Stock market prediction using NLP classification and LSTM model: A comparitive approach

Mihir Padsumbiya
Department of Civil Engineering
Institute of Technology, Nirma University
Ahmedabad, India
17bcl062@nirmauni.ac.in

Proffesor Tejal Upadhyay
Department of Computer Science & Engineering
Institute of Technology, Nirma University
Ahmedabad, India
tejal.upadhyay@nirmauni.ac.in

***Abstract*—**News headlines affect investor psychology that leads to sudden uprise or plunge in the market and can generate huge profits or loss for stock market investors. However, predictive analysis based on technical indicators used in the past is unable to detect patterns based on headlines, which generate the need for sentimental analysis. This research is focused on a comparative analysis between different Natural language processing classification models when trained on news headlines to detect patterns on stock movement and make predictions. Further, an LSTM model is proposed that utilizes technical indicators as well as market sentiment generated using news headlines over a 30-day period, to generate buy/sell calls. It can be concluded that pattern recognition based on Natural Language Processing classification models can be deployed to capture sudden movements in the stock while a time-series analysis using both sentimental and quantitative data in an LSTM model generate better predictions overall.

***Keywords*—**LSTM, Natural Language Processing, Stock Market prediction, Classification, Sentiment Analysis, Neural Network, Text analysis

Abbreviations-NLP: Natural Language Processing, BERT: Bidirectional Encoder Representations from Transformers, LSTM: Long Short Term Memory, RNN: Recurrent Neural Network, CNN: Convolutional Neural Network, SVM: Support Vector Machine, LM: Language Model, ADI: Accumulation Distribution Indicator, ASI: Accumulative Swing Index, MACD: Moving Average Convergence Divergence, MFI: Money Flow Index.

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

1. INTRODUCTION

The process of interpreting, analyzing, and predicting the movement of stocks in the future based on the data available from the past to minimize loss and maximize profit, is termed as stock prediction. Institutions and investors use this process worldwide to predict stock movements well in advance for better decision-making and correct trades. Stock prediction is one of the main tasks a stock market trader performs daily. A company's stock price is affected by both direct and indirect factors and can change its state at any given minute of the trading session. Such a chaotic, sophisticated, and dynamically changing environment calls for proper analysis and staying updated with all the happenings in the world that may affect the stock. Until recent times, a set of fundamental indicators and technical indicators based on past data were used for analyzing the stock and making predictions. Fundamental indicators provide an overall outlook of the stocks earnings and the companies valuation, the company's peer-to-peer stock price comparison and standpoint in the market against competitors, the company's market cap at the current price, and other such factors that are primarily qualitative.

On the other hand, technical indicators are all about the stock movement itself and provide buy/sell calls based on some quantitative calculations on the stock price's past data. Although fundamental and technical indicators are pretty insightful predictions, the very nature of stock markets makes it difficult to get accurate predictions. Apart from fundamental and technical factors affecting the stock movement, other factors such as politics, inflation, company's management dynamics, leadership roles, trend, seasonality, investor psychology, overall market sentiment, and other similar social factors affect the markets daily. Hence, this makes it particularly difficult to produce accurate intra-day calls. Such factors affect the market mainly through the medium of news headlines, which can change investor sentiment.

More than often, such news headlines are related to political dynamics of the country, the country's external affairs and policies that affect international trade, current economic aspects of the country and the world, company's management, product breakthroughs, mergers and acquisitions, natural disasters, Human rights' group activities, and most importantly, company's quarterly earnings report. These headlines fuel investor emotions and future expectations from the company. The overall sentiment is developed in the market for the company that may last for a few days to a full quarter of the business year, or even more. Such sentimental aspects of the investors significantly affect the stock price as it triggers instant demand or supply of stocks in the market suddenly, fueled by greed or fear, respectively.

Hence, this results in a quantum leap or sudden plunge in the price of the stock. Sentiments of the investors take over fundamental and technical aspects of the stock, regardless of the strengths and weaknesses of the stock- fundamentally and technically- the stock price will rise or fall solely based on sentiments of the investors, as a result of news headlines. Such a rising or plunging rally can provide huge profits or may even lead to catastrophic losses, depending on the predictions made

by the investor. Hence, understanding the sentiments of the market participants and analyzing the sentimental aspects of the stock becomes very important for an investor. Although, it is challenging to correctly predict the sentiments of the majority of the market community, as the expectations differ. For instance, a good set of quarterly results with a quarter on quarter and year on year growth in profits may still drive the prices down as investors expect more from the company. Hence, using data-driven machine learning models is a better approach for sentimental analysis.

There have been numerous efforts made in predicting the stock market using different Machine learning models. Traditional Machine learning models such as logistic regression and naïve Bayes generally perform well on quantitative data. However, natural language processing techniques stand out and are most sought after when capturing subjective information. Recently, with the development of state-of-the-art models such as BERT and XLNet, the results have turned even better.

For Machine learning models to perform well, it is imperative to input high-quality data representing real-world scenarios where the model will be tested. For this research, data is scrapped online from the Economic times and Money control from 2016 to 2020. One common approach is to label the data based solely on the overall movement of the stock. Although, that does not represent any 'rally-like' or unusual change in stock price, which can be attributed to the emergence of news headlines. Hence, to address that challenge, the headlines are labeled according to the market's movement that is above or beyond the average movement of the stock price in the past. This would better represent real-life scenarios.

Further, the training of the NLP classification model is approached in two different ways. First, a neural network is trained using word-to-vector embeddings trained solely on the corpus of data available for this research. Second, a state-of-the-art BERT model [1] is fine-tuned for this research. The results are then analyzed and compared thoroughly.

Furthermore, a Long Short Term Memory model is trained which takes buy/sell calls from technical indicators as input, such as Accumulation/Distribution Indicator, Aroon number, Moving Average Convergence Divergence, and Money flow index that in combination represent both the long term and short term trends of the stock, along with the sentiment of that stock for a particular trading session. Such a model can analyze a stock not only based on sentiments but also technical trends. Long Short Term Memory is a type of Recurrent neural network [2]. Traditional RNN's only remember information and insights from the previous hidden state, and as a result, important information may be lost from earlier hidden states. Hence, in a time series application like stock prediction, the LSTM model is the best solution as its architecture allows the model to remove irrelevant information and preserve important information. For this research, the last 30 trading sessions are considered for producing predictions, as generally, it can be assumed that a stock is most likely to be affected by happenings from a 30-day period. Hence, it represents real-life scenarios. The LSTM model that uses both technical indicators and sentimental aspects shows better performance than NLP models that use headlines for predictions.

2. LITERATURE REVIEW

2.1 Related Work

Machine learning is the pioneer of emerging technology and its applications in all domains. The advancements in machine learning algorithms and extreme computational power have made it possible to apply it in each section of the world. Machine learning is the idea of making predictions by analogies from past data for any particular event. Machine learning and its related concepts have applications in many different fields, including finance. In finance, Machine learning is used for various problems such as making investment decisions, detecting frauds, high-speed trading, and overall making better business decisions. While Machine learning methods are aggressively in use for other applications, much research in the past is mainly focused on stock market prediction. The idea of predicting the movement of stock and the price of a stock in the future has provided enormous returns, which has been a motivation for research on this topic. Many different Machine learning algorithms and techniques have been tried and tested for stock market prediction, some even in conjugation with one another. AbdulsalamiSulaiman Olaniyi et al. [3] proposed a linear regression model for stock prediction based on two variables. Hiransha.M et al. [4] compared different versions of a recurrent neural network differing in their architecture, such as Convolutional neural network and LSTM, for predictions on NSE, India, the conclusion being CNN performed best amongst all. Further, General Adversarial networks based on LSTM are used by Zhang. K. et al. [5] to predict the stock prices at market close. Traditional Machine learning models such as Support Vector Machine, Artificial neural network, naïve-Bayes, and random forest were used by Patel. J, et al. [6]. In a nutshell, different machine learning techniques have been used and compared to predict stock markets using quantitative data.

However, in the short term, stock markets are affected mainly by the consensus about a company amongst the market participants. Hence, the machine learning models that only consider the historical price data of the stock lack behind on accurate predictions. Even if the theoretical accuracy is satisfactory, it is hard to define how the model will perform in real-world scenarios. This is because stock markets are affected by numerous other factors such as mergers and acquisitions, changing investor focus on particular sectors of the market, changing commodity prices a company deals in, external affairs of the country, and several other factors that either generate ebullient consent about the company amongst the stakeholders of the stock market, or a panicky environment, which leads to unprecedented movements in the price of that stock. For instance, at the dawn of the coronavirus pandemic worldwide, the stock markets began crashing. In India, trading activities were at a halt for 45 minutes. Such unprecedented movements are impossible to catch through traditional Machine learning models solely based on fundamental and technical numbers. Hence, it is crucial to develop a model to understand the market sentiment based on news headlines, to predict such unforeseen events, and make wiser decisions.

With the development of Natural language processing in recent years, understanding news headlines to predict stock markets using NLP models has attracted some researchers. Ren R et al. [7], Presents SVM based model for sentiment analysis of news events. Zhang. X. et al. [8] used sources like web news events and developed a multiple instance ML classification model. Tejas Mankar et al. [9] uses tweets from Twitter API to analyze stock sentiment and predict stock movement. Chien. Cheng. Lee et al. [10] use a fine-tuned BERT model for sentiment analysis to predict stock movement on US stock markets. The results of these and other work addressing this idea of predicting stock movement using NLP techniques for sentiment analysis are substantial. However, most of the models consider the overall movement of the stock while labeling the data. Unfortunately, such labeling techniques only partially address the problem. A turnaround to this problem is addressed while labeling data for research that will represent real-life scenarios better. News headlines are labeled positive, negative, or neutral based on a threshold cap calculated as the average movement of the stock in the past. Hence, this will better represent whether the emergence of a news headline affected the stock price or not.

2.2 Natural Language Processing

Natural language processing generally deals with analyzing natural language data and allowing computers to understand human language using Machine learning techniques. Natural language processing has wide applications in almost every domain, primarily for automation. One such application is understanding news headlines related to a company and predicting the movement of that company's stock in the future. This is made possible by providing numerical values to different words in the corpus of news headlines. These numeric values, mainly in the form of an array, are called word embeddings. Word embeddings are either pre-trained on large corpora like the word2vec tool created by Google or custom trained using the Gensim library on custom training data available for the project. These vectorial representations of words define sentence-level relationships amongst the words. Hence, word embeddings help understand the context of the sentence to the computer. Word embeddings are used as input for almost all NLP tasks. Word embeddings are a way to represent a word in the vector space. Similar words that mean the same are nearer in the vector space and vice-versa.

There are different algorithms available to train word-embeddings. The embedding layer can be used to train the embeddings if the application is based on neural networks. However, this technique requires a lot of training data, and it is computationally inefficient and slow. Tomas Mikolov et. al.[11] presented a statistical method- Word2vec- for learning a standalone word embedding. The Continuous bag of words model(CBOW) and Skip-gram methods were introduced to train word embeddings based on the importance of context. Its computational efficiency and low time complexity allowed Google to train word-embeddings on a large corpus consisting of billions of words, scrapped from the web. Pennington et al. [12] presented an extended version of the Word2vec model as Global vectors for word representations (GloVe). The word vectors were trained using matrix factorization techniques. GloVe model focuses more on the global representation of a word rather than its local context. Hence, depending on the application, the performance of Word2vec and GloVe can be better than one another. Word2vec embeddings seemed to be the optimum choice for this research as news headlines have a local context that matters more than the global meaning of words.

2.3 BERT

BERT is a state-of-the-art model introduced by google. BERT broke all the limits to any NLP sub-task optimum possible performance for question-answering, summarization, text classification, and other NLP tasks. Figure 1 shows a typical BERT uncased model architecture with 6 encoders and 6 decoders stacked together.

The Transformer is a model that uses the Attention mechanism and has the ability to provide different weights to different parts of the data according to its importance. Transformers use a set of encoders stacked together to encode the context between words or sub-words and then a set of decodes to produce output. A transformer's architecture allows it to learn the context. To make this possible, BERT uses two training strategies Masked LM and Next sentence prediction. The Masked LM strategy allows bidirectional training in models. The masked layer changes approximately 15% of the input tokens to [MASK] token and then attempts to predict the original tokens using the context nearby.

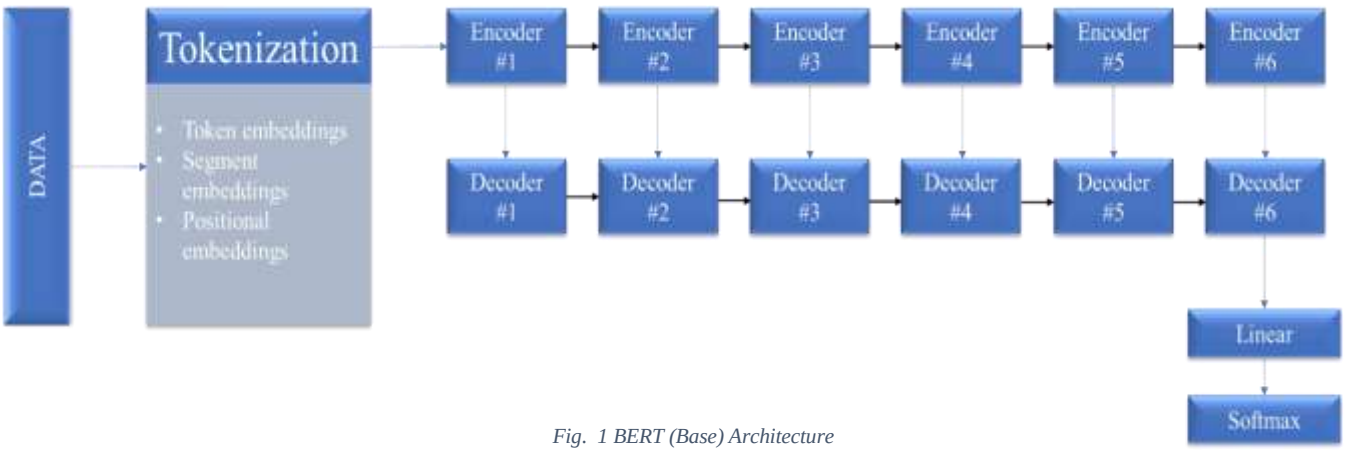

Fig. 1 BERT (Base) Architecture

On the other hand, the Next sentence prediction works at the sentence level, as the name suggests. The model works with input data in pairs so that 50% of pairs of sentences are successive sentences from the text and 50% are not connected. This helps the model to understand the underlying context of the corpus better. The word embeddings for BERT are trained by Google on over billions of words of a corpus formed from the Web. While training BERT for a particular task, there are two ways to go by. First, use the pre-trained models and their weights, or second, fine-tune BERT and stack custom layers over BERT to best suit a problem. For this project, a series of neural network layers are stacked over BERT for the classification of news headlines.

2.4 LSTM

Recurrent neural networks belong to the family of Neural network models, designed to feed information in the model in steps. These networks allow information to persist in the network. The layers in the network are connected in series to each other. RNN's are primarily used to analyze and make predictions based on sequential or time-series data. RNN's are preferred over other neural networks for their ability to make predictions based on non-immediate previous layers, that is, use preserved information from the past, and new information passed at the current layer to make predictions. However, there is a drawback in its performance when preserving information for a long time before fading away. Traditional RNN's face a challenge when it comes to "long-term dependencies." To address this challenge of fading coefficients over extended networks, Hochreiter et al. introduced the Idea of LSTM networks. LSTM networks belong to the family of recurrent neural networks that were introduced to solve long-term dependencies. LSTM's form a chain-like sequence of neural layers similar to RNN's however, with some complex changes. At every step of an LSTM, there are four neural network layers instead of one. All four of them have a specific purpose. LSTM's have unique abilities to remember important information from the past, aiding to the cell state that passes from one step to the other as it is with slight linear interactions at every step, where some information is added and some removed. These minor alterations are made possible using a controlled sigmoid neural net layer and point-wise multiplication operation, called gates. The gates can be interpreted as a switch whose action is the output of the sigmoid function, as in an output of 0 will keep the gates closed and stop the flow of information into the next layer, and an output of 1 will allow full flow of information. At each step of an LSTM, first, the "forget gate" decides the information be removed in the cell state coming from the previous step (eq. 1). The "forget gate" takes as input the hidden state from the previous step and the data fed into that particular step and performs a neural net operation. Next, the "input gate" uses the hidden state from the previous step and input data at the current step, and pass it through a neural net operation and consequently the sigmoid function to decide for each instance whether it be allowed to pass on or not (eq. 2). If allowed, the output is passed through a tanh function (eq. 3), which creates a vector that would be added to the cell state altered as per the "forget gate." At this stage, all the processing of data is complete. The next step is to decide the output. The output is based on a filtered version of the cell state (eq. 4) as expected to preserve information from previous steps. The output (eq. 5) is passed through a layer of point-wise multiplication with the tanh of cell-state to prepare the hidden state for the next step (eq. 6). Overall, it is ensured that information from previous steps forms the base of all the operations, and the same is ensured for the next steps. The cell state acts as a pipeline with gates at each time step, allowing the continuous flow of information and connecting each step.

$$Forget\ gate, f_t = \sigma(W_t.[h_{t-1}\ ,\ x_t\ ] + b_f\ ) \quad (1)$$

$$Input\ gate, i_t = \sigma(W_i.[h_{t-1}\ ,\ x_t] + b_i\ ) \quad (2)$$

$$C_{tt} = \tanh\ (W_c.[h_{t-1}, x_t\ ] + b_c) \quad (3)$$

$$Cell\ state, C_t = f_t * C_{t-1} + i_t * C_{tt} \quad (4)$$

$$Output, o_t = \sigma(W_o.[h_{t-1}\ ,\ x_t\ ] + b_o\ ) \quad (5)$$

$$Hidden\ state, h_t = o_t * \tanh\ (C_t) \quad (6)$$

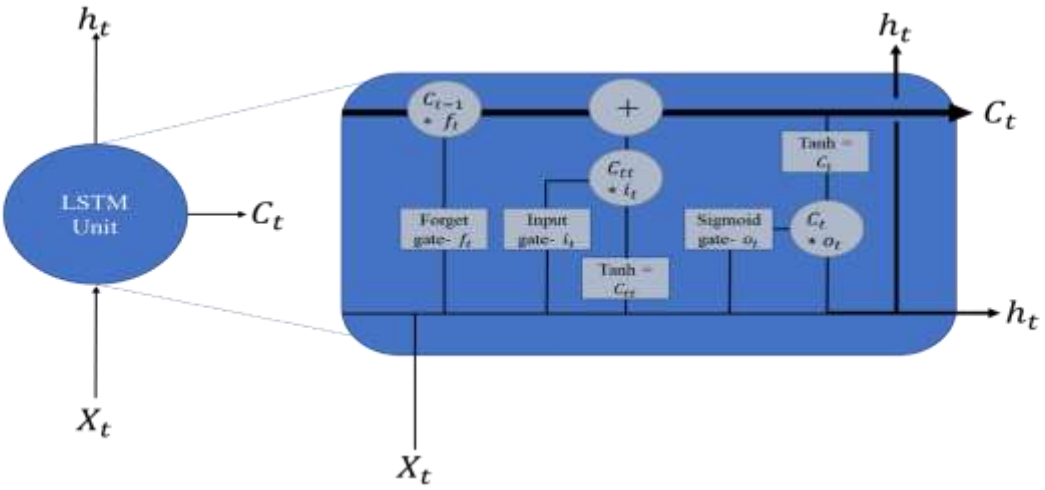

Fig. 2 Typical LSTM Unit

2.5 Technical Indicators

2.5.1 Accumulation Distribution Indicator

As the name suggests, this indicator suggests the short-term change in the price trend for a stock by using its volume and price. It represents the accumulation or distribution of stocks in the market and gives an idea of the supply and demand of the stock. Supply and demand will ultimately decide the price movement of the stock. It is a cumulative indicator and not absolute in itself. Hence, a positive or negative trend is established by comparing ADI values of two successive periods. The ADI indicator uses the high, low, and close price of the stock for a period.

The ADI line or the change in ADI between consequent periods indicates a reversal in the market trend. If the current period ADI is greater than the previous period's ADI value, it is a positive indication for the stock price and vice-versa. If there is no change in ADI value, it indicates a neutral trend for the stock. Equations 7,8,9 , shows the calculations for obtaining ADI for any trading day.

$$MFM = (2 * Close - Low - High)/(High - Low) \quad (7)$$

$$MFV = MFM * Volume \quad (8)$$

$$ADI = Previous\ ADI + MFV \quad (9)$$

Where:

MFM = Money flow multiplier,

MFV = Money flow volume,

ADI = Accumulation Distribution Index,

Close = Closing price for the period,

Low = Low price for the period,

High = High price for the period.

2.5.2 Accumulative Swing Index

Unlike the Accumulation distribution indicator, the Accumulative swing index shows the long-term trend of the stock. ASI uses the opening, closing, high, and low price of the stock for a particular period to indicate a long-term positive, negative, or neutral trend for the stock. The ASI value for a stock is calculated by incorporating the differences in consecutive day opening and closing prices. The ASI values typically lie within the range of -100 to 100. ASI value is calculated based on closing and opening prices of two consecutive trading periods and the variables- R and K. An ASI value greater than 10 indicates a positive long-term trend for the stock, while less than -10 indicates a negative long-term trend for the stock. While an ASI value in the range -10 to 10 indicates a neutral long-term trend for the stock. Equations 10,11, 12 and 11 show the calculations for obtaining ASI for any given day,

$$SI = 50 * (((C_y - O_y + 0.5 * (C_y - O_y) + 0.25 * (C - O)))/R) \quad (10)$$

For calculating variable R,

1. If H – C_y > L – C_y and H – L,

$$R = H - C_y - 0.5 * (L - C_y) + 0.25 * (C_y - O_y) \quad (11)$$

2. If L – C_y > H – C_y and H – L,

$$R = L - C_y - 0.5 * (H - C_y) + 0.25 * (C_y - O_y) \quad (12)$$

3. in all other cases,

$$R = H - L + 0.25 * (C_y - O_y) \quad (13)$$

Where:

SI = Swing index

C = Today's closing price

C_y = Yesterday's closing price

H = Today's highest price

H_y = Yesterday's highest price

K = The larger of $H_y - C$ and $L_y - C$

L = Today's lowest price

L_y = Yesterday's lowest price

O = Today's opening price

O_y = Yesterday's opening price

2.5.3 Aroon Number

Aroon indicator signifies a change in the trend or the trend itself for the price of a stock. It depicts the short to medium-term trends for the price of a stock. Aroon number incorporates the highs and lows for a period of trading days, which is based on the fact that a positive trend in the market will result in higher highs for consecutive trading days while a negative trend in the market will result in lower lows for consecutive trading days. Technically the indicator shows the number of trading days since the highest high in a period of time (for example a period of 20 trading days) and the number of trading days since the lowest low. Aroon up value is calculated using the high price of a stock and Aroon down value is calculated using the low price of a stock. For the purpose of this research, Aroon numbers on a scale of 20 trading days are considered. If the Aroon up value is greater than 50 and the Aroon down value is less than 50, a strong positive trend is confirmed. Similarly, if the Aroon down value is greater than 50 and Aroon up value is lower than 50, a strong negative trend is confirmed. In all other cases, a neutral trend is maintained. Equations 14 and 15 show the calculation for obtaining Aroon numbers,

$$Aroon\ up = 100 * ((25 - H_p)/25) \quad (14)$$

$$Aroon\ down = 100 * ((25 - L_p)/25) \quad (15)$$

Where:

H_p = Days since the highest high in a 25-day period,

L_p = Days since the lowest low in a 25-day period.

2.5.4 Moving Average Convergence Divergence

MACD indicator signifies the short-term trend or trend reversals for the price of a stock. MACD uses multiple exponential moving averages to create a MACD line and a signal line. The MACD line is created using the difference in exponential moving average for the 12-day period and 26-day periods. The signal line depicts the exponential moving average for the 9-day period. Practically, it depicts how different are the long-term and short-term exponential moving averages, and based on that provide buy-sell and hold calls. The exponential moving averages are calculated based on closing prices for the given period of trading days. If the MACD line crosses above the signal line, it depicts a breakout and indicates a positive trend for the price of that stock. Similarly, if the MACD line is below the signal line, it indicates a negative trend for the price of that stock. In all other cases, a neutral trend is indicated. Equations 16, 17, and 18 illustrate the calculations to obtain MACD and Signal line.

For any given day:

$$MACD = EMA_{12} - EMA_{26} \quad (16)$$

For calculating initial EMA_9,

$$EMA_9 = MACD/9 \quad (17)$$

In all other cases,

$$EMA_9 = MACD * (2/10) + (Previous\ EMA_9 * (1 - (2/10))) \quad (18)$$

Where:

EMA_{12} = 12-day Exponential Moving average,

EMA_{26} = 26-day Exponential Moving average,

EMA_9 = 9-day Exponential Moving average,

MACD = Moving average convergence divergence.

2.5.5 Money Flow Index

Money flow index signifies the short term trends or trend reversals for the price of a stock. MFI uses high, low, and close price in a 14-day period of time, along with the volume for trading days in the given period of time to depict the overbought and oversold signals for a stock. MFI typically ranges between 0 and 100. The typical price for any trading day is calculated using the high, low and closing price for that stock on that trading day. This typical price is then compared to the previous day's typical price along with the volume for that trading day to calculate the Raw money flow for a trading day. If the typical price for the current day is higher than the typical price for previous day, the Raw money flow is assigned a positive value, and vice-versa. Money flow ratio is calculated as the ratio of the absolute value of the sum of positive raw money flow in a 14-day period, over the absolute value of the sum of negative raw money flow in a 14-day period. This money flow ratio is the used to calculate the Money flow index for a trading day. If the closing price for a stock for the current day is greater than the closing price for the previous trading day and the current MFI is less than 80 and previous MFI, it indicates a reversal in trend from positive to negative and is labelled negative consequently. Similarly, if the closing price of a stock for the current trading day is lower than the closing price of a stock for the previous trading day and the current MFI is greater than 20 and previous MFI, it indicated a reversal in trend from negative to positive. In all other cases, a neutral trend is indicated. Equations 17, 18, and 19 illustrates the calculations for MFI,

$$Typical\ price = (High + Low + Close)/3 \tag{19}$$

$$Raw\ money\ flow = Typical\ price * Volume \tag{20}$$

$$Money\ flow\ index = (14\ period\ positive\ money\ flow)/(14\ period\ negative\ money\ flow) \tag{21}$$

3. EXPERIMENTAL SETUP

3.1 Data Scraping

One of the most crucial purposes of this research is to analyze and classify news headlines. For training any model, it is important to input data from reliable sources. To address this, data was scrapped on the internet from [15] and [16]. Two different media platforms were chosen to ensure robustness. Both of these are considered the most reliable and complete sources for stock market stakeholders in India, to get all the important news headlines at the right time. Hence, using python libraries such as beautiful soup, and selenium, news headlines related to business and stocks were scrapped from the Economic times archive for the period 2016-2020. Using a custom python function, the headlines were automatically scrapped year-month-day wise. The function operated accordingly if the year was a leap year and gathered all the headlines in the set period. For a particular date, the news headlines were compiled together. Using pandas library in python, the data was visualized as a Data Frame, indexed on the date. The same was carried for earnings news, companies news archive from money control. A total of over 40,000 news headlines of an average word length of 30 words are scrapped through this process.

3.2 Data Pre-processing

The data scrapped in the previous step is extremely raw and it is important to carefully clean and pre-process it before labeling the unstructured data. Moreover, the news headlines are in a mixed format consisting of news not directly related to the fate of any company. Hence, it is important to remove such headlines and choose only those that involve one or more companies. To address the same, regex codes can be used to filter out headlines that have at least one of the chosen companies as a word token. The same is successfully carried by first getting the names of all the companies in the format that publishers at money control and the times network prefer using in their news headlines. Of these further, the companies are filtered out from the list, based on their market cap. Only companies with a market cap of over $2 billion, are chosen. This is because small-cap companies' stocks are often manipulated by the so-called "market sharks". As the number of publicly traded stocks is very less, some people have the power of buying and selling huge chunks of stock at once for their profits which leaves the stock movement unreliable and impossible to predict. A regex code is formed using the names scrapped to filter the headlines. The headlines from both sources are further aligned together with dates in the same format again using regex. Finally, each headline is turned into a tuple consisting of the actual text and the company it is primarily related to, and the headlines are paired up together date-wise in this tuple form.

The previous step generated a dictionary of dates as keys and a list of tuples consisting of headlines and related companies as value. The next challenge is to label this unstructured and unsupervised data. The first step for the same is to collect the tickers for each company under scrutiny and the same were scrapped from money control. Subsequently, the stock prices for each company in the period 2015-2021 were scrapped from yahoo finance.

To label the headlines, all of the open, close, low, and high parameters of the stock for a particular trading session are used, rather than just using opening, and closing prices. It is first important to understand how the headlines affect the trading sessions. As market sentiment can have both short-term and long-term effects on the stocks, it is important to consider that a headline may have affected both the current trading session and the next trading session depending on the

timing of its first press release. Moreover, because of malpractices practiced by some privileged stock market participants, some people have the power to know about the headline before even its first time being released, which may trigger a rally of huge buying or selling of stocks. Also, sometimes the news is such panicking or delightful that market sentiments remain positive or negative for the next trading day too. Hence, to capture all these rallies and market movement, and to consider the overall buying or selling triggers caused by news headlines, it is important to consider the highs and lows of a particular trading session too. The same is carried using a python function in two steps. First, it is crucial to check whether the day of the headline had a trading session or not. This can be further segregated into three different cases- first, if the date is a weekend which would mandatorily mean that it did not have a trading session, second if the day was a weekday and a trading session, and finally if the day was a trading session but did not have a trading session due to market closure. It is considered that if the market did not have a trading session as it is a weekend, the effect of such a headline would be on the next trading session. If the market was a weekday and did not have a trading session then the effect of that headline would be on the next trading session, and if it did have a trading session then the effect of that headline would be on for that day's trading session and the next trading session. All of the same is achieved using timestamps.

After checking which trading sessions will be affected by the headline, the next step is to label the headlines as per the methodology used in this research. The overall attempt is to capture the sudden buy orders or sudden sell orders caused by news headlines. It is a common observation in the stock market that market participants follow a similar sentiment, which leads to huge movements in the stock in the same direction before possible recovery. Hence, it is better to label the headlines based on the difference between opening price and the high or low price of the trading session. If the closing price is greater than the opening price for the session, the difference between the opening price and the high price for the trading session is noted. Similarly, if the closing price is lower than the opening price for the session, the difference between the opening price and the low price for the trading session is noted. This difference is taken as the percentage over the opening price. Next, a threshold is set as the average percentage a stock has moved in the past 60 trading sessions. The headline is consequently labeled positive if the difference taken as a percentage earlier is positive and is higher than the threshold average of the company, to which the headline is primarily related. Similarly, the headline is labeled negative if the difference taken as a percentage earlier is negative and is higher than the threshold average. However, the headline is labeled neutral if the difference taken as a percentage, positive or negative, is lower than the threshold average. In a nutshell, the headline is labeled positive if the stock moves in a positive direction more than the average movement of stock in a 60-day period and negative if the movement is in a negative direction more than the average movement of the stock in a 60-day period, and neutral otherwise. If it is a trading day, then the greater of the different percentages, for the current trading day and the next trading day, is considered.

This methodology is applied for the fact that it is the unusual movement in stocks called "rallies" that would best define the effect of the emergence of news headlines. Moreover, the profits can be maximized if a trade is entered at a correct entry point and exited at the correct exit price point. And most number of times, the high price of the day is representative of such "rallies". Moreover, after such "rallies" traders mostly tend to exit or re-enter the stocks and book their profits, which balances out the overall movement. In such cases, it is extremely difficult to determine the actual effect of the news headline on the stock. Hence, it is better to consider the high or low price for the day its difference with the open price rather than the difference in close price and open price. Such a methodology represents the real-world scenarios better and every machine learning model needs to train on data that is representative of scenarios where it will likely be used.

3.3 Training Word2vec embeddings

For this study, open-source libraries for python were deployed such as Scikit Learn, Matplotlib, Tensorflow, Keras, NumPy, Hugging face transformers, Pytorch, Pandas, Gensim, and SpaCy. The models are set up in Python 3.7. The data pre-processing step provided the data set ready to be analyzed and worked upon for model building. Furthermore, all the pre-processing of datasets will be carried out according to the model being trained. But, one step that is common for all is converting the dataset from text to a form that can be processed by computers. Computers understand numeric language and hence, each instance of data that is each headline shall be converted to a numeric form. The techniques of converting individual word tokens to a real-valued vector in a pre-defined vector space- generally defined by the task in hand and the dataset or "documents" being worked on, based on an assumption that words that mean the same have similar vector representations is known as word-embeddings. There are many different ways to define this vector-to-vector distance but the most widely used technique that works in all dimensions is cosine similarity. Hence, two similar words will have a higher cosine similarity score on their word embeddings. There are many different techniques to train vector representations of words. In traditional Vector space models such as One-Hot encoding and TF-IDF technique, words are represented as a vector of dimensions equal to the size of the vocabulary. This is extremely inefficient to train and often leads to sparse vector representations. Hence, distributed word representation techniques that use a neural network to learn representations, work better. Tomas Mikolov, et. al., proposed two neural architectures- the Continuous Bag of Words (CBOW) and Skip Gram- to learn word embeddings. The difference in both these methods is that CBOW works in a way that predicts the current word from a sliding window consisting of neighboring words as context, while the Skip-gram method predicts the surrounding words in the window by using the current word as context. Each word in the context window is represented in the form of One-Hot vectors and will form the input data for the input layer of the neural

network. There is a hidden layer and an output layer to complete the neural network. Using this type of architecture ignores the sequence of words, which is according to the Bag-of-Words assumption. The CBOW method is more efficient and trains the embeddings more quickly, however, the Skip-gram method handles and learns better embeddings for rarely occurring words.

For this research, as the dataset or "documents" consist of news headlines, that consist of company names, which occur rarely considering the dataset as a whole, it is better to use the skip-gram technique. Word-embeddings for this research are learned using the Gensim library in Python. The dataset is trained with all the data instances regardless of its label together to maintain robustness in the embeddings. The word2vec embeddings learned using the Skip gram technique had the parameters set as- embedding vector size of 300 which resembles the number of neurons in the hidden layer, context window of size 7 to maintain a balance between the train time and train accuracy, and a total of 5 noise words be drawn. The model is trained on a total of 38407 words generated by splitting each headline by tab. The model is trained for a total of 1001 epochs to maintain optimum convergence.

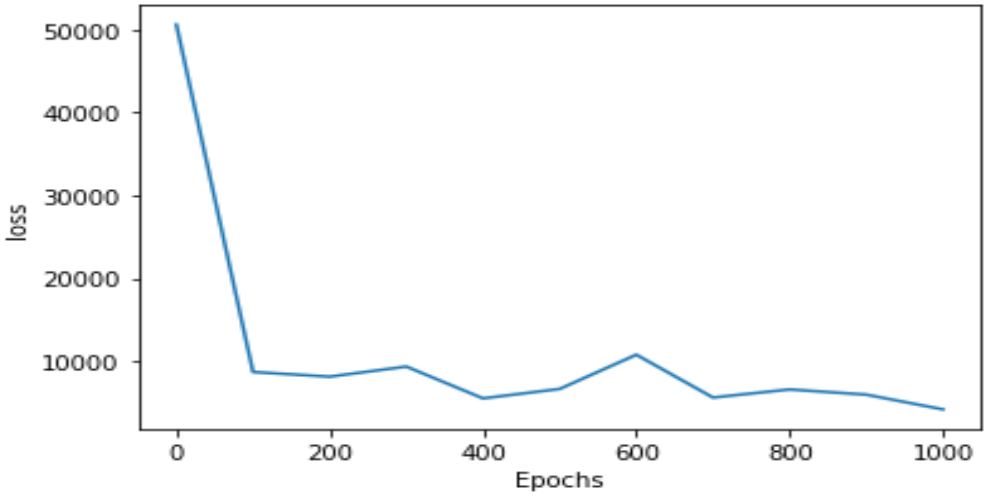

Fig. 3 Training loss for Word2vec embeddings

The model is trained to converge from a loss of 50578 to 4205 with a training time of 572.56 seconds. The model can build a vocabulary for 19912 unique word embeddings. Figure 3, shows the training loss occurred during training of word2vec embeddings. These trained word2vec embeddings are in the form of a 300 feature array. The values of this array is a representative for the similarity of two words that are used in same context. Figure 4 and 5, Illustrates a heat map for the words “Profit” and “loss”. As visible, the two words are similar at some ends, although the heat map differs in some areas showing that these words are typically used in the same context but for different purposes. This concludes the training of word embeddings which we will be using for training the machine learning models. Figure 6 illustrates the vectorial form of sample words from the learned corpus. These word2vec embeddings shown in vectorial form shows relationships between different words such as the words “*Liladhar prabhudas*” and “*Sharekhan*” are located together, as both these words are names for brokerage firms and hence, they tend to appear in a similar context.

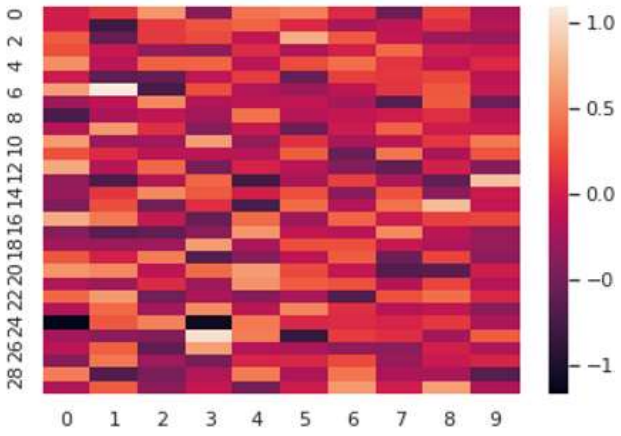

Fig. 4 Heat map for word2vec embedding of "loss"

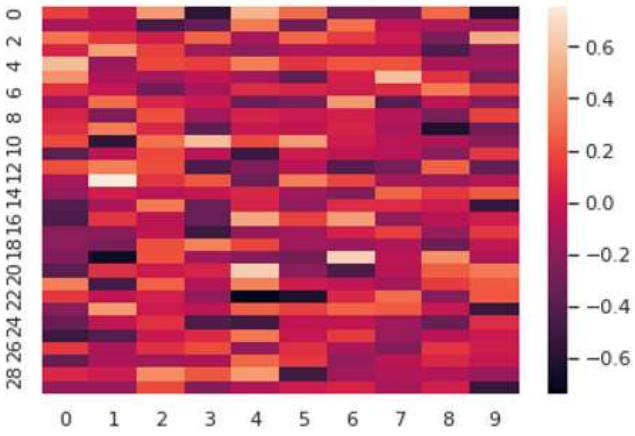

Fig. 5 Heat map for word2vec embedding of "profit"

3.4 Training of ML Classifiers

To train a model on the headline, the headlines must themselves be in a vectorial form. The word embeddings trained will represent the words in each headline and provide a vectorial representation for each headline. This is executed using the SpaCy library in python which allows using pre-trained word2vec embeddings on tokens and generating a vector for any document. An imbalance in the dataset based on the number of data instances available for each class can lead to skewed results for any Machine learning model. The dataset used for this research is skewed towards neutral headlines. There are a total of 10035 negative labeled headlines, 9297 positive labeled headlines, and 19075 neutral labeled headlines. Hence, to offset and balance the dataset, 10000 neutral labeled headlines were randomly eliminated from the dataset. Hence, a total of 28,407 headlines were used to train any model consisting of 10035 negatively labeled, 9297 positively labeled, and 9075 neutral labeled headlines. Each headline in the dataset is now converted to a document using a blank 'en' model with the word embeddings trained earlier. Each headline is converted to a vectorial representation of dimension 60 x 5 consisting of float values. The dataset is further divided into training and testing sets. The ratio of division is set to 8:2, which leads to a total of 20 % of the total dataset being set aside for testing the model. A total of 22725 headlines were used for training the model and 5682 headlines were used for testing the model performance.

The vectorial representations of the headlines achieved using word2vec embeddings trained on our dataset will be fed into different Machine learning models, to analyze and compare, to choose the best performing model. The models trained are Support Vector Machine, K-Nearest Neighbors, and ensemble models such as Random forests classifier and Gradient Boosting. The parameters for each model were chosen using the best of the given parameters in Grid Search. Although, the basic purpose is to compare and analyze the best model. Support vector machine is a machine learning model that identifies a hyperplane to classify supervised data into their respective classes. For this research, the number of classes does not allow a linear hyperplane and hence, a Radial Basis Function is chosen as a hyperplane parameter. K-nearest neighbor is another machine learning model that is deployed here. K-nearest neighbor looks at similar data points in the dataset to predict the class of the data point under scrutiny. The most important hyper-parameter to choose here is the number of neighbors to look at or the value of K. For this research, owing to the immense volatility of the stock market, the best results are achieved with the value of K as 1. The ensemble models deployed here are, Random-forest classifier and Gradient Boosting. Ensemble models are built from decision tree models where the trees are added one after the other iteratively improving the errors made by the preceding iteration. For this research, the hyper-parameters set for gradient boosting are- exponential as loss function, the learning rate of 0.05, max-tree depth of 5, the minimum number of samples required to split a node as 12, the minimum number of samples required to be at a leaf node as 12, the maximum number of features for best split as 17, and Friedman mean square error to calculate the quality of a split. Random forest classifier is another supervised ensemble machine learning model which uses bagging to create different decision trees which are fit different bootstrapped samples from the training data. The hyper-parameters chosen for the Random-forest classifier here are- maximum tree depth as 8, Gini impurity to measure the quality of split, number of features for best split as 17, and the number of trees as 200.

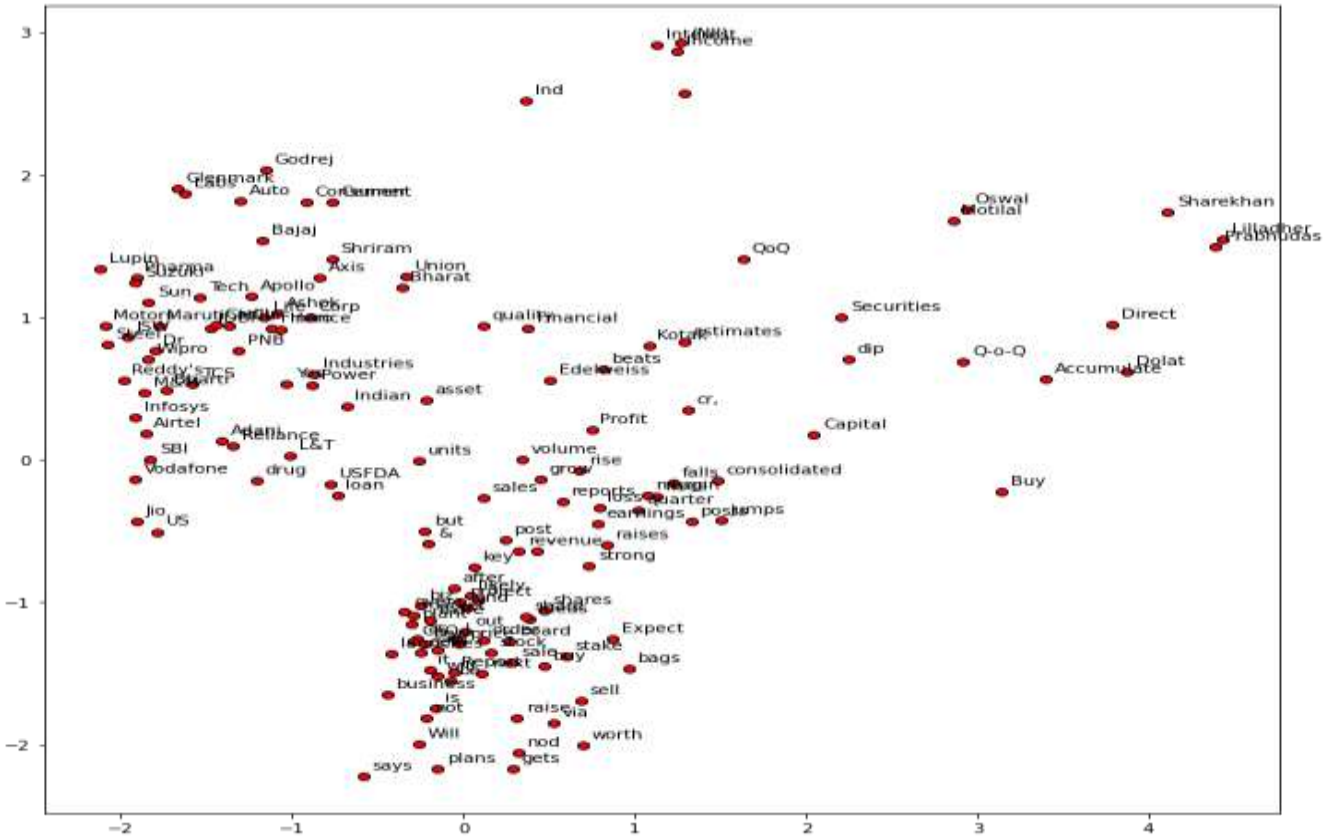

Fig. 6 Word2vec embeddings in vectorial form

3.5 Training FINE-tuned BERT model

The abilities of state-of-the-art BERT model with transformer architecture are phenomenal. Using the Hugging face and Pytorch libraries in Python, a Fine-Tuned BERT model is deployed for classification of headlines as positive, negative or neutral. All the headlines are tokenized to give input-ids, token-type-ids, and attention-masks, according to the pre-trained embeddings of BERT model. The headlines are also truncated according to a maximum headline length of 25 tokens. To perfectly balance out the frequency of headlines, a class weight of 0.9436 was computed for negative labelled headlines, 1.0433 for neutral labelled headlines, and 1.0185 for positively labelled headlines. This concludes the pre-processing of headlines to BERT suitable form. Finally, the pre-trained BERT model was fine-tuned by stacking an Artificial Neural Network at the bottom end. The model architecture consisted of BERT transformer, followed by six neural net layers, followed by the output layer. The output of BERT transformer acts as an input for the first fully connected feed forward layer with 256 neurons. The first layer takes up ReLu function as activation function. To prevent overfitting, dropout layer is deployed. This layer is followed by the first hidden layer with 256 neurons and Leaky ReLu as activation function, again followed by a dropout layer. Similarly, four hidden layers stack up below the first hidden layer with 128, 64, 32, and 32 neurons respectively. All hidden layers' output is computed by a Leaky ReLu function. The second, third and fourth hidden layers are also followed by a dropout layer to prevent overfitting. Dropout layer works in a way such that at each layer, it drops down some input units to 0 and scales up other units in order to offset them. The overall result is a robust model with less overfitting. Leaky ReLu is chosen to avoid a dead end in case of negative input. The final hidden layer is followed by an output layer, where the outputs are computed by log-softmax function. For backpropagation, the loss function deployed is negative log likelihood as it is a multi-class classification problem. The model is optimized using Adam weight decay optimizer. The model is trained for a total of 20 epochs. A detailed architectural representation of the fine-tuned BERT model is shown in Figure 7.

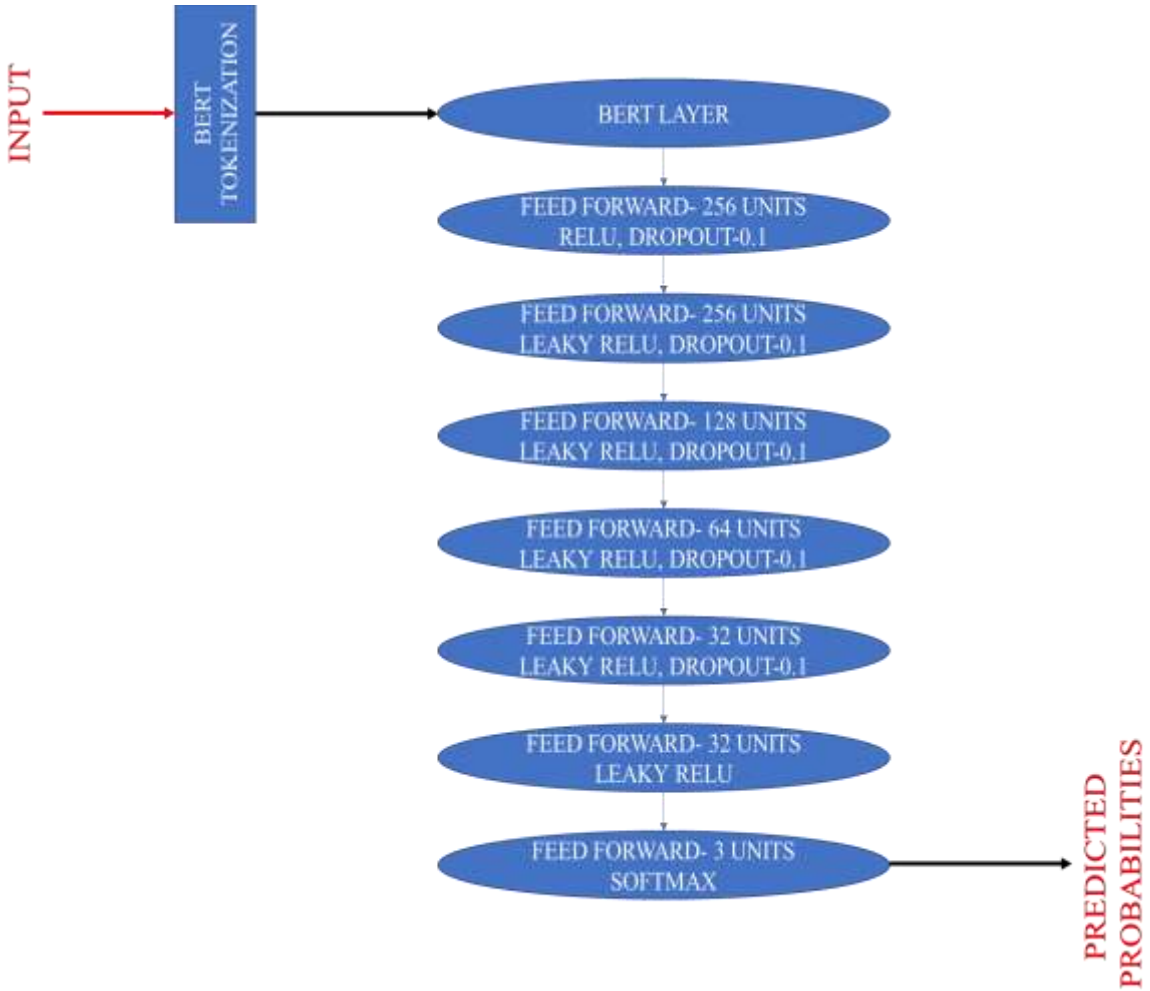

Fig. 7 Fine-Tuned BERT architecture

3.6 Training LSTM

LSTM models fit well on time series data, in a way to represent the real-world scenarios. In stock markets, the prices of a stock are often affected by both long term trends and short term trends. A model may fit data representing short term trends with more accuracy but that would be misrepresenting the real world scenarios. Similarly, a model may fit data representing long term trends with more accuracy but that would again be misrepresenting real world scenarios. Hence, it is important to

include both short term and long term trends while creating a stock prediction model. For the purpose of this project, five technical indicators- ADI, ASI, Aroon Numbers, MACD, and MFI- are used for representing the long term and short term technical trends. Moreover, polarity scores of news headlines representing the sentimental trend of the market participants, is used as an added feature for the input data. The input data for LSTM model will follow a different methodology to NLP models trained earlier. The time period for calculating each indicator for any trading session varies.

For each trading session, all the technical indicator and sentiment calls for the last 30 trading sessions will be used as input data. Hence, all of the technical indicators are calculated based on the open, low, high, close, and volume quantities for any trading session. To get a numeric value representing the market sentiments for any trading session, Vader sentiment library in Python is deployed. Using Vader sentiment analysis tool, the polarity scores for all headlines are calculated. All the stocks are provided with a base sentiment value of 2.0 for any trading session. If there is a news headline for a given stock on a given trading session, the polarity scores of the previous trading session are altered accordingly as the previous day's indicators will represent the movement of that stock on the current trading session. However, the effect of any news can also be seen on the next trading session hence, it is important to update the polarity scores of the current session too. Moreover, if the news headline emerges on a non-trading day, the polarity scores of immediate previous trading session are updated. Hence, the technical indicators and the polarity scores are calculated for each trading session.

Each data instance of the training data prepared consists a total of 6 features for each trading session, with a total of 30 trading sessions making a total of 180 features. The labels are generated as per the closing and opening prices of the stock on that trading session, differing the methodology used for NLP models, as the long-term trends are also being focused on. Moreover, the LSTM model shall be trained individually on each stock as the trained weights will differ, considering the difference in the nature of each stock. Moreover, if a stock was listed on the stock market at a later date than the first date under scrutiny, it will have lower amount of data instances. Hence, it is not viable to train each stock on same weights.

The LSTM architecture deployed here consists of five LSTM layers. The first layer of LSTM has input shape 30 x 6 with output dimensions of 512. This layer is followed by second layer of LSTM network with 512 units again, followed by two layers of LSTM network with 256 units each, followed by the final LSTM layer with 128 units. A dropout layer is stacked to prevent overfitting. Following the dropout layer, a feed forward fully connected layer is deployed with 128 neurons and sigmoid activation function. Finally, the output layer uses sigmoid activation function. The model uses Binary cross entropy to calculate the losses and learn the weights using back propagation, and Adam optimizer. In the model, a total of 4,688,770 parameters are trainable. The model uses loss as performance metric. A detailed structure of the LSTM model deployed for this research is represented in Figure 8.

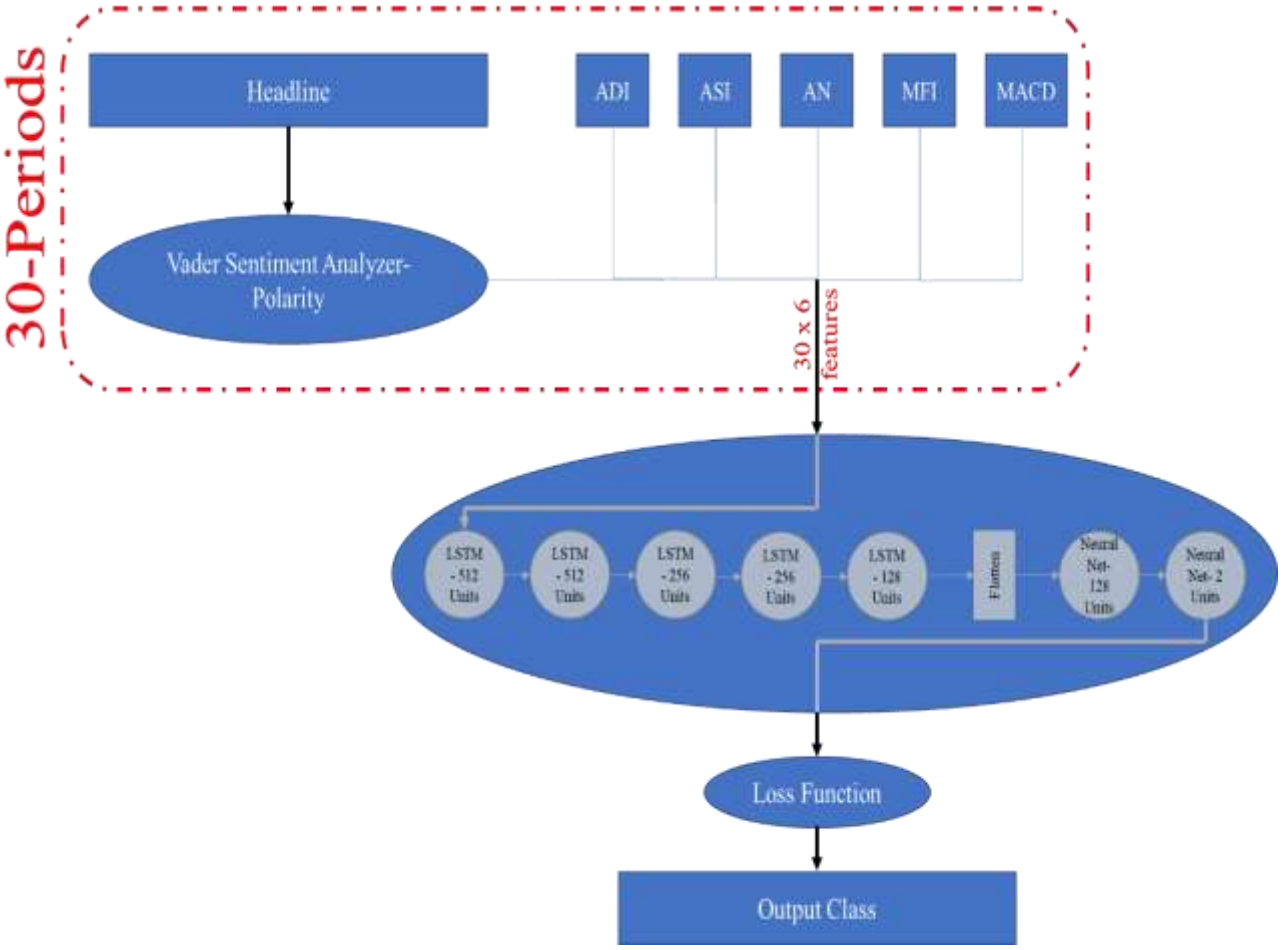

Fig. 8 LSTM model architecture

4. RESULTS AND ANALYSIS

Word embeddings were trained using the word2vec technique on dataset consisting news headlines. These embeddings were used to train different Machine Learning models to classify a headline as positive, negative or neutral. The predicted values trained by each machine learning model are compared with the true values and models are compared on various performance metrics. The Support vector classifier trained with regularization parameter as 1, radial basis function as kernel (to support non-linear data), and gamma value 1/300.

The SVM model trained on a set of 22,725 headlines, was then tested on a test dataset consisting of 5682 headlines. The SVM model performed satisfactorily with a testing accuracy of 43%. The overall performance of the model based on different metrics for headlines with each label is summarized in table 1. The confusion matrix for predicted classes and true classes is given in Figure 9. Further, a K-nearest neighbor classifier was trained on a set of 22,725 headlines and tested on a test dataset consisting 5682 headlines. K-nearest neighbor performed well considering the volatility of stock markets and the similarity of different labelled headline vectors. The overall performance for K-nearest neighbor with metric values is given in table 2. The confusion matrix for predicted and true labels is given in Figure 10. The gradient boosting ensemble model trained and tested on the same training set as SVM and K-NN models, performed better than the two. The overall performance is summarized in table 3. The confusion matrix for predicted and true labels is given in Figure 11. Random-forest classifier gives the best precision scores amongst all. The performance metrics are summarized in table 4. The confusion matric is represented in Figure 12. It can be observed that apart from K-nearest neighbors the recall and f-1 score for positively labelled headlines is extremely low. Hence, other than K-nearest neighbors classifier, all other models fail to generalize well and form a decision boundary that fits all. Hence, although the overall precision score for K-nearest neighbors is lower than Random-forest classifier, K-nearest neighbors clearly performs well on testing data.

BERT model is a transformer- based model that has given state of the art performance on almost all NLP sub-tasks. An artificial neural network is stacked underneath the pre-trained BERT uncased model. The entire fine-tuned BERT model is trained on news headlines tokenized by BERT mechanism. The batch size is set as 64 and the model is trained for 20 epochs. The performance of BERT model on our dataset is as expected. As the problem statement addressed in this research project of classifying news headlines is extremely volatile, not absolute, and vulnerable to be affected by numerous other factors, a fine-tuned BERT model couldn't learn the underlying patterns better than machine learning models trained on custom word embeddings. The fine-tuned BERT model was trained for 20 epochs. After training for 20 epochs, the validation loss converged to 1.080 which is satisfactory considering the problem statement in hand. The fine-tuned BERT model gives an accuracy of 43% on test data. The performance metrics for Fine-tuned BERT model is summarized in table 5.

Table 1 Performance Metrics for SVM Classifier

Label	Precision	Recall	f-1 score
0	0.42	0.61	0.49
1	0.58	0.56	0.52
2	0.39	0.14	0.20
Weighted Average	0.43	0.44	0.41

Table 2 Performance metrics for Gradient Boosting Classifier

Label	Precision	Recall	f-1 score
0	0.45	0.57	0.50
1	0.48	0.60	0.53
2	0.45	0.21	0.29
Weighted Average	0.46	0.46	0.44

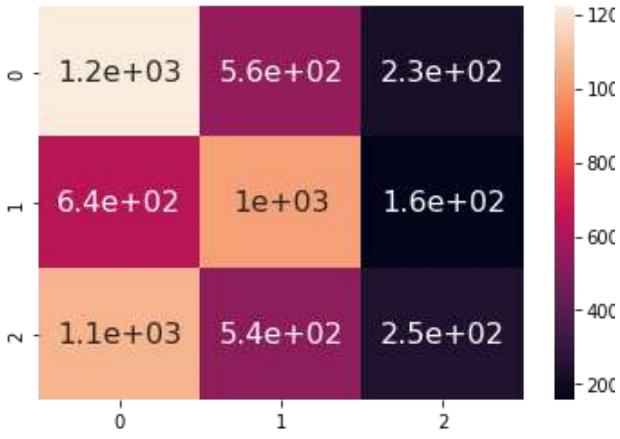

Fig. 9 Confusion Matrix for SVM Classifier

Fig. 10 Confusion Matrix for Gradient Boosting Classifier

Table 3 Performance metrics for K-Nearest Neighbor Classifier

Label	Precision	Recall	f-1 score
0	0.5	0.52	0.51
1	0.49	0.46	0.47
2	0.46	0.47	0.46
Weighted Average	0.48	0.48	0.48

Table 2 Performance metrics for Random-forest Classifier

Label	Precision	Recall	f-1 score
0	0.45	0.58	0.51
1	0.45	0.69	0.55
2	0.66	0.10	0.18
Weighted Average	0.52	0.46	0.41

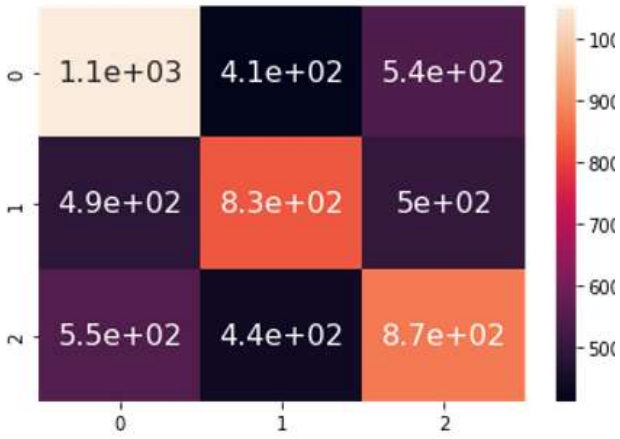

Fig. 10 Confusion Matrix for K-NN Classifier

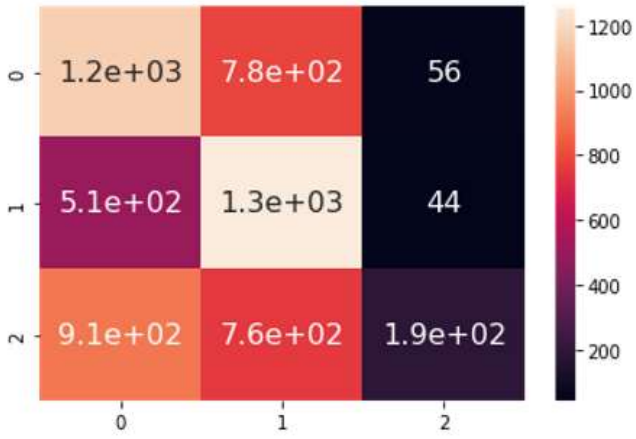

Fig. 11 Confusion Matrix for Random-forest Classifier

Table 3 Performance metrics for Fine-tuned BERT classifier

Label	Precision	Recall	f-1 score
0	0.45	0.55	0.49
1	0.44	0.65	0.53
2	0.41	0.12	0.18
Weighted Average	0.43	0.44	0.40

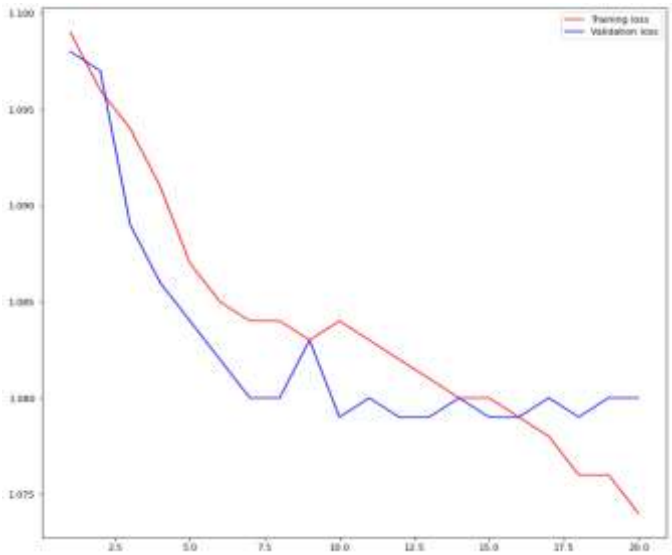

Fig. 12 Training-Validation loss for Fine-tuned BERT classifier

The fine-tuned BERT classifier fit well on the training data with a validation loss in the range 1.075 to 1.08. The model was able to converge losses. A graphical representation summarizing the training and validation process of the fine -tuned BERT model is given in figure 13.

However, the precision, recall, f-1 scores appear lower. This can be reasoned by the size of dataset which is extremely small. For any NLP problem, the size of dataset required is large, for the model to learn patterns successfully and generalize the learned weights better. Moreover, the models deployed for this research only uses the headlines as data with an average word length of 25 words. This essentially means that the length of each data point in the dataset is not enough for the model to detect the underlying context and learn the patterns better. Also, the manipulative and volatile

nature of stock markets with a forever changing trend in the pattern of price movements, hinders the learning process. Despite all of the above lying issues, the performance of each model is satisfactorily good.

Upon comparing the results of the best model trained on the custom word2vec embeddings trained on the headline dataset used for this research and the Fine-tuned BERT model, it can be concluded that K-nearest neighbor performed slightly better. However, given a larger, clearer dataset, the performance for each of the model under scrutiny will improve. The above conclusion can be reasoned on the dataset itself. It is observed that the headlines themselves are extremely volatile and vulnerable to other factors. For example, if a stock's price has faced a downward trend for a significant amount of time, it has already created a negative sentiment in the market. In such a case then, a set of bad financial quarterly results making headlines for the company will not affect the stock price negatively as the market participants were already expecting worser results. Hence, the market sentiment for a stock effectively works in a relative manner.

However, as the number of training samples and the overall size of dataset available is less, the model is not being able to generalize. Hence, it can be hypothesized that a larger dataset on the same fine-tuned BERT model with similar parameters will perform significantly better.

The Machine learning models and the fine-tuned BERT model trained represented the short-term immediate trends caused by a surfacing of news headline. However, stock markets have both long term and short-term trends that represent the market dynamics better. Instituting both long term and short-term trends in the market will help depict a more real-life scenario of stock market dynamics for any stock. To address this, an LSTM model which fits perfect on time-variate data and captures long-term effects of trends on the price of a stock was trained on the basis of interpretations from technical indicators, and the market sentiment based on news headlines. The LSTM model performed as expected. Because of more context being provided to the model with technical trends- both long and short term- along with sentimental trend, the model converges to better losses. The model is trained for a total of 200 epochs for any stock. The average loss achieved for any company's stock from the top 200 companies based on market cap, is in the range 0.65-0.7. While the average testing accuracy for any company's stock from the top 200 companies based on market cap is in the range 40%-60%. The model performs better as more context is provided. A better than average loss number and average accuracy range signifies that the model is able to understand the underlying patterns while training. However, aiding to lower amount of training features and the overall size of the dataset, the model is not being able to generalize those patterns. A better model with weights that are more robust is approachable with a larger dataset. The results of training loss and validation loss- for a sample of 10 companies that are one of the largest companies listed on Indian stock markets on the basis of market cap- is given in figure 14 and figure 15 respectively. It can be seen that the model is able to generalize well on each of the given stock and there are no outliers. The Binary cross entropy loss on testing data for each company, and the accuracy on predictions made on testing data for each of the 10 companies is given in figure 16. The graph shows a constant loss on the testing dataset for each of the 10 companies, which is even comparable to the training and validation losses for each company respectively. Hence, the model fits well on training data and it is able to learn weights that is able to make consistent predictions for different companies. A better accuracy and lower loss value can be further achieved with a larger dataset.

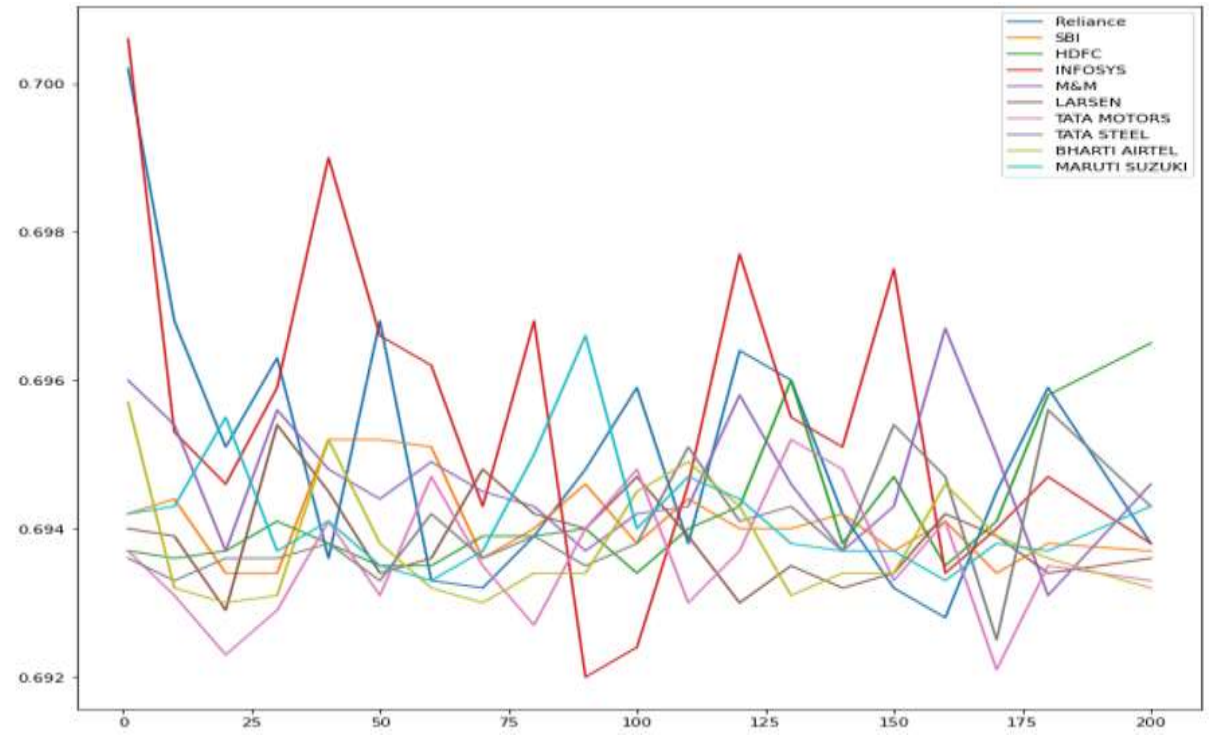

Fig. 13 Training loss on LSTM model

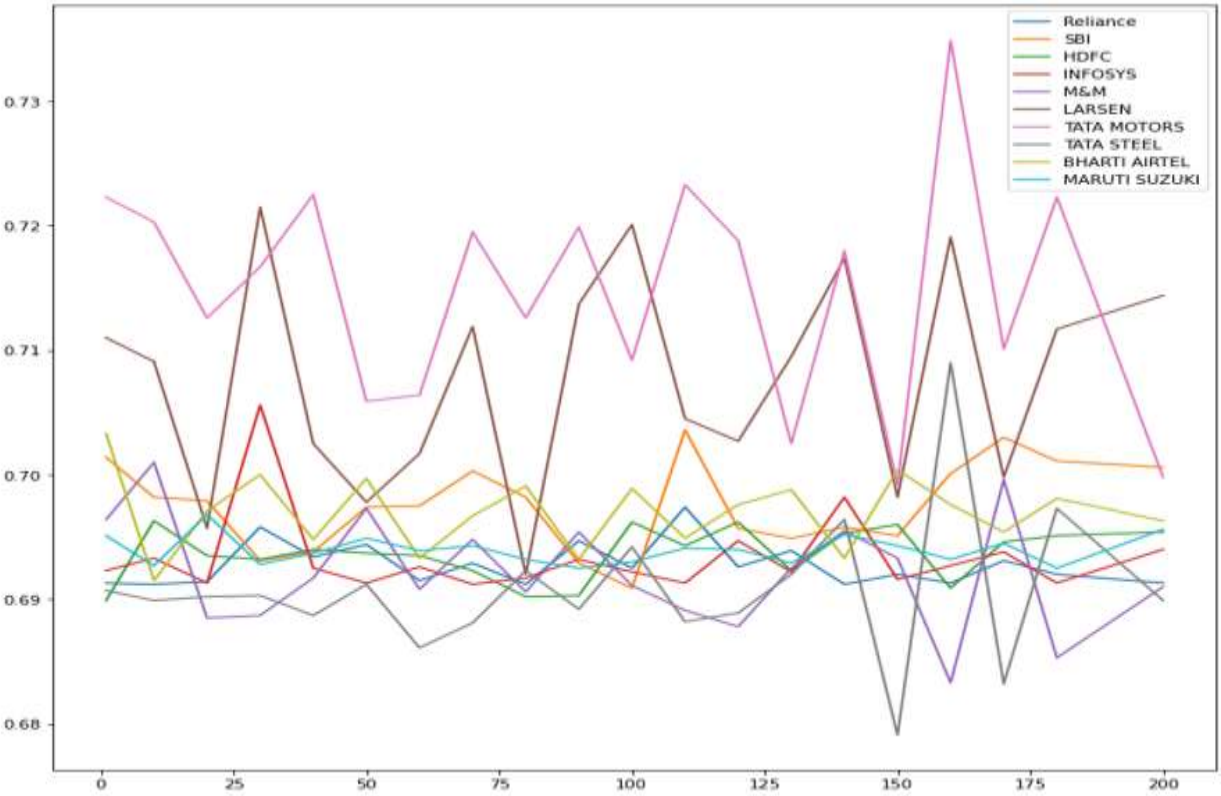

Fig. 14 Validation loss on LSTM model

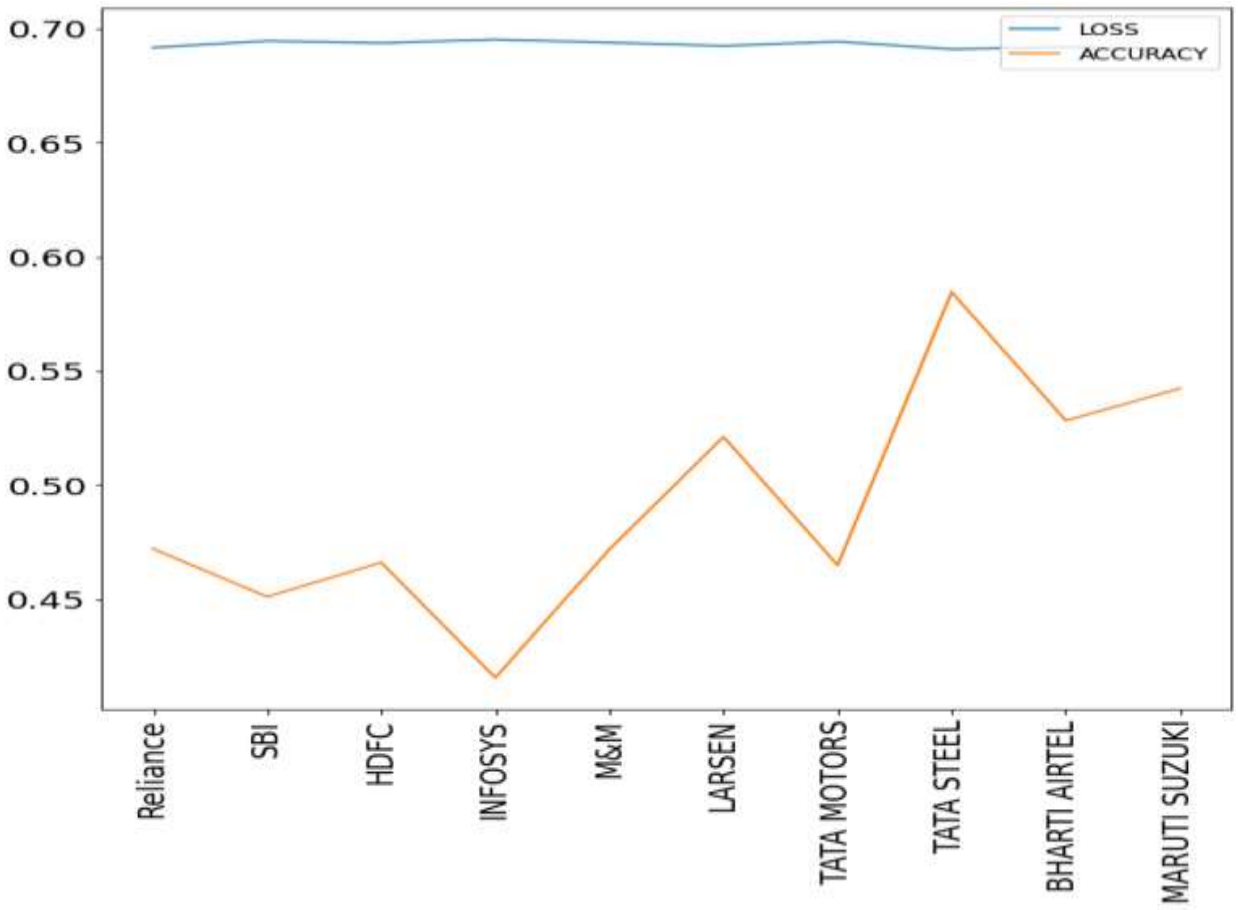

Fig. 15 Performance of LSTM model on test data

5. CONCLUSION

The purpose of this research is to analyze the effect of news headlines on the price of a stock. The short-term sentiment in the market as a result of the news headline directly related to a particular stock can lead to bigger than usual sudden jolt in the price of the stock which can lead to huge profits or losses. The current methods that deal with short term fundamental and technical indicators fail to capture these rallies. Moreover, most of the research focused on this problem statement fail to address the fact that sentimental effect on the price of a stock can never be captured using the opening and closing prices of the stock. This is because the sentimental effect on any stock is short lived and tends to evaporate by the end of trading session. Hence, it is best to use the high, low and opening price of the stock to label the dataset.

The problem of analyzing news headlines in the form of text data is best served by NLP practices. Hence, different NLP techniques are used to classify headlines. The problem statement addressed is a multi-class classification problem and machine learning models are used accordingly. The solution is approached in two different ways- by training custom word2vec embeddings on the textual data available for this research, and by using pre-trained BERT embeddings. The word2vec embeddings were trained on the news headlines used for this research, to capture better context of the happenings of Indian market.

NLP classification models are deployed for this research to classify headlines into three classes- positively affecting, negatively affecting, and neutral. Different machine learning models-SVM, K-Nearest neighbor, Random-forest classifier, and Gradient boosting- are executed. Amongst all, K-nearest neighbor performed the best with a precision score of 48%. Further, BERT model was trained on the dataset. The BERT model was fine-tuned with an artificial neural network stacked with BERT. The fine-tuned BERT classifier used the pre-trained BERT embeddings as input data. This gave a precision score of 43% and a loss of 1.080, which is lower than K-nearest neighbor classifier. Considering the manipulative and volatile nature of stock market which is affected by numerous different direct and indirect factors, both pathways- using custom trained word2vec embeddings and using pre-trained BERT embeddings-used to achieve an NLP classification model, yielded satisfactory results. However, the performance of NLP classifier namely K-nearest neighbor classifier trained on custom trained word2vec embeddings yielded comparatively better results. This is because the size of dataset is far too small for the state-of-the-art BERT model to learn better weights and study robust patterns that accurately captures the context of Indian markets. While, custom word2vec embeddings that were specifically trained on the dataset, was better accustomed to the contextual features of the headlines related to Indian stock market. Moreover, the length of each data point was small to accurately train the weights and capture a wider context of the issue highlighted in the headline.

Further, to provide a wider context of the direct factors affecting the price of a stock, an LSTM model is trained on a 30-day period. The LSTM model uses interpretations from five different technical indicators on the stock data along with market sentiment for the stock using news headlines, that captures both the long-term trend and the short-term trend on the price of stock. As expected, the LSTM model performs better with an average loss in the range 0.65 to 0.7 and an average accuracy in the range 0.40 to 0.60. Moreover, the proposed LSTM model is definitive in itself as both long-term and short-term indicators are considered in training which accurately represents real-world scenarios. Hence, it can be concluded that LSTM model trained on wider context of the direct factors affecting the price of a stock is best for pattern recognition and predicting the overall movement in the price of a stock while NLP classifier K-Nearest Neighbor can be used for predicting the short-term movements in the price of a stock.

6. FUTURE SCOPE

- The accuracy of the NLP classification models can be improved significantly with the use of full news articles. Furthermore, the performance can be improved with a larger and better-quality dataset.
- The accuracy of LSTM model can be improved by analyzing indirect factors that affect the stock of a company and feed it into the LSTM model. Such factors such as change in ruling government, environment policies, market indices performance, and other factors relating to international markets and movements in stock markets world-wide, can greatly affect the price of a stock. One way to possibly achieve this is by pooling stocks in different groups in a way that the companies grouped together have the maximum tendency of influencing the business of each other.

7. REFERENCES

1. Jacob Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", arXiv:1810.04805v2 [cs.CL].
2. S. Hochreiter et al. "Long Short-Term Memory" , Neural Computation, Vol.9-8.
3. S Abdulsalam Sulaiman Olaniyi et al. R. G," Stock Trend Prediction Using Regression Analysis – A Data Mining Approach", ARPN Journal of Systems and Software Volume 1-4 (JULY 2011), Brisbane, Australia.
4. Hiransha. M et al. "NSE stock market predict ion using deep-learning models", Procedia computer science, Vol. 132, pp. 1351-1362, 2018.
5. Zhang K. et al. "Stock Market Prediction Based on Generative Adversarial Network", Procedia computer science, Vol. 147, pp. 400-406, 2019.
6. Patel. J et al. "Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques", Expert Systems with Applications, Vol. 42(1), pp.259-268, 2015.
7. Ren R et al. "Forecasting stock market movement direct ion using sentiment analysis and support vector machine", IEEE Systems Journal, Vol. 13(1), pp. 760-70, Mar 2018.
8. Zhang. X et al. "Stock market predict ion via multi-source multiple instance learning", IEEE Access, Vol. 6, pp. 50720-50728, 2018.
9. Tejas Mankar et al. "Stock Market Prediction based on Social Sentiments using Machine Learning", ICSCET, January 2018.
10. Chien. Cheng. Lee et al. "BERT-Based Stock Market Sentiment Analysis", ICCE-Taiwan- 2020.
11. Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space", arXiv:1301.3781v3 [cs.CL]
12. J. Pennington et al. "GloVe: Global Vectors for Word Representation", EMNLP, pp. 1532–1543, October 2014.
13. https://economictimes.indiatimes.com/
14. https://www.moneycontrol.com/