Stock Price Prediction using Sentiment Analysis

1 Introduction

The stock market and its movements are highly volatile in the financial world. Researchers are drawn to it in order to capture the volatility and anticipate its next moves. Investors and market analysts research market behaviour and devise buy and sell strategies based on their findings. Since the stock market generates a vast amount of data every day, it is extremely difficult for an investor to consider both current and historical data when forecasting a stock's future trend. There are primarily two approaches for predicting consumer patterns, technical analysis and fundamental analysis. Technical forecasting uses price and volume data from the past to forecast future trends. Fundamental analysis of a company, on the other hand, entails examining financial data in order to gain insight. Our aim is to predict the future trend of a stock by using yahoo conversations about the stock as primary data and attempting to classify the conversation as good (positive) or poor (negative). If the sentiment is optimistic, the stock price is more likely to rise; if the sentiment is negative, the stock price is more likely to fall. We will select the stocks with the highest positive score and predict their prices using fundamental analysis and historical data.

2 Sentiment Analysis

In order to create the dataset for performing the sentiment analysis to select the stock, we have selected the most active stocks with the largest trading volume per day from the site `https://in.finance.yahoo.com/most-active`. We have scrapped the conversations of all these stocks from their corresponding site on yahoo finance using selenium library which supports the automation of web browsers, and stored them in the form of text files.

These text files were pre-processed using SpaCy. SpaCy is a free, open-source library for NLP in Python. It facilitates sentence detection, thereby providing structured data. All punctuations and special characters were removed. It also provides for lemmatization wherein the inflected forms of a word are reduced to their root words, so that the word with the same meaning will be assigned the same token. Lemmatization helps you avoid duplicate words that have similar meanings.The final preprocessed data is overwritten in the corresponding text files.

We have used a pre-trained sentiment analysis model from the flair library [1]. It is one of the most powerful pre-trained models. It's an NLP framework built on top of PyTorch. The flair sentiment classifer was originally trained on IMDB movie review dataset. Flair gives a positive/negative label and gives a score from 0 to 1 indicating how much positive/negative the sentiment is.
For example, for a sentence: " *You know there is a reason that Microsoft and Google hit the ball out of the park Grand Slam Homerun on Datacenter Revenue! Answer is AMD Epyc! AMD is headed for much higher ground, and its innovating over Intel. And it will continue for years to come...* " our model outputs a positive sentiment score of 0.9974676370620728.
For another sentence, "*A lot of investors complain that Apple and Cook have lost their innovative touch.its been a long time since they came out with the iphone*" our model outputs a positive sentiment score of 0.9994353652000427

The table below illustrates sample sentiment scores for the most active stocks.

Stock	Sentiment Score	Positive/negative
AAL	0.9995208978652954	NEGATIVE
AAPL	0.9998970031738281	NEGATIVE
AMC	0.9997380375862122	NEGATIVE
AMD	0.9984862208366394	POSITIVE
BAC	0.9995003938674927	NEGATIVE
F	0.9991517066955566	NEGATIVE
GE	0.9983514547348022	NEGATIVE
GSAT	0.9344609975814819	NEGATIVE
INTC	0.5814518332481384	NEGATIVE
ITUB	0.9846652150154114	NEGATIVE
MVIS	0.997525155544281	NEGATIVE
NIO	0.759212076663971	POSITIVE
NOK	0.5432959794998169	POSITIVE
OCGN	0.9777657389640808	NEGATIVE
PBR	0.9980466365814209	NEGATIVE
PLTR	0.9920461177825928	POSITIVE
PLUG	0.9970833659172058	POSITIVE
SIRI	0.9995372295379639	NEGATIVE
SKLZ	0.9999772310256958	NEGATIVE
SNAP	0.9909734129905701	POSITIVE
T	0.7431989908218384	POSITIVE
TSLA	0.9994969367980957	NEGATIVE
VALE	0.8566350936889648	NEGATIVE
VIAC	0.999734103679657	NEGATIVE
WFC	0.9320763349533081	NEGATIVE

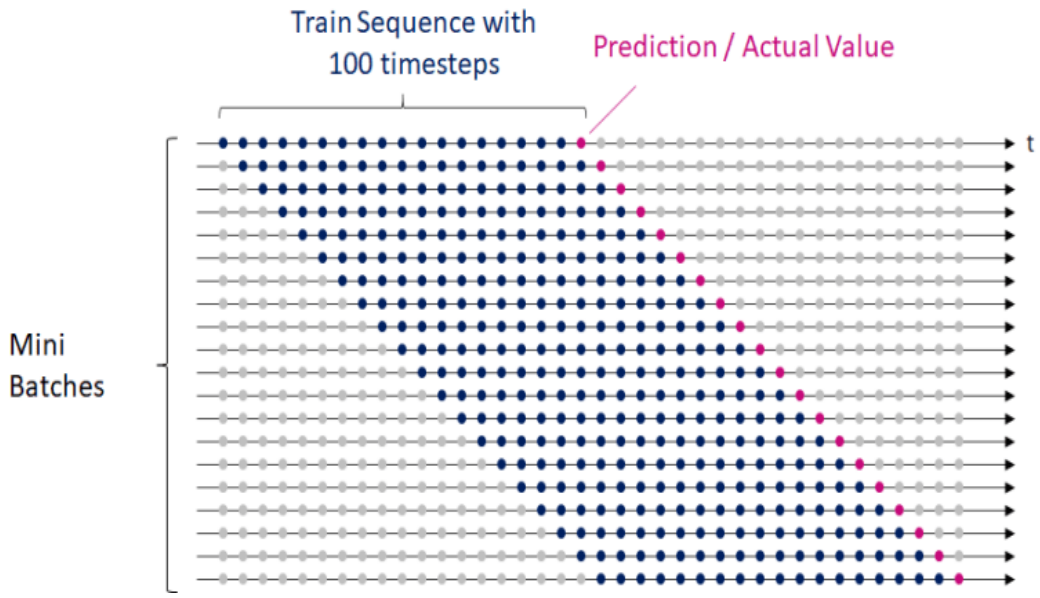
3 Models

After performing sentiment analysis we predict the stock prices for the stock with the most positive score. We took historical data for the past 10 years from alphavantage api with the following features: high , low, open, close, exponential moving averages for the last 5, 30 and 180 days, relative strength index(RSI), on balance volume(OBV) , moving average convergence/divergence (MACD), earnings per share and sentiment score of the stock corresponding to the date.

1. **LSTM**

   The above mentioned data is appropriately scaled and split as train and test data in an 80:20 ratio. Now we train an LSTM model with the train data. The Long Short Term Memory neural network is a type of a Recurrent Neural Network (RNN) which uses previous time events to inform the later ones. As the prediction of stock prices requires long term dependencies, LSTMs are preferred over RNNs.

   In multivariate time series prediction, the model is trained on many sequences to predict the value of the time steps following the sequences. Accordingly, the data structure required for this is three-dimensional. The first dimension is the number of sequences, the second dimension is number of the time steps in a sequence and the third dimension is the number of features.



We have divided the training set into mini batches of size 100 where each mini batch has 7 features. The number of neurons for the hidden layer of the model is equal to the product of timestep and features per mini batch.

Now we find multiple stocks and implement our trained model on these stocks one by one and extract features from the last layer of our model. Next we cluster the stocks based on these features extracted so that we can finally train our model on the cluster containing the chosen stock. In order to cluster the stocks we use the K-means clustering method. It groups similar data points together, discovers underlying patterns and distributes the stocks into k number of clusters. A centroid is the imaginary or real location representing the center of the cluster. The K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

After this we train our model on the stocks in the cluster containing the chosen stock and make our predictions on the test dataset. We plot the predicted values and the actual values, and calculate the mean absolute percentage error (MAPE) and the median absolute percentage error (MDAPE).

2. **ARIMA**

   (a) AR: (Auto Regressive) means that the model uses the dependent relationship between an observation and some predefined number of lagged observations (also known as "time lag" or "lag").

   (b) I:(Integrated) means that the model employs differencing of raw observations (e.g. it subtracts an observation from an observation at the previous time step) in order to make the time-series stationary.

   (c) MA: (Moving Average) means that the model exploits the relationship between the residual error and the observations.

   The standard ARIMA models expect as input parameters 3 arguments i.e. p,d,q. p is the number of lag observations. d is the degree of differencing. q is the size/width of the moving average window. These parameters have been calculated with the help of Auto ARIMA

4 Results

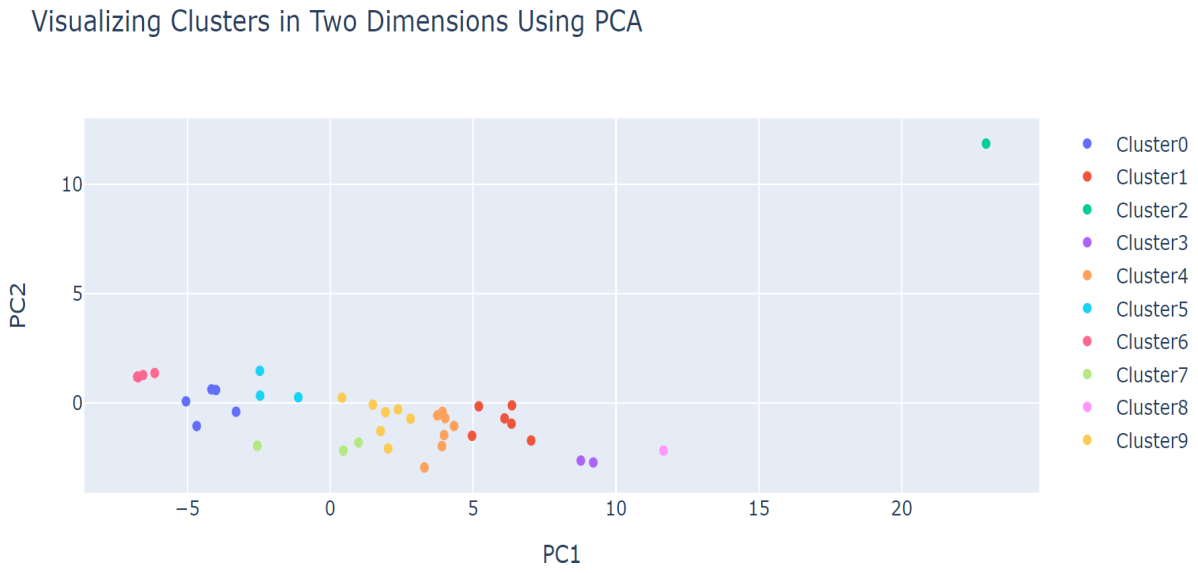
While making predictions for ***AAPL*** using LSTM:

The clusters are found to be as shown in the table below.

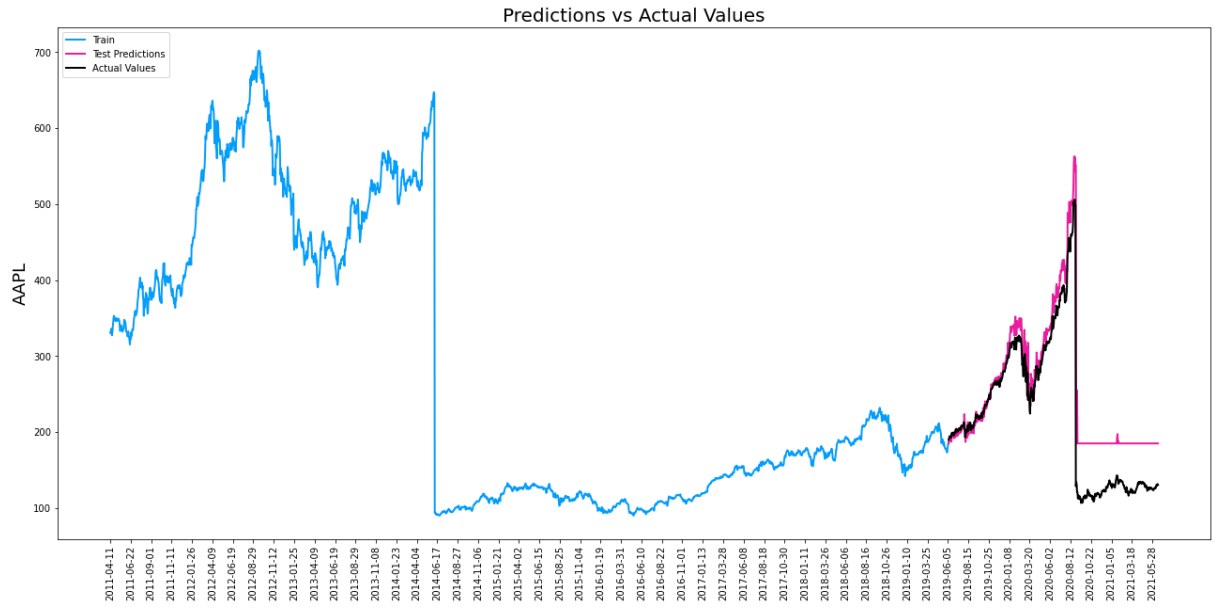
Label	Stock
AAPL	0
TSLA	0
EA	0
SYMC	0
MA	0
UAL	1
CSCO	1
MSFT	1
ABT	1
AXP	1
MCD	1
FUBO	2
WMT	3
AMD	3
BSX	4
VLO	4
TXN	4

Label	Stock
PEP	4
KO	4
BAC	4
SNE	4
JNJ	5
V	5
GM	5
BBBY	6
AAL	6
NOK	6
SWN	6
BB	6
EBAY	6
OXY	6
HPE	6
HMC	6
NAV	6

Label	Stock
WBA	6
F	6
GE	6
XOM	6
IBM	6
T	7
TM	7
GOOG	7
BA	8
NOC	9
DIS	9
INTC	9
NVDA	9
AMZN	9
CVX	9
LMT	9



$TSLA, EA, SYMC, MA, AAPL$ were grouped in the same cluster with AAPL. On training the model on these stocks and predicting values for AAPL, we got the resulting graph which can be seen in fig 5.



Mean Absolute Percentage Error (MAPE): 22.77

Median Absolute Percentage Error (MDAPE): 7.41

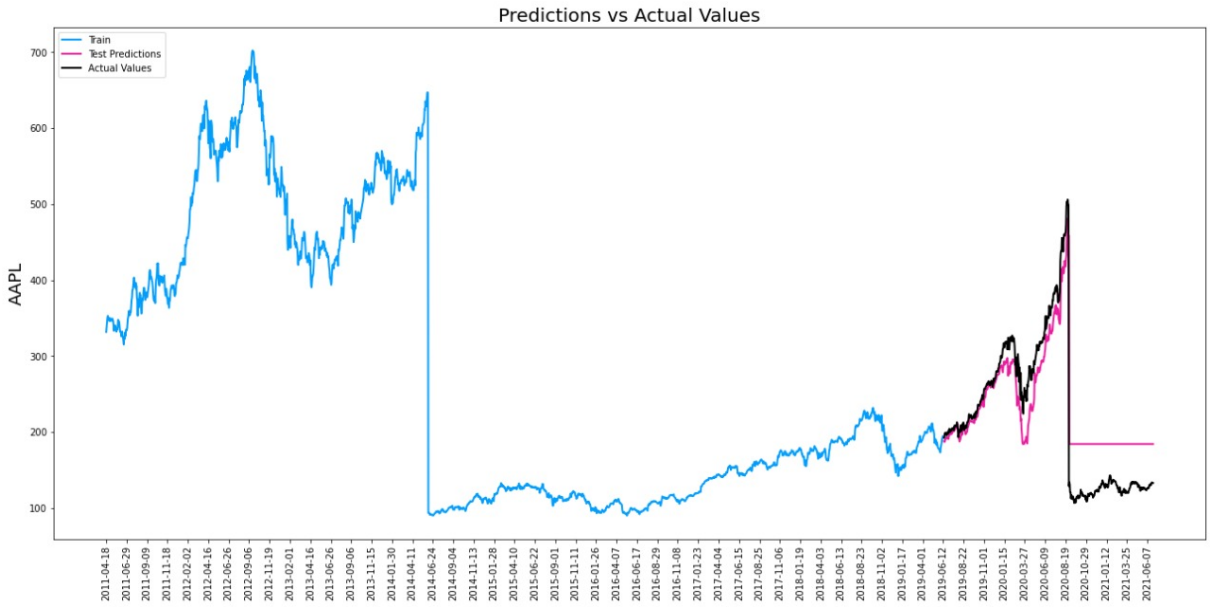
5 Comparison

Here we will compare the accuracy and predictions for the various methods we have deployed to predict stock price data for AAPL.

Method 1: **RNN without clustering**

Here we took historical data for the past 10 years from alphavantage api with the following features: high , low, open, close, exponential moving averages for the last 5, 30 and 180 days corresponding to the date, RSI, OBV, MACD and quarterly EPS values

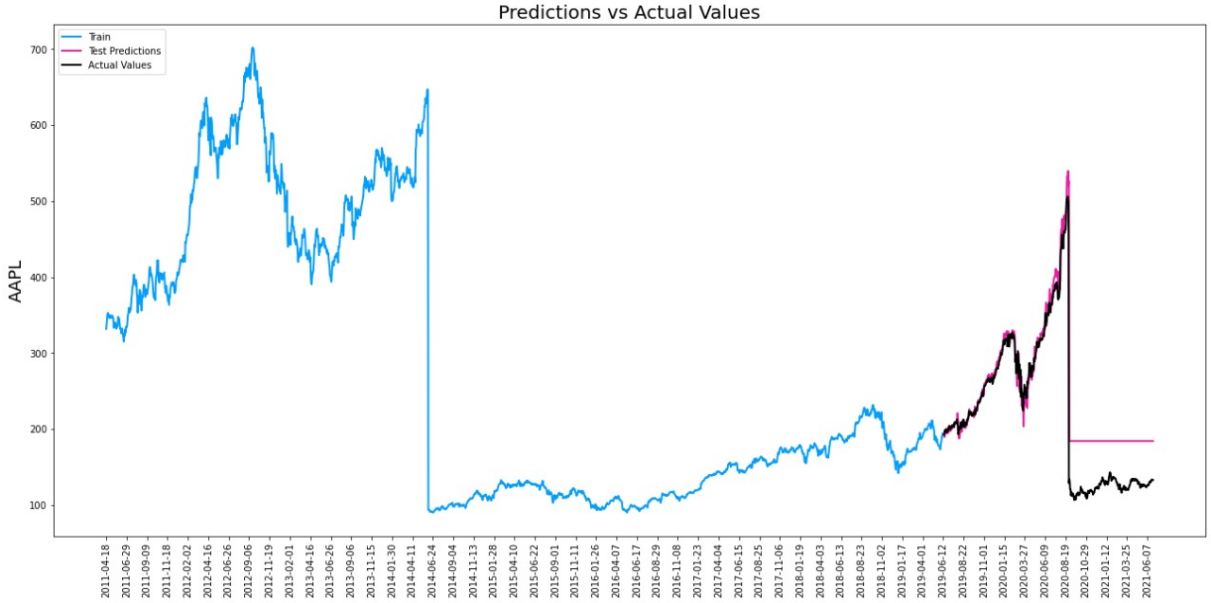
The above mentioned data is appropriately scaled and split as train and test data in an 80:20 ratio. Now we train an LSTM model with the train data. On making predictions for the test data the following graph was obtained.



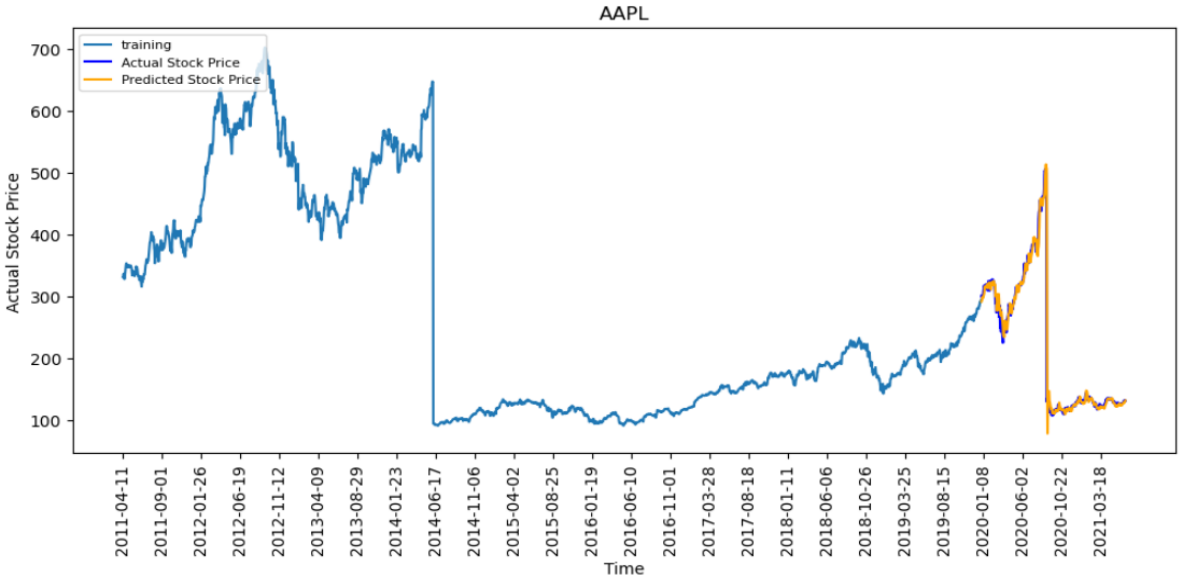
Method 2: **RNN with clustering**

In this method we trained our model on the target stock and then implemented this model on a list of stocks. We extracted features from the last layer of RNN for these stocks and clustered them based on these features. Finally, we trained our model on the cluster containing our target stock and got this result-

$TSLA$ and EA were clustered in the same stock as $AAPL$.



Method 3: ARIMA model In this method we trained our model on the target stock data for the past 10 years. It uses data upto the previous day to make predictions.



Error	Method 1	Method2	Method 3
Mean Absolute Percentage Error (MAPE)	24.31 %	21.58 %	3.86 %
Median Absolute Percentage Error (MDAPE)	11.6 %	4.44 %	2.04%

6 Conclusion

As we can see from the above comparisons, that ARIMA model provides better predictions with a better accuracy as compared to the LSTM Model. Also, ARIMA is relatively fast as compared to LSTM in terms of training/fitting time and complexity.

References

[1] Alan Akbik et al. "FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP". In: (June 2019), pp. 54–59. DOI: 10.18653/v1/N19-4010. URL: https://aclanthology.org/N19-4010.