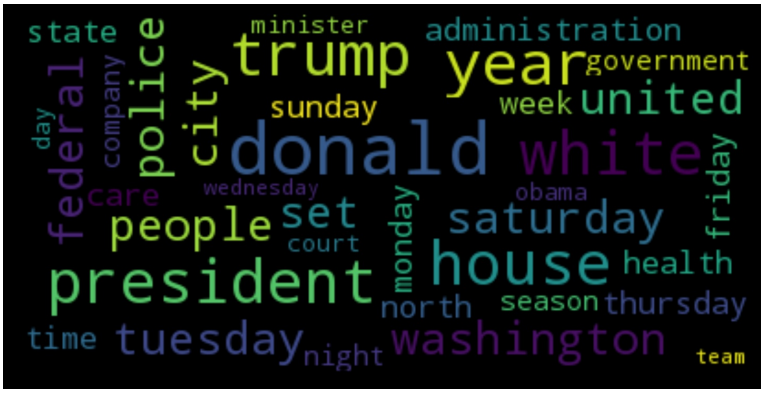

Figure 3.4: Word-Cloud plot where size of word represents tf-idf score of each word.

3.2.3 Stock Price, Headlines and Sentiment Co-relation

To label the data, there has to be some metric according to which positive and negative sentiments can be defined. For this purpose, previous stock price was used to gauge how well the stock has performed given its current price. If the previous stock price is lower than the current stock price, this indicates an increase in the stock valuation. The assumption here is that the positive sentiment from the news is driving this growth of the stock.

The headlines were labelled using this assumption. If a stock on day t has an increase in its price compared to its price on day *t-1*, the headlines on day t are labelled are 1 (showing a positive sentiment). Similarly, 0 would co-relate to a loss of stock price. This defines the model as a binary classifier, as it seeks to predict the loss/gain (0 or 1) for a stock price for a future date given it's current headlines.

3.2.4 Data-set split

The data-set split is 80:10:10, where 80% of the data is used for training and rest for testing and validation. Although the split used in [1] is 3:1:1, however, their training set consisted of 157,033 headlines, so they could use more headlines for test and validation to achieve a greater accuracy. In my model, this split was

found to be optimum as it resulted in minimum variance in parameter-estimation and performance statistics, given the number of instances. The data-set consists of 6189 financial news headlines from the New York Times Api for the top 5 IT corporations listed on S % P 500 index (Google, Microsoft, Facebook, Apple Inc, Amazon). Each headline in the training set is labelled as a 0 or 1.

3.2.5 Neural Network Classifier

All the headlines are converted into averaged vectors using the tf-idf scores of individual word vectors. These vectors are then fed into a Artificial Neural network for training.

Design and specifications

A simple 2-layer neural network is defined using Keras Sequential API. The design and specifications of the optimum model achieved is provided in figure 3.5 below.

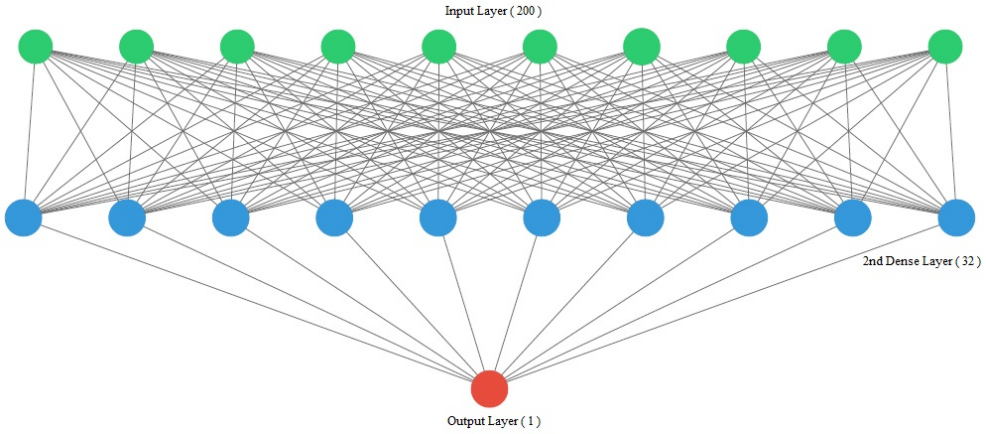

Figure 3.5: Neural Network architecture for model M2.

The input layer takes in an array with 200 dimensions according to the size of our vectors defined using Word2vec. The first layer is the dense layer which means that each neuron from this layer are fully connected to the subsequent layer. The layer outputs a 32 dimensional array (This dimension number is subject to experiment but is usually in a power of 2). The second layer is also a dense layer. It

takes in the 32 dimensional input from the first layer and outputs a 1 dimensional array.

For the activation functions, ReLu, Sigmoid and Softmax were chosen for the 2 layers. Sigmoid worked the best (based on results shown below) and was chosen for both the layers as this was a binary classification task, with few layers. As there were no hidden layers, further options were not explored.

Training Parameters

For the loss function, binary-cross-entropy was selected (see equation 10.1). Loss function shows how good or bad the predicted values are and based on that, the parameters are updated. As this is not a regression task, we don't need to use Mean Square Error here, hence binary-cross-entropy was the obvious choice.

For the optimizer, Adagrad was chosen with the intial learning rate set to 0.02. The data from the tf-idf is sparse as the more important words that carries more weight are fewer compared to the words that are frequent and do not represent the overall sentiment. Adagrad provides more learning rate to the weights that are not updated frequently.

Batch size is set to 10 and epochs are 100. Due to the small data-set size, I experimented with smaller batch sizes to get a faster convergence on the loss.

Results

For the results, I focused on the following 4 metrics to evaluate my predictions. The results are summarized below.

Accuracy	Precision	Recall	F1-Score
0.587	0.643	0.605	0.623

Table 3.2: Results for Model M2.

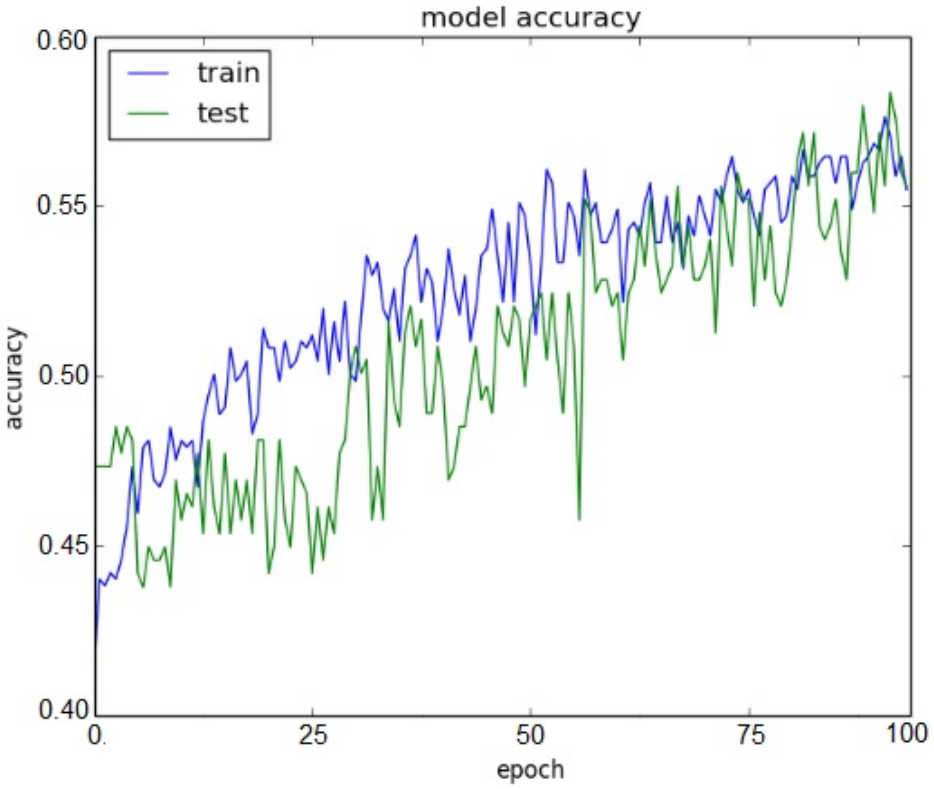

Figure 3.6: Accuracy graph of model M2.

3.3 Model M3: Stock Price prediction using VADER Lexicon and LSTM Neural Network

Sentiment Analysis is a difficult task and it is still an open-ended question in NLP because language is very subjective. A lot of context is lost in text, as words alone cannot determine the sentiment behind them, so computers can be misled. Furthermore, a piece of text can contain long dependencies between two concepts being described. More than one sort of sentiments can also be present in a text, which can further mislead the analysis model.

For example, the sentence, *Google's performance was average at best in the second quarter*, can lead the MNB model to output a positive sentiment for it. This is due to the word *best* occurring in the sentence which is usually associated with positive sentiment but in this case, it is not necessarily pointing towards a positive sentiment.

To overcome this problem, I used NLTK's Vader sentiment lexicon, which is a

predefined open source library for sentiment analysis. I used the sentiment scores generated from Vader with a LSTM neural network, in hopes to learn the long range dependencies within text. This leads to prediction of the actual stock prices and not just the directional change.

3.3.1 Vader Sentiment Analysis

VADER sentiment analysis works by mapping a sentiment score to every lexical feature encountered in a text. A word, punctuation mark and even an emoticon, is considered as a lexical feature and contributes to the overall sentiment of the sentiment. These sentiment scores are predefined and normalized from a large group of people that ranked the sentiment for each lexical feature, in order to normalize any bias.

Vader Scores

Consider the sentence, *this bike is awesome!*. The Vader Polarity Scores for this sentence are given in the table below.

Positive	Neutral	Negative	Compound
0.769	0.303	0.0	0.783

Table 3.3: Vader Sentiment Polarity Scores

- The positive, negative and neutral scores range from 0 to 1 and represents the proportion of each sentiment in the sentence.
- The compound score returns a value from -1 (most negative) to +1 (most positive). It calculates the sum of all the 3 scores from above and normalizes them using the equation below by Hutto [14].

$$\frac{x^2}{\sqrt{x^2 + \alpha}} \tag{15.1}$$

where $\boldsymbol{x}$ represent the sum of all the scores in the text that are normalized

between 0 to 1. α is a normalization parameter with a value of 15. A compound value of 0.783 represents a very positive sentence.

3.3.2 Pearson correlation between stock prices and sentiment score

The sentiment score for each headline is calculated using the Vader sentiment library and fed into a data-frame as shown below.

	prices	compound	neg	neu	pos	Headline
0	312	0.4767	0	0.849	0.151	In Allowing Ad Blockers, a Test
1	305	0.4019	0	0.912	0.088	Google Moves to Keep Its Lead as

Figure 3.7: Vader sentiment scores

Before I begin with my machine learning model, I need to visualize and analyze the co-relation between with change in stock price and the sentiment scores. Below is a graph which shows Google's Series A stock against the Vader sentiment score calculated on New York Times Headlines on Google in the same time-frame.

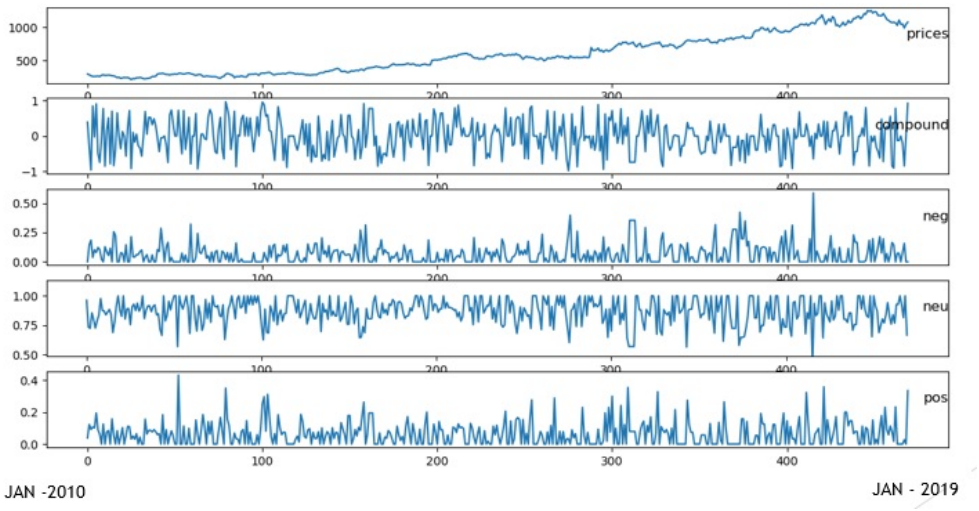

Figure 3.8: Visualization of sentiment scores with respect to stock price. Stock data from Google Series A stock from Jan 2010 - Jan 2019.

To make sense of this data, I used Pearson correlation coefficient, to get some initial idea of the relationship between compound scores and the stock price. As the compound score should increase with the increase in stock price and decrease

with the decrease in stock price, some linearity should be reflected in this analysis. The formula used is defined below.

$$p = \frac{N \sum XY - (\sum X \sum Y)}{\sqrt{\sum x^2 - (\sum x)^2][N \sum y^2 - (\sum y)^2]}} \tag{16.1}$$

Where X is the stock price, Y is the corresponding compound score and N is the number of samples.

Pearson correlation test returns a value from -1 to 1, with positive values showing direct proportionality between the 2 variables and the negative values showing inverse proportionality. The results of the Pearson Correlation test in this case was 0.2064. This means that the two variables, stock price and compound scores are directly related but the relationship is weak. In general, a value of 0.7 and above is considered good, but this is subjective. This at least confirms that there is not a sufficient linear relationship present between the 2 variables and non-linearity must also be taken into account.

Similarly, this test was also conducted for positive, negative and neutral sentiments for which the p value came out to be 0.334, 0.275 and 0.119 respectively. On the basis of this result, the neutral value was discarded as it had the least impact on the overall stock prices among the 4 sentiment features.

3.3.3 LSTM neural network model for Vader scores

This model takes the input from the Vader sentiment library and trains on the compound, negative and positive scores. It also takes in the previous day stock price. The predicted value is the stock price, given the Vader sentiment scores from the headlines and the stock price from a previous day.

Converting into a supervised learning problem

Using the model from [15], I frame the supervised learning problem as, predicting the stock price for the current day t given the sentiment scores from headlines gathered at a previous time-step, day *t-1*. Similarly the stock price from day *t-1*

is extracted. This way we can transform the data-set from series to supervised. A sample for the data is given in the figure below.

prices (t)	compound (t-1)	neg (t-1)	neu (t-1)	pos (t-1)	prices (t-1)
312	0.4767	0	0.849	0.151	305
319	0.4019	0	0.912	0.088	312

Figure 3.9: Training Data-set sample. t is the current day, while t-1 is the previous day.

Design and specifications

A 2 layered LSTM neural network is implemented. It is defined using Keras Sequential and LSTM API. The design and specifications of the optimum model achieved is provided in figure 3.10 below

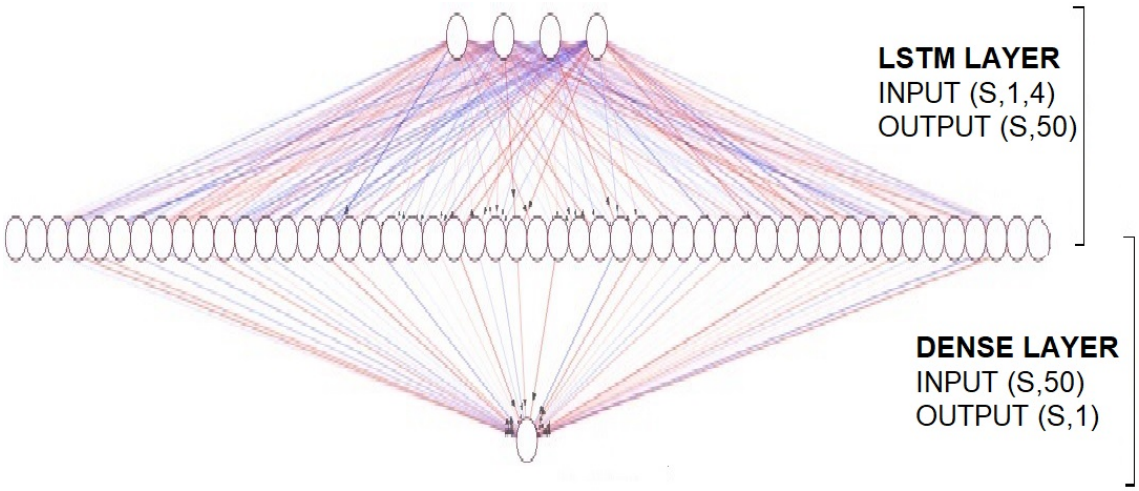

Figure 3.10: LSTM architecture for Model M3.

The input LSTM layer takes a 3D input equivalent to (samples size, time-steps,features). In this case, we only have 1 time-step as we are only considering previous day prices. Features are 4 according to the 3 sentiment scores from the Vader library and the previous day stock price. The LSTM layer outputs a 2D array of size (samples, 50) which becomes the input for the Dense layer. This is the output layer that outputs in the form (sample size, 1).

ReLu is chosen as the activation function for the LSTM layer. While linear activation function gave the best result for the output layer.

Data-set split

Different data-splits were experimented such as 80:10:10, 60:20:20 and 70:15:15. Based on my relatively small data-set (LSTMs benefit from very large data-sets) of 10000 samples (Headlines from Google from Jan - 2010 to Jan - 2019) compared to the one used in [5], a split of 80:10:10 was selected. For larger data-sets, it is better to have more room for test/validation.

Training Parameters

As this is a regression problem, Mean Square Error (MSE), is selected as the loss function. For the optimizer, Adam and RMSprop, both seem to perform equally well with MSE loss function and little tuning was required to the parameter. The learning rate was set at 0.001. Batch size is kept at 72 and the number of epochs is 50.

Results

The RMSE value obtained from this model is 45.45 US$. This is a very high error value which is reflected by the equally unimpressive MAPE score of 5.42%. Compared to the results from similar studies, such as [9], they had the best MAPE score of 1.83%.. However, this model acts as a good baseline for my final regression model M4.

Figure 3.11: M3 result: Real vs Predicted price for Google stock from Jan - 2017 to Jan 2019.

However, to evaluate the accuracy on par with the other directional models, directional change is also calculated. Directional change is calculated as follows.

- For each price n predicted on day t, the predicted price $n\theta$ on day *t-1* is subtracted.
- The same procedure is repeated for the actual prices.
- If the value is positive, a positive change occurred and there was a price gain, otherwise, its a price loss.
- A value of 1 is assigned to all the prices (predicted and actual) if the change is positive, otherwise 0 is assigned.

After this process, the directional metrics can be used to evaluate this model. Below are the results from this Model.

Accuracy	Precision	Recall	F1-Score
0.521	0.544	0.568	0.558

Table 3.4: Directional results for Model M3.

3.4 Model M4: LSTM NN Regression Model using Word2vec

For the final model, I performed a regression based analysis in which I combined the economic indicators with the sentiment analysis model. This is an experimental model as there has not been much work that is done related to it. Most studies tend to either go for completely textual or completely technical analysis.

What previously worked for me well is the model M2 that used Word2vec. Following this direction, I built upon the model M2 and instead of using a MLP, I decide to go for a LSTM NN for directional prediction.

3.4.1 Model M4-A: Binary sentiment classifier using Word2vec embedding from Model M2 and LSTM NN

Previous attempts at sentiment classification has been very domain specific. The Model M3 which used Vader library, has predefined scores for each textual feature and MNB works on occurrences of words based on class. These was certainly a need for a universal approach that could work well in the context of financial news. As words tend to work in conjunction with each other (a single occurrence of word does not likely define the sentiment of a whole text), for this purpose I decided to further my model M2 which used Word2vec for word embeddings, and train it on a LSTM NN as opposed to a Multi-Layer-Perceptron (MLP). This is done to learn the dependencies among words that are crucial to define the sentiment of a sentence.

Word2vec embedding from Model M2

Since the task of converting the headlines into vectors was achieved in model M2, those word embeddings are used here with the same data-set. These embeddings

are in a matrix, such that each word maps onto its embedded vector. This will be used as in input into the LSTM for sentiment classification (positive and negative).

LSTM NN architecture

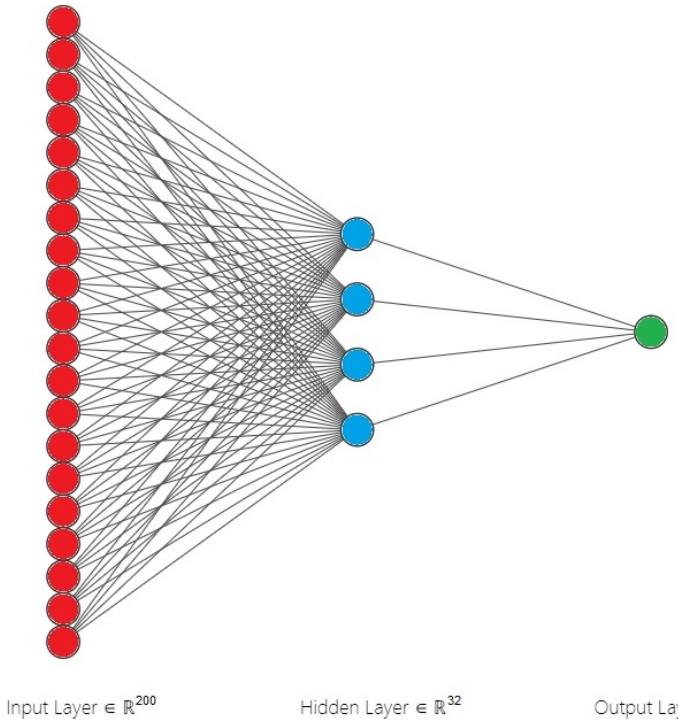

Figure 3.12: LSTM NN architecture for sentiment classifier.

The following LSTM NN is defined using Keras's LSTM Api. It consists of the following layers:

- Embedding Layer: It is Input layer that that maps each word into a vector. It takes a 3D input of size (Sample size, max length of input = 2678, embedding dimension = 200).
- LSTM Layer: Hidden layer that takes input from the embedding layer and outputs a 2D array of size (Sample size, 32).
- Dense Layer: Output Layer that consolidates the input from LSTM layer into a value between 0 and 1 using sigmoid function for the all the values in the sample.

Data-split and training Parameters

To get consistent results, the data-split was kept the same at 80:20. For the loss function, *binary crossentropy* was chosen, as defined in equation (5). *Adam* was again chosen as the optimizer, as it showed the fastest loss convergence. Another reason for faster model performance was that the embedded layer used pre-trained word embeddings, so this layer was not trainable. Epoch, batch-size and learning rate were kept at 50, 64 and 0.01 respectively. *Sigmoid* was chosen as the activation function for this binary classification task.

Results

This model performed the best by far among the binary classifiers. The combination of word2vec word embeddings and LSTM NN worked well as expected. Theses results are further discussed in section 4.

Accuracy	Precision	Recall	F1-Score
0.659	0.675	0.682	0.677

Table 3.5: Binary classification results for Model M4-A.

3.4.2 Model M4-B: LSTM NN for stock prediction using regression

The results from the regression model M3 were not up to the par with the more recent results on stock prediction using regression [16]. For this reason, the final regression model borrows it's sentiment analyzer from the binary classifier model M4-A. The idea is that, the model will take as input, the predicted binary sentiment score as an input feature for the predictions. However, in the case of training, I can already calculate the direction change in the historical stock price, so that is not taken from the model's output. This is done to improve the training accuracy. Another noteworthy feature is the addition of momentum based economic indicators. These indicators can capture the stock trend over a certain period of time. The LSTM NN can benefit from this fact as it is specially designed to learn

time series data. These indicators will be supplemented with the binary sentiment score.

LSTM NN architecture

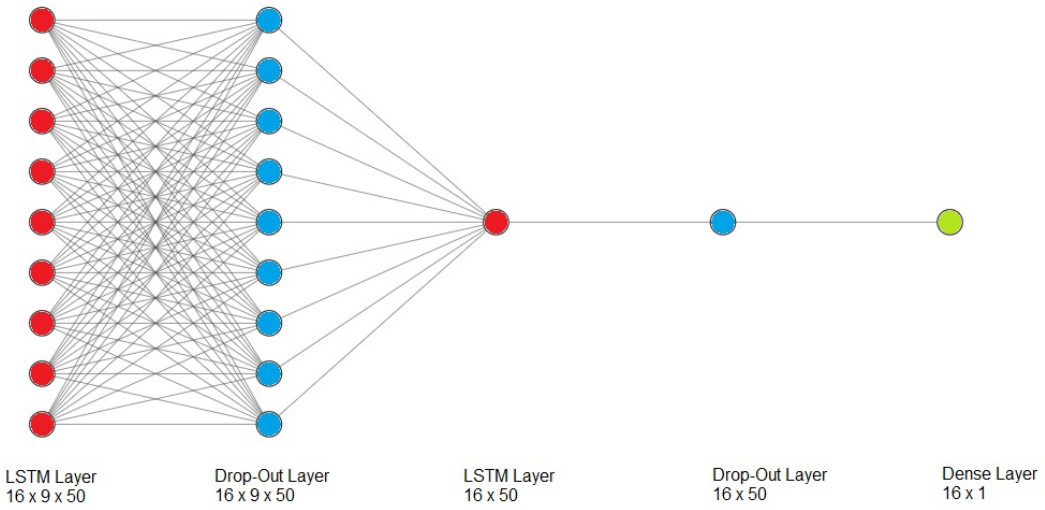

Figure 3.13: LSTM architecture for stock price predictor.

The LSTM consists of 5 layers:

1. LSTM Layer: Input layer that takes a 3D array consisting of *16 batches*, *9 time-steps* and *50 sequence length* .
2. Drop-out Layer: Outputs the same 3D array as input.
3. LSTM Layer: Outputs a 2D array consisting of *16 batches*, *9 units*.
4. Drop-out Layer: Outputs the same 2D array as input. regularization.
5. Dense Layer: Outputs a 2D array consisting of *16 batches*, *1 unit*.

Data Preparation and split

The input data consists of 10,000 Google headlines scrapped from NewsApi using the data scrapper. The headlines run from Jan - 2010 to Jan 2019. Corresponding daily-adjusted-closing stock prices for Google's Series A stock are also loaded.

Momentum indicators, RSI (Relative Strength Index) for 12 days, MACD (Moving Average Convergence Divergence) for 12 days and Bollinger-bands for 20 days, are also calculated for the daily stock price. Lastly, sentiment classification from the Model M4-A classifier, is also taken for predictions to form a feature set as shown below.

Figure 3.14: LSTM architecture for stock price predictor.

Data is split into a 80:10:10 ratio. This is the optimum ratio used in all previous models.

Parameter tuning

Parameters were selected based on the RMSE value (see equation 11.1) from predicted and actual prices. This was done on the validation data-set to select the parameters options that would output the least RMSE value. Below is a list containing different parameters and the different values that were considered. The best performing parameters are shown in the figure below.

Parameter	After Tuning	From
N	30	range(3,50)
LSTM Units	64	10, 50, 64, 128
Dropout Prob	0.6	0.5, 0.6, 0.7, 0.8, 0.9, 1
Optimizer	adamax	adam, sgd, rmsprop, adamax, adadelta
Epochs	60	1, 10, 20, 30, 40, 50, 60
Batch Size	16	8, 16, 32, 64, 128

Table 3.6: Optimum parameter values from the tuning set.

For the activation functions, *ReLu* is selected for the LSTM layers. Linear activation function is the default choice for the output layer. For the loss function, *mean-squared-error* is used.

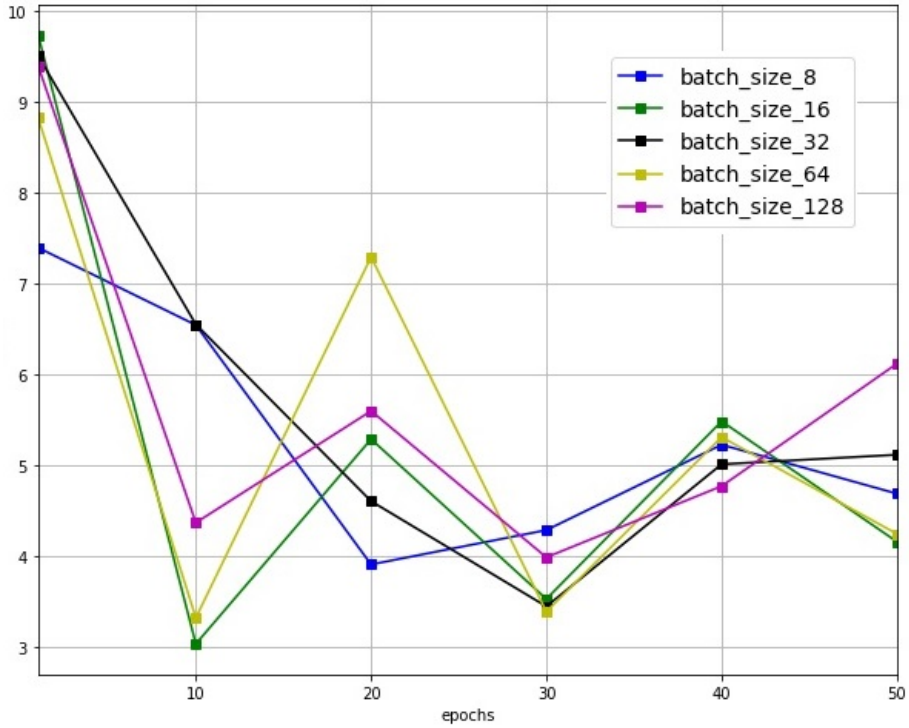

Figure 3.15: Selecting best performing Batch size based on RMSE.

.

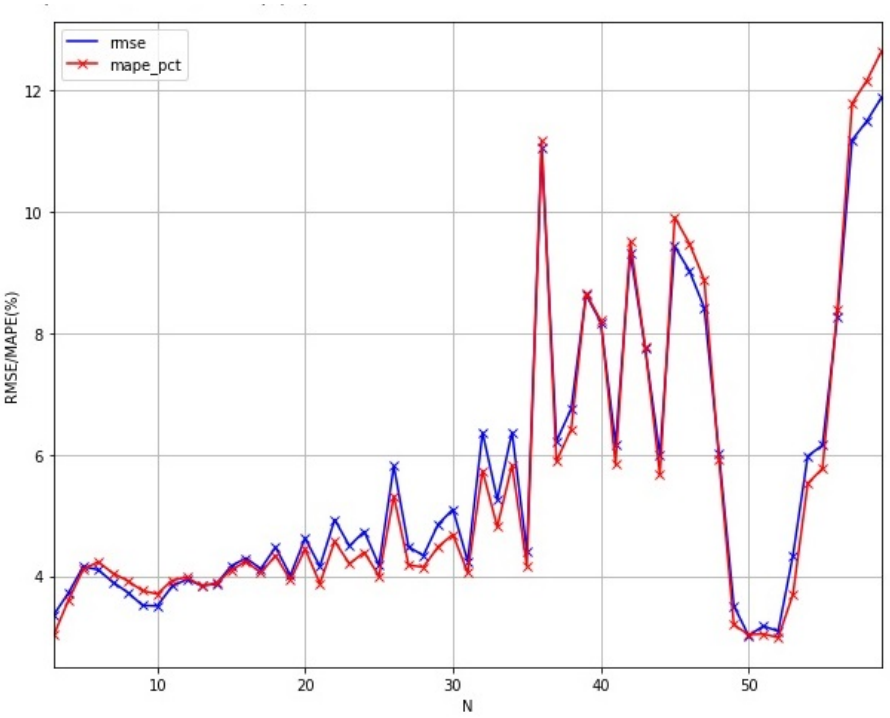

Figure 3.16: Selecting best performing N based on RMSE.

.

Results

This model performed quite impressively. It achieved a RMSE score of 6.73 US $ and a MAPE score of 5.42%, beating both, the model M3 and the MAPE score of 1.83% from [9]. The results are further discussed in section 4.

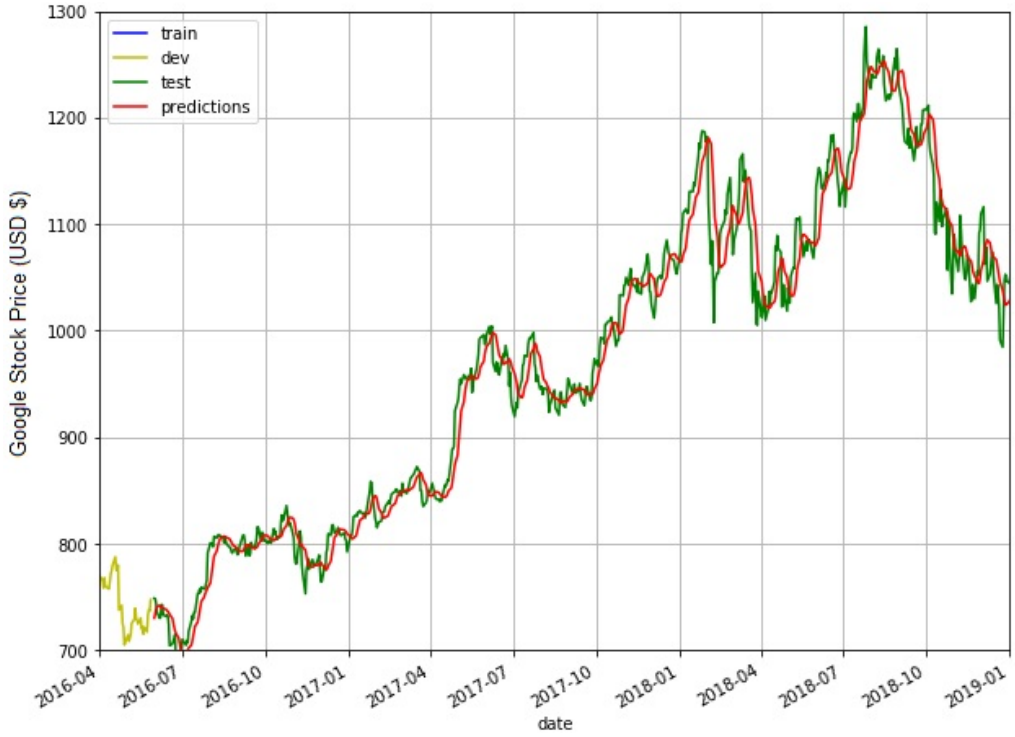

Figure 3.17: Prediction over actual stock price graph.

Accuracy	Precision	Recall	F1-Score
0.622	0.602	0.646	0.642

Table 3.7: Binary classification after conversion using method on page 31.

CHAPTER 4

DISCUSSION

The results from the models are categorized into main categories. The first category is the binary directional classification results which only measures the directional change of the stock market the next day. The second category is the stock price prediction using regression which estimates the stock price rather than just the directional result. These results are summarized below.

4.1 Binary classification results

Binary Directional Classification				
Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
M1	0.531	0.574	0.504	0.586
M2	0.587	0.643	0.605	0.623
M4-A	0.659	0.675	0.682	0.677

Table 4.1: Results for binary directional classification.

Model M1 uses a Multinomial-Naive-Bayes model to predict the sentiment from a headline. This model has been previously shown to work well with twitter sentiment classification as shown in [17], achieving 80% classification accuracy. This was the motivation behind using this model, however stock market headlines contain context specific words and sentence structure. Hence, singled out words alone do not necessarily define the overall sentiment of a sentence. A bi-gram model was used, but this did not sufficient captured the semantic structure of the entire headline. The accuracy of 53% still indicates a positive result compared to the directional accuracy of 57% achieved by [1]. These results show that the stock direction is predictable. Subsequent models use these results as a baseline to be measured against as this is the simplest and purely statistical model.

In this regard, Model M2 is a considerable improvement from Model M1. M2 uses Word2vec word-embeddings that transforms text into a vector representation. The idea was that these embeddings will help to capture the context that arise from different arrangement of words. As sentence structures tend to form long range dependencies, coming from Model M1, it is very naive to consider that they don't. This idea was further cemented by using neural networks for predictions. [18] cites several advantages of using neural networks over statistical approaches with context-based learning being one of them. This improvement is reflected in the results from Model M2 with a greater accuracy of 0.587, performing better than the bag-of-words model of [1]. A significantly greater precision than accuracy is indicating that some data (headlines) are classified as x but has words more associated with class y. This limitation is there as the headlines are significantly shorter in size compared to articles.

Model M4-A builds upon the M3 by using the word-embeddings and training them on a LSTM NN rather than a MLP (Multi Layer Perceptron) as the case is in [6]. RNNs (LSTM's super-class) have proven better performance in NLP related tasks over other neural over neural network architectures, due to sequential nature of text [19]. The results obtain match the best directional accuracy obtained by [4] at 66%. These are the best binary classification results obtained in my research. This is still considerably lower than the best directional accuracy achieved by [20] of 87%. The limitation, I believe occur from the homogeneous nature of data as it only comes from one source, New York Times. However, [20] focuses on the whole Dow Jones Industrial Average (DJIA) index, whereas my analysis focus on the top 5 IT corporations listed on the S&P 500 index. Hence, their data-set should contain very less homogeneity as different stocks perform differently, and most of the time, are inversely related within the same sector.

4.2 Regression results

Stock Price using Regression		
Model	**RMSE**	**MAPE**
M3	45.45 $	5.42%
M4-B	6.73 $	1.15%

Table 4.2: Results for stock price using regression

The first regression model M3, uses NLTK's Vader sentiment lexicon that gives the sentiment scores for a given piece of text. These scores were calculated for every headline and used as features along with the previous day stock prices, for training the LSTM NN. The idea here was the the previous day stock prices will indicate the momentum of the stock price and the noise in the stock price will be indicated by the sentiment going on around that particular stock. The results from this model were not very optimistic. [20] achieved a lesser MAPE of 1.83% compared to 5.42% from my analysis. This can be attributed to several factors, however the major factor that I want to highlight is that the Vader scores are very generic scores. They are more suited for generic bodies of text that are textually very simple. Headlines are often misleading and contain double meaning, so the scores from the Vader lexicon do not necessarily reflect the sentiment generated by the headlines. This fact is further proven by the low value of pearson-correlation-coefficient between the Vader scores and the stock prices. The RMSE value for this model is 45.45 US $. This is a relative score and can't be used like the MAPE value to compare with other studies that I have reviewed. So this, RMSE score is used as a basis for comparison between this and the subsequent model M4-B, as the dataset is shared between these two models.

The final model M4-B, builds upon the models M4-A and M3. It follows a similar architecture to [21], where they incorporated textual and technical data to

form a single normalized feature to become the input of an LSTM NN. My model combines the binary output from the M4-A and uses that as a feature when it makes the next day prediction. It also incorporates momentum based - technical indicators to capture the trend of the stock price more precisely than M3. The results are very promising with the RMSE value of 6.73 US $, it performs almost 7 times better than model M3. This is reflective in the predicted vs actual stock price graph (figure 3.17), where two lines more closely map to each other. The MAPE value of 1.15% is lower than the best of 1.86% achieved by [20]. This can be attributed to the inclusion of momentum based indicator as suggested in [6]. One striking feature in this model is the N parameter, that defines the number of time-steps that are used to make the next prediction. Figure 3.16 plots different N values over RMSE scores for model M4-B. This was done to select the best perform N value. Model M3 had an N value of 1, compared to 30, of this model. This is an important distinction between the 2 models as this value has to be set optimally. As mentioned in [6], news flashes such as headlines takes time to propagate and impact the stock performance of a commodity. Too low of an N value can negatively affect the machine learning model. That is why they achieved greater inter-day accuracy compared to intra-day. However, too big of an N value can also increase the predication errors as forecasting starts to use already predicted values.

4.3 Regression to Binary classification results

Binary Directional Classification				
Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
M3	0.521	0.544	0.568	0.558
M4-B	0.584	0.592	0.609	0.599

Table 4.3: Results for binary directional classification after conversion from regression results using the method on page 31.

The predictions from the two regression models, M3 and M4-B, are transformed

to binary outputs, indicating directional flow of stock prices compared to the previous day. This is done to compare the results from the regression models with the directional models along similar evaluation metrics. The process of this calculation is explained in equation XXX. The results for the model M3 are not as optimistic as expected from the low RMSE and MAPE achieved by the model. It performed worse than the baseline directional model M1, with an accuracy of 52% at best. The model M4-B performed much better with an accuracy of 58% and similar scores for the rest of the metrics. However, these scores are at par with the Model M2, which is the second best performing directional model.

These results are definitely interesting and led to two important discoveries. The first is that, at low MAPE scores, there is not a strong enough co-relation between MAPE scores and directional accuracy. This means that even if a model achieves a near 1% MAPE score, it cannot be claimed for certain that the model is a reliable directional indicator. This claim is further proven by the results achieved in [9], according to which, their best accuracy of 86% was achieved by the 3rd best MAPE score of their model.

Another important realization at this point is that, both, regression and directional models achieve different purposes and should be used according to the requirement of the application. In hindsight, the directional model M4-A, has performed the best according to the measure of accuracy. However, regression models provide the bigger picture in the data. They have the capacity to predict the future trends and in the context of prediction of stock prices, these are valuable insights for investors that are buying/selling on a day-to-day basis. Given that the daily price variation for Google's Series A stock is 9.7 US $ [22], a RMSE of 6 US$ is quite reasonable. Measuring these models on the basis of the the directional metrics used above, does not provide a complete picture of their performance. The question that leads these results is that, what algorithm is used for trading purposes that implements the above predictive models.

CHAPTER 5
CONCLUSION AND FUTURE DIRECTIONS

In this paper I have shown that a strong co-relation exists between stock prices and the sentiment extracted from textual data regarding a particular stock. For this purpose, I used a combination of several language and machine learning models to best predict the stock prices based on the direction of stock movement and the actual stock prices.

The best performing model for directional classification of stock prices, is the model M3, which uses news headlines represented in a vector space using word2vec form. This form of representation allowed the model to learn dependencies present among words in a headline, a feature that other language models lacked. For machine learning, LSTM's outperformed Multi-nomial Naive Bayes and Multi-layered Perceptrons, with a best of 65% directional accuracy. The regression based analysis followed a similar pattern, with the combination of word2vec as the language model and LSTM NN as the machine learning model, producing the best MAPE score 1.15%.

In terms of future direction, one particular field of research is algorithmic trading using the results from my prediction models. Simulated trading, is one such domain that can be looked upon as presented in [1], that used it as an evaluation metric for their stock prediction models. In my case, I can build this algorithm upon the technical indicator that I have already incorporated in my analysis, Essentially, it will trigger a buy/sell signal upon reaching a certain threshold value. This can further be used as a common metric to evaluate the best performing regression and directional models.

It would also be advantageous to expand the variety of stocks selected for prediction. Stocks listed on S&P 500 index tend to be interlinked, especially if they belong to a similar category. These entire groups can be targeted, which will

not only lead to a bigger textual data-set, but also the LSTMs NN can benefit from a wider pool of technical data, such as stock performance of rival stocks.

Finally, I want to leave the readers with some caveats. Surely the results of my research are interesting and ambitious, however, I do recognize the fact that they come from a small data-set of news headlines. Incorporating a greater variety of textual-data, not limited by length of text, source of news, geographical location and language, would aid removing any biases present in the market associated with a packed time-frame. For example, the impacts of cyclic stocks, income reports, mergers/acquisitions and other sudden shocks.

Appendices

APPENDIX A

SOURCE CODE

Source code is uploded on github. Use the following link:

github.com/talalz94/FYP—Prediction-of-Financial-Commodities-using-AI

REFERENCES

[1] R. P. Schumaker and H. Chen, "Textual analysis of stock market prediction using breaking financial news: The azfin text system," *ACM Transactions on Information Systems (TOIS)*, vol. 27, no. 2, p. 12, 2009.

[2] E. F. Fama, "The behavior of stock-market prices," *The journal of Business*, vol. 38, no. 1, pp. 34–105, 1965.

[3] B. Qian and K. Rasheed, "Stock market prediction with multiple classifiers," *Applied Intelligence*, vol. 26, no. 1, pp. 25–33, 2007.

[4] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Deep learning for event-driven stock prediction," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[5] D. M. Nelson, A. C. Pereira, and R. A. de Oliveira, "Stock market's price movement prediction with lstm neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2017, pp. 1419–1426.

[6] L. dos Santos Pinheiro and M. Dras, "Stock market prediction with deep learning: A character-based neural language model for event-based trading," in *Proceedings of the Australasian Language Technology Association Workshop 2017*, 2017, pp. 6–15.

[7] H. GÜNDÜZ, Z. Cataltepe, and Y. Yaslan, "Stock daily return prediction using expanded features and feature selection," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 25, no. 6, pp. 4829–4840, 2017.

[8] D. Kahneman and A. Tversky, "Prospect theory: An analysis of decision under risk," in *Handbook of the fundamentals of financial decision making: Part I*, World Scientific, 2013, pp. 99–127.

[9] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of computational science*, vol. 2, no. 1, pp. 1–8, 2011.

[10] K. Chen, Y. Zhou, and F. Dai, "A lstm-based method for stock returns prediction: A case study of china stock market," in *2015 IEEE International Conference on Big Data (Big Data)*, IEEE, 2015, pp. 2823–2824.

[11] A. Mittal and A. Goel, "Stock prediction using twitter sentiment analysis,"

[12] V. Parikh and P. Shah, "Stock prediction and automated trading system,"

[13] G. D. I. Anghel, "Stock market efficiency and the macd. evidence from countries around the world," *Procedia economics and finance*, vol. 32, pp. 1414–1431, 2015.

[14] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Eighth international AAAI conference on weblogs and social media*, 2014.

[15] J. Brownlee, *How to convert a time series to a supervised learning problem in python*, 2017.

[16] M. A. Shah and C. Bhavsar, "Predicting stock market using regression technique,"

[17] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision,"

[18] J. A. Benediktsson, P. H. Swain, and O. K. Ersoy, "Neural network approaches versus statistical methods in classification of multisource remote sensing data," 1990.

[19] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of cnn and rnn for natural language processing," *arXiv preprint arXiv:1702.01923*, 2017.

[20] B. Liu, "Sentiment analysis and opinion mining," *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[21] R. Akita, A. Yoshihara, T. Matsubara, and K. Uehara, "Deep learning for stock prediction using numerical and textual information," in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, IEEE, 2016, pp. 1–6.

[22] M. Dybek, *Alphabet inc. (goog) — capm*, 2019.