STOCK MARKET PREDICTION USING OPTIMISED LSTM MODEL

Project Report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Prakhar Srivastava (191509)

Under the supervision of

Dr Pardeep Kumar

To

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan 173234, Himachal Pradesh

CERTIFICATE

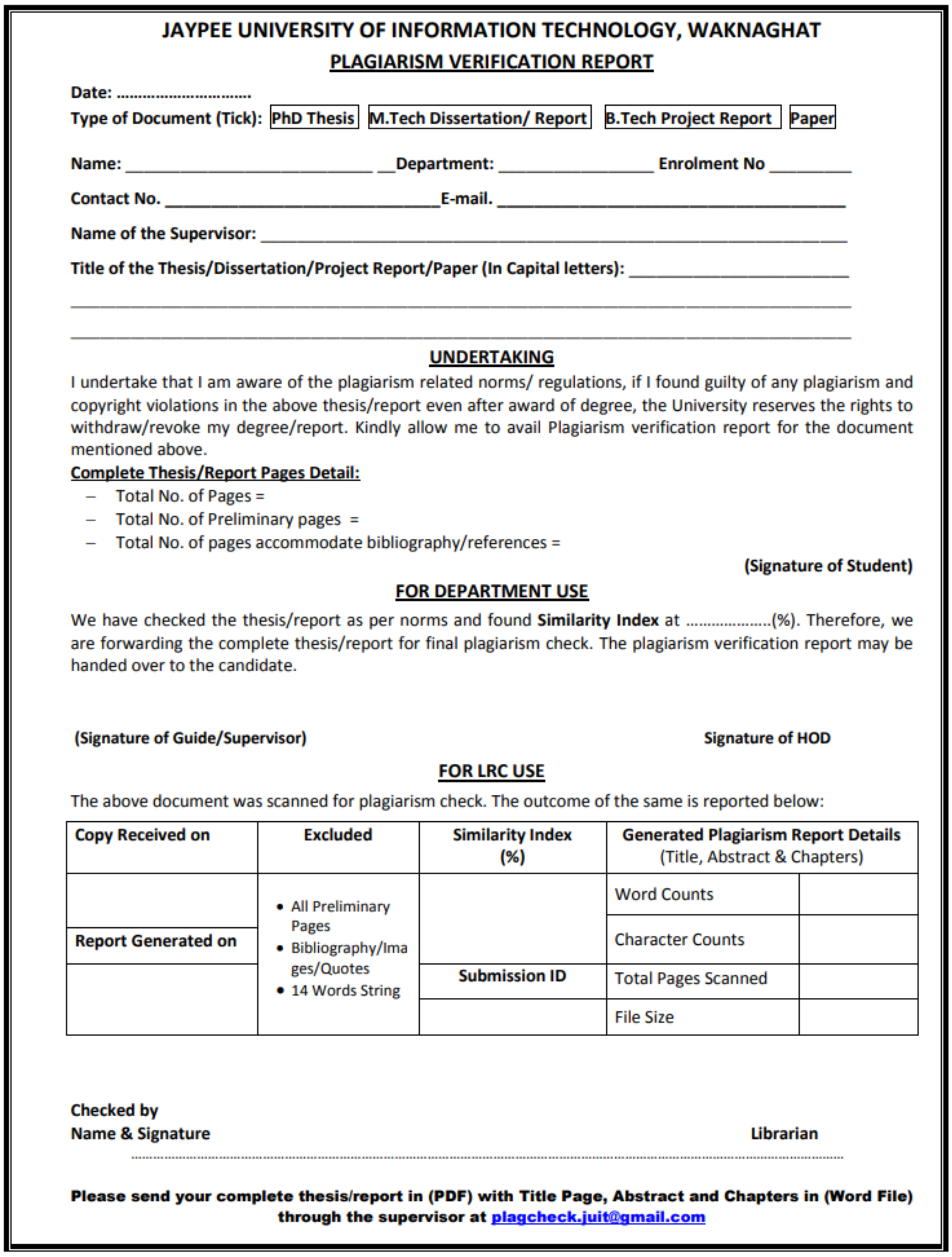
I hereby declare that the work presented in this report entitled "**Stock Market Prediction Using Optimised LSTM Model**" in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from March 2023 to June 2023 under the supervision of Dr Pardeep Kumar, Associate Professor, Department of CSE & IT. I also authenticate that I have carried out the above-mentioned project work under the proficiency stream Data Science. The matter embodied in the report has not been submitted for the award of any other degree or diploma.

PRAKHAR SRIVASTAVA, 191509

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

DR PARDEEP KUMAR,
ASSOCIATE PROFESSOR,
DEPARTMENT OF CSE&IT
Dated:

PLAGIARISM CERTIFICATE



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: ____________________________ __**Department:** _________________ **Enrolment No** _________

Contact No. _______________________________**E-mail.** _______________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ________________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

– Total No. of Pages =
– Total No. of Preliminary pages =
– Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENTS

Firstly, I express my heartiest thanks and gratefulness to almighty God for his divine blessing in making it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Dr. Pardeep Kumar**, Assistant Professor, Department of CSE & IT, Jaypee University of Information Technology, Wakhnaghat. Deep knowledge & keen interest of my supervisor in the field of "Data Science" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Pardeep Kumar,** Assistant Professor, Department of CSE & IT, for his kind help in finishing my project.

I would also generously welcome each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with respect the constant support and patience of my parents.

PRAKHAR SRIVASTAVA (191509)

TABLE OF CONTENT

Title	**Page No.**
Certificate	i
Plagiarism Certificate	ii
Acknowledgement	iii
List of Abbreviations	v
List of Figures	vi
List of Graphs	vii
List of Tables	viii
Abstract	x
Chapter - 01 (Introduction)	1
Chapter - 02 (Literature Survey)	11
Chapter - 03 (System Design & Development)	19
Chapter - 04 (Experiments & Result Analysis)	39
Chapter - 05 (Conclusion)	42
Reference	44
Appendices	46

LIST OF ABBREVIATIONS

Abbreviation	**Definition**
LSTM	Long-Short Term Memory
SVM	Support Vector Machine
S&P500	Standard and Poor’s 500
SEC	Securities and Exchange Commission
RNN	Recurrent Neural Network
RMSE	Root Mean Square Error
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
MPA	Mean Prediction Accuracy
MDA	Movement Direction Accuracy
CNN	Convolutional Neural Network

LIST OF FIGURES

Figure No.	Title	Page No.
1.4.1	Architecture of LSTM model	4
1.4.2	LSTM equations	5
1.4.3	Random Forest Architecture	6
1.4.4	Ensemble Learning Architecture	7
2.1.1	Schematic model of the research paper	11
2.1.2	Architecture overview of the research paper	12
2.1.5	Proposed model of 2D - CNNPred	15
2.1.6	Proposed model of 3D - CNNPred	15
2.1.7	Generic Scheme of the research paper	17
3.1.20	LSTM model summary	33
3.1.23	LSTM model shape	37

LIST OF GRAPHS

Graph No.	Title	Page No.
2.1.4	Graphs of F1 score of comparison metrics	14
3.1.16	Historical View of Closing Price	31
3.1.17	Historical View of Volume Traded	31
3.1.18	Moving Averages of stock	32
3.1.21	Train Loss and Validation Loss	34
3.1.22	Graph of model without sentiment	34
3.1.24	Train Loss and Validation Loss (sentiment)	38
3.1.25	Graph of model with sentiment	38

LIST OF TABLES

Table No.	Title	Page No.
2.1.3	Performance metrics comparison	13
3.1.1	Scrapped News Headline Dataset	20
3.1.2	Sentiment Score of News Headlines Dataset	21
3.1.3	Normalized Sentiment Score News Headline	22
3.1.4	Scrapped Subreddit Dataset	23
3.1.5	Merged Subreddit Dataset	23
3.1.6	Lower-cased Subreddit Posts	24
3.1.7	Sentence-tokenized Subreddit Posts	25
3.1.8	Word-tokenized Subreddit Posts	25
3.1.9	Subreddit Posts Stop words removed	26
3.1.10	Lemmatized Subreddit Posts	26
3.1.11	Cleaned Social Media Dataset	27
3.1.12	Social Media Dataset with Sentiment Score	28
3.1.13	Normalized Social Media Dataset	29

3.1.14	Google Stock Dataset	30
3.1.15	Final Merged Dataset (including sentiment score)	30
3.1.19	Normalized Stock Data	32
4.1	Result table	40
4.2	Simple LSTM predictions	40
4.3	LSTM (after Regularization) predictions	41
4.4	LSTM (after Cross-validation) predictions	41
4.5	LSTM (with sentiment score) predictions	41
4.6	LSTM (with sentiment score and Regularization) predictions	41

ABSTRACT

In today’s world there have been many technologies evolved that can efficiently predict the stock market data for a given period of time, the need to accurately predict the stock market data can also be achieved using various Machine Learning algorithms or classifiers, such as - Decision Tree Classifier, SVM, K-nearest Neighbors, Logistic Regression, etc. The problem with these types of Machine Learning algorithms is that, although they give a pretty good result on the training dataset, however for the testing dataset the accuracy is not up to the mark, that is often times they have high variance. Apart from the training model used for the classification, the type of dataset used also plays an important role in determining the accuracy of the prediction, the stock market prediction dataset is a combination of various features (Close Stock Price, Open Stock Price, Highest Stock Trading, Lowest Stock Trading, Volume Traded, Dividends, Stock Splits) which helps in determining whether the stock price will go up or go down on the following day; in addition to these features, there are numerous factors affecting stock prices these factors include - Financial News related to the company, Newsletters of the organization, Annual Report of the organization, dividends, launches, and the current market scenario. Natural disasters and epidemics can also affect stock prices. For instance, the COVID-19 pandemic significantly changed the stock prices of large technology companies, and we can see a decline in those prices as well. In contrast, the stock market prices of bio-medical companies increased as a result of an uptick in demand and strong business performance in that segment.

For the purpose of forecasting stock prices, I developed a machine learning model for the companies, that tracks the stock prices of companies listed on the stock exchange. The model uses Stacked LSTM model, to improve the accuracy of the model I have used the sentiment score of News Headlines related to the company, also I have used the sentiment score of social media posts to better train the model. In the end, I compared the predicted score with the actual score.

Chapter - 01
INTRODUCTION

1.1 Introduction

This project basically comes under the umbrella of Data Science and Machine Learning field. I have created a machine-learning prediction model that helps in the prediction of the stock trend of a company and also predicts whether the stock price will go up/down in the following days. The Stacked LSTM model, which is utilized in a variety of stock price prediction algorithms, was used to develop this model. Along with the Stacked LSTM model I have also included potential features like News Headlines related to the particular company, Social Network posts, to improve the prediction results. The timeframe of the dataset is of 69 days from March 30, 2023 to June 07, 2023 with a timestamp gap of 5 minutes. I tried comparing the results of the models where I used the sentiment scores with the model with where I have just used the Stacked LSTM model on the stock dataset.

The purpose of predicting stock market prices is to know whether one should buy or sell their stocks, and help them to take decisions based on predicted values. It is a well-known statistic that 60 per cent of the total of all stock transactions in the U. S. are carried out by bots and algorithms. As a result, Wall Street employs a ton of data scientists, one of whose major objectives is to address the issue of predicting when the market rises and falls.

The key factor in determining the crest and the nadir of a stock depends upon the horizon of the dataset we take into consideration, that is, what is the forecast of a stock in minutes, hours, days, weeks, and even years. The majority of investors base their decisions on metrics like revenue, costs, profits, and other fundamental information from financial statements and SEC filings; however, a data analyst can set this problem up by leveraging what's known as alternative data, which are metrics like a company's social media activity, customer reviews, credit card purchases, and other fundamental information of that nature that are available publicly; but one of the main issues is the lack of data, in my opinion. The second problem is that, for firms that have been publicly traded for, say, 5 years, we only really have data for 52 weeks in a year or 260 weeks in total. One approach that some researchers have taken is that if they want sales, they'll take weekly sales divided by seven to get mean and then add some statistical noise just to capture some of the natural variations that happen with sales. This strategy was used in a recent MIT research, which demonstrated that they could 57% of the time estimate quarterly results better than Wall Street.

1.2 Problem Statement

The majority of stock market prediction models use Decision Tree Classifier, SVM, Logistic Regression, K-nearest Neighbor, etc.; however, the problem with these base models is that they do not accurately predict the stock market prices because they are generally overfitted models, such that, they have a high level of variance; the precision on the training data is quite good (training error is low), but when we compare it with the precision on the test dataset, it is not as good (the testing error is high). A lot of these attempts to predict the stock market does seem to fall a little bit flat. According to economist Burton Malkiel, a stock's price is better defined by a random walk because each day-to-day variation in the price is random and unpredictable because all the information that is currently known about a stock's price has already been factored into it.[1] As a result, any change in the price would only portray the release of new info or random noise. Outside factors also affect the stock prices trends, the 2016 geopolitical tension with North Korea, Iran's negotiation, and trade deal with China, and the global pandemic COVID - 19, all of these things had dramatic impacts on the stock market and in order to have even a remotely accurate prediction over the relevant time window, we would have to correctly predict these things too. Investors were naturally quite anxious about the tensions with

North Korea in 2017 and with China in 2019. As a result, there were many ups whenever things looked impressive and many downs whenever things looked awful. In the end, everything depends on what people do and speak.

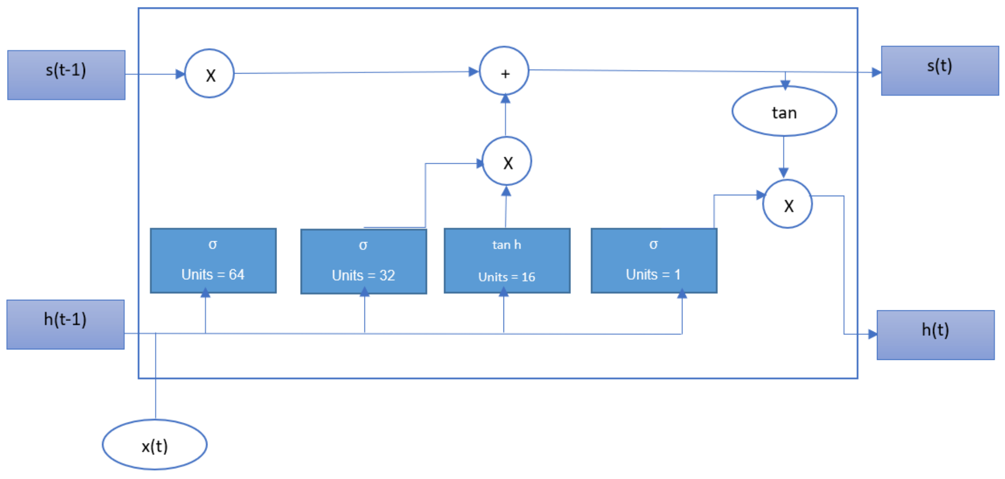
In almost the majority of traditional models, a direct correlation between the data is assumed. This presumption forcefully calls into doubt the reliability of the traditional time series data, which are frequently nonlinear. In order to accurately calculate the value of a stock, it's crucial to pick an appropriate historical window size. Too big of a window can cause you to overlook vital information. If the window size is too small, the prediction will be impacted by a lot of information or sentiments before the window time frame.

1.3 Objectives

The objective of developing this machine learning model for the stock market price prediction is to have a better understanding of the stock market trends beforehand so that it could help investors to know the trend of the stock prices in the present scenario.[2] The major emphasis is on doing technical analysis, which is studying historical price movement in order to predict accurately what the market may do next. Technical analysis is how we do this since it is focused entirely on price, which is the basis for all technical analysis. We are aware that there are many intricate financial indicators and that the stock market fluctuates wildly, but as technology advances, more opportunities to build a steady financial foundation through the stock market are opening up. Additionally, it helps experts identify the most comprehensive indicators so they can provide better predictions for people who are not familiar with stocks. Stocks provide you with the right to a piece of a company's assets and profits; they are essentially equity investments that represent a percentage of a company's ownership or organization.

Because one has to make investments in a stock that will see a growth in value over the years rather than a fall, the market value forecast is crucial in assisting in maximizing the profits of stocks. Recurrent neural networks, or RNNs, are among the finest models for handling sequential data. The memory cell, a computational unit, from the long short-term memory model, takes the role of typical artificial neurons in the network's hidden layer. These memory cells allow networks to effectively link memories to stock values. The basic goal of generating stock price forecasts is to realize big returns. It's difficult to predict how the stock market will change. Any outside factor has the potential to affect the stock price. These include controversial issues, financial news, stock movements of rival companies or industries, and other hard-to-define things like rumors, anxiety, and other behavioral elements. The challenge of gathering all these data makes machine learning a possible contender for stock market prediction. It can evaluate the patterns and how they relate to a predictor variable using a lot of inputs.

1.4 Methodology



1.4.1 Architecture of LSTM Model

$$\Gamma_f^{(t)} = \sigma(W_f\ [\ a^{(t-1)}\ , x^{(t)}\ ] + b_f\ )$$

$$\Gamma_u^{(t)} = \sigma(W_u\ [a^{(t-1)}\ , x^{(t)}\ ] + b_u\ )$$

$$\bar{c}^{\ (t)} = tanh(W_c\ [a^{(t-1)}\ , x^{(t)}\ ] + b_c\ )$$

$$c^{(t)} = \Gamma_f^{(t)} \circ\ c^{(t-1)} + \Gamma_u^{(t)} \circ\ \bar{c}^{(t)}$$

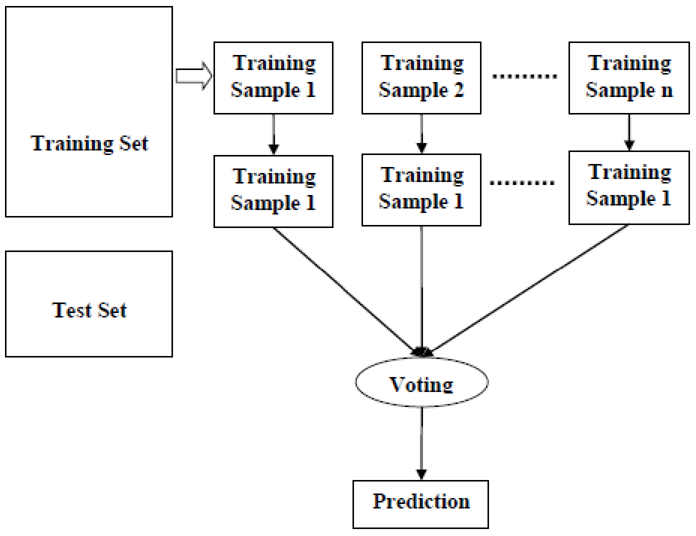
$$\Gamma_{\circ}^{(t)} = \sigma(W_{\circ}\ [a^{(t-1)}\ , x^{(t)}\ ] + b_{\circ}\ )$$

$$a^{(t)}\ = \Gamma_{\circ}^{(t)} \circ\ tanh(c^{(t)})$$

1.4.2 LSTM equations

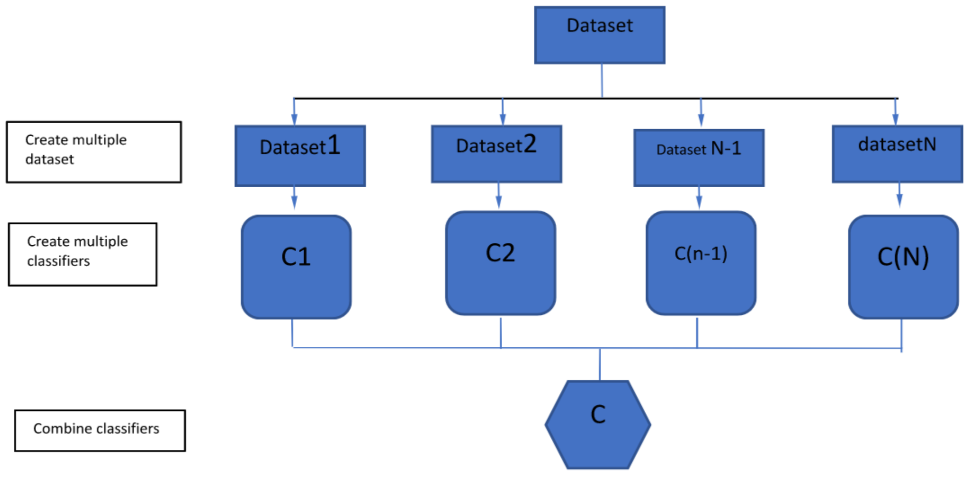
Apart from the basic Recurrent Neural Network architecture, the Long Short Term Memory architecture consists of forget gate that keeps the historical data in memory (fig. 1.4.1), for the sake of illustration, Let's imagine that when reading, we wish to use our long short-term memory to keep track of grammar patterns like the singular or plural nature of the subject. In order to delete the previously saved memory content of the single or multiple states when the subject switches from a singular term to a plural term, we must find out how to use a long short-term memory model's forget gate. Another new term in the Long Short-Term Memory is the "W_f" are weights that govern the forget gate's behavior, it concatenates the value in the brackets $a^{(t-1)}$, $x^{(t)}$ and multiplies with the weights, and the equation results in a vector called *tau F* with values ranging between zero and one. This Forget gate vector will be multiplied element-wise by the previous cell state so if one of the values of *tau F* is zero or close to zero then it means that the Long Short Term Memory model should remove that piece of information in the corresponding component. If one of the values is in one then it will keep the information next. When we lose track of the fact that the topic under discussion is single, we must find a means to update the gate to reflect the fact that the new topic is plural; there are a number of formulae for updating the gate in a manner similar to the forget gate. In order to update the new subject, we must generate a new vector of values that we can add to our prior cell state in order to produce our final new state. *tau U* is once more a vector of values between zero and one, and this will be multiplied element-wise with *c(t)* to compute the value. To decide which outputs, we are going to use, we will be using two formulae *tau* $^{t}{}_{o}$ helps to decide what to output using a sigmoid function and a^{t} is multiplied with *tan h* of the previous state. In contrast to the more basic model of an RNN, which struggles to acquire knowledge retention over longer periods of time, LSTMs are essentially expressly built to avoid long-term reliance problems. They nearly always recall information for longer lengths of time.[3]

The Random Forest Classifier is the other traditional model that we used in the blended model for the prediction of stock prices. It is one of the many potent tools in the machine learning library, and it has many uses, including remote sensing. For example, it is employed in the ETM (Enhanced Thematic Mapper) devices, which are used on satellites that see far outside the human spectrum for having a look at land masses and they acquire images of them. Machine learning's Random Forest is a classification tool. Classification is a type of issue where the results are categorical in nature, such as "yes" or "no," "true" or "false," or "zero" or "one."



1.4.3 Random Forest Architecture

In contrast to other traditional machine learning models, the Random Forest Classifier does not overfit the data, which means that it considers the outputs of multiple trees to reduce the risk of overfitting. Another benefit of using the Random Forest Classifier is that it requires less training time. Overfitting occurs when the data is fitted too close to the sample that we focus on the oddities rather than being able to predict the entire data set. In the big data world of today, this is crucial, and this is where random forest really shines. Data in the modern world is quite messy, thus if we have a random forest, it can keep precision when a significant amount of the data is absent. It guesses missing data. This means that if we have data that comes in from five or six various sets of statistics so they have some of the same shared information but one of them is missing, it will look at that individually and build a model for it. Multiple decision trees are built using the Random Forest method, and the method chooses the majority decision of the trees as the final decision. A decision tree has generally four terminologies - entropy, entropy is the measure of randomness; information gain is the measure of the decrease in the entropy after the dataset is split what that means here is that we've gone from one set which has a very high entropy to a lower set of entropy; leaf node carries the classification or the decision.



1.4.4 Ensemble Learning Architecture

Individual models are combined using the ensemble learning approach to increase the model's predictability and stability. In this method, numerous machine learning models are combined into a single predictive model in order to reduce variance using bagging, reduce bias using boosting, or enhance predictions using stacking. Ensemble allows for greater predictive performance versus a single model. Instead of learning a single sophisticated model, train numerous basic models and integrate their output to arrive at the final choice. We discover that certain models are good at modelling one component of the data while others are good at modelling another. In ensemble learning, the cumulative model strength balances out the biases and variances of each individual model. A composite prediction from ensemble learning will have a final accuracy that is higher than the performance of the individual models. Ensemble techniques may be separated into two categories - parallel ensemble method and sequential ensemble method. The fundamental goal of sequential approaches is to take advantage of the reliance between the base learners. By giving previously mislabeled instances more weight, a model's total performance may be improved, and base learners are created sequentially in this case, called "AdaBoosting".

Anywhere the basic learners are built in parallel, such as with a random forest classifier, parallel techniques are employed since the mistakes are typically dramatically reduced by averaging. The primary goal of parallel techniques is to use diversity between base learners; we are utilizing a parallel ensemble approach where the many models are integrated based on which ones perform better with certain kinds of data, in the ensemble model, the most useful features of each model are taken into account based on the input data, and an ensemble learning model is then used to generate predictions. The use of numerous models in an ensemble model improves the performance of a single model by addressing two key issues: accuracy and robustness, ensemble models' robustness comes from including predictions from all of the base learners' models. All weak models may be effectively merged to form an ensemble model, which can then be used to produce more accurate and reliable models.

By merging data from several modelling methodologies, you may also create an ensemble of well-designed, robust, and diversified models; ensemble models are much more accurate and resilient than a single model that has been carefully optimized. Model averaging is a method of ensemble learning in which each member of the group makes an equal contribution to the prediction. The ensemble prediction for regression is obtained as the mean of the member projections, the prediction is computed as that of the mode of the member projections when predicting a class label, and the average maximum of the accumulated probabilities for each target class may be used to determine the prediction in the case of class probability. This method's drawback is that each model contributes equally to the ensemble's final forecast. Despite the fact that some models are shown to perform significantly better or less effectively than other models an improvement on a model-averaging ensemble is a weighted average ensemble, where each member's contribution to the outcome is weighed according to the model's performance. Since each model's weights are modest positive numbers that add up to 1, they may be used to calculate each model's predicted performance or reliability. Combining the findings of many models to produce a general outcome from a single model is the principle underlying bagging, by averaging several estimates, bagging or bootstrap aggregation lowers variations of an estimate. Creating randomly picked data sets from the initial training data and bootstrapping are the first two of the three processes in the bagging process. The second stage is to construct and fit various classifiers to each of these various copies. The third stage is to average the guesses to come up with a final, overall projection.

1.5 Organization

Like any other Stock Market prediction model, the developed model also focuses on the importance of the datasets that have been taken in order to predict the stock prices. It has been observed that changes in social media mood are correlated with changes in the financial markets. Companies may have accumulated mountains of consumer input in the highly data-overwhelmed environment of today, but it is still hard for mere humans to manually assess the data without biases or inaccuracy. In data analysis, time series forecasting and modelling are crucial. Analytics frequently employ time series analysis, a specialized area of statistics.

Chapter – 02

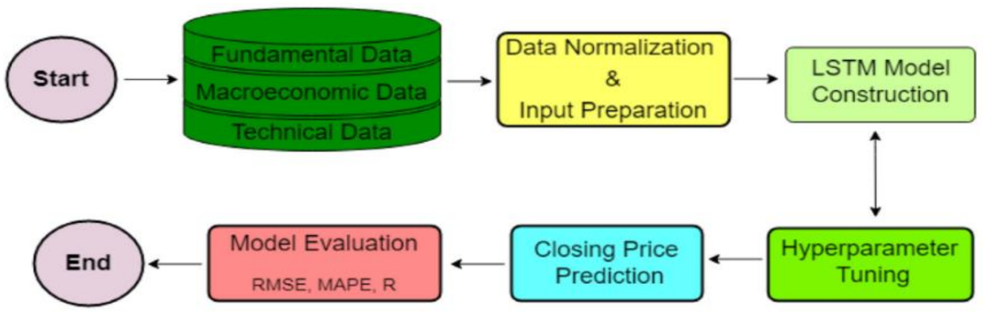
LITERATURE SURVEY

2.1 Related Information

[4] - Predicting stock market index using LSTM

The research was done by Hum Nath Bhandari, Binod Rimal, Nawa Raj Pokhrel, Ramchandra Rimal, Keshab R. Dahal, Rajendra K.C. Khatri. The paper got published in the year May 2022.

The technology used in this paper is Long-Short Term memory using both single-layer as well as multi-layer LSTM. The dataset on which this model was applied is the S&P 500 index along with features including Fundamental data - such as news of the organization, macroeconomic data, and technical indicators. Performance evaluation is done using various metrics such as - RMSE (Root Mean Square Error), MAPE (Mean Absolute Percentage Error), and Correlation Coefficient.



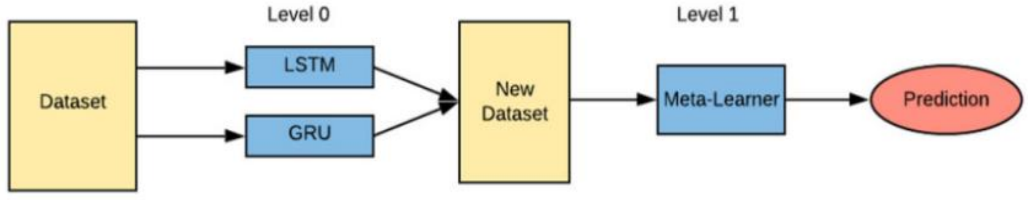
2.1.1 Schematic model of the research paper

RNN is more suitable for sequential data modelling since stock market data is typically non-linear. This study makes a contribution by carefully choosing the variables and considering how they could affect larger marketplaces and other parts of the economy. A good model created using the ideal collection of features may adequately forecast stock prices and provide better market information. Almost all traditional models presume a linear connection between the variables. Since the time series data in real-world applications are frequently nonlinear, this assumption sharply calls into question the reliability of the traditional time series models. A really well deep learning method in RNN for time series forecasting is LSTM. For instance, LSTM is employed for classification and regression issues in both the modelling of rainfall runoff and stock market predictions.

[5] - A novel ensemble deep learning model for stock prediction based on stock prices and news

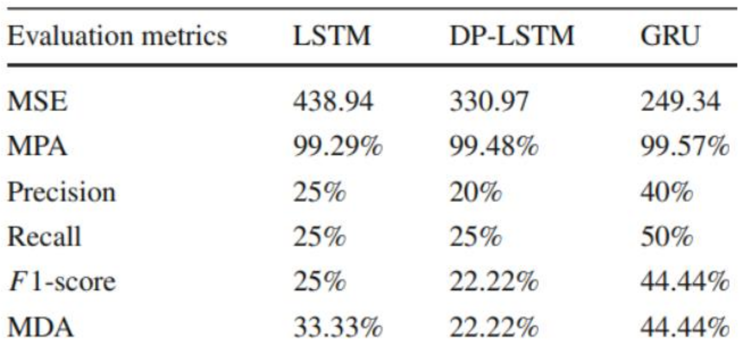
The research was done by Yang Li, Yi Pan, the research paper got published in the year September 2021.

The technology used in this paper is an ensemble learning method to combine two Recurrent Neural Networks followed by a fully connected Neural Network. The dataset used for the development of this project is S&P 500, along with news data from various news websites such as - CNBC.com, Reuters.com, WSJ.com, and Fortune.com. Performance evaluation is done using various metrics such as - MSE (Mean Squared Error), MPA (Mean Prediction Accuracy, and MDA (Movement Direction Accuracy) for the ensemble model.



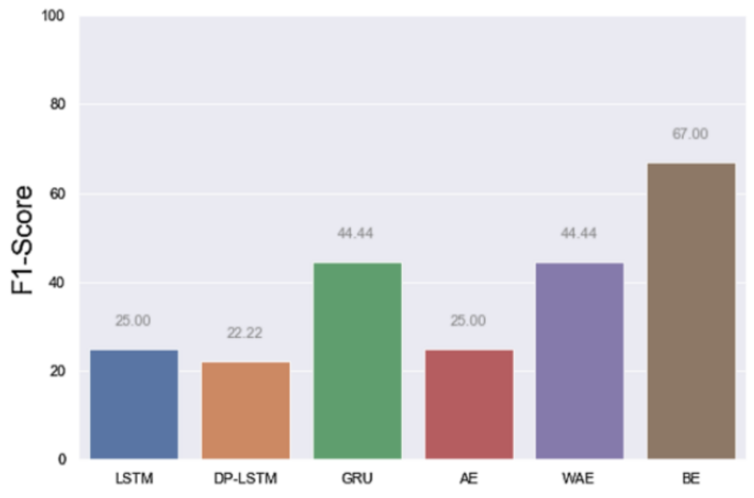
2.1.2 Architecture overview of the research paper

The challenge with future stock forecasting is that there are far too many variables at play that simultaneously impact the magnitude and frequency of stock price rises and falls. Stock prices fluctuate as a result of market forces, which are supply and demand in the financial markets. If more individuals want to buy a stock than sell it, the price of that stock will increase. In a situation when there is a greater supply than demand for an item, the price will fall. As the price increases, the quantity sought lowers. Demand rises in proportion to the price decline. This is the Demand Law. The stock price would undoubtedly decline if demand dropped. This work makes the following key contributions: First, we use sentimental analysis to get sentimental ratings from several news agencies. Second, we derive estimates based on both ratings and historical data by combining two complementary contemporary deep learning architectures. Third, in order to further enhance the predictions, we combine the judgments using a clever ensemble neural network. To build the blending ensemble model, a dataset is necessary. The LSTM model and the GRU model are level 1 sub-models that are trained using the training data. Following the first training phase, we utilize the level 1 models that have been trained to draw conclusions on the validation data, which is essentially the level 2 model's training set. Additionally, the test results are utilized to calculate the final prediction's accuracy.



Evaluation metrics	LSTM	DP-LSTM	GRU
MSE	438.94	330.97	249.34
MPA	99.29%	99.48%	99.57%
Precision	25%	20%	40%
Recall	25%	25%	50%
$F1$-score	25%	22.22%	44.44%
MDA	33.33%	22.22%	44.44%

2.1.3 Performance metrics comparison

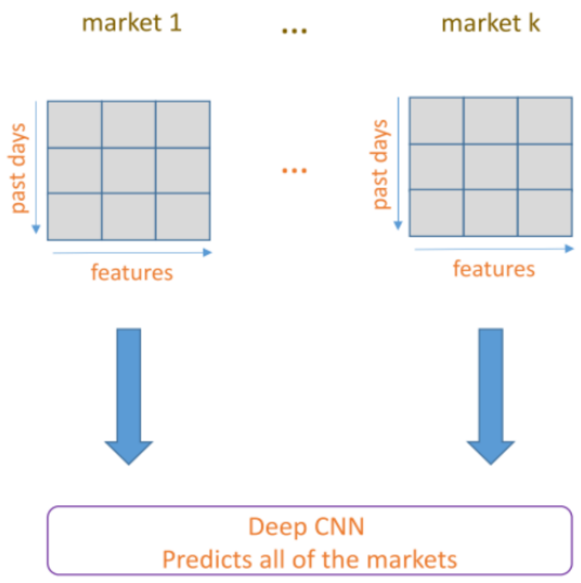


2.1.4 F1 Score of different comparison metrics

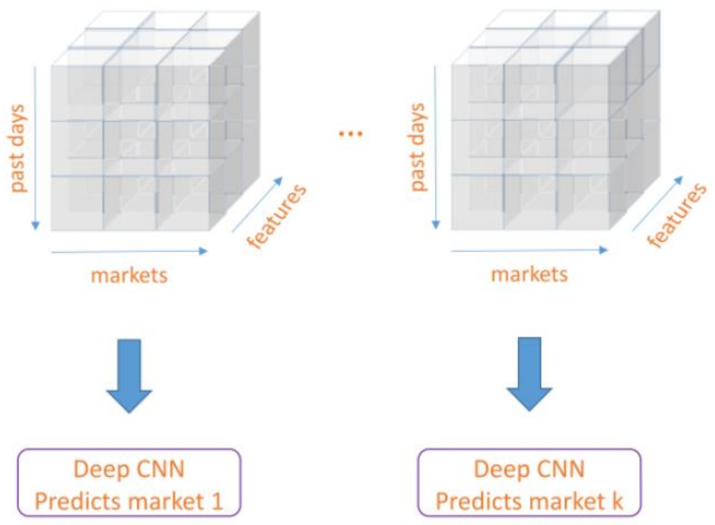
[6] - CNNpred: CNN - based stock market prediction using a diverse set of variables

This research was done by Ehsan Hoseinzade, Saman Haratizadeh, the research paper was published in the year March 2019.

The technology used in this research is a Convolutional Neural Network for the prediction of stock prices. The building of a model for feature extraction from a number of variables that comprise data from historical records of pertinent marketplaces is the primary topic of this research. This information contains initial fundamental factors such as plain historical prices, technical indications, or changes in those variables during the previous day or two. An algorithm like CNN looks to be a viable solution for this feature extraction challenge given the variety of the input data and potential complexity of the feature space that might be necessary for a decent prediction.



2.1.5 Proposed model of 2D – CNNPred



2.1.6 Proposed model of 3D – CNNPred

CNNPred is a broad framework for stock market forecasting based on CNN. The terms 2D - CNNPred and 3D - CNNPred relate to the two variants of

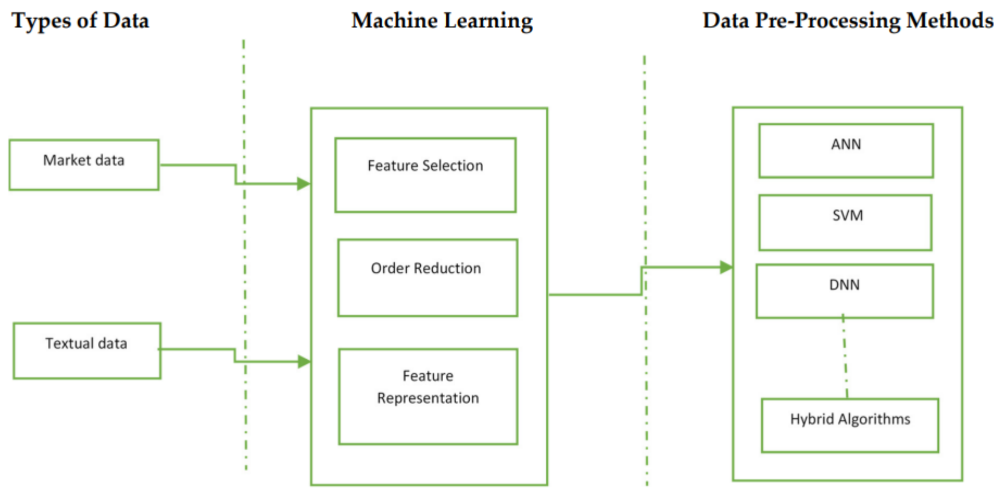
CNNPred. We break down the structure into four main sections. input data visualization, daily feature extraction, durational feature, and outcome prediction. Finding a general model to connect a market's past with its present fluctuations is the aim of the first strategy. By general model, we refer to a model that holds true across many markets. In other words, we presume that for many markets, the true mapping function from the past to the future is the one that is accurate. To do this, we first create a single model that can forecast a market's future based on its own historical data; but, in order to obtain the appropriate mapping function, this model must be trained on samples from many markets. The second method, 3D - CNNPred, on the other hand, makes the assumption that different models are required for generating predictions in various markets, but that each forecasting model can draw data from a variety of markets' historical data.

[7] – Stock Market Prediction Using Machine Learning Techniques: A Decade Survey on Methodologies, Recent Developments, and Future Directions

This research was done by Nusrat Rouf, Majid Bashir Malik, Tasleem Arif, Sparsh Sharma, Saurabh Singh, Satyabrata Aich, and Hee-Cheol Kim, the research paper was published in the year November 2021.

The opening of the study emphasizes the growing importance of machine learning methods for stock market forecasting. By outlining the drawbacks of conventional strategies and highlighting the demand for cutting-edge methodologies to capture the complexity of stock market data.

The researchers then go on to discuss alternative machine learning-based stock market forecast approaches. They examine more sophisticated approaches like neural networks, ensemble methods, and deep learning models as well as more conventional techniques like regression-based models, decision trees, and support vector machines. To give a thorough grasp of each method's application in stock market forecasting, its advantages and disadvantages are analyzed. In order to properly prepare the input data for prediction models, the survey also includes feature selection and preprocessing strategies. The researchers investigate several strategies, elaborating on their effects on the performance of prediction models, including principal component analysis, wavelet transformations, and data normalization.



2.1.7 Generic Scheme of the research paper

We go into great detail on evaluation criteria and performance measurements that are used to judge how well stock market prediction algorithms work. The study provides widely used measures, including mean absolute error, F1-score, recall, accuracy, and precision. Additionally, it looks at more sophisticated metrics like profit-based assessment and risk-adjusted returns, which include the practical effects of using prediction models in actual trading scenarios.

The use of ensemble methods and hybrid models is emphasized as successful stock market forecasting strategies. The researchers talk about ensemble approaches, which may combine numerous models for better accuracy. Examples include bagging, boosting, and stacking. In order to improve forecast accuracy, they also investigate hybrid models, which combine machine learning with other strategies including sentiment analysis, technical analysis, and macroeconomic aspects.

The study also discusses current research in machine learning-based stock market prediction. It addresses recent developments including explainable AI, transfer learning, and deep reinforcement learning. Additionally, as a way to supplement traditional market data, the integration of other data sources - such as social media sentiment, news articles, and web scraping - is studied.

Chapter - 03

SYSTEM DESIGN AND DEVELOPMENT

The proposed system uses Long-Short Term Memory as a model for predicting stock prices, it also takes into account company-related news and social media posts about the company. I performed sentiment analysis on the news headlines and social media posts and used it as a feature along with the LSTM model.

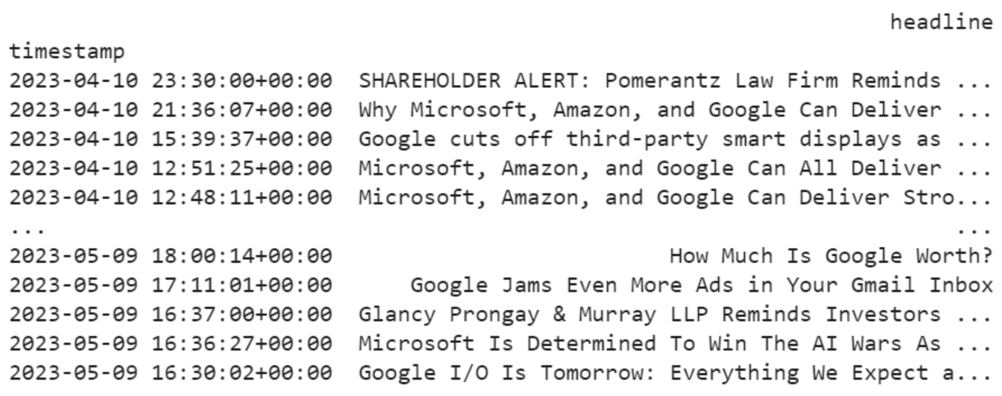
3.1 Model Development

3.1.1 Data source

3.1.1.1 Company-related News

The news headlines related to the company are gathered using Application Programming Interface (API). The Application Programming Interface I used for obtaining the news headlines is “NewsAPI” which provides worldwide news related to a particular company or organization or in general daily news along with the date and timestamp on which the news was posted, the module used for obtaining news is “newsapi-python”. For this project, I obtained news headlines of “Google” in the timeframe “2023-04-10” to “2023-06-01”.

The limitation of using NewsAPI or any Application Programming Interface is that we get data only in a limited timeframe. After obtaining the data, I sorted the news using the date and time. Also, I filtered the dataset so that the headlines should contain the words “earnings”, “revenue”, “profit”, and “lawsuit”, which are relevant to the stock price of Google.



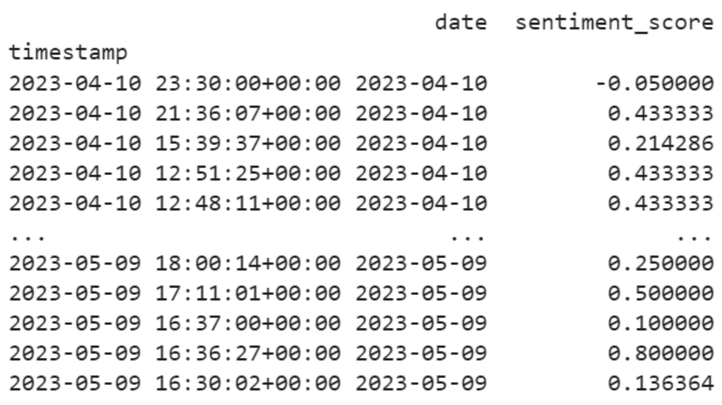
```
                                                                         headline
timestamp
2023-04-10 23:30:00+00:00  SHAREHOLDER ALERT: Pomerantz Law Firm Reminds ...
2023-04-10 21:36:07+00:00  Why Microsoft, Amazon, and Google Can Deliver ...
2023-04-10 15:39:37+00:00  Google cuts off third-party smart displays as ...
2023-04-10 12:51:25+00:00  Microsoft, Amazon, and Google Can All Deliver ...
2023-04-10 12:48:11+00:00  Microsoft, Amazon, and Google Can Deliver Stro...
...                                                                           ...
2023-05-09 18:00:14+00:00                              How Much Is Google Worth?
2023-05-09 17:11:01+00:00      Google Jams Even More Ads in Your Gmail Inbox
2023-05-09 16:37:00+00:00  Glancy Prongay & Murray LLP Reminds Investors ...
2023-05-09 16:36:27+00:00  Microsoft Is Determined To Win The AI Wars As ...
2023-05-09 16:30:02+00:00  Google I/O Is Tomorrow: Everything We Expect a...
```

3.1.1 Scrapped News Headline Dataset

The above figure (fig._3.1.1) shows the news headlines that are extracted using the NewsAPI. For the timeframe – “April 10, 2023 to May 09, 2023” around 340 news headlines were extracted, that is, eleven headlines per day. Additionally, for the timeframe – “May 09, 2023 to June 01, 2023” around 202 news headlines were scrapped, that is, ten headlines per day.

After extracting the news headlines, I calculated the sentiment score for each news headline, for this I used “TextBlob sentiment polarity”. TextBlob library from the textblob package is used for the calculation of sentiment scores. To carry out sentiment analysis, the TextBlob library combines rule-based and pattern-based methods. It uses a sentiment analysis model that has already been trained and is based on a vocabulary of words with assigned sentiment ratings. A text's sentiment polarity is determined by adding up the emotion ratings of all the words it contains, then normalizing the result.

On a continuous scale from -1.0 to 1.0, where negative values indicate negative sentiment, positive values indicate positive sentiment, and values near to zero suggest neutral emotion, the sentiment polarity score shows the sentiment or opinion indicated in the headline.

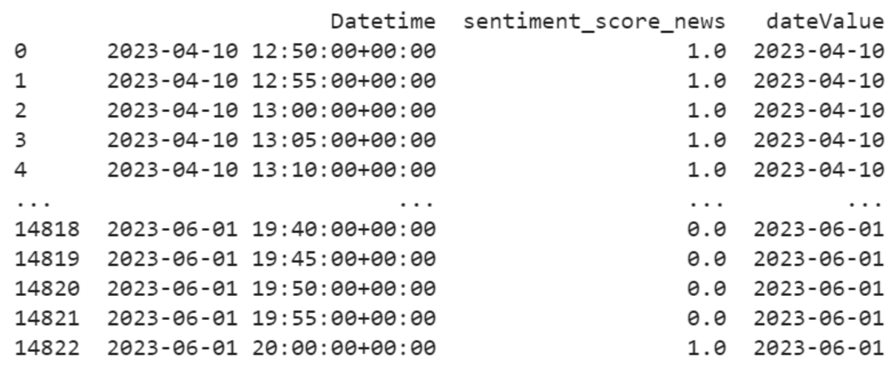

```
                                 date  sentiment_score
timestamp
2023-04-10 23:30:00+00:00 2023-04-10        -0.050000
2023-04-10 21:36:07+00:00 2023-04-10         0.433333
2023-04-10 15:39:37+00:00 2023-04-10         0.214286
2023-04-10 12:51:25+00:00 2023-04-10         0.433333
2023-04-10 12:48:11+00:00 2023-04-10         0.433333
...                                 ...              ...
2023-05-09 18:00:14+00:00 2023-05-09         0.250000
2023-05-09 17:11:01+00:00 2023-05-09         0.500000
2023-05-09 16:37:00+00:00 2023-05-09         0.100000
2023-05-09 16:36:27+00:00 2023-05-09         0.800000
2023-05-09 16:30:02+00:00 2023-05-09         0.136364
```

3.1.2 Sentiment Score of News Headline Dataset

The above figure (fig._3.1.2) shows the calculated sentiment score for the timeframe - “April 10, 2023 to May 09, 2023” along with the timestamp of each news headline. Similarly, sentiment scores were calculated for the timeframe - “May 09, 2023 to June 01, 2023”. Both the DataFrames were merged.

Since, the stock dataset is in the timestep of 5 minutes, the news headlines dataset needs to be also normalized, so I resampled the merged DataFrame with the time frequency of 5 minutes. Using the forward-fill approach, the missing values in the 'sentiment_score' column are filled with the most recent non-missing value in the same column. It is helpful when working with time series data or where consistency of numbers is sought in later computations or analysis. The resulting DataFrame contains 14,822 rows.

After calculating the sentiment scores of the news headlines and merging the DataFrames, the sentiment scores are normalized in the range [-2, 2], where, -2 represents extremely negative news, -1 represents negative news, 0 represents neutral news, +1 represents positive news, and +2 represents extremely positive news. After normalizing the dataset looks like (fig._3.1.3)



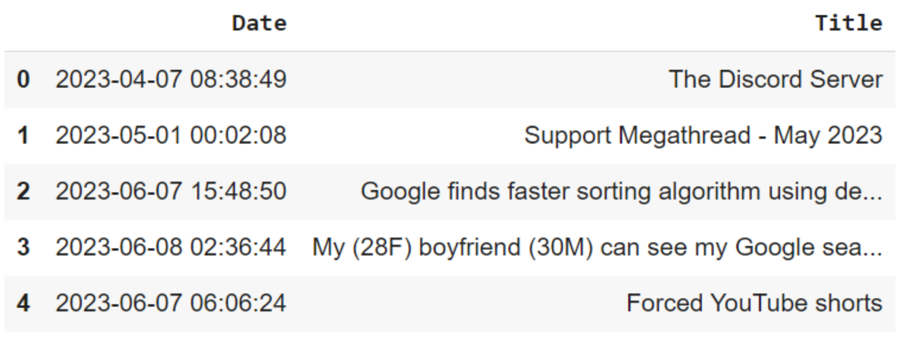
```
                        Datetime  sentiment_score_news   dateValue
0      2023-04-10 12:50:00+00:00                   1.0  2023-04-10
1      2023-04-10 12:55:00+00:00                   1.0  2023-04-10
2      2023-04-10 13:00:00+00:00                   1.0  2023-04-10
3      2023-04-10 13:05:00+00:00                   1.0  2023-04-10
4      2023-04-10 13:10:00+00:00                   1.0  2023-04-10
...                          ...                   ...         ...
14818  2023-06-01 19:40:00+00:00                   0.0  2023-06-01
14819  2023-06-01 19:45:00+00:00                   0.0  2023-06-01
14820  2023-06-01 19:50:00+00:00                   0.0  2023-06-01
14821  2023-06-01 19:55:00+00:00                   0.0  2023-06-01
14822  2023-06-01 20:00:00+00:00                   1.0  2023-06-01
```

3.1.3 Normalized Sentiment Score News Headline

3.1.1.2 Social Media Posts

The social media platform I have used for extracting the posts is “Reddit”. It is a website where people can post anything like images, audio, and text posts just like Twitter. For scraping posts from Reddit, I used a web scraping tool called “PhantomBuster” and “Reddit Application Programming Interface (API)”, which allows users to scrape subreddit posts for the website for a particular timeframe. It scrapes posts, along with the date and time on which it was posted.

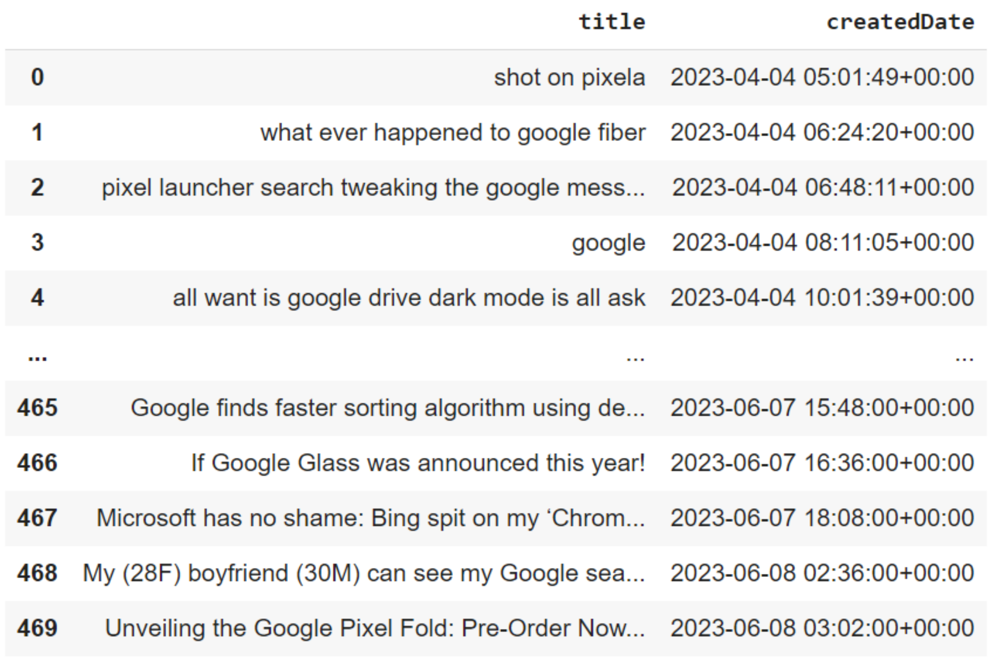
For this project, I obtained subreddit posts of “r/Google” in the timeframe “2023-04-04” to “2023-06-08”. For this I used “praw” package, Python package called PRAW (Python Reddit API Wrapper) offers a simple way to communicate with the Reddit API. It makes it easier for developers to construct application, bots, or scripts that interact with Reddit programmatically, which in turn streamlines the process of accessing and obtaining data from the Reddit platform.



	Date	Title
0	2023-04-07 08:38:49	The Discord Server
1	2023-05-01 00:02:08	Support Megathread - May 2023
2	2023-06-07 15:48:50	Google finds faster sorting algorithm using de...
3	2023-06-08 02:36:44	My (28F) boyfriend (30M) can see my Google sea...
4	2023-06-07 06:06:24	Forced YouTube shorts

3.1.4 Scrapped Subreddit Dataset

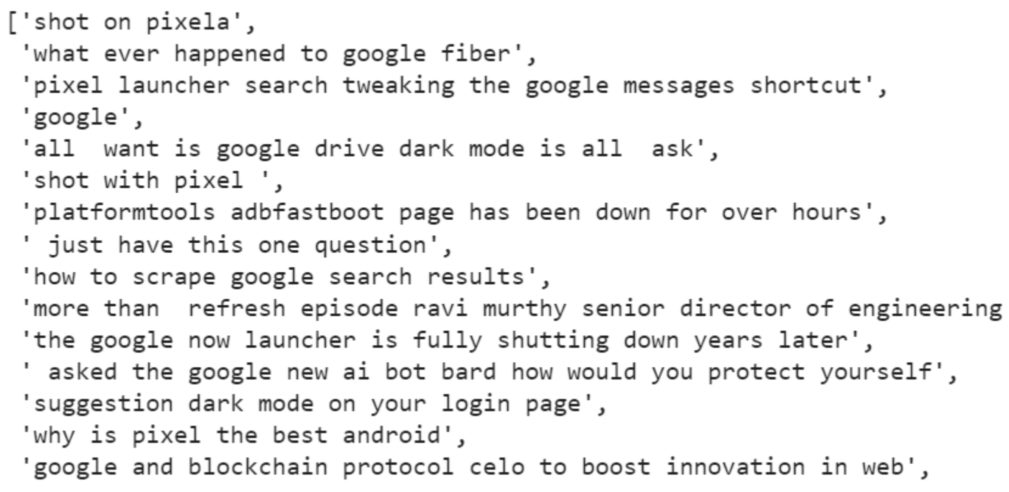
The above (fig._3.1.4) shows the subreddit posts that are extracted from "r/Google" using Reddit API. For the timeline – "April 04, 2023 to May 09, 2023" around 267 subreddit posts are scrapped using an online platform – "PhantomBuster", that is, around seven social media posts per day. For the timeline – "May 10, 2023 to June 08, 2023" a total of 206 posts were extracted using Reddit API, that is, around eight posts per day. Both the datasets are merged in a single DataFrame that has a total of 470 subreddit posts ranging from – "April 04, 2023 to June 08, 2023" (fig._3.1.5)



	title	createdDate
0	shot on pixela	2023-04-04 05:01:49+00:00
1	what ever happened to google fiber	2023-04-04 06:24:20+00:00
2	pixel launcher search tweaking the google mess...	2023-04-04 06:48:11+00:00
3	google	2023-04-04 08:11:05+00:00
4	all want is google drive dark mode is all ask	2023-04-04 10:01:39+00:00
...	...	...
465	Google finds faster sorting algorithm using de...	2023-06-07 15:48:00+00:00
466	If Google Glass was announced this year!	2023-06-07 16:36:00+00:00
467	Microsoft has no shame: Bing spit on my 'Chrom...	2023-06-07 18:08:00+00:00
468	My (28F) boyfriend (30M) can see my Google sea...	2023-06-08 02:36:00+00:00
469	Unveiling the Google Pixel Fold: Pre-Order Now...	2023-06-08 03:02:00+00:00

3.1.5 Merged Subreddit Dataset

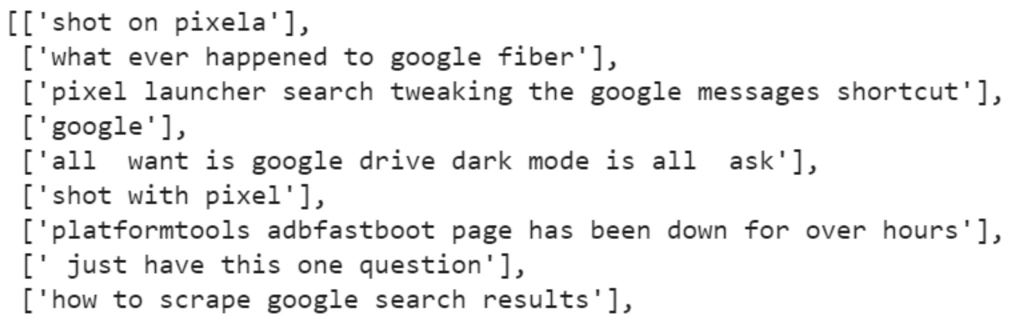
The subreddit posts are natural language in nature, so in order to get the accurate sentiment score for the posts, we need to perform Natural Language Processing (NLP) tasks on the posts. For that, the first pre-processing task is to convert all the text to lower case. (fig._3.1.6)



```
['shot on pixela',
 'what ever happened to google fiber',
 'pixel launcher search tweaking the google messages shortcut',
 'google',
 'all  want is google drive dark mode is all  ask',
 'shot with pixel ',
 'platformtools adbfastboot page has been down for over hours',
 ' just have this one question',
 'how to scrape google search results',
 'more than  refresh episode ravi murthy senior director of engineering
 'the google now launcher is fully shutting down years later',
 ' asked the google new ai bot bard how would you protect yourself',
 'suggestion dark mode on your login page',
 'why is pixel the best android',
 'google and blockchain protocol celo to boost innovation in web',
```

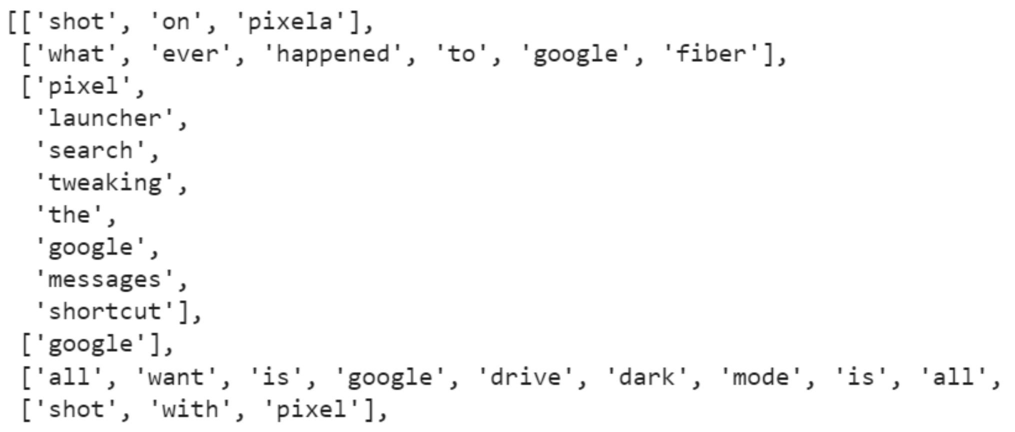
3.1.6 Lower-cased Subreddit Posts

The next task is – Tokenization. The practice of breaking up text or a string of characters into tokens is known as tokenization. Depending on the particular tokenization approach employed, tokens can also represent sub words, characters, or other significant units in addition to words. Because it helps transform unstructured text input into a structured format that machine learning algorithms can analyze, tokenization is a vital step in NLP. Tokenizing text makes it feasible to analyze, comprehend, and extrapolate meaning from each of the text's constituent parts. For this I used “sent_tokenize” and “word_tokenize” from the package – “nltk.tokenize”.



```
[['shot on pixela'],
 ['what ever happened to google fiber'],
 ['pixel launcher search tweaking the google messages shortcut'],
 ['google'],
 ['all  want is google drive dark mode is all  ask'],
 ['shot with pixel'],
 ['platformtools adbfastboot page has been down for over hours'],
 [' just have this one question'],
 ['how to scrape google search results'],
```

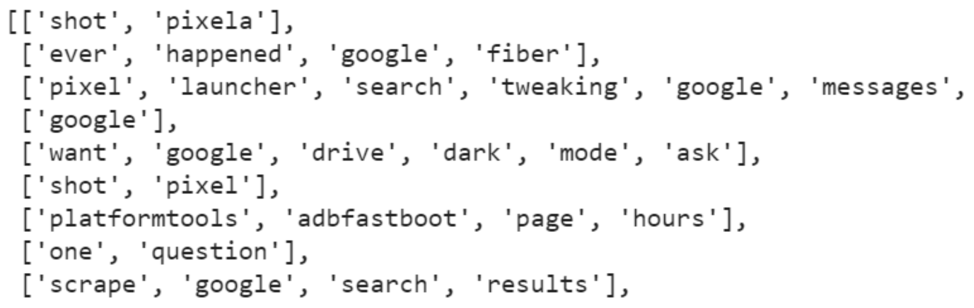
3.1.7 Sentence -Tokenized Subreddit Posts



```
[['shot', 'on', 'pixela'],
 ['what', 'ever', 'happened', 'to', 'google', 'fiber'],
 ['pixel',
  'launcher',
  'search',
  'tweaking',
  'the',
  'google',
  'messages',
  'shortcut'],
 ['google'],
 ['all', 'want', 'is', 'google', 'drive', 'dark', 'mode', 'is', 'all',
 ['shot', 'with', 'pixel'],
```

3.1.8 Word – Tokenized Subreddit Posts

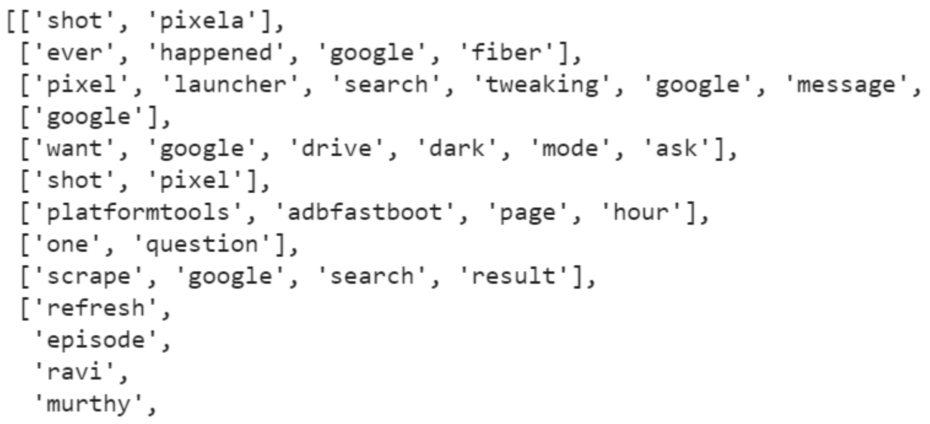
The next task is the removal of the Stop words. Stop words are often used words that are thought to contribute little to nothing significantly to the overall meaning of a phrase or text in Natural Language Processing (NLP). Words like "and", "the", "is", "in", "a", and others commonly occur in a particular language and are included in this category. A frequent preprocessing step in NLP is the elimination of stop words, which is removing certain words from the text prior to further analysis or modelling. The Dataset after removing the Stop words (fig._3.1.9)



```
[['shot', 'pixela'],
 ['ever', 'happened', 'google', 'fiber'],
 ['pixel', 'launcher', 'search', 'tweaking', 'google', 'messages',
 ['google'],
 ['want', 'google', 'drive', 'dark', 'mode', 'ask'],
 ['shot', 'pixel'],
 ['platformtools', 'adbfastboot', 'page', 'hours'],
 ['one', 'question'],
 ['scrape', 'google', 'search', 'results'],
```

3.1.9 Subreddit Posts Stop words removed

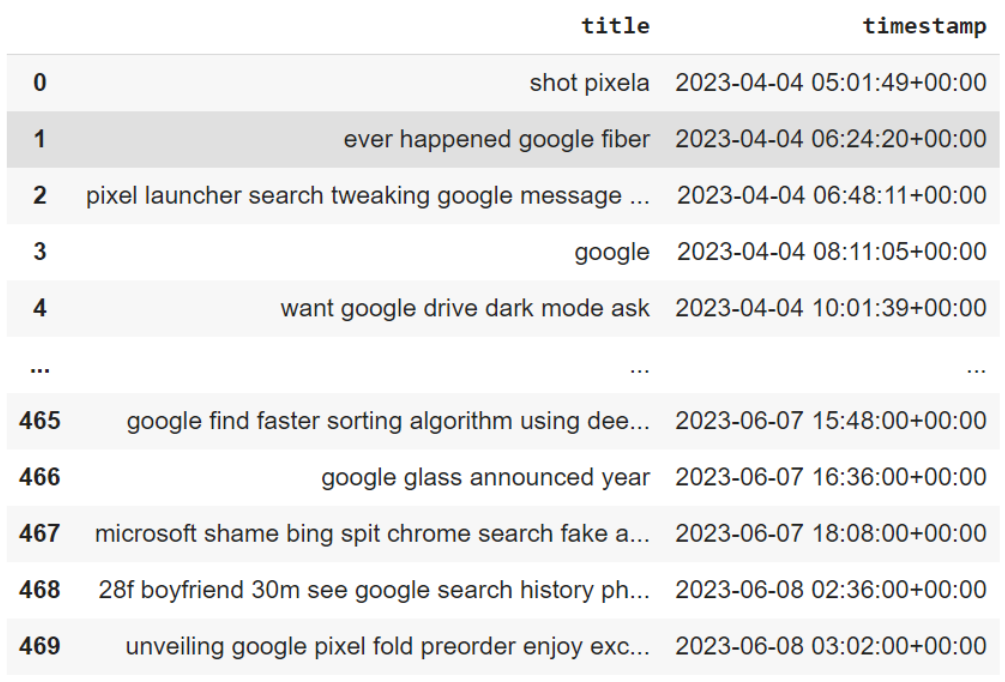
The next task is Lemmatization. Lemmatization in Natural Language Processing (NLP) is the act of breaking down words into their lemma, or root form. The lemma reflects a word's canonical form and conveys its essential meaning without taking into account tense, case, gender, plurality, or other inflections or variants. As an illustration, the lemma of the words "driving", "drive", and "ran" is "run". Lemmatization's main goal is to normalize words so that all possible spellings of a word map to the same representation. Reasons why Lemmatization is performed: “Text Standardization”, “Vocabulary Reduction”, “Improved Information Retrieval”. For the purpose of Lemmatization I used “WordNetLemmatizer” from the package – “nltk.stem.wordnet”. The output after Lemmatization is (figure)



```
[['shot', 'pixela'],
 ['ever', 'happened', 'google', 'fiber'],
 ['pixel', 'launcher', 'search', 'tweaking', 'google', 'message',
 ['google'],
 ['want', 'google', 'drive', 'dark', 'mode', 'ask'],
 ['shot', 'pixel'],
 ['platformtools', 'adbfastboot', 'page', 'hour'],
 ['one', 'question'],
 ['scrape', 'google', 'search', 'result'],
 ['refresh',
  'episode',
  'ravi',
  'murthy',
```

3.1.10 Lemmatized Subreddit Posts

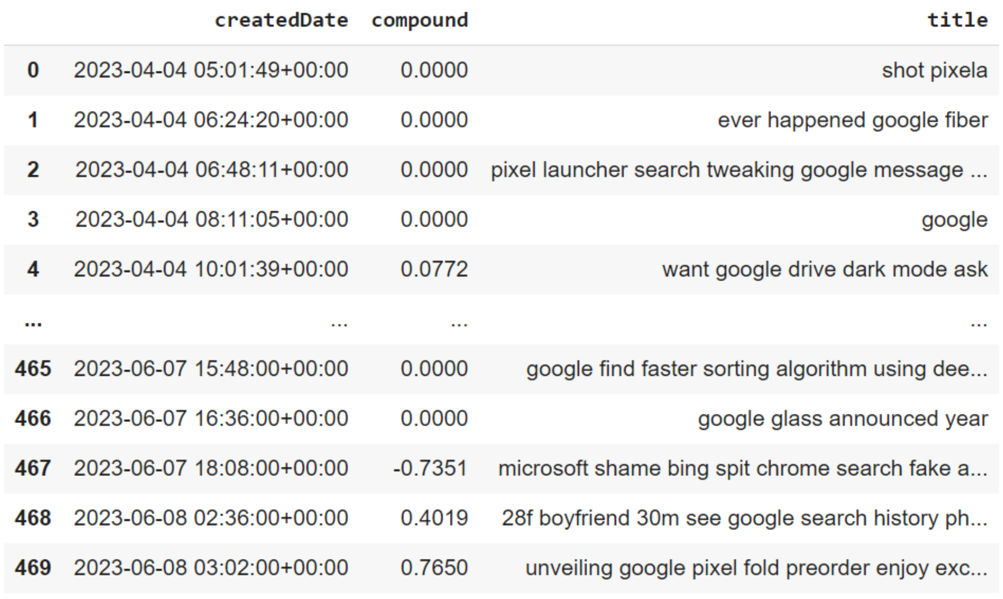
After performing all the Natural Language Processing (NLP) tasks, that is, Lowercasing, Tokenization, Stop words removal, Lemmatization, the final processed dataset looks like (fig._3.1.11)



	title	timestamp
0	shot pixela	2023-04-04 05:01:49+00:00
1	ever happened google fiber	2023-04-04 06:24:20+00:00
2	pixel launcher search tweaking google message ...	2023-04-04 06:48:11+00:00
3	google	2023-04-04 08:11:05+00:00
4	want google drive dark mode ask	2023-04-04 10:01:39+00:00
...	...	...
465	google find faster sorting algorithm using dee...	2023-06-07 15:48:00+00:00
466	google glass announced year	2023-06-07 16:36:00+00:00
467	microsoft shame bing spit chrome search fake a...	2023-06-07 18:08:00+00:00
468	28f boyfriend 30m see google search history ph...	2023-06-08 02:36:00+00:00
469	unveiling google pixel fold preorder enjoy exc...	2023-06-08 03:02:00+00:00

3.1.11 Cleaned Social Media Dataset

Now that the subreddit social media posts are processed, we can calculate the sentiment scores for each subreddit posts. For this, I use “SentimentIntensityAnalyzer” from the package – “nltk.sentiment.vader”. I calculate the polarity score for each post that ranges from [-1, 1], where -1 indicated negative post, 0 indicated neutral post, and +1 indicated positive post. Using a pre-defined vocabulary of words and phrases with corresponding sentiment scores, the SentimentIntensityAnalyzer in NLTK's VADER (Valence Aware Dictionary and sEntiment Reasoner) does sentiment analysis. The analyzer determines the sentiment polarity of a text by taking into account the strength and valence of each word, as well as the use and punctuation in the text's context. In addition to an overall compound emotion score, the resultant sentiment scores reveal the text's positive, negative, and neutral sentiments.

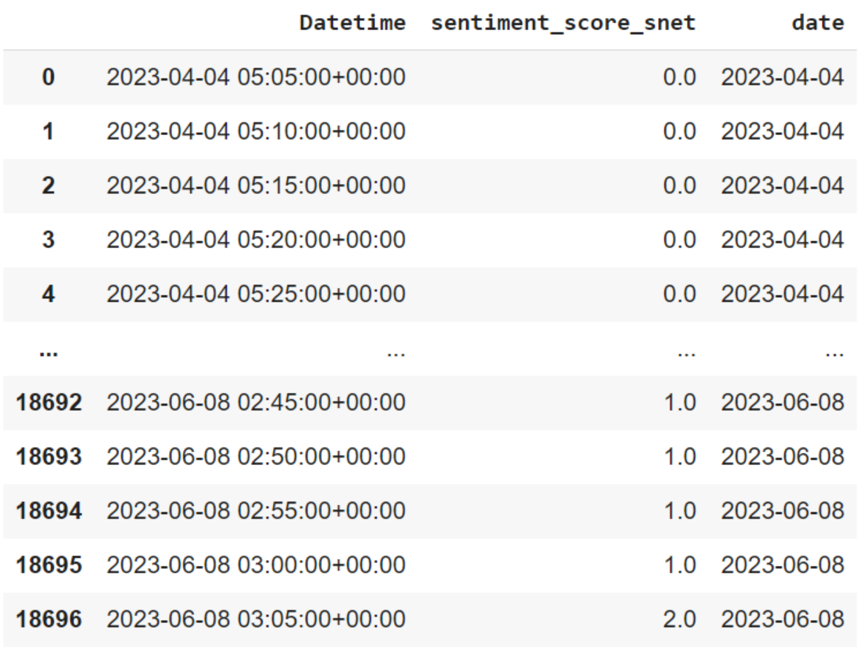


	createdDate	compound	title
0	2023-04-04 05:01:49+00:00	0.0000	shot pixela
1	2023-04-04 06:24:20+00:00	0.0000	ever happened google fiber
2	2023-04-04 06:48:11+00:00	0.0000	pixel launcher search tweaking google message ...
3	2023-04-04 08:11:05+00:00	0.0000	google
4	2023-04-04 10:01:39+00:00	0.0772	want google drive dark mode ask
...	...	...	...
465	2023-06-07 15:48:00+00:00	0.0000	google find faster sorting algorithm using dee...
466	2023-06-07 16:36:00+00:00	0.0000	google glass announced year
467	2023-06-07 18:08:00+00:00	-0.7351	microsoft shame bing spit chrome search fake a...
468	2023-06-08 02:36:00+00:00	0.4019	28f boyfriend 30m see google search history ph...
469	2023-06-08 03:02:00+00:00	0.7650	unveiling google pixel fold preorder enjoy exc...

3.1.12 Social Media Dataset with Sentiment Score

Similar to the news headlines dataset, the social media posts dataset needs to be also normalized, so I resampled the merged DataFrame with the time frequency of 5 minutes. Using the forward-fill approach, the missing values in the 'compound' column are filled with the most recent non-missing value in the same column. The resulting DataFrame contains 18,696 rows.

After calculating the sentiment scores of the social media posts and merging the DataFrames, the sentiment scores are normalized in the range [-2, 2], where, -2 represents extremely negative post, -1 represents negative post, 0 represents neutral post, +1 represents positive post, and +2 represents extremely positive post. After normalizing the dataset looks like (figure)



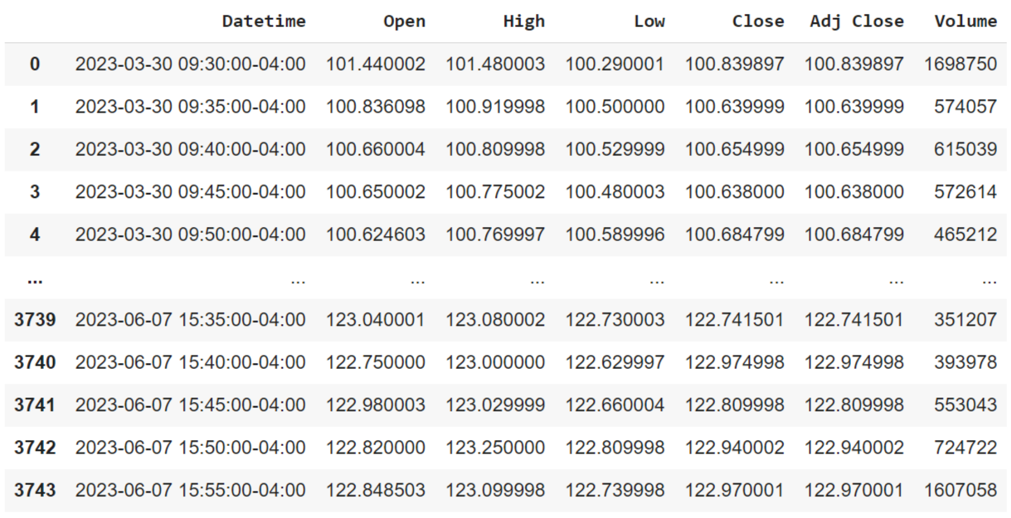
	Datetime	sentiment_score_snet	date
0	2023-04-04 05:05:00+00:00	0.0	2023-04-04
1	2023-04-04 05:10:00+00:00	0.0	2023-04-04
2	2023-04-04 05:15:00+00:00	0.0	2023-04-04
3	2023-04-04 05:20:00+00:00	0.0	2023-04-04
4	2023-04-04 05:25:00+00:00	0.0	2023-04-04
...	...	...	...
18692	2023-06-08 02:45:00+00:00	1.0	2023-06-08
18693	2023-06-08 02:50:00+00:00	1.0	2023-06-08
18694	2023-06-08 02:55:00+00:00	1.0	2023-06-08
18695	2023-06-08 03:00:00+00:00	1.0	2023-06-08
18696	2023-06-08 03:05:00+00:00	2.0	2023-06-08

3.1.13 Normalized Social Media Dataset

3.1.1.3 Stock Dataset

The data used in this project is obtained using the “yfinance” module in Python, one of the well-known Python modules for gathering web data is “yfinance”, I obtained news headlines of Google frames of the “yfinance” module to obtain and gather financial information about the firm (such as financial ratios, etc.), as well as the history of marketing data.

The timeframe is from “2023-03-30” to “2023-06-07” with a timestamp gap of 5 minutes. The dataset has a total of 3,743 rows. (fig._3.1.14)

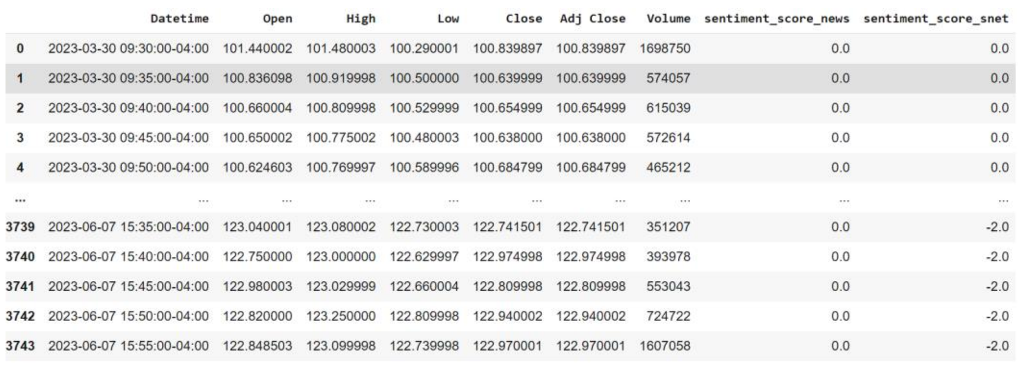


	Datetime	Open	High	Low	Close	Adj Close	Volume
0	2023-03-30 09:30:00-04:00	101.440002	101.480003	100.290001	100.839897	100.839897	1698750
1	2023-03-30 09:35:00-04:00	100.836098	100.919998	100.500000	100.639999	100.639999	574057
2	2023-03-30 09:40:00-04:00	100.660004	100.809998	100.529999	100.654999	100.654999	615039
3	2023-03-30 09:45:00-04:00	100.650002	100.775002	100.480003	100.638000	100.638000	572614
4	2023-03-30 09:50:00-04:00	100.624603	100.769997	100.589996	100.684799	100.684799	465212
...	...	...	...	...	...	...	...
3739	2023-06-07 15:35:00-04:00	123.040001	123.080002	122.730003	122.741501	122.741501	351207
3740	2023-06-07 15:40:00-04:00	122.750000	123.000000	122.629997	122.974998	122.974998	393978
3741	2023-06-07 15:45:00-04:00	122.980003	123.029999	122.660004	122.809998	122.809998	553043
3742	2023-06-07 15:50:00-04:00	122.820000	123.250000	122.809998	122.940002	122.940002	724722
3743	2023-06-07 15:55:00-04:00	122.848503	123.099998	122.739998	122.970001	122.970001	1607058

3.1.14 Google Stock Dataset

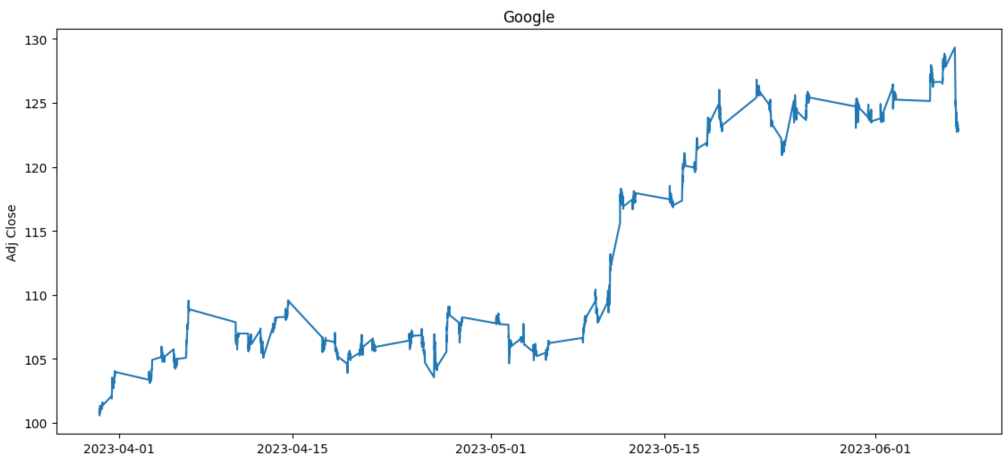
3.1.2 Preparing Data

After calculating the sentiment scores of the news headlines and the social media posts, and resampling both the datasets – "News Headlines", and "Subreddit Posts" with timestamp of frequency 5 minutes. I merged the "News Headlines Sentiment" dataset and the "Subreddit Posts Sentiment" dataset with the "Stock Price" dataset and filled 0 for all the missing values. The final dataset looks like (fig._3.1.15)

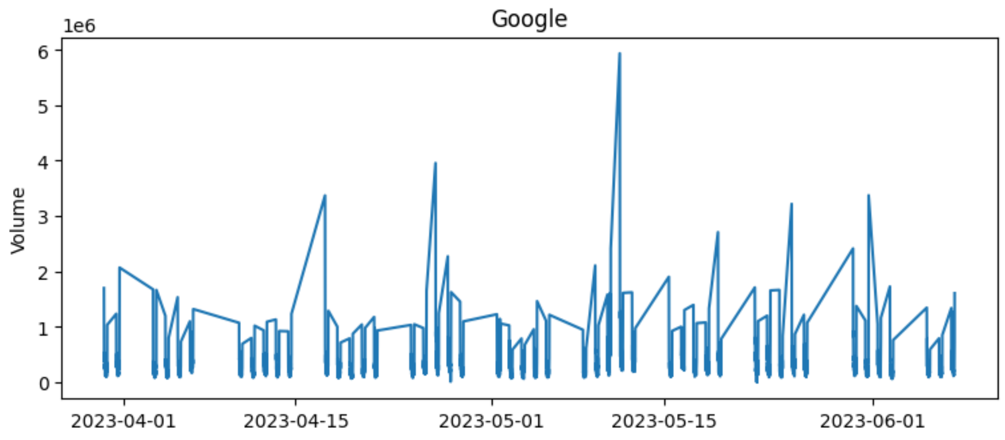


	Datetime	Open	High	Low	Close	Adj Close	Volume	sentiment_score_news	sentiment_score_snet
0	2023-03-30 09:30:00-04:00	101.440002	101.480003	100.290001	100.839897	100.839897	1698750	0.0	0.0
1	2023-03-30 09:35:00-04:00	100.836098	100.919998	100.500000	100.639999	100.639999	574057	0.0	0.0
2	2023-03-30 09:40:00-04:00	100.660004	100.809998	100.529999	100.654999	100.654999	615039	0.0	0.0
3	2023-03-30 09:45:00-04:00	100.650002	100.775002	100.480003	100.638000	100.638000	572614	0.0	0.0
4	2023-03-30 09:50:00-04:00	100.624603	100.769997	100.589996	100.684799	100.684799	465212	0.0	0.0
...	...	...	...	...	...	...	...	...	...
3739	2023-06-07 15:35:00-04:00	123.040001	123.080002	122.730003	122.741501	122.741501	351207	0.0	-2.0
3740	2023-06-07 15:40:00-04:00	122.750000	123.000000	122.629997	122.974998	122.974998	393978	0.0	-2.0
3741	2023-06-07 15:45:00-04:00	122.980003	123.029999	122.660004	122.809998	122.809998	553043	0.0	-2.0
3742	2023-06-07 15:50:00-04:00	122.820000	123.250000	122.809998	122.940002	122.940002	724722	0.0	-2.0
3743	2023-06-07 15:55:00-04:00	122.848503	123.099998	122.739998	122.970001	122.970001	1607058	0.0	-2.0

3.1.15 Final Merged Dataset (including sentiment score)

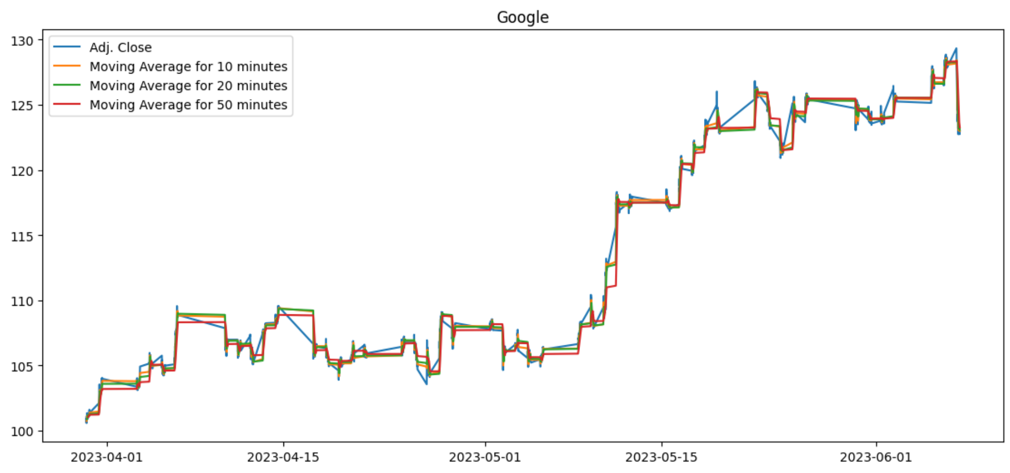


3.1.16 Historical view of Closing Price



3.1.17 Historical view of Volume Traded

Moving averages are frequently used in time series analysis to eliminate outliers and spot underlying trends or patterns in the data. It generates a fresh set of averages by determining the average value of a string of succeeding data points falling inside a certain window or period. The moving average aids in noise reduction and draws attention to the time series' overall pattern of behavior. The original time series' fluctuations are smoothed out by the moving average, making it simpler to spot long-term trends and patterns. A group of data points can be substituted with a single average value to decrease noise or short-term volatility and highlight the time series' general behavior. (fig._3.1.18)

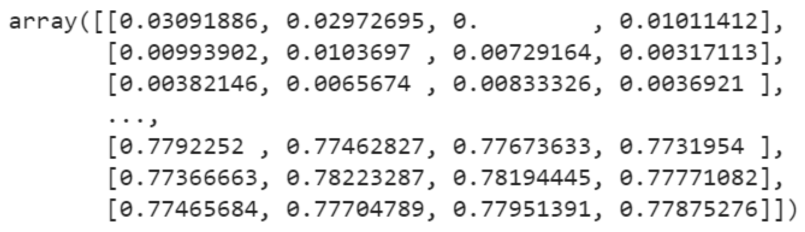


3.1.18 Moving Averages of Stock

3.1.3.1 Training the model (without Sentiment Dataset)

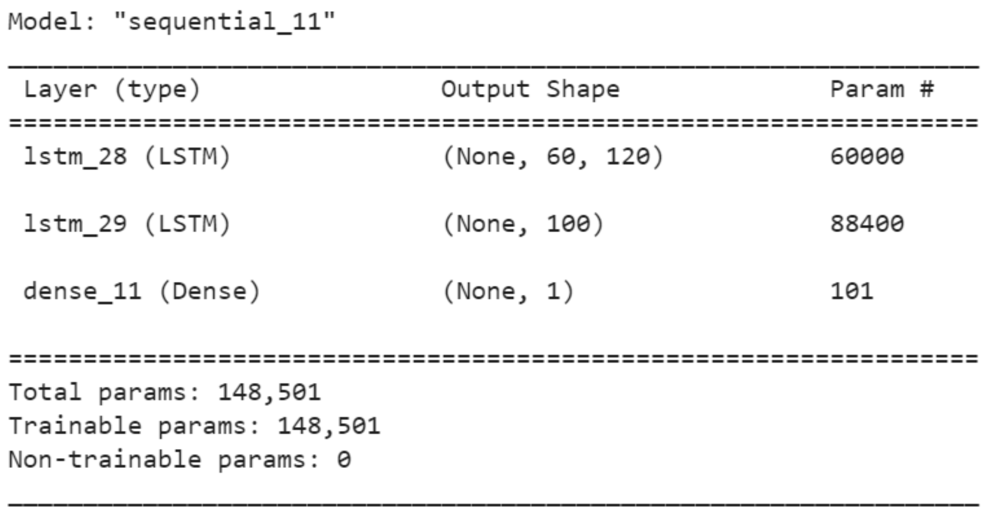
Firstly, we train the Long-Short Term Memory (LSTM) model on the dataset that does not contain the sentiment score of the news headlines and social media posts related to the company.

For the LSTM model, I use MinMaxScaler to standardize the sentiment score dataset, with values ranging between 0 and 1 (fig._3.1.19). After that, we use the stacked LSTM model to train the dataset.



```
array([[0.03091886, 0.02972695, 0.        , 0.01011412],
       [0.00993902, 0.0103697 , 0.00729164, 0.00317113],
       [0.00382146, 0.0065674 , 0.00833326, 0.0036921 ],
       ...,
       [0.7792252 , 0.77462827, 0.77673633, 0.7731954 ],
       [0.77366663, 0.78223287, 0.78194445, 0.77771082],
       [0.77465684, 0.77704789, 0.77951391, 0.77875276]])
```

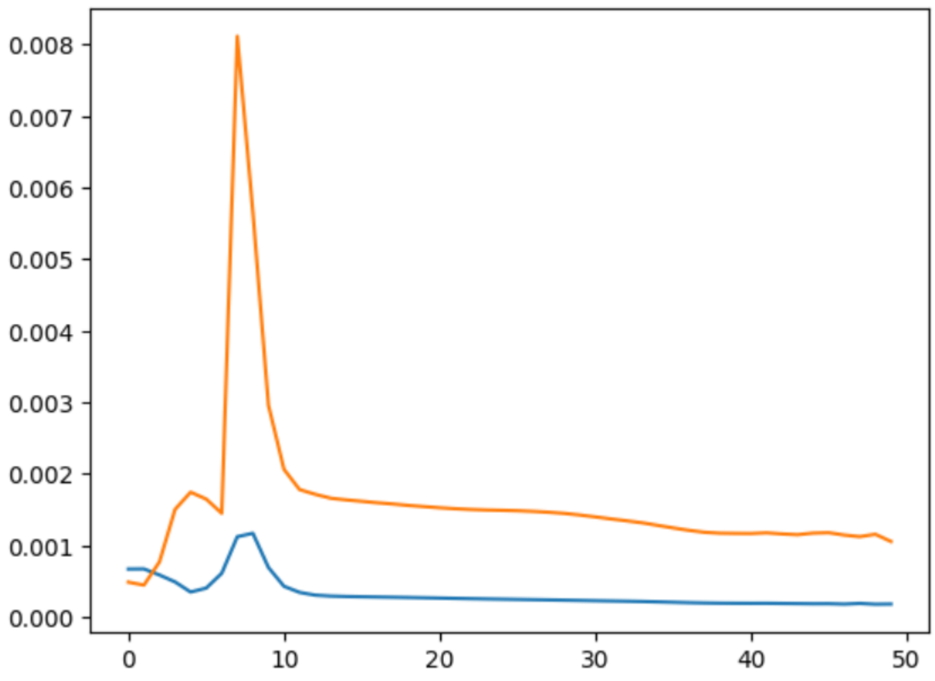
3.1.19 Normalized Stock Data



```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_28 (LSTM)              (None, 60, 120)           60000

 lstm_29 (LSTM)              (None, 100)               88400

 dense_11 (Dense)            (None, 1)                 101

=================================================================
Total params: 148,501
Trainable params: 148,501
Non-trainable params: 0
_________________________________________________________________
```

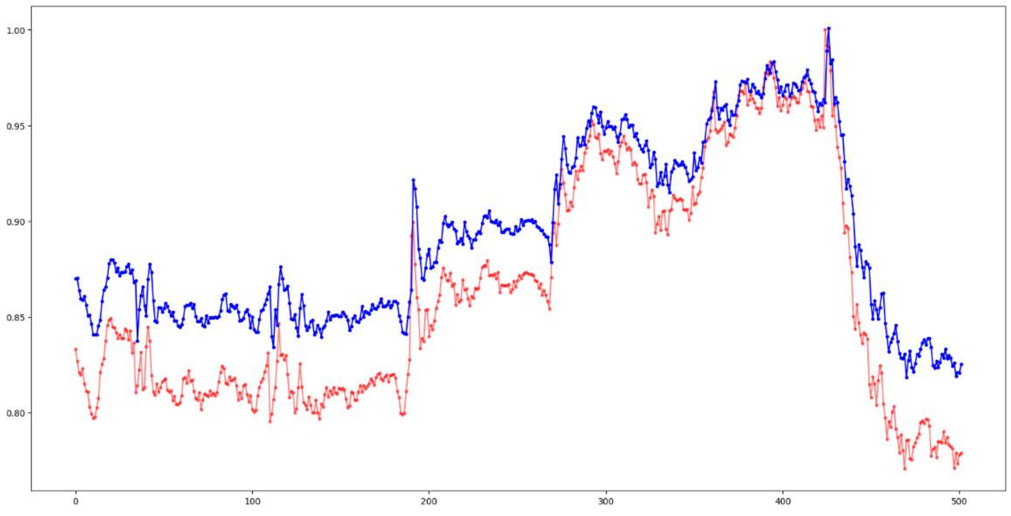
3.1.20 LSTM model summary

For the LSTM model architecture, I have used a Sequential LSTM model with a linear stack of layers, the first layer has 120 units and returns sequences and the second layer has 100 units, a Dense layer with a single unit is also added. The loss function I have used for the model is – “Mean-Squared Error” and the optimizer is set to “adam” optimizer. I have also used “Early Stopping” so that the training process stops if the loss does not improve beyond a threshold.



3.1.21 Train Loss and Validation Loss

We plot the predictive model graph, with the orange line indicating training dataset prediction, and the blue line indicating the dataset on which the prediction is done. (fig._3.1.22)



3.1.22 Graph of model without sentiment score

To make sure that the model does not get overfit, I used Regularization. In machine learning and neural networks, regularization is used to prevent overfitting, which happens when a model learns to fit the training data too well but is unable to generalize to new, unforeseen data. Overfitting can result in subpar performance and a diminished capacity for making precise forecasts.

The three LSTM layers in this model architecture are each followed by a dropout layer and a dense output layer. After each LSTM layer, dropout regularization is used to reduce overfitting by arbitrarily removing some of the connections between neurons. 20% of the connections will have their values randomly set to 0 during training, according to the dropout rate of 0.2.

Additionally, I also used Cross-validation to confirm and validate that the LSTM model does not get overfitted. Resampling is a machine learning and model evaluation approach called cross-validation. The available data is divided into a number of folds, or subsets, with each fold serving as both a training set and a validation set. Cross-validation is a technique used to quantify a model's performance on a different dataset and assess how effectively it generalizes to new data.

Three "gate" structures make up the unique network structure known as LSTM. A forgetting gate, an output gate, and an input gate are the three gates that make up an LSTM unit. At the time it enters the LSTM network, information may be selected by rules. The forgetting gate will delete information that does not follow the algorithm, leaving only the information that does.

Forget gate - A forget gate eliminates data from the cell state. The data that is no longer required for the LSTM to understand anything or that is of lesser significance is removed by multiplying a filter. This is required to boost the functionality of the LSTM network. These gates' two inputs are h_t-1 and x_t. The hidden state from the previous cell, or its output, is h_t-1, and its input at that particular time step is x_t.

Input gate - The input gate's sigmoid function determines which values should be included in the cell state. This is basically quite similar to the forget gate and serves as a filter for all the data from hi-1 and x t. The construction of a vector containing each value that might be added to the cell state (as determined by h t-1 and x t). This is accomplished using the tanh function, which generates values between -1 and +1. after multiplying the value of the regulatory filter (the sigmoid gate) by the generated vector, adding this useful information to the cell state (the tanh function).

Output gate: A vector is produced after the values are scaled applying the tanh function to fall between -1 and +1. Using the values of h t-1 and x t, one may create a filter that can regulate the values that must be created from the vector constructed above. Once more, this filter makes use of the sigmoid function. The vector created in Step 1 is multiplied by the value of this regulatory filter before being sent to the output and concealed state of the succeeding cell.

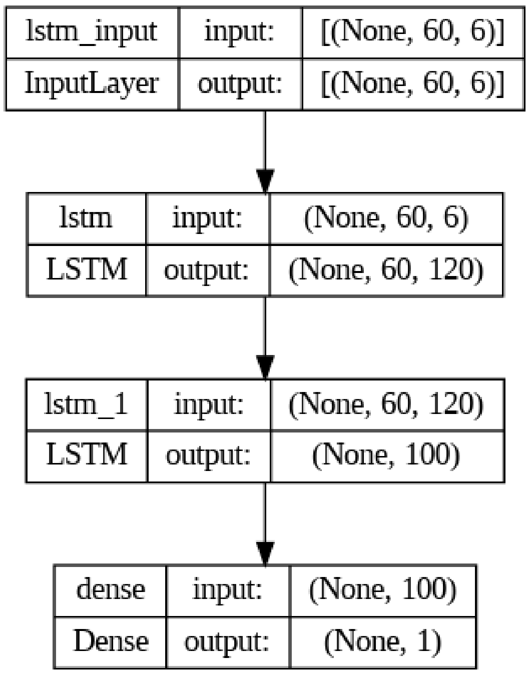
3.1.3.2 Training the model (with Sentiment Dataset)

To check how the sentiment score of the news headlines and the sentiment scores of the social media posts behaves with the Stock Dataset of a company. I trained the Long Short-Term Memory (LSTM) model with the dataset containing the sentiment scores related to the company.

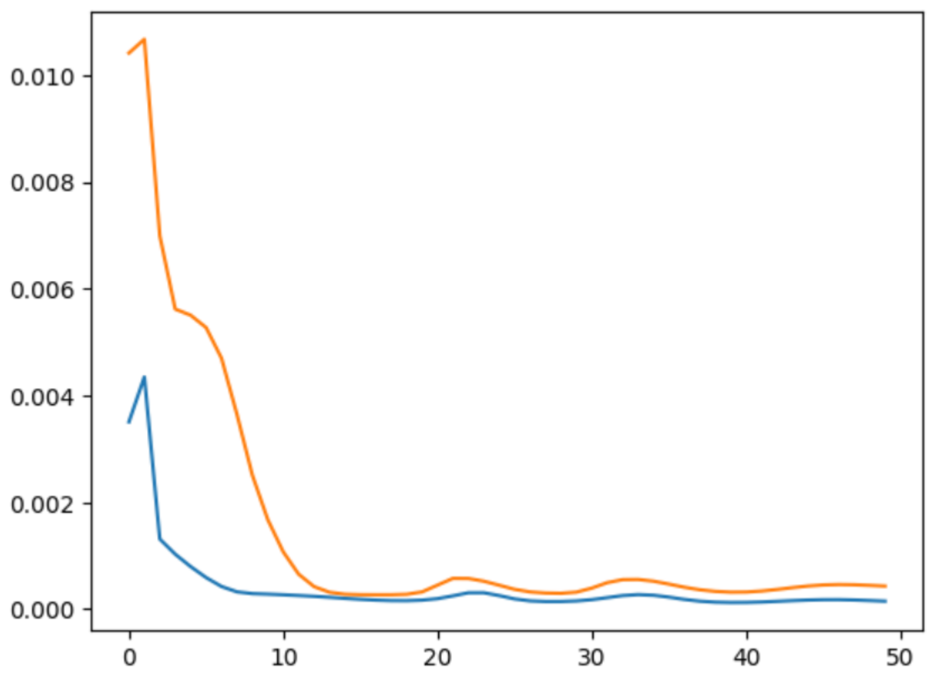
To standardize the stock dataset, I used MinMaxScaler on the dataset. A normalization method called MinMaxScaler is often employed in machine learning and data preparation. Features are transformed by scaling them to a predetermined range, usually between 0 and 1. The MinMaxScaler' goal is to scale the feature values to a common scale, which is advantageous for some algorithms and models.

I divided the data array or DataFrame into three parts: the training set, which covers 70% of the data's length or up to its beginning; the validation set, which covers 70% to 90% of the length; and the test set, which covers 90% of the data's length or up to its end.

The LSTM model is same as that of the previous model, that is, the architecture is a Sequential LSTM model, the first layer has 120 units and returns sequences and the second layer has 100 units, a Dense layer with a single unit is also added. The loss function I have used for the model is – “Mean-Squared Error” and the optimizer is set to “adam” optimizer. I have also used “Early Stopping” so that the training process stops if the loss does not improve beyond a threshold.

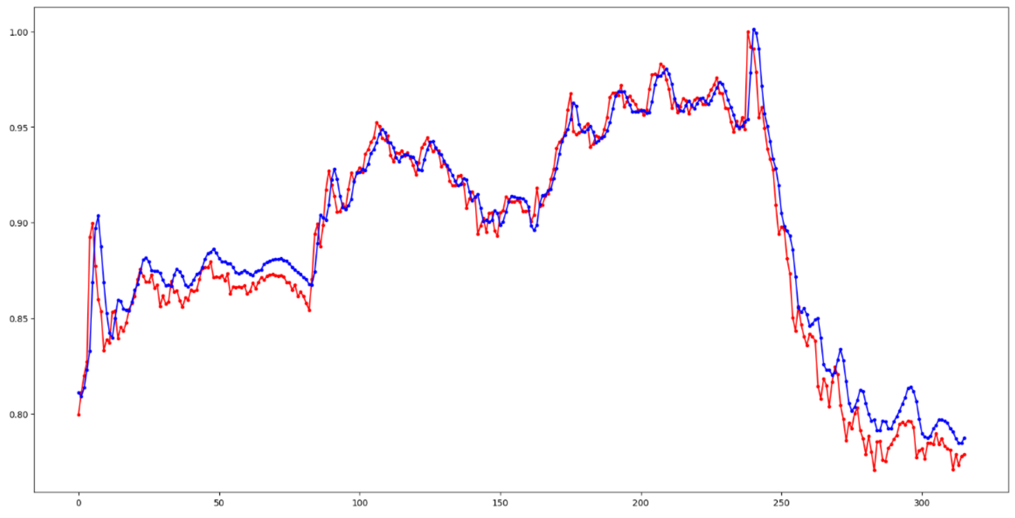


3.1.23 LSTM model shape



3.1.24 Train Loss and Validation Loss (sentiment)

We plot the predictive model graph, with the orange line indicating training dataset prediction, and the blue line indicating the dataset on which the prediction is done.



3.1.25 Graph of model with sentiment score

To make sure that the model does not get overfit, I used Regularization, as well as Cross-Validation.

Chapter - 04

EXPERIMENTS AND RESULT ANALYSIS

4.1 System Configuration

4.1.1 Hardware requirements

- RAM: 16 GB
- Storage: 500 GB
- CPU: 2.3 GHz or faster
- Architecture: 64-bit processor

4.1.2 Software requirements

- The model is trained and the data is pre-processed using Python 3.5 in a Jupyter notebook.

- OS: Windows 10 or higher.

In order to compare the projected and real stock prices and determine the MSE values, we utilize the forecasted stock price. The projected values are then used to determine the price variation of the stock on the prediction day; if the anticipated stock price rises, the output is 1, and if the predicted stock price declines, the output is 0.

The MSE, Accuracy, Precision, Recall, F1 Score are defined as -

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

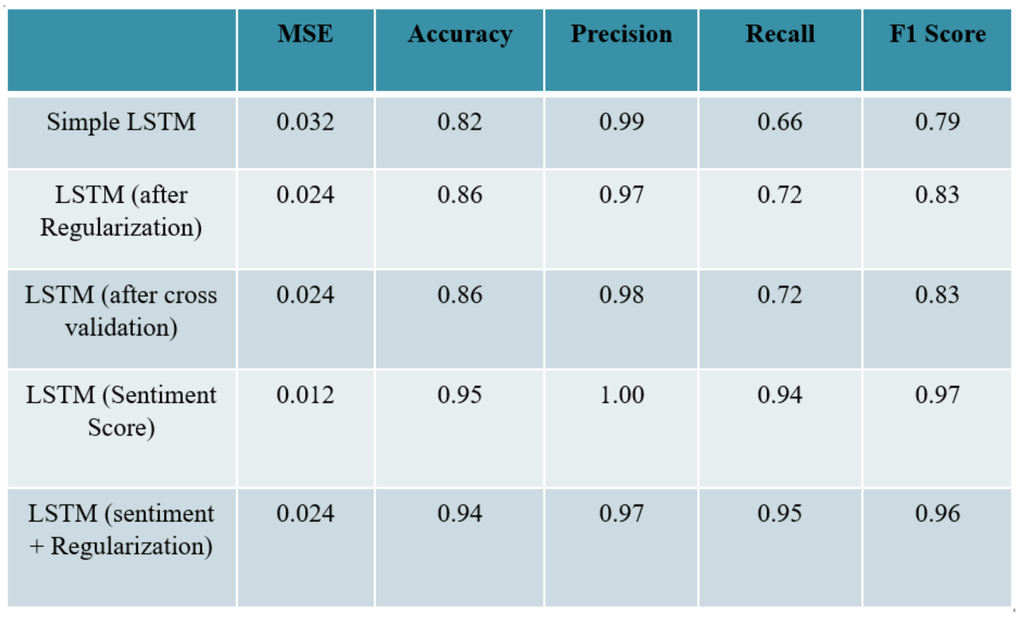
Accuracy = Number of Correct Predictions / Total Predictions

Precision = True Positive / (True Positive + False Positive)

Recall = True Positive / (True Positive + False Positive)

F1 Score = 2 x Precision x Recall / (Precision + Recall)

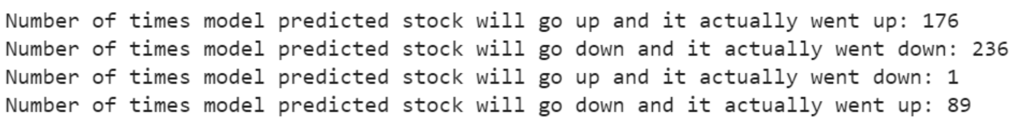
For the LSTM model, we got the following results –



	MSE	Accuracy	Precision	Recall	F1 Score
Simple LSTM	0.032	0.82	0.99	0.66	0.79
LSTM (after Regularization)	0.024	0.86	0.97	0.72	0.83
LSTM (after cross validation)	0.024	0.86	0.98	0.72	0.83
LSTM (Sentiment Score)	0.012	0.95	1.00	0.94	0.97
LSTM (sentiment + Regularization)	0.024	0.94	0.97	0.95	0.96

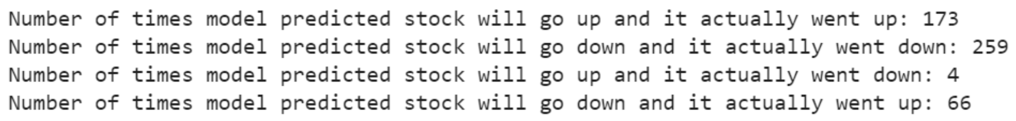
4.1 Result Table

We got the following predictions for the LSTM model –



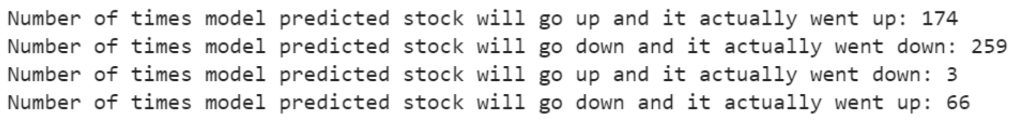
```
Number of times model predicted stock will go up and it actually went up: 176
Number of times model predicted stock will go down and it actually went down: 236
Number of times model predicted stock will go up and it actually went down: 1
Number of times model predicted stock will go down and it actually went up: 89
```

4.2 Simple LSTM predictions



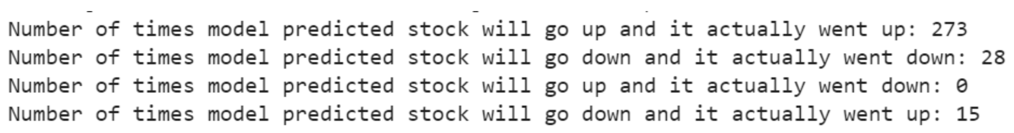
```
Number of times model predicted stock will go up and it actually went up: 173
Number of times model predicted stock will go down and it actually went down: 259
Number of times model predicted stock will go up and it actually went down: 4
Number of times model predicted stock will go down and it actually went up: 66
```

4.3 LSTM (after Regularization) predictions



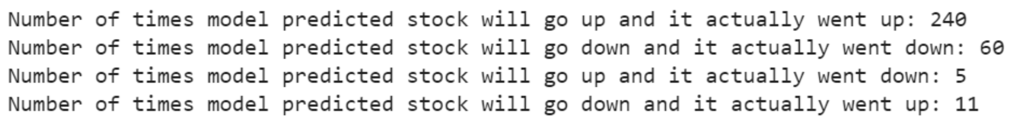
```
Number of times model predicted stock will go up and it actually went up: 174
Number of times model predicted stock will go down and it actually went down: 259
Number of times model predicted stock will go up and it actually went down: 3
Number of times model predicted stock will go down and it actually went up: 66
```

4.4 LSTM (after Cross-Validation) predictions



```
Number of times model predicted stock will go up and it actually went up: 273
Number of times model predicted stock will go down and it actually went down: 28
Number of times model predicted stock will go up and it actually went down: 0
Number of times model predicted stock will go down and it actually went up: 15
```

4.5 LSTM (with sentiment score) predictions



```
Number of times model predicted stock will go up and it actually went up: 240
Number of times model predicted stock will go down and it actually went down: 60
Number of times model predicted stock will go up and it actually went down: 5
Number of times model predicted stock will go down and it actually went up: 11
```

4.6 LSTM (with sentiment score and Regularization) predictions

Chapter - 05

CONCLUSION

5.1 Conclusion

The LSTM (Long-Short Term Memory Model) with Sentiment Analysis of News Headlines and Social Media Posts outperformed the simple Long-Short Term (LSTM) model.

5.2 Future Scope

- We want to expand on this project by adding unstructured textual data to the model, such as Press releases, earnings reports of the underlying firms, news about policies, and research studies from market experts.

- We intend to include other base models as well as LSTM and other neural networks.

- Adjusting the parameters will further increase the model's accuracy.

5.3 Applications

- Time Series Forecasting: Based on previous data, LSTM models may be used to project stock values. The model may be used to forecast future stock prices after being trained using historical stock prices and other market data.

- Pattern Recognition: Historical stock price patterns may be examined using LSTM models. Then, future stock price projections can be based on these patterns.

- Portfolio Optimization: A portfolio of investments may be optimized using LSTM models based on past stock prices and other market variables. The ideal combination of stocks, bonds, and other assets may be determined using the model in order to maximize profits while lowering risk.

- Risk Management: To identify possible dangers in the stock market and create solutions to reduce such risks, LSTM models can be employed. The model, for instance, may be used to pinpoint equities that are susceptible to being impacted by cyclical or other occurrences, as well as to create risk-hedging plans.

REFERENCES

[1] A. Mansouri, A. Nazari, and M. Ramazani, “A Comparison of Artificial Neural Network Model and Logistics Regression in Prediction of Companiess Bankruptcy (A Case Study of Tehran Stock Exchange),” *SSRN Electronic Journal* , 2016, doi: https://doi.org/10.2139/ssrn.2787308.

[2] Y. Kara, M. Acar Boyacioglu, and Ö. K. Baykan, “Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange,” *Expert Systems with Applications*, vol. 38, no. 5, pp. 5311–5319, May 2011, doi: https://doi.org/10.1016/j.eswa.2010.10.027.

[3] S. Hochreiter, “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 06, no. 02, pp. 107–116, Apr. 1998, doi: https://doi.org/10.1142/s0218488598000094.

[4] H. N. Bhandari, B. Rimal, N. R. Pokhrel, R. Rimal, K. R. Dahal, and R. K. C. Khatri, “Predicting stock market index using LSTM,” *Machine Learning with Applications*, p. 100320, May 2022, doi: https://doi.org/10.1016/j.mlwa.2022.100320.

[5] Y. Li and Y. Pan, “A novel ensemble deep learning model for stock prediction based on stock prices and news,” *International Journal of Data Science and Analytics*, Sep. 2021, doi: https://doi.org/10.1007/s41060-021-00279-9.

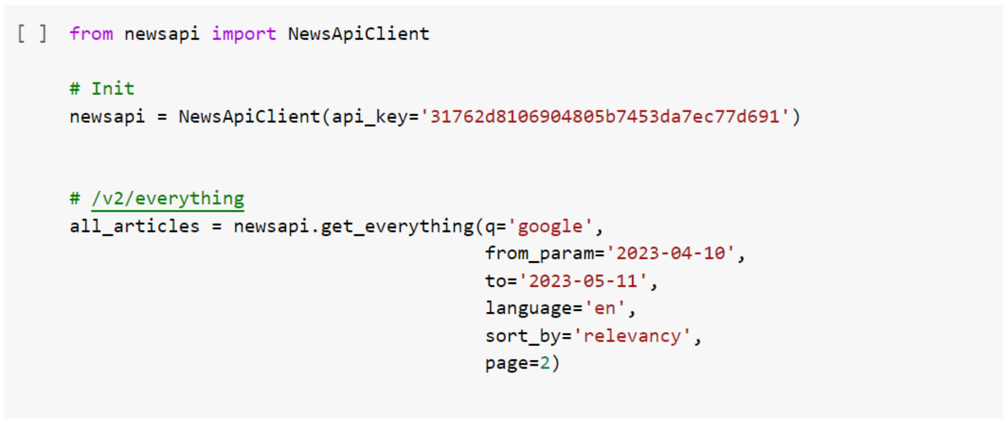
[6] E. Hoseinzade and S. Haratizadeh, “CNNpred: CNN-based stock market prediction using a diverse set of variables,” *Expert Systems with Applications*, vol. 129, pp. 273–285, Sep. 2019, doi: https://doi.org/10.1016/j.eswa.2019.03.029.

[7] Rouf, N. et al. (2021) ‘Stock market prediction using Machine Learning Techniques: A decade survey on methodologies, recent developments, and Future Directions’, Electronics, 10(21), p. 2717. doi:10.3390/electronics10212717.

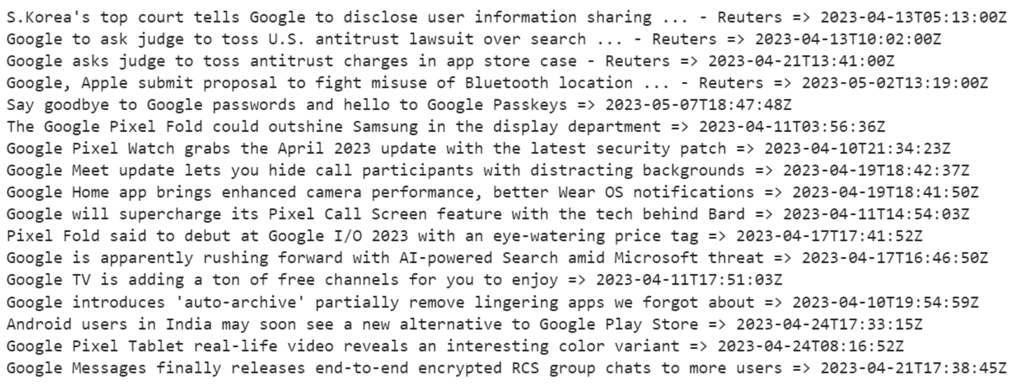
APPENDICES

- News Headlines Dataset –

Extracting News Headlines

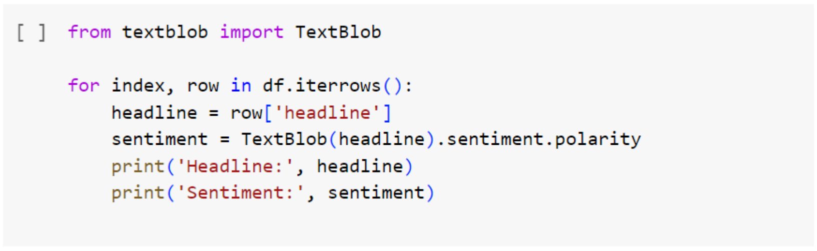


```
[ ]  from newsapi import NewsApiClient

     # Init
     newsapi = NewsApiClient(api_key='31762d8106904805b7453da7ec77d691')


     # /v2/everything
     all_articles = newsapi.get_everything(q='google',
                                           from_param='2023-04-10',
                                           to='2023-05-11',
                                           language='en',
                                           sort_by='relevancy',
                                           page=2)
```



```
S.Korea's top court tells Google to disclose user information sharing ... - Reuters => 2023-04-13T05:13:00Z
Google to ask judge to toss U.S. antitrust lawsuit over search ... - Reuters => 2023-04-13T10:02:00Z
Google asks judge to toss antitrust charges in app store case - Reuters => 2023-04-21T13:41:00Z
Google, Apple submit proposal to fight misuse of Bluetooth location ... - Reuters => 2023-05-02T13:19:00Z
Say goodbye to Google passwords and hello to Google Passkeys => 2023-05-07T18:47:48Z
The Google Pixel Fold could outshine Samsung in the display department => 2023-04-11T03:56:36Z
Google Pixel Watch grabs the April 2023 update with the latest security patch => 2023-04-10T21:34:23Z
Google Meet update lets you hide call participants with distracting backgrounds => 2023-04-19T18:42:37Z
Google Home app brings enhanced camera performance, better Wear OS notifications => 2023-04-19T18:41:50Z
Google will supercharge its Pixel Call Screen feature with the tech behind Bard => 2023-04-11T14:54:03Z
Pixel Fold said to debut at Google I/O 2023 with an eye-watering price tag => 2023-04-17T17:41:52Z
Google is apparently rushing forward with AI-powered Search amid Microsoft threat => 2023-04-17T16:46:50Z
Google TV is adding a ton of free channels for you to enjoy => 2023-04-11T17:51:03Z
Google introduces 'auto-archive' partially remove lingering apps we forgot about => 2023-04-10T19:54:59Z
Android users in India may soon see a new alternative to Google Play Store => 2023-04-24T17:33:15Z
Google Pixel Tablet real-life video reveals an interesting color variant => 2023-04-24T08:16:52Z
Google Messages finally releases end-to-end encrypted RCS group chats to more users => 2023-04-21T17:38:45Z
```

Calculating Sentiment Scores



```
[ ]  from textblob import TextBlob

     for index, row in df.iterrows():
         headline = row['headline']
         sentiment = TextBlob(headline).sentiment.polarity
         print('Headline:', headline)
         print('Sentiment:', sentiment)
```

```
Headline: Google releases Android 14 Beta 1 for Pixel phones
Sentiment: 0.0
Headline: Microsoft, Amazon, and Google to Gain From Cloud Growth, Says Analyst
Sentiment: 0.0
Headline: Google Disappoints, But They Are Still A Good Company
Sentiment: 0.7
Headline: Phenix Joins Google Cloud Marketplace
Sentiment: 0.0
Headline: Opera brings its free VPN to iOS to rival Apple and Google’s paid alternatives
Sentiment: 0.4
Headline: Snap Hires Google Ad Executive To Help Improve Ad Performance
Sentiment: 0.0
```

Merged News Headlines with Sentiment

```
[ ]  print(merged_df)
```

```
                        Datetime  sentiment_score_news   dateValue
0      2023-04-10 12:50:00+00:00                   1.0  2023-04-10
1      2023-04-10 12:55:00+00:00                   1.0  2023-04-10
2      2023-04-10 13:00:00+00:00                   1.0  2023-04-10
3      2023-04-10 13:05:00+00:00                   1.0  2023-04-10
4      2023-04-10 13:10:00+00:00                   1.0  2023-04-10
...                          ...                   ...         ...
14818  2023-06-01 19:40:00+00:00                   0.0  2023-06-01
14819  2023-06-01 19:45:00+00:00                   0.0  2023-06-01
14820  2023-06-01 19:50:00+00:00                   0.0  2023-06-01
14821  2023-06-01 19:55:00+00:00                   0.0  2023-06-01
14822  2023-06-01 20:00:00+00:00                   1.0  2023-06-01
```

- Subreddit Dataset -

Extract using Reddit API

```
[ ]  import praw

     user_agent = "Scraper 1.0 by /u/imPrakharS"
     reddit = praw.Reddit(
         client_id = "oWSqkTMRuriViwdVz8MVIg",
         client_secret = "QDQBqqpT-w8wvBZnOfGhAUb-R1Pw_g",
         user_agent = user_agent
     )
```

Loading Subreddit CSV File

```
[ ]  dataSB = pd.read_csv('subreddit.csv')

     dataSB.head()
```

	title	score	commentCount	upvoteRatio	createdDate
0	If Google made a sneaker	1209	99	0.94	2023-04-30T06:17:58.000Z
1	41 million views and local guide for 10 years ...	997	223	0.94	2023-04-16T09:25:44.000Z
2	Pixel 7a leaked	848	94	0.96	2023-04-28T17:32:13.000Z
3	What just happened to google? Only seeing spon...	791	259	0.94	2023-04-19T01:14:54.000Z
4	Shot on Pixel4a.	699	27	0.92	2023-04-04T05:01:49.000Z

Step 1 - convert to lower text

```
[ ]  clean_text_1 = []

     def to_lower_case(data):
       for words in merged_df.title:
         clean_text_1.append(str.lower(words))
```

```
[ ]  to_lower_case(merged_df.title)
```

Step 2 - Tokenize

```
[ ]  clean_text_2 = []

     from nltk.tokenize import sent_tokenize, word_tokenize

     import nltk
     nltk.download('punkt')
```

```
     [nltk_data] Downloading package punkt to /root/nltk_data...
     [nltk_data]   Package punkt is already up-to-date!
     True
```

```
[ ]  sent_tok = []

     for sent in clean_text_1:
       sent = sent_tokenize(sent)
       sent_tok.append(sent)
```

Step 3 - Word Tokenize

```
[ ]  clean_text_2 = [word_tokenize(i) for i in clean_text_1]
```

```
[ ]  clean_text_2
```

```
     [['shot', 'on', 'pixela'],
      ['what', 'ever', 'happened', 'to', 'google', 'fiber'],
      ['pixel',
       'launcher',
       'search',
```

Step 4 - Stopword removal

```
[ ]  import nltk
     nltk.download('stopwords')
```

```
     [nltk_data] Downloading package stopwords to /root/nltk_data...
     [nltk_data]   Package stopwords is already up-to-date!
     True
```

```
[ ]  from nltk.corpus import stopwords
```

```
[ ]  clean_text_4 = []

     for words in clean_text_3 :
       w = []
       for word in words:
         if not word in stopwords.words('english'):
           w.append(word)
       clean_text_4.append(w)
```

Step 5 - Stemming

```
[ ]  from nltk.stem.porter import PorterStemmer
```

```
[ ]  port = PorterStemmer()
```

```
[ ]  # "reading, washing, wash, Driving" => "read, wash, wash, drive"

     clean_text_5 = []

     for words in clean_text_4:
       w = []
       for word in words:
         w.append(port.stem(word))
       clean_text_5.append(w)
```

Step 6 - lemmetization

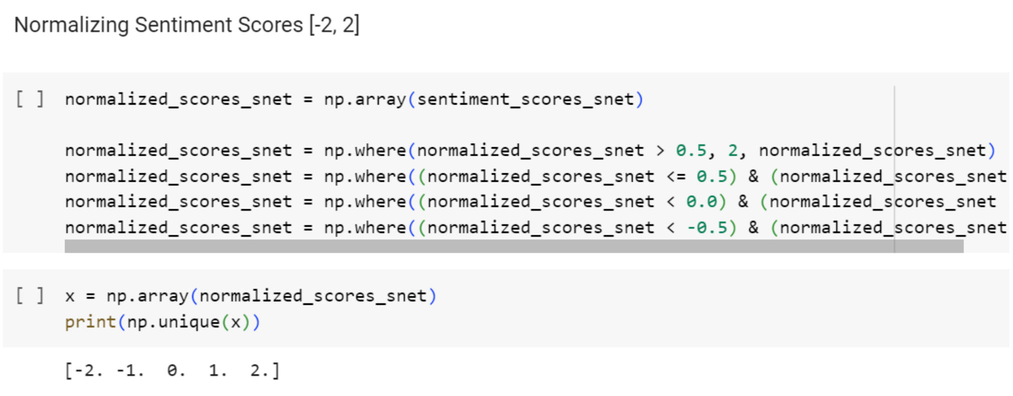
```
[ ]  from nltk.stem.wordnet import WordNetLemmatizer
```

```
[ ]  wnet = WordNetLemmatizer()
```

```
[ ]  import nltk
     nltk.download('wordnet')
```

```
     [nltk_data] Downloading package wordnet to /root/nltk_data...
     [nltk_data]   Package wordnet is already up-to-date!
     True
```

```
[ ]  lem = []
     clean_text_5 = []
     for words in clean_text_4:
       w = []
       for word in words:
         w.append(wnet.lemmatize(word))
       lem.append(w)
       clean_text_5.append(w)
```

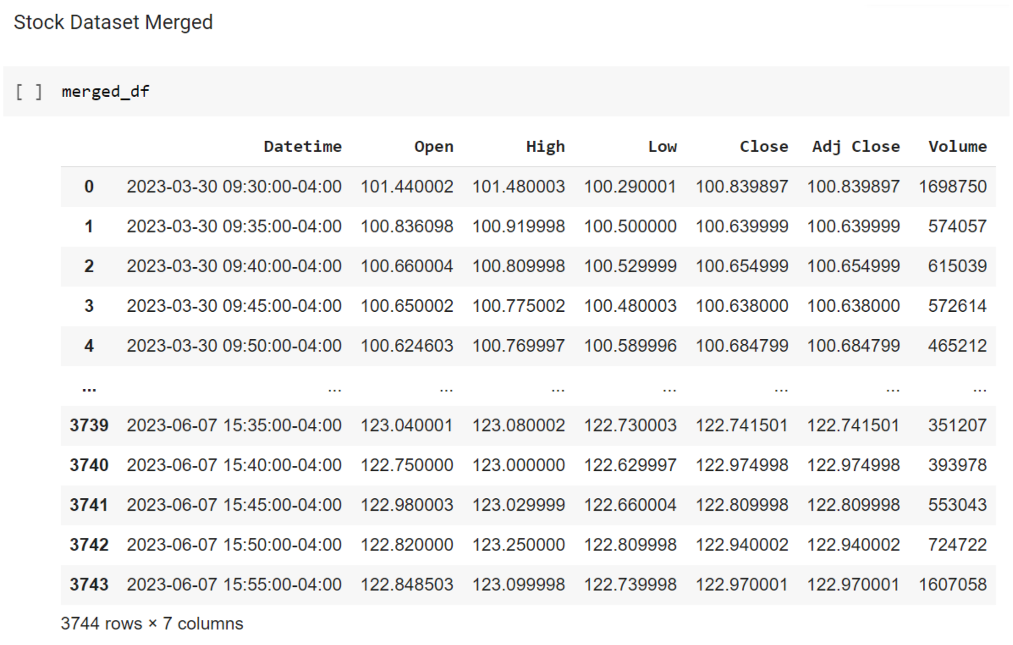


Normalizing Sentiment Scores [-2, 2]

```
[ ]  normalized_scores_snet = np.array(sentiment_scores_snet)

     normalized_scores_snet = np.where(normalized_scores_snet > 0.5, 2, normalized_scores_snet)
     normalized_scores_snet = np.where((normalized_scores_snet <= 0.5) & (normalized_scores_snet
     normalized_scores_snet = np.where((normalized_scores_snet < 0.0) & (normalized_scores_snet
     normalized_scores_snet = np.where((normalized_scores_snet < -0.5) & (normalized_scores_snet
```

```
[ ]  x = np.array(normalized_scores_snet)
     print(np.unique(x))

     [-2. -1.  0.  1.  2.]
```

- Stock Dataset –



Stock Dataset Merged

```
[ ]  merged_df
```

	Datetime	Open	High	Low	Close	Adj Close	Volume
0	2023-03-30 09:30:00-04:00	101.440002	101.480003	100.290001	100.839897	100.839897	1698750
1	2023-03-30 09:35:00-04:00	100.836098	100.919998	100.500000	100.639999	100.639999	574057
2	2023-03-30 09:40:00-04:00	100.660004	100.809998	100.529999	100.654999	100.654999	615039
3	2023-03-30 09:45:00-04:00	100.650002	100.775002	100.480003	100.638000	100.638000	572614
4	2023-03-30 09:50:00-04:00	100.624603	100.769997	100.589996	100.684799	100.684799	465212
...	...	...	...	...	...	...	...
3739	2023-06-07 15:35:00-04:00	123.040001	123.080002	122.730003	122.741501	122.741501	351207
3740	2023-06-07 15:40:00-04:00	122.750000	123.000000	122.629997	122.974998	122.974998	393978
3741	2023-06-07 15:45:00-04:00	122.980003	123.029999	122.660004	122.809998	122.809998	553043
3742	2023-06-07 15:50:00-04:00	122.820000	123.250000	122.809998	122.940002	122.940002	724722
3743	2023-06-07 15:55:00-04:00	122.848503	123.099998	122.739998	122.970001	122.970001	1607058

3744 rows × 7 columns

- LSTM model (without Sentiment Dataset) –

```
[ ]  train_ind = int(0.85*len(df))
     train = data[:train_ind]
     test = data[train_ind:]
```

```
[ ]  train.shape, test.shape

     ((3182, 4), (562, 4))
```

```
[ ]  xtrain,ytrain,xtest,ytest = train[:,:4],train[:,3],test[:,:4],test[:,3]
     xtrain.shape, ytrain.shape,

     ((3182, 4), (3182,))
```

```
[ ]  lookback = 60
     n_features = 4
     output_dim = 4
     train_len = len(xtrain) - lookback
     test_len = len(xtest) - lookback
```

```
[ ]  import tensorflow as tf

     model1.compile(loss='mse', optimizer='adam', metrics=['accuracy',
                                                 tf.keras.metrics.Precision(),
                                                 tf.keras.metrics.Recall(),])

     earlystop = EarlyStopping(monitor='val_loss', min_delta=0.0001, patience=80,  verbose=1, mode='min')
```

```
[ ]  start = time()
     print("start:",0)
     model1.fit(x_train,y_train, epochs = 50, batch_size=30,
              validation_data=(x_test,y_test),verbose = 1,
              shuffle = False, callbacks=[earlystop])
     print("end:",time()-start)
```

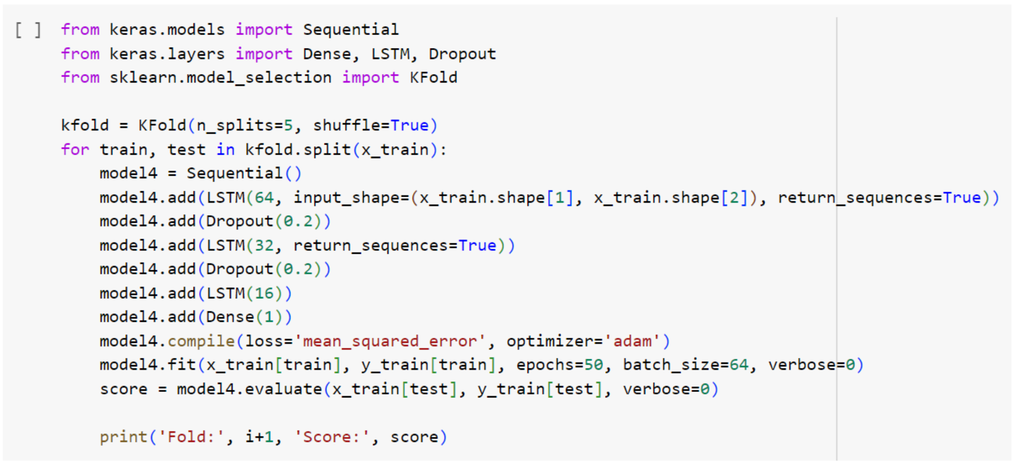
Regularization

```
[ ]  from keras.models import Sequential
     from keras.layers import Dense, LSTM, Dropout

     model2 = Sequential()
     model2.add(LSTM(120, input_shape=(x_train.shape[1], x_train.shape[2]), return_sequences=True))
     model2.add(Dropout(0.2))
     model2.add(LSTM(100, return_sequences=True))
     model2.add(Dropout(0.2))
     model2.add(LSTM(80))
     model2.add(Dense(1))
```

```
[ ]  model2.compile(loss='mean_squared_error', optimizer='adam')
     earlystop = EarlyStopping(monitor='val_loss', min_delta=0.0001, patience=80,  verbose=1, mode='min')
```

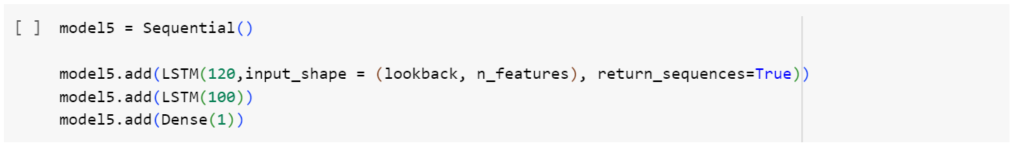
```
[ ]  start = time()
     print("start:",0)
     model2.fit(x_train,y_train, epochs = 50, batch_size=30,
              validation_data=(x_test,y_test),verbose = 1,
              shuffle = False, callbacks=[earlystop])
     print("end:",time()-start)
```

Cross - Validation

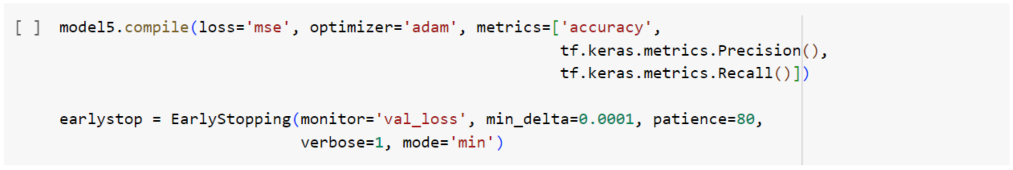


```
[ ]  from keras.models import Sequential
     from keras.layers import Dense, LSTM, Dropout
     from sklearn.model_selection import KFold

     kfold = KFold(n_splits=5, shuffle=True)
     for train, test in kfold.split(x_train):
         model4 = Sequential()
         model4.add(LSTM(64, input_shape=(x_train.shape[1], x_train.shape[2]), return_sequences=True))
         model4.add(Dropout(0.2))
         model4.add(LSTM(32, return_sequences=True))
         model4.add(Dropout(0.2))
         model4.add(LSTM(16))
         model4.add(Dense(1))
         model4.compile(loss='mean_squared_error', optimizer='adam')
         model4.fit(x_train[train], y_train[train], epochs=50, batch_size=64, verbose=0)
         score = model4.evaluate(x_train[test], y_train[test], verbose=0)

         print('Fold:', i+1, 'Score:', score)
```

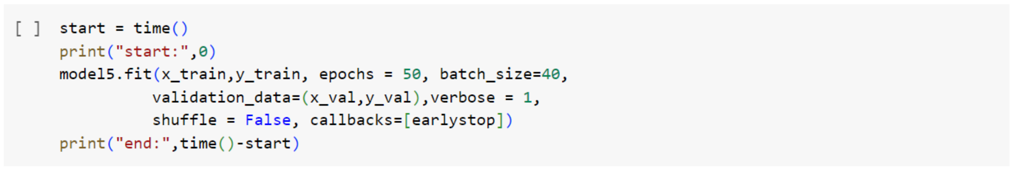
- LSTM model (with Sentiment Dataset) –



```
[ ]  model5 = Sequential()

     model5.add(LSTM(120,input_shape = (lookback, n_features), return_sequences=True))
     model5.add(LSTM(100))
     model5.add(Dense(1))
```

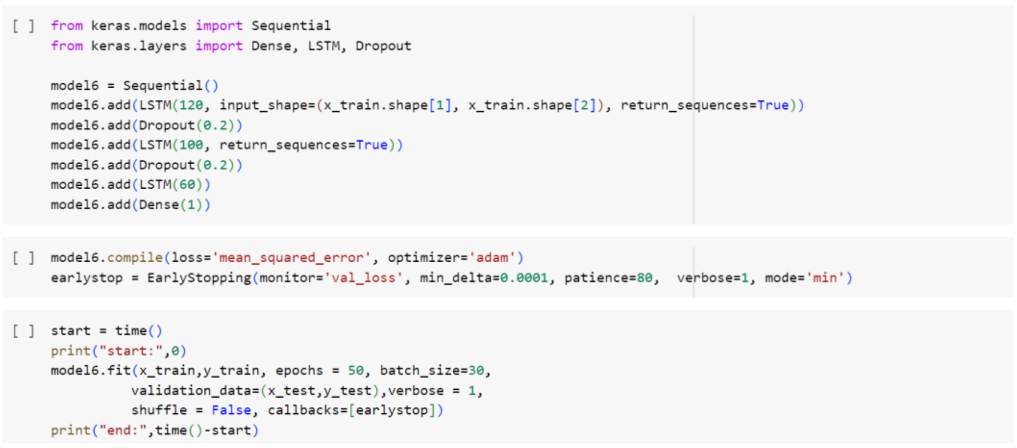


```
[ ]  model5.compile(loss='mse', optimizer='adam', metrics=['accuracy',
                                                             tf.keras.metrics.Precision(),
                                                             tf.keras.metrics.Recall()])

     earlystop = EarlyStopping(monitor='val_loss', min_delta=0.0001, patience=80,
                               verbose=1, mode='min')
```



```
[ ]  start = time()
     print("start:",0)
     model5.fit(x_train,y_train, epochs = 50, batch_size=40,
                validation_data=(x_val,y_val),verbose = 1,
                shuffle = False, callbacks=[earlystop])
     print("end:",time()-start)
```

Regularization



```
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout

model6 = Sequential()
model6.add(LSTM(120, input_shape=(x_train.shape[1], x_train.shape[2]), return_sequences=True))
model6.add(Dropout(0.2))
model6.add(LSTM(100, return_sequences=True))
model6.add(Dropout(0.2))
model6.add(LSTM(60))
model6.add(Dense(1))
```

```
model6.compile(loss='mean_squared_error', optimizer='adam')
earlystop = EarlyStopping(monitor='val_loss', min_delta=0.0001, patience=80,  verbose=1, mode='min')
```

```
start = time()
print("start:",0)
model6.fit(x_train,y_train, epochs = 50, batch_size=30,
          validation_data=(x_test,y_test),verbose = 1,
          shuffle = False, callbacks=[earlystop])
print("end:",time()-start)
```