Stock Price Prediction using Sentiment Analyses of Tweets

Anand, Mayank
my321532@dal.ca

Bhupathiraju, Akhilesh Varma
ak445438@dal.ca

December 2021

1 Abstract

The analysis of the stock market has been very profound among the machine learning community. It is well known that is it not very easy to predict the future stock price of a company. There are a variety of intrinsic and extrinsic factors that might influence the market. Many predictive approaches ranging from statistical models to advanced deep learning models have been developed. Most people use historical price data and other technical measures to analyze the stock movement. It is also interesting to see that the news about the company has some kind of influence over the choices of an investor. This paper presents and compares a range of prediction techniques that can be used to predict the stock market. To begin with, the stock data along with that the news of a stock company were extracted from Yahoo Finance and Twitter respectively, and then sentiment scores were extracted from the text using BERT (Bidirectional Encoder Representations from Transformers) which is a state of the art language model. Following that, deep learning models namely LSTM (Long Short Term Memory) and GAN (Generative Adversarial Network) were built and these results were compared with a simpler ARIMAX (Auto-Regressive Integrated Moving Average) model.

2 Introduction

Researchers have been predicting Stock Market using machine learning like ARIMA by taking the past data for a while now. In addition to the past data, they have also included other features, as it is well known that various other factors affect the stock price. Moreover, with the recent advancements in deep learning space, neural networks specially RNNs and GANs have been proven to perform better than previous ones. In our paper, we tried to expand one such idea of using LSTMs and GAN as implemented by Priyank Sonkiya [10] and looked at particularly how the sentiments of tweets related to the stock company would be effective in predicting the stock price. As Anshul Mittal discussed [7] in his research where he used POMS(Profile of Mood States) score along with several tweets containing the stock name which showed how tweets emotions can affect stock prices movement, a similar thing was tested out where emotions were extracted from text using a tool. In addition to this, to get the sentiment scores the sentiment analysis of tweets was performed from text data retrieved from Twitter accounts of CEO/CTO, company news, public news gotten with the help of keyword search. To obtain the sentiment scores flair library was used which has been trained on State-of-the-Art models like BERT which is a transformer-based machine learning method for natural language processing (NLP) pre-training developed by Google. As for the actual prediction of future stock prices, we explored various models from machine learning (ARIMAX) to deep learning (LSTM and GAN) and tested out the results, and compared them. Our main primary model was LSTM which uses RNN (Recurrent Neural Network) architecture and it overcomes the problem of vanishing gradient faced by them. LSTM is known for understanding the long-term and short-term dependencies present in the data, hence they are ideal for modeling time series data. The sentiment score was passed to the model along with previous 10-day data for each row and predictions were made for the 11th day. For implementing GAN we have passed the previous day closing prices of stock to train the model, and the sentiment scores were passed in as a latent. GAN works by creating synthetic data and tries to classify whether the data is real or fake. Even though GAN were primarily built for the generation of fabricated images. As performed in [10] we have decided to pass the 1-d time series data as input. The data for our purpose included stock market data NVIDIA which is part of SP 50 stock companies extracted from the Yahoo Finance website. As for the tweets, we will be extracting them from Twitter from different sources like the Company news account, CEO/CTO official account, company keyword search, and ticker keyword search. The stock market being highly volatile, various things can trigger a sudden rise or fall of the price, one of them being the real-time news related to that company. For instance, a highly optimistic tweet from the CEO of the company could grab the eyeballs of many investors around the world leading to a spike in the price. This is why looking at the sentiments of the tweets can help us understand the market change and may be useful in predicting the price of a company's stock. Although in the end, only the ticker keyword was able to give meaningful results.

3 Related Work

There have been a plethora of methodologies used for the prediction of the stock price from basic statistical models like simple moving averages to machine learning models like random forest regression. Analysts like Barboza et al [8] have used past prices along with other technical indicators as part of their data set. Researchers like Chetan [4] have used sources like Twitter to scrap news data and used them for retrieving sentiment values. Bharadwaj et al [6] described the need to incur sentiment analysis while predicting stock prices. Anshul Mittal [7] in his paper discussed how emotions and moods of individuals can affect their decision-making process, thus, showcasing whether there is a direct correlation between public sentiment and market sentiment. They have used Twitter data to get the public opinion and classify them using something called POMS(Profile of Moods State). Although there were standard tools like OpinionFinder, SentiWordnet available for this purpose, the authors decided to build their sentiment analyzer using custom POMS. These POMS were grouped into 4 classes - Calm, Happy, Alert, and Kind. Using this, different prediction models were built to test the actual vs predicted stock movement.

Priyank Sonkiya [10] has proposed that GAN (Generative Adversarial Network) showed very promising results stock price predictions. He and his fellow researchers have experimented on different models and they concluded the GAN was performing the best among in terms of RMSE (Root Mean Square Error). They have made use of the news and headlines from the Seeking Alpha website and performed sentiment analysis on it by passing them through finBERT transformer model. Subsequently, different models were built - ARIMA, LSTM, GRU, and GAN. Among these GAN gave the best results. In this model, GRU was used as the generator and 1-d CNN was as the discriminator. The sentiment results were passed in as a latent vector to the model, by doing this it was observed that the model converged much quicker than when assigned with random weights. The author and his team had not only used the past price of the stock as the main feature but a total of 37 technical indicators to predict the target price. Before passing the data to the model, timestamps were created. In this respect, the team has created a 3-day timestamp for their time-series data, where fundamentally past 3 days data was used to predict the value of the 4th day.

4 Problem Definition and Methodology

In this, we are essentially trying to see whether the daily news related to the company can add influence in predicting the next day's closing price of a stock. To extract the meaning from the headlines and news, sentiment analysis is considered one of the best tools. These sentiments might have a direct impact on the emotions of investors. For instance, if there is negative talk about a company because of its poor performance or if there is a recent fallout with a partnering company; people are going to start pulling their investments from the company and this may cause the share price to fluctuate downwards. Once the sentiment values are obtained, they are passed to a prediction model along with other parameters like the past price of the stock for forecasting future prices. This way traders or investors who want to analyze the news data can regard these results as an influential attribute for speculation.

4.1 Data Collection and Preparation

A file containing S&P 50 companies with their stock names and ticker symbol was created. Using this file, the stock data was fetched from the Yahoo finance website by iterating through the ticker one by one. The start and end dates were also given. By using the yfinance [2] library in python, the day-wise stock data was retrieved for 1 year. For our use case, we have fixed to taking only NVDIA's stock data. For extracting the tweets from Twitter numerous implementations have been tested by us but none of them were giving out the exact results that we wanted. Finally, an open-source library called snscrape [6] was used which gets the tweets data based on search keywords along with the start date and end date. For the keyword search, the ticker of the stock was used for example: AMZN, NVDA.

4.1.1 Challenges associated with extracting tweets from Twitter :

- Tweepy – the most commonly used library for tweet extraction. However, it has quite a few limitations. The main one being the cap for requests is 15 minutes and only the recent 3200 tweets are retrieved every time we try to query. It works on old authentication v1.1, and it is incompatible with changing the search query as per our needs e.g. passing in the dates. The limit of a normal developer account on Twitter is 500,000 tweets per month.

- Postman – to overcome the drawbacks of tweepy, we tried passing in different twitter search URLs that are needed for our scenario https://api.twitter.com/2/tweets/search/all. Using this we were able to extract the tweets, but again there was a limit to it.

- Academic researcher access for twitter's developer account- By activating this, although we were able to use the full achieve search and our monthly cap increased. Getting the required tweets was quite a problem. As the tweets obtained were very slow and every time we have to call and pass in the next token to get the next set of results. This causes unnecessary delays in getting the results.

- Company name keyword search – The amount of data that is obtained by entering a stock's company name is really huge. But the amount of stock-related tweet data that is needed for our purpose is very minute and it might get lost among several other tweets.

- User handles including that of CEO/CTO – Not tweet given out by the company's handle or by their CEO/CTO might be that important for our use case, for e.g A tweet containing "Happy Thanksgiving" is not useful, although it gives out the positive sentiment. One more thing that was observed was that resulting tweets were not that proper as most of the tweets were related to replies or retweets which may not add much meaning to our use case.

- Stock news Twitter handle – Although these handles post only stock-related tweets, not every day do we get news or updates related to a particular stock company. So, the data that we end up getting might result in a sparse dataset. This has been checked using fuzzy keyword matching (fuzzywuzzy library in python) which uses Levenshtein/edit distance metric for string matching. By taking the company's name as the search string, the tweet data for each day was searched and the results obtained have shown that there were days when there were no tweets about the company.

4.2 Data Prepossessing

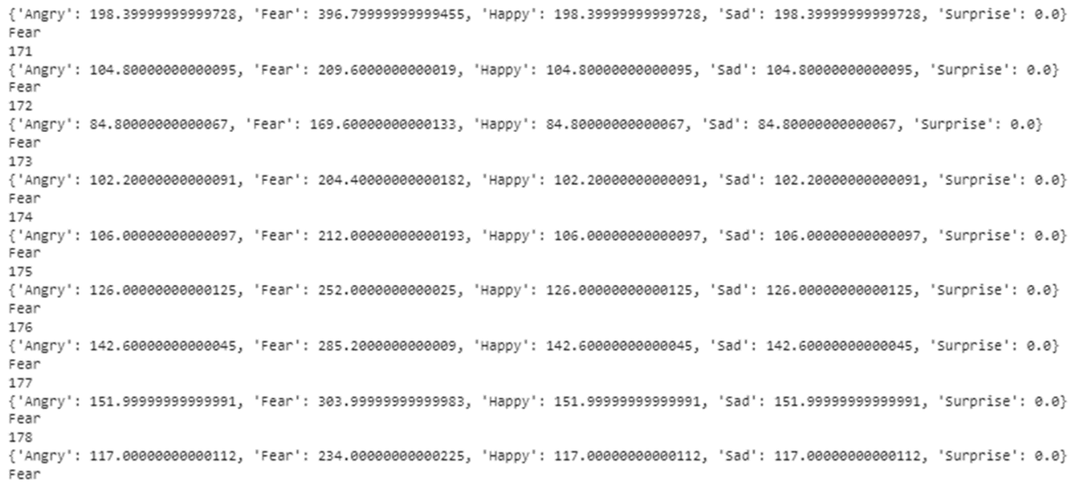
Since there are two datasets now and, in the tweets, data frame, there are multiple tweets for a given day. These multiple tweets were converted to lists for a single day first by removing the time and keeping only the date of the tweet and then doing a group by on the table. Now, the data frame will be having all the tweets for each day in a single row. After this, merging was performed on the updated tweets data frame and the original stock dataframe based on date. As one of our main sources of data will be from twitter and we are expecting to have a lot of noise in it. The shape of the data frame after merging is (249, 8)

Cleaning Text data – Before using text data as it is from Twitter, some cleaning has to be performed to remove all the noise from the text. For this purpose, scrub words like ASCII characters, links, new lines, extra spaces, HTML markups were removed using regex. Also, contraction words were expanded to their original form. Stemming was not performed on text although it might have removed some data redundancy as sending the words in their true form to the sentiment analyzer might be more meaningful. Stop words have been removed before passing the data for sentiment analysis to the Vader library. However, they weren't removed when the BERT model was used for sentiment analysis.

4.3 Sentiment Analysis

For predicting the sentiments of our tweets, we have explored three approaches. One would be POMS(Profile of Mood States), VADER (Valence Aware Dictionary for Sentiment Reasoning) which is a rule-based (lexicon based) analysis tool that helps us get sentiment score which can be either positive, negative, neutral, or compound based on the twitter sentence that we pass. Another one would be a deep learning transformer-based model called BERT by Google [3] that has been trained on a huge corpus of data. In addition to BERT, there is finBERT [6] which is a fine-tuned version of BERT trained on financial data. We would be comparing the results of these two.

POMS (Profile of Mood States) - To extract the POMS tag from the tweets, text2emotion library in python was used which classifies text into 5 emotions Happy, Angry, Sad, Surprise, and Fear. However, it didn't add much value as a feature, as emotions of most of the days were predicted as having "fear" as the highest score. In the end, we were getting the majority (99%) of the tags as fear. So, we had decided not to use this feature.



```
{'Angry': 198.3999999999728, 'Fear': 396.7999999999455, 'Happy': 198.3999999999728, 'Sad': 198.3999999999728, 'Surprise': 0.0}
Fear
171
{'Angry': 104.8000000000095, 'Fear': 209.6000000000019, 'Happy': 104.8000000000095, 'Sad': 104.8000000000095, 'Surprise': 0.0}
Fear
172
{'Angry': 84.8000000000067, 'Fear': 169.6000000000133, 'Happy': 84.8000000000067, 'Sad': 84.8000000000067, 'Surprise': 0.0}
Fear
173
{'Angry': 102.2000000000091, 'Fear': 204.4000000000182, 'Happy': 102.2000000000091, 'Sad': 102.2000000000091, 'Surprise': 0.0}
Fear
174
{'Angry': 106.0000000000097, 'Fear': 212.0000000000193, 'Happy': 106.0000000000097, 'Sad': 106.0000000000097, 'Surprise': 0.0}
Fear
175
{'Angry': 126.0000000000125, 'Fear': 252.000000000025, 'Happy': 126.0000000000125, 'Sad': 126.0000000000125, 'Surprise': 0.0}
Fear
176
{'Angry': 142.6000000000045, 'Fear': 285.200000000009, 'Happy': 142.6000000000045, 'Sad': 142.6000000000045, 'Surprise': 0.0}
Fear
177
{'Angry': 151.9999999999991, 'Fear': 303.9999999999983, 'Happy': 151.9999999999991, 'Sad': 151.9999999999991, 'Surprise': 0.0}
Fear
178
{'Angry': 117.0000000000112, 'Fear': 234.0000000000225, 'Happy': 117.0000000000112, 'Sad': 117.0000000000112, 'Surprise': 0.0}
Fear
```

Figure 1: Output probabilities of emotion tag from text2emotion library in python

Vader's Sentiment – Initially SentimentIntensityAnalyzer from Vader in python's nltk library was used. But the results obtained from the model were not that good. In most of the classes, nearly 98% were coming either as positive or neutral, and less than 2% were coming as negative for a company. One reason for this could be as Vader is lexical based and uses pre-calculated sentiment scores for each word, so this fails to capture the semantic meaning at a sentence level. The same was the case with the TextBlob library.

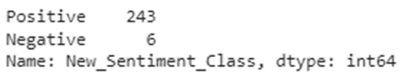

```
Positive    243
Negative      6
Name: New_Sentiment_Class, dtype: int64
```

Figure 2: Value counts of sentiment class using Vader

BERT - To overcome the shortcomings of the above libraries, we wanted to try the FinBERT transformer model which is a pre-trained model from Google trained on financial data. However, training the transformer model was very resource-consuming. So, we used a powerful model called Flair [1] present in python which is a pre-trained sentiment model used to predict the sentiments of the tweets. Flair uses DistilBERT model to make predictions, which is a smaller yet powerful model of the BERT transformer model containing nearly 66 million parameters.

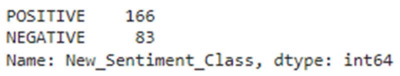
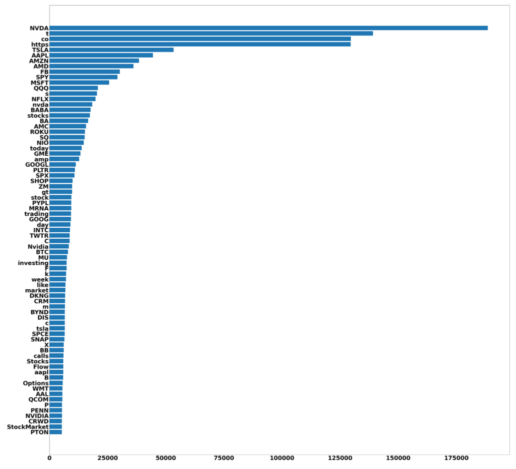

```
POSITIVE    166
NEGATIVE     83
Name: New_Sentiment_Class, dtype: int64
```

Figure 3: Value counts of sentiment class using flair

For predicting the sentiment score, each tweet was sent to the flair model and a sentiment class (positive or negative) was obtained. However, as there are multiple tweets nearly 700 -800 average tweets per day and each tweet will be having its own sentiment class, the overall sentiment value of the entire day was considered. For this, the most frequently occurring sentiment class for the day was considered the final sentiment class for that particular day. These sentiment classes are passed to the LSTM model as an additional feature and also to the generator model of the GAN as a latent vector similar to the one done in the paper [10]

4.4 Exploratory data analysis on the text

To understand more about the text data that we have, we performed some basic text analysis like looking at the word count, removing stop words, stemming/ lemmatization, tokenization. After this, we have decided to perform K-means clustering on the text data to find similar clusters, for this, we converted our tokens which are obtained from tokenization to Tf-idf vectors. To extract topics from the text data, we also performed topic modeling using SVD (Singular Value Decomposition)


(a) Highest frequency of words with the stopwords removed

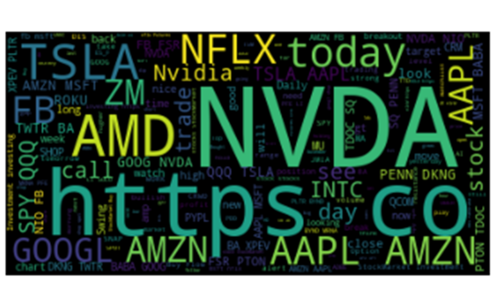
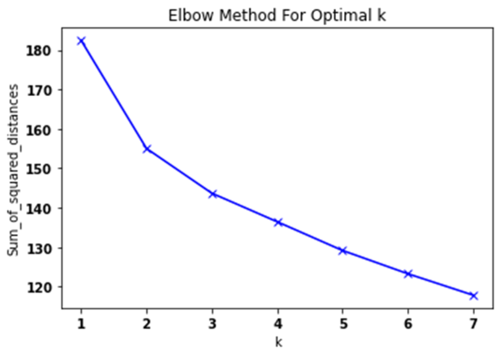
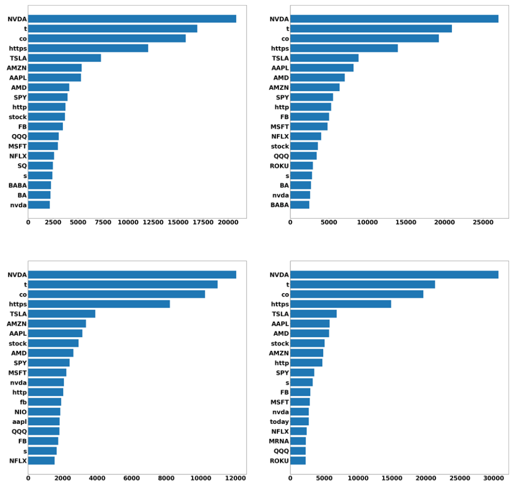

(b) Word cloud of the same frequency distribution of words

Figure 4: Frequency distribution and word cloud

K means clustering was performed using Euclidean distance by taking the vectors obtained after converting the text in the tweets into Tf-idf values. The optimal number of clusters were found using elbow curve. After clustering, it is observed that the words in the clusters are more or less similar indicating that the clusters are not well defined.



(a) Elbow Curve to choose the right value for k in k-means clustering



(b) Getting the top words from each of the clusters

Figure 5: K-means Clustering

SVD works on the mathematical principle that a matrix can be broken down into product of Eigen values (diagonal matrix) and Eigen vectors (orthogonal matrices). M=U*S*V. These singular matrices usually represented by U, V contains document-topic representations and term-topic representations. The diagonal matrix represented by S or Sigma can be tuned to adjusting the k value indicating the number of topics we want to extract from the text. Here in our data, the words present in the topics may not make much sense but if the text corpus is huge enough, we can obtain some well defined topics from our text.

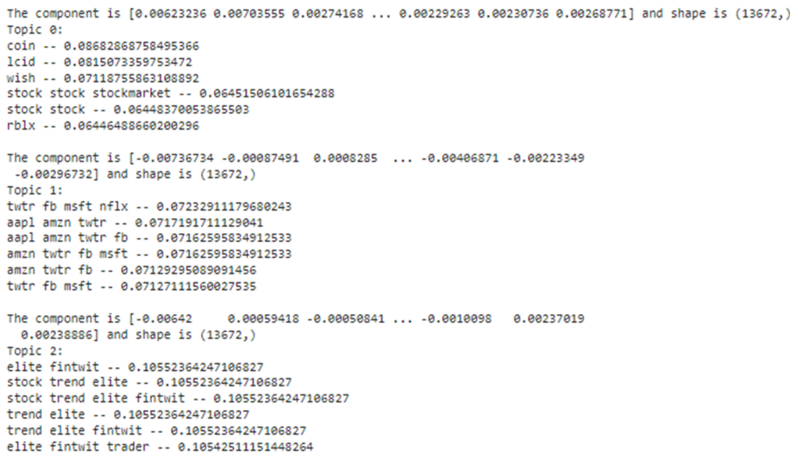
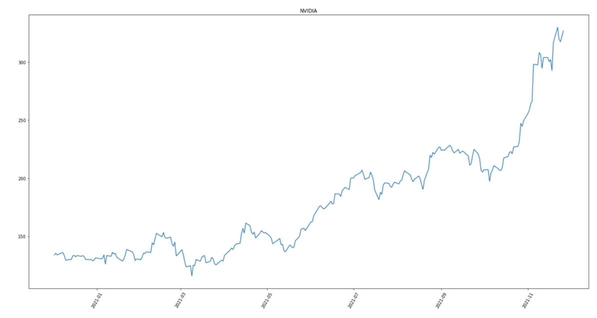
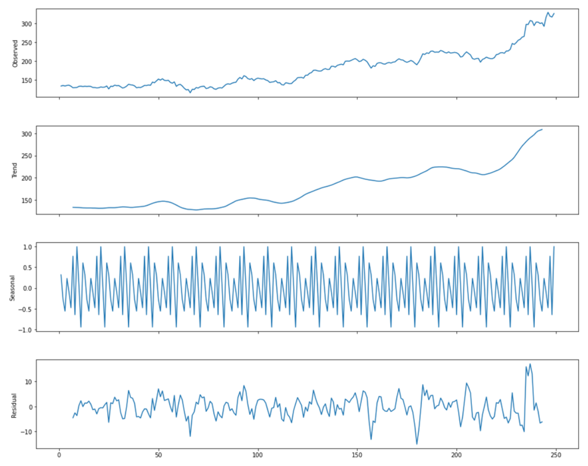

```
The component is [0.00623236 0.00703555 0.00274168 ... 0.00229263 0.00230736 0.00268771] and shape is (13672,)
Topic 0:
coin -- 0.08682868758495366
lcid -- 0.0815073359753472
wish -- 0.07118755863108892
stock stock stockmarket -- 0.06451506101654288
stock stock -- 0.06448370053865503
rblx -- 0.06446488660200296

The component is [-0.00736734 -0.00087491  0.0008285  ... -0.00406871 -0.00223349
 -0.00296732] and shape is (13672,)
Topic 1:
twtr fb msft nflx -- 0.07232911179680243
aapl amzn twtr -- 0.0717191711129041
aapl amzn twtr fb -- 0.07162595834912533
amzn twtr fb msft -- 0.07162595834912533
amzn twtr fb -- 0.07129295089091456
twtr fb msft -- 0.07127111560027535

The component is [-0.00642     0.00059418 -0.00050841 ... -0.0010098   0.00237019
  0.00238886] and shape is (13672,)
Topic 2:
elite fintwit -- 0.10552364247106827
stock trend elite -- 0.10552364247106827
stock trend elite fintwit -- 0.10552364247106827
trend elite -- 0.10552364247106827
trend elite fintwit -- 0.10552364247106827
elite fintwit trader -- 0.10542511151448264
```

Figure 6: Some of the topics obtained from SVD with top occurring words in each of them

Having a closer look at the closing price of the NVIDIA stock from December 2020 to November 2021, it is can be seen that there is a clear upward trend in the price and a bit of seasonality too. This can be detected even better after plotting the decomposition of the target price. Also, there is some amount of randomness spotted in the residual portion.



(a) Closing price of NVIDIA stock from December 2020 to November 2021



(b) Closing price of NVIDIA stock from December 2020 to November 2021

Figure 7: Understanding target variable

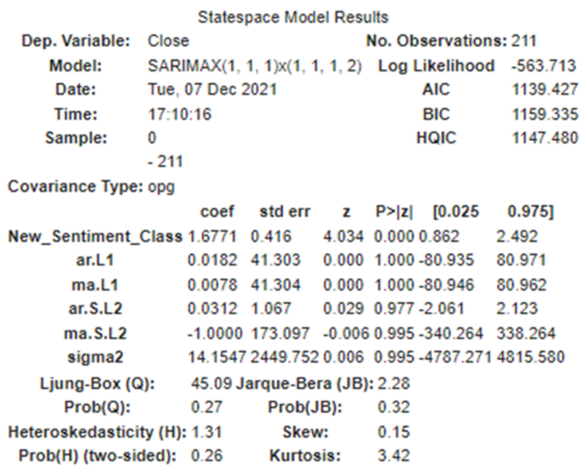
4.5 Potential Models

4.5.1 ARIMAX

ARIMA stands for Auto-regressive integrated moving average. The ARIMA model is represented by (p,d,q), where p relates to the autoregressive model's order (number of time lags), d is the number of times differencing is done to make the data stationary, and q is the number of lags seen in the moving-average model. The Seasonal ARIMA model is represented by (p,d,q)(P, D, Q)m. The capital P, D, Q refers to the autoregressive, differencing, and moving average terms for the seasonal part of the ARIMA model.

ARIMAX stands for Auto-regressive integrated moving average and The X stands for exogenous, which is nothing but the additional variable that is a separate outside variable that helps us measure the endogenous variable it is included for getting better prediction. In the ARIMAX model, we will be passing in the historical data of the closing price, along with other key parameters that we believe will add value to our model prediction.

For our use case, we will be using the SARIMAX model as there is some seasonality seen in the target variable. And for the exogenous variable, we will be passing in the sentiment scores that we have generated from the tweets data. Before sending our data to the model, the target variable has been shifted by 1 indicating that the current day prices will be used for predicting the next day's price.



```
                         Statespace Model Results
 Dep. Variable:  Close                              No. Observations: 211
        Model:   SARIMAX(1, 1, 1)x(1, 1, 1, 2)  Log Likelihood   -563.713
         Date:   Tue, 07 Dec 2021                          AIC   1139.427
         Time:   17:10:16                                  BIC   1159.335
       Sample:   0                                        HQIC   1147.480
                 - 211
Covariance Type: opg
                          coef     std err   z       P>|z|  [0.025     0.975]
New_Sentiment_Class  1.6771   0.416     4.034   0.000  0.862      2.492
          ar.L1      0.0182   41.303    0.000   1.000  -80.935    80.971
          ma.L1      0.0078   41.304    0.000   1.000  -80.946    80.962
          ar.S.L2    0.0312   1.067     0.029   0.977  -2.061     2.123
          ma.S.L2    -1.0000  173.097   -0.006  0.995  -340.264   338.264
          sigma2     14.1547  2449.752  0.006   0.995  -4787.271  4815.580
     Ljung-Box (Q):      45.09  Jarque-Bera (JB): 2.28
          Prob(Q):       0.27         Prob(JB):   0.32
Heteroskedasticity (H):  1.31            Skew:    0.15
 Prob(H) (two-sided):    0.26        Kurtosis:    3.42
```

Figure 8: SARIMAX Model Summary

After looking at the decomposition of the target variable (close price). It is obvious that there is seasonality and trend associated with the close price of the stock. So, while building the SARIMAX model, we try to incorporate them in the p,d,q. The resultant model after being plotted in ACF (autocorrelation) and PACF (partial autocorrelation) and both trend and seasonality is been dealt with.

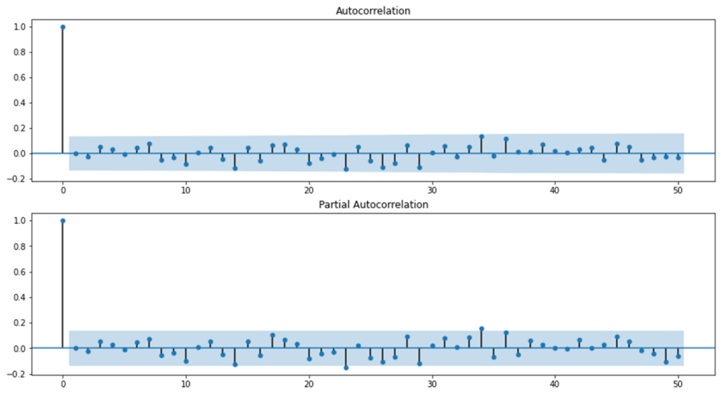
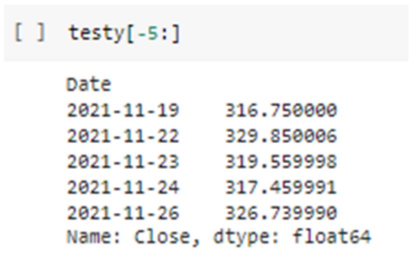


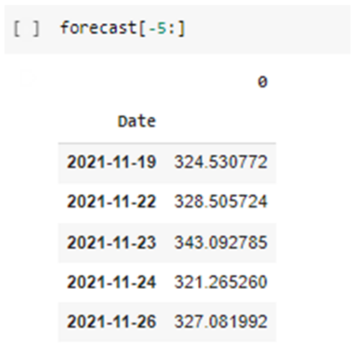
Figure 9: ACF and PACF plots

In order to automate the process of selecting the p and q and finding the differencing degree d, the auto Arima model can be used which automatically selects the parameters that best fit the data. The dataset containing one year of stock prices of NVIDA is divided into 85% of training and 15% of testing based on the date. So, finally, the training data was fitted using the auto Arima model having both seasonality and exogenous variable.



```
[ ]  testy[-5:]

     Date
     2021-11-19    316.750000
     2021-11-22    329.850006
     2021-11-23    319.559998
     2021-11-24    317.459991
     2021-11-26    326.739990
     Name: Close, dtype: float64
```

(a) Actual Price

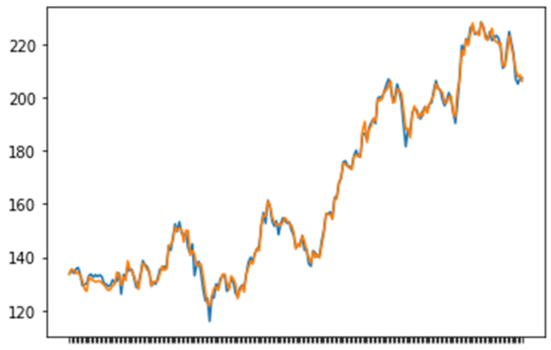


```
[ ]  forecast[-5:]
```

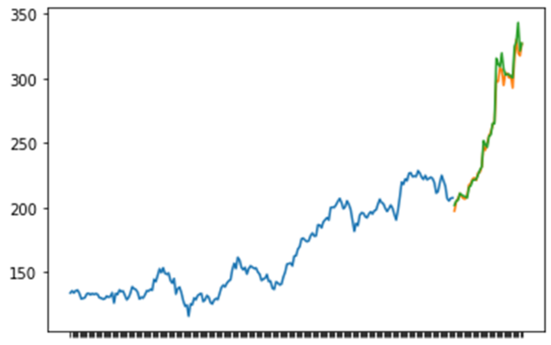
Date	0
2021-11-19	324.530772
2021-11-22	328.505724
2021-11-23	343.092785
2021-11-24	321.265260
2021-11-26	327.081992

(b) Forecasted Price

Figure 10: Actual price values vs forecasted price values



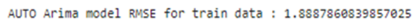
(a) Model performance on training set



(b) Model performance on testing set

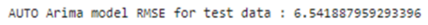
Figure 11: Auto Arima model findings

Interpreting results from ARIMAX/Auto arima - RMSE (Root mean squared error) was used to see the model performance on the training and test data. After plotting the graphs of the original and predicted data, it is obvious that the model was able to capture the overall trend and also accommodate a seasonality for most of the data.



```
AUTO Arima model RMSE for train data : 1.8887860839857025
```

(a) Error metrics for train data



```
AUTO Arima model RMSE for test data : 6.541887959293396
```

(b) Error metrics for test data

Figure 12: Error metrics for Auto Arima

4.5.2 Long Short Term Memory (LSTM)

LSTM models were created as a solution of short-term memory because of the vanishing gradient problem of Recurring Neural Networks(RNN) and perform extremely well with time series data for generating sequences. Internally LSTM has three gates Forget, Input and Output. Forget gate decides what information should be thrown away or kept. Information from the previous hidden state and information from current input is passed through the sigmoid function which gives a value between 0 to 1. A value closer to 0 will be thrown away. Input gate is used to update cell state and the Output gate decides which information will be propagated to the next cell.

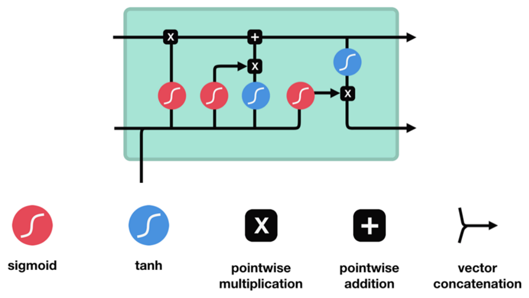
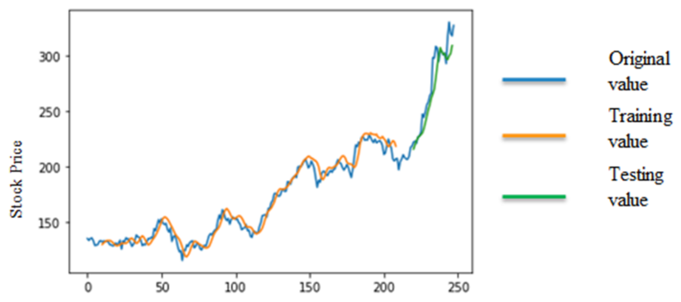


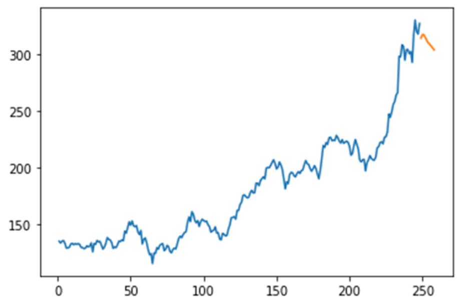
Figure 13: LSTM Model architecture **Source:** Michael Phi [9]

LSTM networks are a modified version of RNN (Recurrent Neural Networks). LSTM's were mainly used to overcome the vanishing gradient problem that occurs in RNN. LSTM's have something called as a cell state where the information is passed through. It also has gates that regulate the flow of information meaning they have the capability to remove and add information to the cell state. It consists of a neural network layer with a sigmoid activation function and a pointwise multiplication operation. The sigmoid function converts the input as one or zero. Zero indicating 'nothing is passed' one indicating 'everything is passed'. There are three gates in LSTM namely, input gate, forget gate, and output gate. These are gates contain parts like tanh and pointwise addition which is used to send the old state and forget some information. This helps the model remember long-term dependencies present in the data. Since we are using time-series data along with sentiments, it is apparent that the past values are going to have some kind of influence over the future predictions. So, using LSTM neural network makes sense as we want to capture these dependencies associated with time. Before sending the data to the model, min-max scaling was performed on the closing price. After this, the data (only the closing) was divided into train and test like before. The train and test data were now transformed in such a way that each of the records contains its previous ten-day values. That is, a time step of ten was used to generate this dataset, indicating that for predicting the value of the current day, the previous 10-day values will be used. Now, the sentiment class vector is attached to the train and test data increasing the features. The data is finally sent to the LSTM model containing 3 stacked LSTM with 64 neurons in the first layer and X neurons in the second layer and

Y neurons in the last layer. A dropout layer is also present in the network. And l2 regularization is applied. Adam optimizer is being used with a learning rate of x. A batch size of 64 is used and that network is trained on 100 epochs. For building and training, the neural network Keras powered by TensorFlow was used.



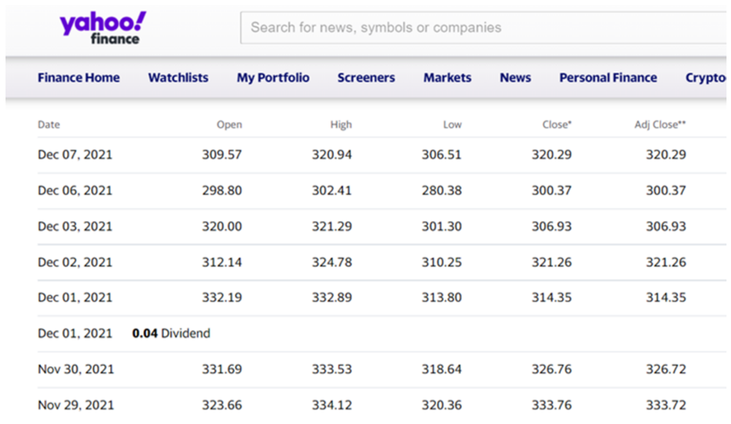
(a) LSTM Model fitted on train data and test data



(b) Forecasting 10 days future stock prices of NVIDIA

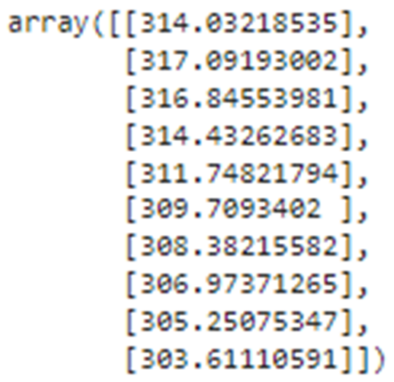
Figure 14: LSTM findings

Interpreting the results from LSTM– The plot of the original data and predicted data shows that the LSTM model was able to capture the overall trend and seasonality pretty well. For testing and comparing the model performance, root mean squared error is once again used. For train data, the RMSE was 0.023 and for test data, the RMSE was 0.085. This shows that the model's performance was better than arimax. Now looking at the future forecasting of the stock prices. It is observed that although there is some error in predicting the exact price of the stock. The model was able to capture and predict the variations in the trend, considering that whenever there was a rise in the stock the model has predicted an upward trend and whenever there was a drop, it was able to predict the downward trend.



Date	Open	High	Low	Close*	Adj Close**
Dec 07, 2021	309.57	320.94	306.51	320.29	320.29
Dec 06, 2021	298.80	302.41	280.38	300.37	300.37
Dec 03, 2021	320.00	321.29	301.30	306.93	306.93
Dec 02, 2021	312.14	324.78	310.25	321.26	321.26
Dec 01, 2021	332.19	332.89	313.80	314.35	314.35
Dec 01, 2021	**0.04** Dividend				
Nov 30, 2021	331.69	333.53	318.64	326.76	326.72
Nov 29, 2021	323.66	334.12	320.36	333.76	333.72

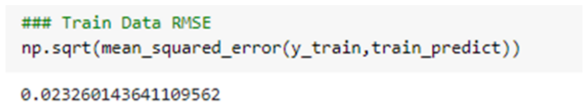
(a) Original values from Yahoo finance website



```
array([[314.03218535],
       [317.09193002],
       [316.84553981],
       [314.43262683],
       [311.74821794],
       [309.7093402 ],
       [308.38215582],
       [306.97371265],
       [305.25075347],
       [303.61110591]])
```

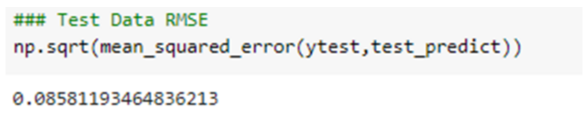
(b) 10 days forecasted values

Figure 15: Orignal vs Forecasted values



```
### Train Data RMSE
np.sqrt(mean_squared_error(y_train,train_predict))

0.023260143641109562
```

(a) Training Error



```
### Test Data RMSE
np.sqrt(mean_squared_error(ytest,test_predict))

0.08581193464836213
```

(b) Testing Error

Figure 16: Evaluation metric for LSTM.

5 Conclusion and Future Work

We have looked at how news sentiment from Twitter can add influence to the stock price and can even be used to predict it. Clearly, LSTM was showing better predictions than the ARIMAX model. Currently, we are working on GAN (Generative Adversarial Network) which is a deep learning model designed by Ian Goodfellow in 2014 [5]. GAN contains two neural networks called Discriminator and Generator which compete with each other in a min-max game. Zhang et al [11] has taken MLP (multi-layer perceptron) as the discriminator and LSTM as the generator for forecasting the future closing prices. Priyank Sonkiya [10] on the other hand has taken 1D CNN as the discriminator and GRU as the generator. He has passed the sentiment values as a latent vector instead of passing in some random weights so that the model would converge much faster. Both the researchers have shown some exciting results for the prediction of the stock market price. Influenced by their work we will be stacking LSTM and GRU as the generator and for the discriminator, we will be trying both MLP and 1D CNN and compare the results. Our model will be different from the other two in such a way that as we are stacking the different RNN models and using additional features such as sentiment scores. Stock data with additional variables are passed to the discriminator while in the generator the sentiment scores of the tweets will be sent as a latent vector.

References

[1] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, 2019.

[2] Ran Aroussi. yfinance. https://github.com/ranaroussi/yfinance, 2021.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[4] Chetan Gondaliya, Ajay Patel, and Tirthank Shah. Sentiment analysis and prediction of indian stock market amid covid-19 pandemic. *IOP Conference Series: Materials Science and Engineering*, 1020:012023, 01 2021.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[6] JustAnotherArchivist. snscrape. https://github.com/JustAnotherArchivist/snscrape, 2021.

[7] Anshul Mittal and Arpit Goel. Stock prediction using twitter sentiment analysis.

[8] Felipe Barboza Oriani and Guilherme P. Coelho. Evaluating the impact of technical indicators on stock forecasting. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2016.

[9] Michael Phi. Illustrated guide to lstm's and gru's: A step by step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, 2018.

[10] Priyank Sonkiya, Vikas Bajpai, and Anukriti Bansal. Stock price prediction using bert and gan, 2021.

[11] Kang Zhang, Guoqiang Zhong, Junyu Dong, Shengke Wang, and Yong Wang. Stock market prediction based on generative adversarial network. *Procedia Computer Science*, 147:400–406, 2019. 2018 International Conference on Identification, Information and Knowledge in the Internet of Things.