CS464 – Introduction to Machine Learning

Project Final Report

Stock Price Prediction Using Twests and Historical Data

Y.Kaan Baycan	22103383
Kimya Ghasemlou	21801412
Uğur Sorar	22102408
Yusuf Doğan	21702970
Deniz Aydemir	22001859

Table Of Contents

1.Introduction

In a dynamic and ever-evolving financial world, forecasting stock market trends is a highly interesting and challenging issue. The advent of social media has changed the financial research landscape, leading to changes in important results: investor sentiment. Share prices can be greatly affected by public opinion and reaction, especially on platforms such as Twitter. This work delves into the challenging task of harnessing the power of sentiment analysis derived from Twitter data using the traditional method of analyzing historical stock data
Our approach includes a combination of sophisticated natural language processing (NLP) techniques to interpret and quantify sentiment from Twitter feeds, and time-series analysis to understand patterns in historical stock data With that combining these two disparate data sources presents a unique challenge: qualitative of social media content, often To bridge the gap between subjective in nature, and quantitative, objective historical stock data.

2.Problem Description

The main challenge in this field is accurate forecasting of stock price movements by combining insights from two different data sources: historical trends in stock prices and sensitivity revealed on social media. The work seeks to reveal the complex interaction between public sentiment and stock price volatility as expressed on Twitter. The complexity is further increased by the nature of the data – structured, statistical stock market data pitted against unstructured data, text from Twitter.
Our effort includes not only the technical aspects of data preprocessing, model training, and evaluation but also in-depth theoretical analysis of how sentiment influences market behavior. We think there is a substantial relationship between public sentiment and stock market performance, and our goal is to capture this relationship quantitatively. The successful execution of this project promises not only to provide valuable insights into stock forecasting rather it will contribute to a broader understanding of how social media can affect financial markets. The main questions we are trying to answer in this work are: How good are the models cited in the forecasts and which ones perform better? whether sentiment actually affects stock prices; If yes, how hard is it? What methods work best to analyze a data set? What kind of unexpected result will we have from the model results?

3.Methods

3.1 Dataset

We started the explanatory data analyses by importing two data sets ‘Tweet.csv’ and ‘Company_Tweet.csv’[1]. The Tweet.csv data set contains the 6 columns such as tweet_id, writer, body etc. The Company_Tweet data set contains 2 columns which are tweet_id and ticker_symbol which indicates to what company that tweet is related to. We merged these two data sets based on tweet_id to create the ‘tweets’ data frame. This new data frame also contains the ticker_symbol column to relate the tweets to companies and it has 1425013 rows in total as tweet entries. These tweets are all between January 1st 2015 and December 31st 2019.

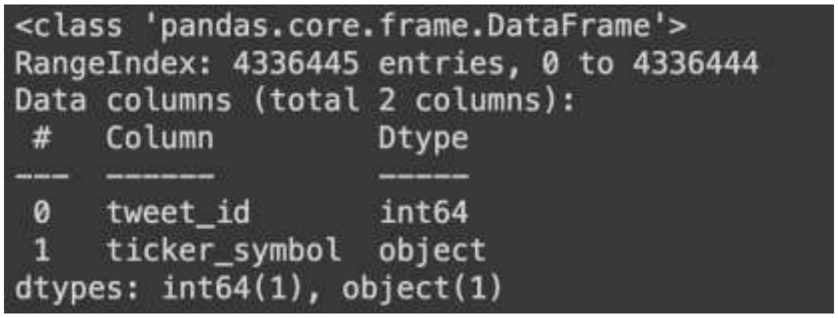

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4336445 entries, 0 to 4336444
Data columns (total 2 columns):
 #   Column         Dtype
---  ------         -----
 0   tweet_id       int64
 1   ticker_symbol  object
dtypes: int64(1), object(1)
```

Fig. 1 (Tweets Dataframe)

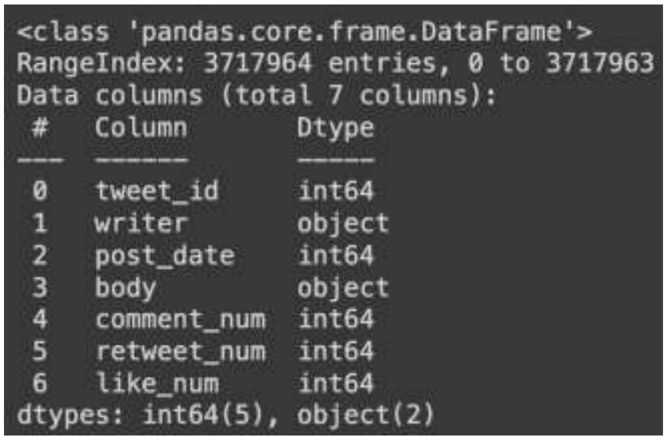

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3717964 entries, 0 to 3717963
Data columns (total 7 columns):
 #   Column       Dtype
---  ------       -----
 0   tweet_id     int64
 1   writer       object
 2   post_date    int64
 3   body         object
 4   comment_num  int64
 5   retweet_num  int64
 6   like_num     int64
dtypes: int64(5), object(2)
```

Fig. 2 (Company Tweets Dataframe)

This dataset includes tweets about five different companies: GOOGL, TSLA, MSFT, AAPL, AMZN. We have filtered AAPL tweets based on the ticker symbol to analyze separately. After filtering the tweets we have got 1425013 data. These tweets are all between January 1st 2015 and December 31st 2019.

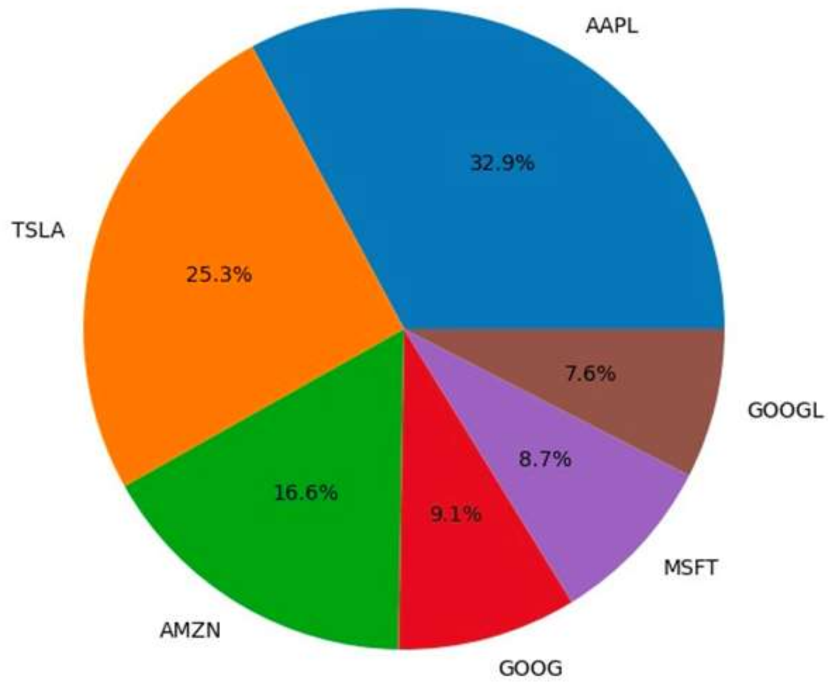
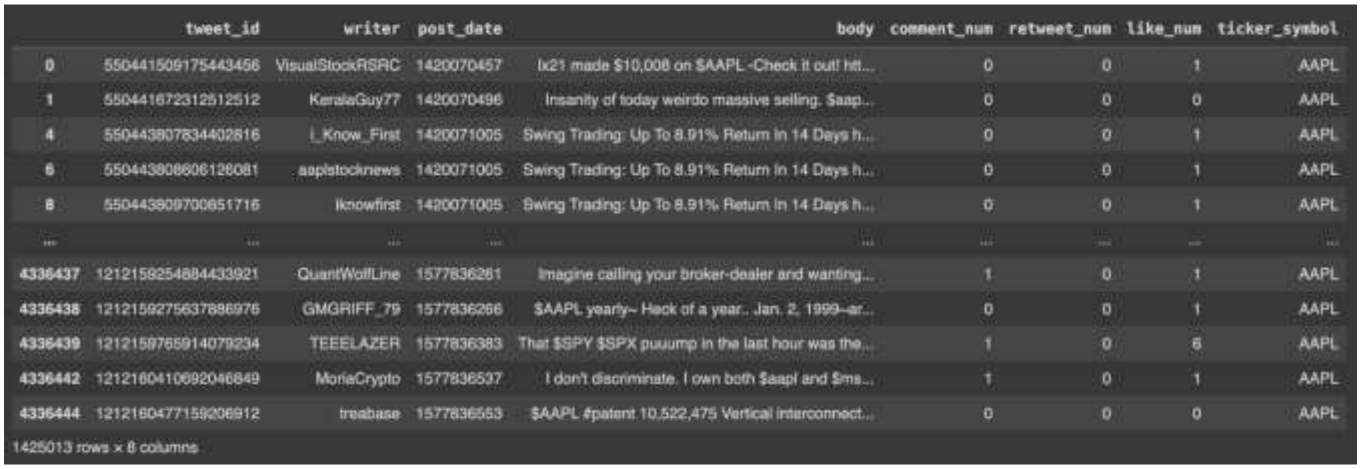


Fig. 3 (Company Distribution Pie Chart)

The resulting dataframe after merging two dataframes is shown in Fig. X.



	tweet_id	writer	post_date	body	comment_num	retweet_num	like_num	ticker_symbol
0	550441509175443456	VisualStockRSRC	1420070457	lx21 made $10,008 on $AAPL -Check it out! htt...	0	0	1	AAPL
1	550441672312512512	KeralaGuy77	1420070496	Insanity of today weirdo massive selling. $aap...	0	0	0	AAPL
4	550443807834402816	i_Know_First	1420071005	Swing Trading: Up To 8.91% Return In 14 Days h...	0	0	1	AAPL
6	550443808606126081	aaplstocknews	1420071005	Swing Trading: Up To 8.91% Return In 14 Days h...	0	0	1	AAPL
8	550443809700851716	iknowfirst	1420071005	Swing Trading: Up To 8.91% Return In 14 Days h...	0	0	1	AAPL
...	...	...	...	...	...	...	...	...
4336437	1212159254884433921	QuantWolfLine	1577836261	Imagine calling your broker-dealer and wanting...	1	0	1	AAPL
4336438	1212159275637886976	GMGRIFF_79	1577836266	$AAPL yearly~ Heck of a year.. Jan. 2, 1999~ar...	0	0	1	AAPL
4336439	1212159765914079234	TEEELAZER	1577836383	That $SPY $SPX puuump in the last hour was the...	1	0	6	AAPL
4336442	1212160410692046849	MoriaCrypto	1577836537	I don't discriminate. I own both $aapl and $ms...	1	0	1	AAPL
4336444	1212160477159206912	treabase	1577836553	$AAPL #patent 10,522,475 Vertical interconnect...	0	0	0	AAPL

1425013 rows × 8 columns

Fig. 4 (Combined Tweets and Company Tweets)

We splitted the data frame as 80% training data and 20% percent test data. We also divided the half of the test data set to use as validation data, currently the division is based on dates, as the earlier 80% training, mid 10% validation and latest 10% test. For validation, we will employ k-fold-cross-validation. We will compare the models based on their Root Mean Squared Error (RMSE) and R^2 values.

3.2 Preprocessing

3.2.1 TextBlob

TextBlob[2] is a python library built on top of NLTK(Natural Language Toolkit) and Pattern library. It provides a simplified process of working with NLP tasks. TextBlob is a pretrained model on labeled data and each text corresponds to a positive, negative or a natural sentiment. It gives polarity and subjectivity scores, polarity representing the tone of the text, and subjectivity representing the measure of subjectivity of the text. We have used the polarity scores. These sentiments are represented through values between [-1,1]. -1 representing the negative tone, 0 neutral and 1 positive tone. The values have been rounded to the closest integer and appended to the data frame shown in Fig. X to each tweet.

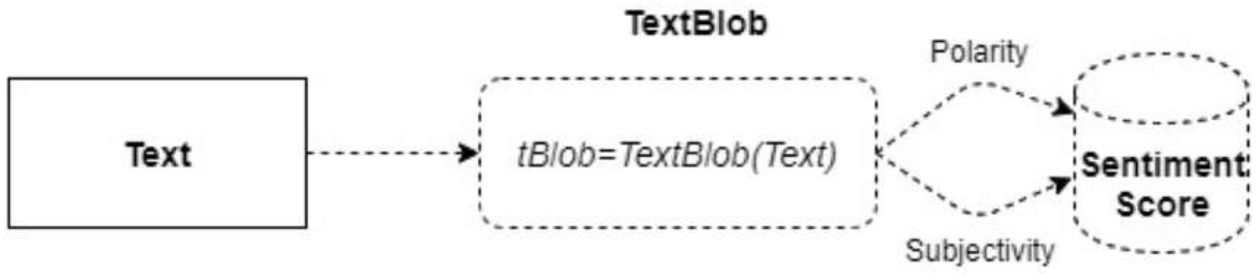
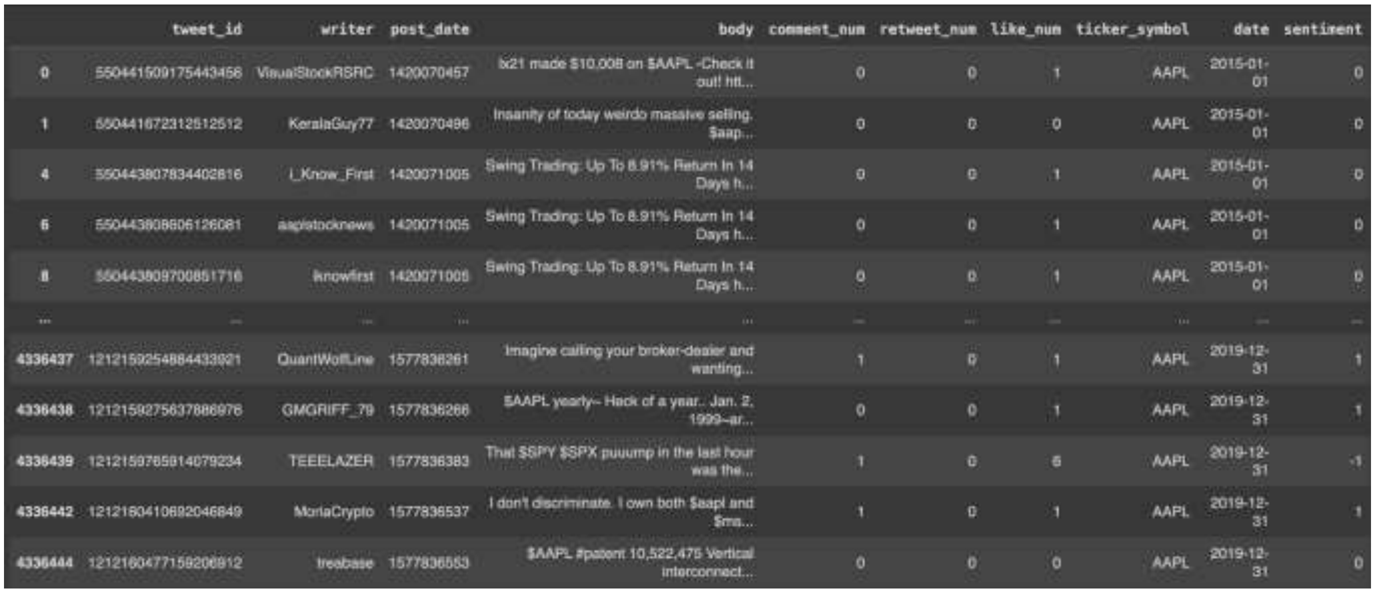


Fig. 5 (TextBlob)



	tweet_id	writer	post_date	body	comment_num	retweet_num	like_num	ticker_symbol	date	sentiment
0	550441509175443456	VisualStockRSRC	1420070457	lx21 made $10,008 on $AAPL -Check it out! htt...	0	0	1	AAPL	2015-01-01	0
1	550441672312512512	KeralaGuy77	1420070496	Insanity of today weirdo massive selling. $aap...	0	0	0	AAPL	2015-01-01	0
4	550443807834402816	i_Know_First	1420071005	Swing Trading: Up To 8.91% Return In 14 Days h...	0	0	1	AAPL	2015-01-01	0
6	550443808606126081	aaplstocknews	1420071005	Swing Trading: Up To 8.91% Return In 14 Days h...	0	0	1	AAPL	2015-01-01	0
8	550443809700851716	iknowfirst	1420071005	Swing Trading: Up To 8.91% Return In 14 Days h...	0	0	1	AAPL	2015-01-01	0
...	...	...	...	...	...	...	...	...	...	...
4336437	1212159254884433921	QuantWolfLine	1577836261	Imagine calling your broker-dealer and wanting...	1	0	1	AAPL	2019-12-31	1
4336438	1212159275637886976	GMGRIFF_79	1577836266	$AAPL yearly-- Heck of a year.. Jan. 2, 1999--ar...	0	0	1	AAPL	2019-12-31	1
4336439	1212159765914079234	TEEELAZER	1577836383	That $SPY $SPX puuump in the last hour was the...	1	0	6	AAPL	2019-12-31	-1
4336442	1212160410692046849	MoriaCrypto	1577836537	I don't discriminate. I own both $aapl and $ms...	1	0	1	AAPL	2019-12-31	1
4336444	1212160477159206912	treabase	1577836553	$AAPL #patent 10,522,475 Vertical interconnect...	0	0	0	AAPL	2019-12-31	0

Fig. 6 (After Appending Sentiment score)

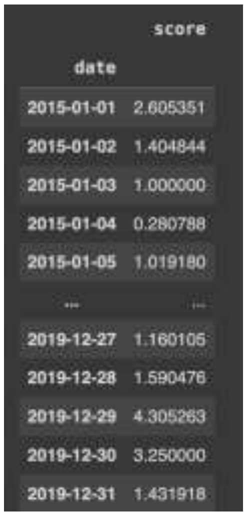
After obtaining the sentiment score, comment, retweet and like count has been used to calculate an interaction score and it has been combined with the sentiment score.

Interaction_score = like_num + comment_num + retweet_num

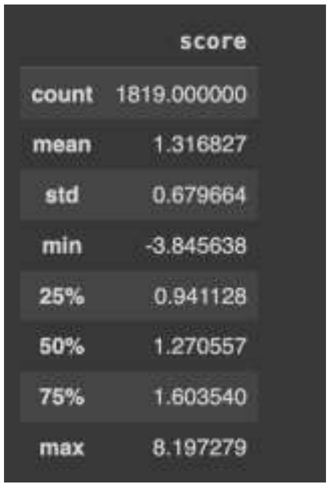
Score = sentiment + sentiment*log(interaction_score+1)

The score has been normalized by the formula below:

Score = (sentiment - mean) / std



	score
date	
2015-01-01	2.605351
2015-01-02	1.404844
2015-01-03	1.000000
2015-01-04	0.280788
2015-01-05	1.019180
...	...
2019-12-27	1.160105
2019-12-28	1.590476
2019-12-29	4.305263
2019-12-30	3.250000
2019-12-31	1.431918



	score
count	1819.000000
mean	1.316827
std	0.679664
min	-3.845638
25%	0.941128
50%	1.270557
75%	1.603540
max	8.197279

Fig. 7 (Final Score for each Day) *Fig. 8 (Dataframe Info after Normalization)*

This score gives us the final score for each tweet based on interaction and sentiment analysis, then based on tweet dates the average score for each company in a day is calculated to be used in the next steps.

3.2.2 Bert

BERT stands for Bidirectional Encoder Representations from Transformers and is a model used for sentiment analysis in NLP tasks. It is trained on 2.500M words from Wikipedia and 800M words from BooksCorpus. We have used the pretrained "distilbert-base-uncased" which is 2-layer, 768-hidden, 12-heads, 110M parameter neural network architecture.[3] The model is trained on general data thus the best approach would be to fine-tune the model, however since the tweets dataset is an unlabeled dataset, this task was not possible to perform, and we have proceeded using the original pretrained model.

The result of sentiment analysis was a value between [0,1], 0 representing the negative, 0.5 neutral and 1 positive.

After applying the sentiment analysis on tweets data it is observed that the result was a huge tendency towards labeling them as positive. Fig. X is a bar chart representing the positive and negative count for tweets with maximum daily interaction score.

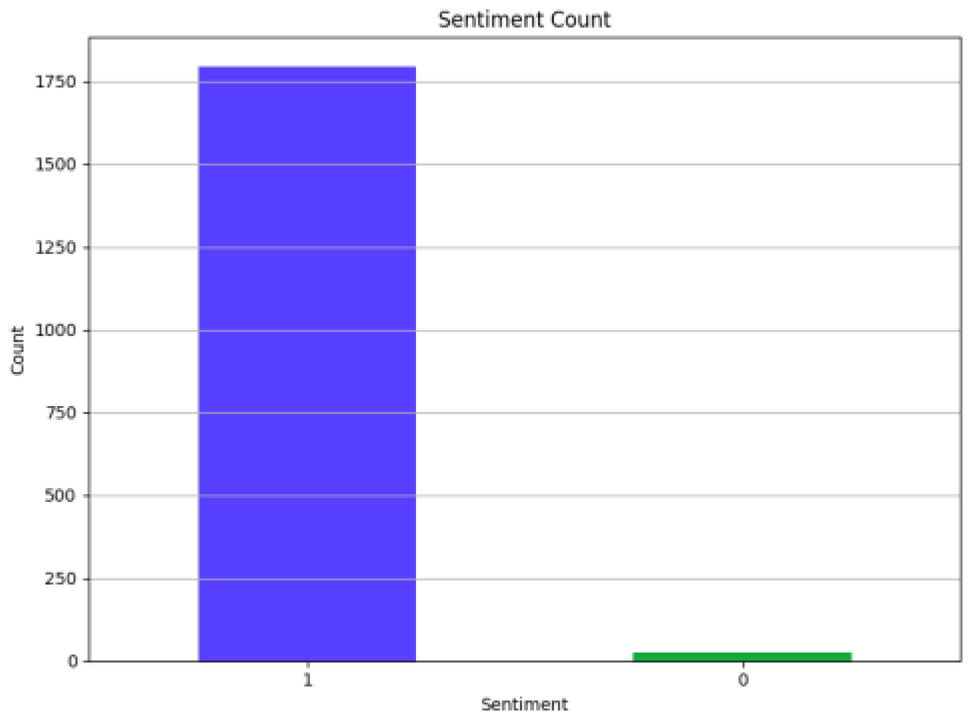
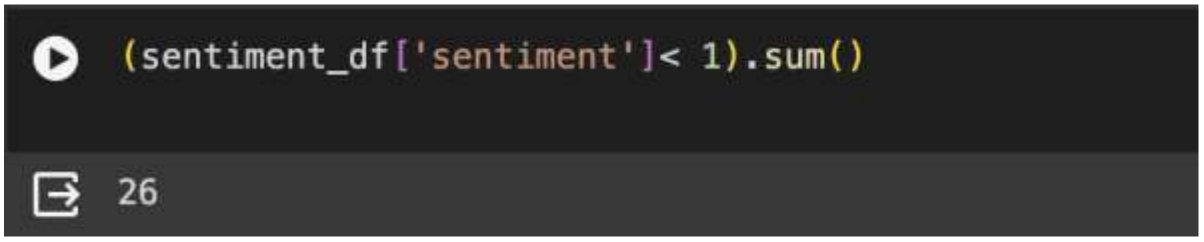


Fig. 9 (Bert Bar Chart for Daily Max Interaction Score Tweets)

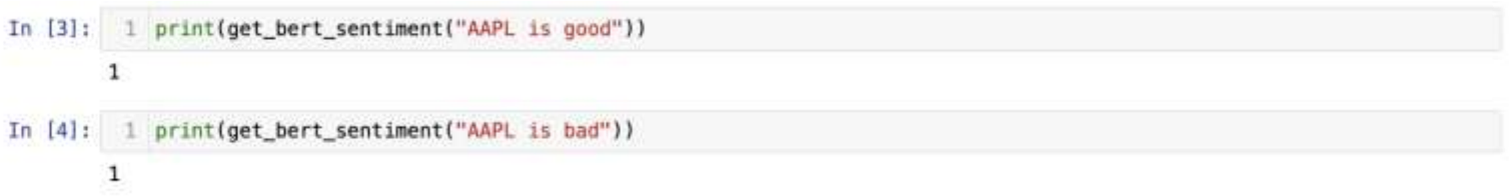


```
(sentiment_df['sentiment']< 1).sum()
```

```
26
```

Fig. 10 (Bert Daily Max Interaction Score Tweets Positive Count)

There was a possibility that the tweets data originally contained higher amounts of positive than negative ones but after experimenting it has been seen that the model was biased toward being positive.



```
In [3]: print(get_bert_sentiment("AAPL is good"))
1

In [4]: print(get_bert_sentiment("AAPL is bad"))
1
```

Fig. 11 (Bert Bias Test)

After comparing it with TextBlob we have decided to continue with scores obtained from TextBlob. Fig. X shows the bar chart for values obtained in the tweet sentiment analysis using TextBlob step, the values were between [-1,1] and has been converted to [0,1] for comparison.

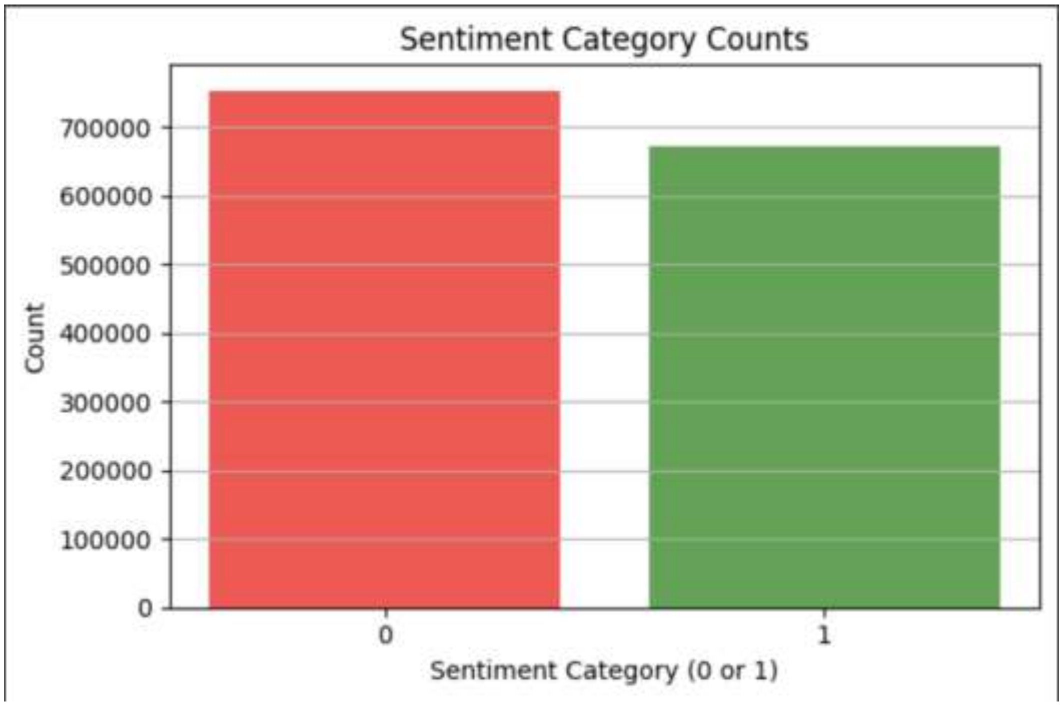
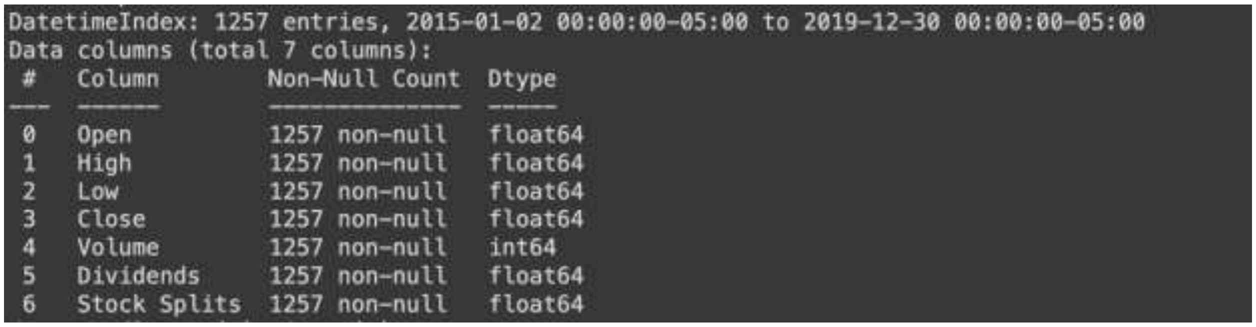


Fig. 12 (TextBlob Bar Chart)

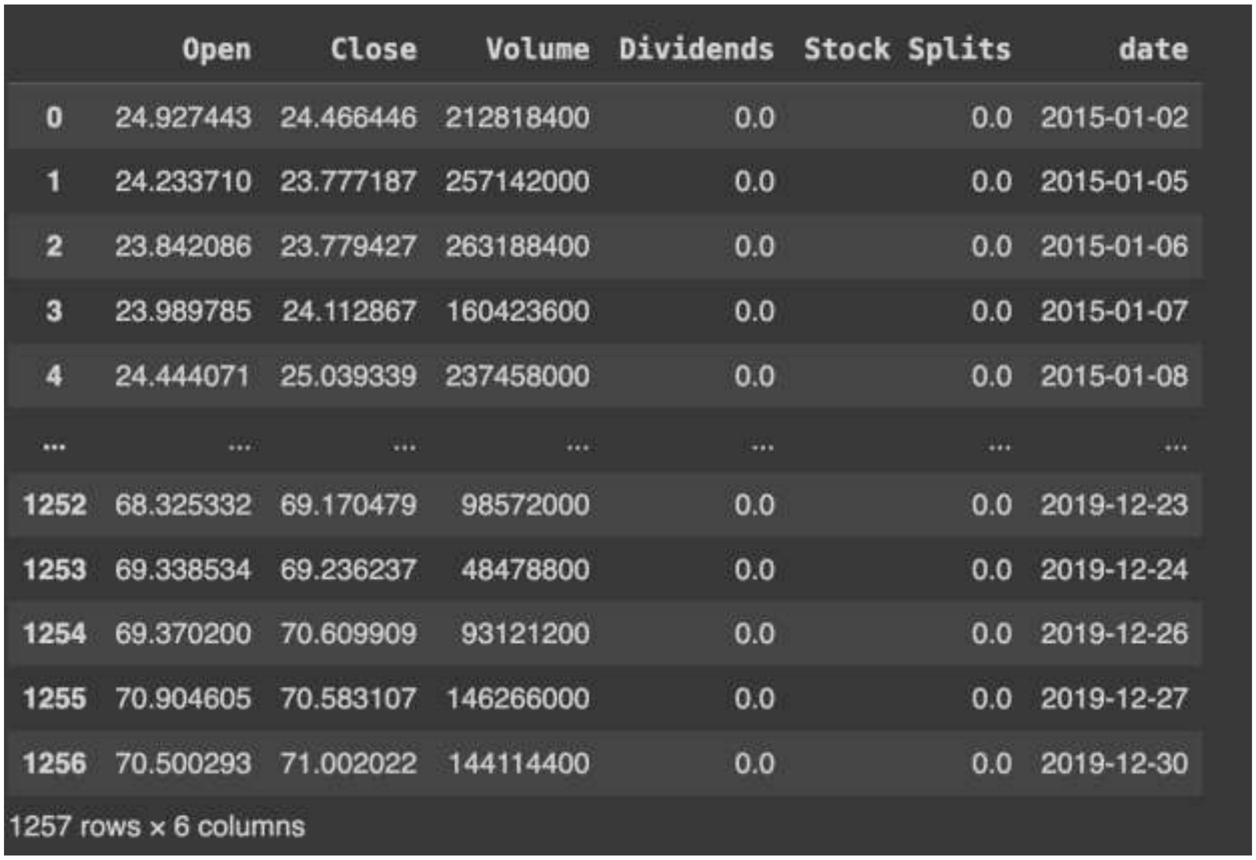
3.2.3 Yfinance

After obtaining the tweets data frame, we used the yfinance[4] library to scrape the stock prices information between January 2015 and December 2019 of AAPL that will be analyzed. We altered the date type of the stock prices data frame to be the same as the sentiments data frame. We also added one day to the dates of the sentiment data frame in order to use the sentiment scores of the previous day in forecasting current day’s stock price.



```
DatetimeIndex: 1257 entries, 2015-01-02 00:00:00-05:00 to 2019-12-30 00:00:00-05:00
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Open          1257 non-null   float64
 1   High          1257 non-null   float64
 2   Low           1257 non-null   float64
 3   Close         1257 non-null   float64
 4   Volume        1257 non-null   int64  
 5   Dividends     1257 non-null   float64
 6   Stock Splits  1257 non-null   float64
```

Fig. 13 (yfinance Dataframe)

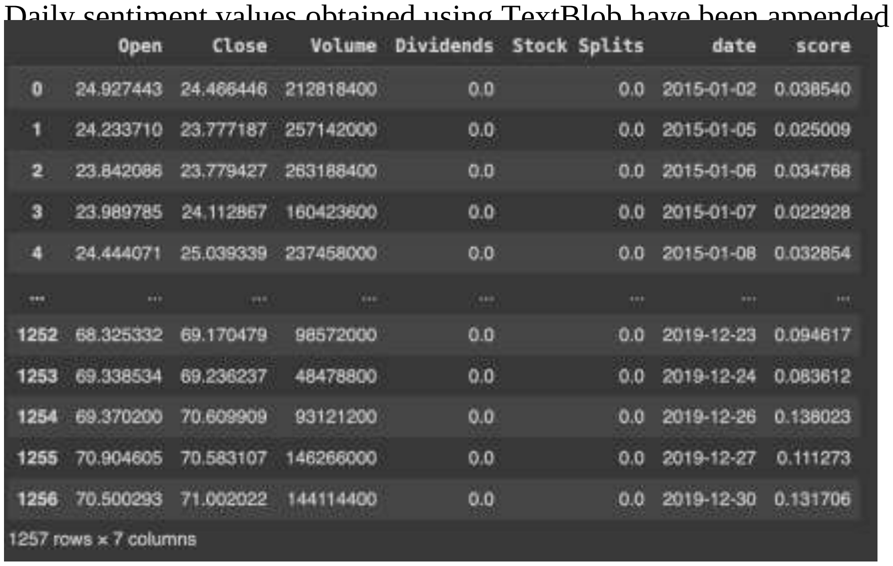


	Open	Close	Volume	Dividends	Stock Splits	date
0	24.927443	24.466446	212818400	0.0	0.0	2015-01-02
1	24.233710	23.777187	257142000	0.0	0.0	2015-01-05
2	23.842086	23.779427	263188400	0.0	0.0	2015-01-06
3	23.989785	24.112867	160423600	0.0	0.0	2015-01-07
4	24.444071	25.039339	237458000	0.0	0.0	2015-01-08
...	...	...	...	...	...	...
1252	68.325332	69.170479	98572000	0.0	0.0	2019-12-23
1253	69.338534	69.236237	48478800	0.0	0.0	2019-12-24
1254	69.370200	70.609909	93121200	0.0	0.0	2019-12-26
1255	70.904605	70.583107	146266000	0.0	0.0	2019-12-27
1256	70.500293	71.002022	144114400	0.0	0.0	2019-12-30

1257 rows × 6 columns

Fig. 14 (yfinance Data)

Daily sentiment values obtained using TextBlob have been appended to the dataframe.



	Open	Close	Volume	Dividends	Stock Splits	date	score
0	24.927443	24.466446	212818400	0.0	0.0	2015-01-02	0.038540
1	24.233710	23.777187	257142000	0.0	0.0	2015-01-05	0.025009
2	23.842086	23.779427	263188400	0.0	0.0	2015-01-06	0.034768
3	23.989785	24.112867	160423600	0.0	0.0	2015-01-07	0.022928
4	24.444071	25.039339	237458000	0.0	0.0	2015-01-08	0.032854
...	...	...	...	...	...	...	...
1252	68.325332	69.170479	98572000	0.0	0.0	2019-12-23	0.094617
1253	69.338534	69.236237	48478800	0.0	0.0	2019-12-24	0.083612
1254	69.370200	70.609909	93121200	0.0	0.0	2019-12-26	0.138023
1255	70.904605	70.583107	146266000	0.0	0.0	2019-12-27	0.111273
1256	70.500293	71.002022	144114400	0.0	0.0	2019-12-30	0.131706

1257 rows × 7 columns

Fig. 15 (yfinance Combined with Sentiment Score)

3.3 Neural Networks

In our study, we employed both Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) neural network architectures for a time series forecasting task enriched with sentiment-related data. Both LSTM and GRU models are types of recurrent neural networks (RNNs) capable of capturing long-range dependencies in sequential data, making them suitable for our predictive analysis.

LSTM is a type of RNN architecture designed to handle the shortcomings of traditional RNNs in capturing long-range dependencies. It utilizes a gating mechanism with input, forget, and output gates, allowing it to selectively retain or forget information over varying time intervals. On the other hand, GRU is another variant of RNNs similar to LSTM but with a simplified structure. It combines the memory unit and output gate into a single gate, enabling it to control the flow of information more efficiently with fewer parameters compared to LSTM.

Both the LSTM and GRU models we constructed consisted of five layers of the respective recurrent units followed by Dense layers for predicting future values in a time series dataset. The sequential model architectures were similar, with multiple LSTM or GRU layers, each containing 20 units and configured to return sequences, facilitating information flow between successive layers. The final LSTM or GRU layer was connected to a Dense layer with a single unit, aimed at forecasting a single continuous value. The depth of both LSTM and GRU models, with a series of recurrent layers, was chosen to enable the extraction of intricate temporal patterns from the input sequences. However, we acknowledge the potential risk of overfitting due to the complex architecture, especially with limited training data.

We opted for the Adam optimizer with a learning rate of 1e-3 due to its adaptive nature, beneficial for both LSTM and GRU architectures. Adam's adaptability often leads to faster convergence and better performance across various problems.We used Mean Squared Error (MSE) as our loss function, which aligns with the regression task for both LSTM and GRU models. Our choice of a batch size of 32 and training for 20 epochs remains consistent for both LSTM and GRU models to balance computational efficiency and model generalization.

We standardized the input data to a similar range using a scaler previously fitted on the training data, aiding the model's convergence; then, generated test sequences and corresponding output values by constructing input sequences and output values from the dataset. Our detailed findings and comparisons between LSTM and GRU models are explained in the “Results” chapter.

3.4 Ensemble Methods

3.4.1 XGBoost

In the initial implementation of the XGBoost algorithm, we endeavored to forecast stock closing prices utilizing parameters: opening price, volume, dividends, stock split and score; the tree algorithm was unable to capture the nuances of sequential variations in closing price trends. When we did the predictions for the whole data set, we saw that the model was overfitted to its training data and could not catch the sequential changes in the test data. We saw that for the test data the predictions were stable and did not reflect any volatility on the stock price values.

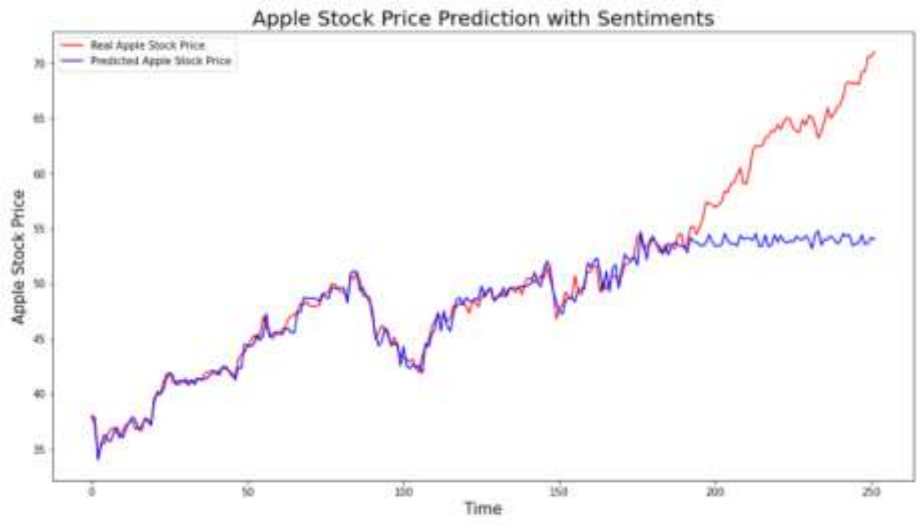


Fig 16. XgBoost Model with Close Target

To ensure that the algorithm catches the sequential relation, we modified our training method. In our modified approach, the model was trained to predict not the closing prices but the stock price changes for consecutive days. The training inputs remained the same. After obtaining the closing price difference predictions, to obtain the closing price predictions we implied the following algorithm: for the first day in the test data the closing price prediction was equal to the closing price itself. For the upcoming days the closing prices were predicted as the closing price difference prediction for the current day added to the closing price prediction for the previous day. With this prediction algorithm we obtained the following prediction on the validation data:

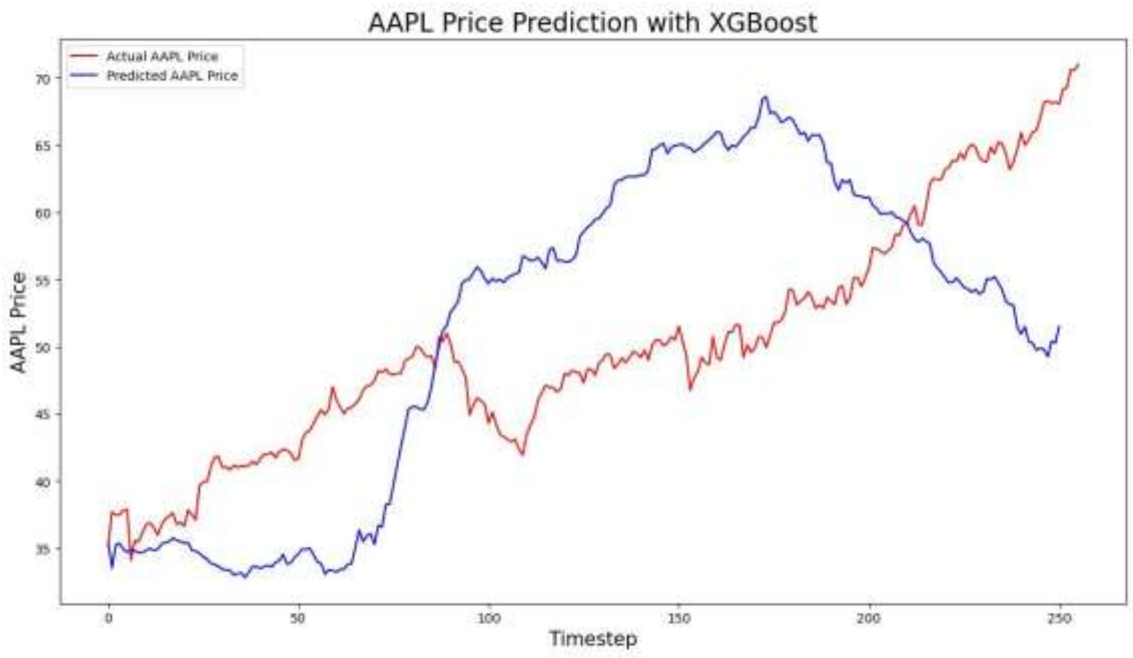


Fig 17 Intial XgBoost Model with Price Difference Target

The prediction results were not accurate. Although we obtained our aim to see a sequential pattern on the predictions, the models predictions for the closing prices were highly inaccurate. Also it was observed that models direction predictions were also off. One of the main reasons was once the model's prediction and the actual closing price value were far from each other automatically the upcoming predictions were becoming inaccurate since the predictions were recursively following each other. To fix this issue we decided to update our prediction algorithm once again. The predictions were now made by adding the closing price difference prediction of the tree algorithm to the actual closing price of the previous day. When we applied this new strategy the model's predictions became highly over-fitted to the data and results were not a realistic reflection of the models actual performance. Since the closing price differences were actually very small for most of the days, the tree algorithm was making almost perfect predictions of the actual stock prices.

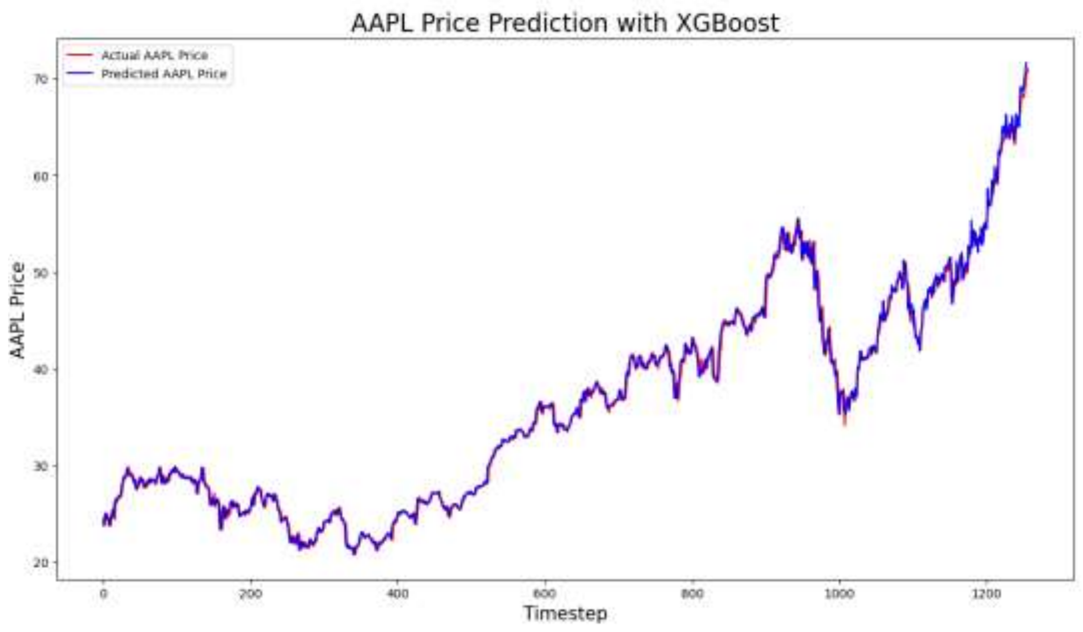


Fig 18 Overfitting XgBoost Model

After failing once again, we changed our testing method. After training the tree model with the train data we tested the model on the whole data set. With this application, although the predictions were over-fitted for the training data we also obtained prediction results with better accuracy for the test data and overall better direction accuracy for the test data.

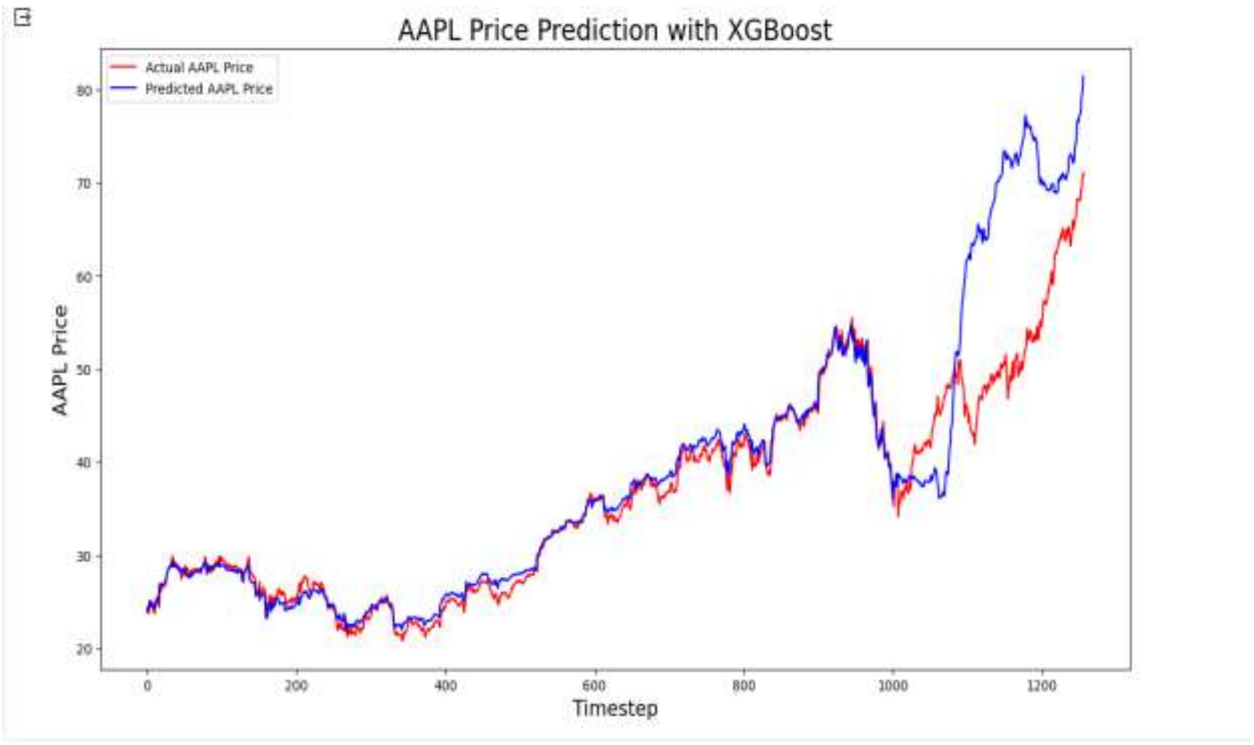


Fig 19. XgBoost Performance vs Real Data

After deciding that this method gave the best observed accuracy scores for the XGBoost model, we performed a grid to fine-tune the hyperparameters. The following parameters were included in the search space:

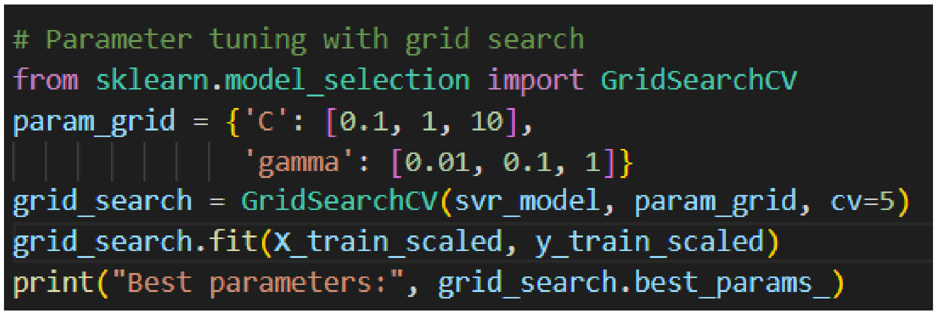
- Number of estimators: {1000, 1500}
- Maximum depth: {3, 5}
- Learning rate: {0.1, 0.01, 0.001}
- Subsample: {0.7, 0.9}, which is the fraction of samples to be used for fitting each individual base learner.
- Column sample by tree: {0.7, 0.9}, which is the fraction of features to be used for each tree.

After the grid search observed optimal hyperparameters were 1000 estimators, 3 for maximum depth, 0.1 as learning rate and 0.7 for subsample and column sample by tree.

3.5 SVR

Support Vector Regression (SVR) is recognized for its proficiency in regression tasks, effectively handling both linear and nonlinear models. Motivated by SVR's notable performance in similar applications, we preferred an SVR approach after failing on ensemble methods for our project on predicting AAPL stock prices, utilizing a dataset spanning 1,252 days. Our preprocessing involved generating lagged features of the target column “Close” and standardizing the data, a crucial step given SVR's sensitivity to input scale. This process aimed to ensure data uniformity, aligning the stock price information for effective regression analysis. In a novel approach, we integrated historical stock prices with sentiment analysis data from Twitter, hypothesizing that this combination could provide a more comprehensive understanding of market dynamics.

We opted for the RBF (radial basis function) kernel in our SVR model and embarked on hyperparameter tuning using a grid search, specifically targeting the 'C' and 'gamma' parameters. The grid search, conducted through sklearn, identified the best parameters as C=10 and gamma=0.01, indicating a preference for a narrow margin.



```
# Parameter tuning with grid search
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [0.1, 1, 10],
              'gamma': [0.01, 0.1, 1]}
grid_search = GridSearchCV(svr_model, param_grid, cv=5)
grid_search.fit(X_train_scaled, y_train_scaled)
print("Best parameters:", grid_search.best_params_)
```

Fig 20. Grid Search Code Snippet

Our evaluation of the SVR model spanned two scenarios: with and without the inclusion of sentiment data. We employed various metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and accuracy in predicting price movement directions. Intriguingly, the inclusion of sentiment analysis led to a higher MSE, suggesting that while sentiments offer valuable insights, they may also introduce additional complexity and noise.

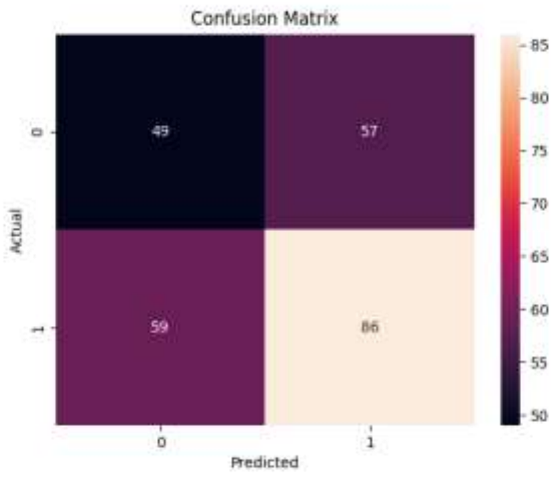


Fig 21. Confusion Matrix

To provide a clear and comparative perspective, we visualized the actual versus predicted closing prices under both scenarios. Additionally, we conducted a detailed assessment of the model's precision, recall, F-score, and support using precision_recall_fscore_support from sklearn, complemented by a confusion matrix visualization. This comprehensive analysis not only highlighted the model's predictive capabilities but also shed light on its effectiveness in accurately discerning upward and downward trends in stock prices. Such insights are pivotal in understanding the intricate interplay between quantitative stock data and qualitative sentiment indicators in financial market predictions.

4.Coding Environment Summary

Our coding environment predominantly comprises Jupyter Notebook[5] and Google Colab[6]. These platforms facilitate collaborative coding endeavors and serve as our primary tools for code sharing. As of our current workflow, we have adopted Python version 3.10.12. Within this framework, we leverage several libraries to augment our tasks. Notably, we employ Matplotlib for the visualization of graphs and plots, Pandas for robust data manipulation and analysis, and NumPy for comprehensive numeric operations. Regarding specific models, our toolkit incorporates TextBlob for Natural Language Processing (NLP) and sentiment analysis, TensorFlow for implementing Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, CatBoostRegressor for CatBoost algorithm, and xgb for XGBoost implementation and after progress meeting we added one more model SVR (support vector regressor). We used the SVR algorithm of the “sklearn” library. For accessing historical and current financial data such as stock prices or dividends, we rely on the Yahoo Finance API [4].

5.Evaluation of the Models

In our stock price prediction project, we employed a comprehensive set of evaluation metrics and carefully considered data splitting strategies to ensure the robustness and relevance of our model assessments. The primary metrics we used were Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Accuracy.
The formula for MAE, which measures the average magnitude of errors in a set of predictions, without considering their direction, is:

$$MAE = \frac{1}{n}\sum|y_i - \hat{y}_i|$$

Where $y_i$is the true value while $\hat{y}_i$is the predicted value.

MSE, which provides insight into the variance of the predictions and the magnitude of the errors, is calculated as:

$$MSE = \frac{1}{n}\sum(y_i - \hat{y}_i)^2$$

RMSE, an extension of MSE, gives the error magnitude on the same scale as the data, calculated by taking the square root of MSE.
Accuracy, which measures the proportion of correct predictions (particularly for directionality in stock prices), is given by:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

For parameter tuning and model selection, we used a grid search approach with 5-fold cross-validation (cv=5). Notably, we did not shuffle the data during cross-validation, acknowledging the importance of the sequential order in time series data. Shuffling could have disrupted the chronological sequence, leading to unrealistic training and potentially misleading outcomes. By preserving the time order, we ensured that our evaluation accurately mirrored real-world, time-sensitive conditions in the financial market, thus aiding us in pinpointing the most effective parameters and models for stock price prediction.

6.Results

6.1 Neural Networks

6.1.1 LSTM

The Long Short-Term Memory (LSTM) model's performance analysis, considering different window sizes and the integration of sentiment analysis, reveals intricate patterns in predicting AAPL closing prices using historical stock data spanning 1,252 days.

When incorporating a larger window size of 20 along with sentiments, the LSTM model exhibited a Mean Squared Error (MSE) of 16.30 and an accuracy of 56.0%. Moreover, without sentiments, the LSTM model exhibited a Mean Squared Error (MSE) of 16.06 and an accuracy of 57.6%. This larger window size aimed to encompass an extended historical context, providing a broader dataset for analysis. However, the higher MSE suggests that this model, despite capturing long-term trends, may have incorporated more irrelevant or noisy data, thus impacting its ability to accurately predict shorter-term trends. Additionally, the marginally reduced accuracy indicates that the incorporation of sentiments added complexity, leading to a slight decrease in prediction precision.

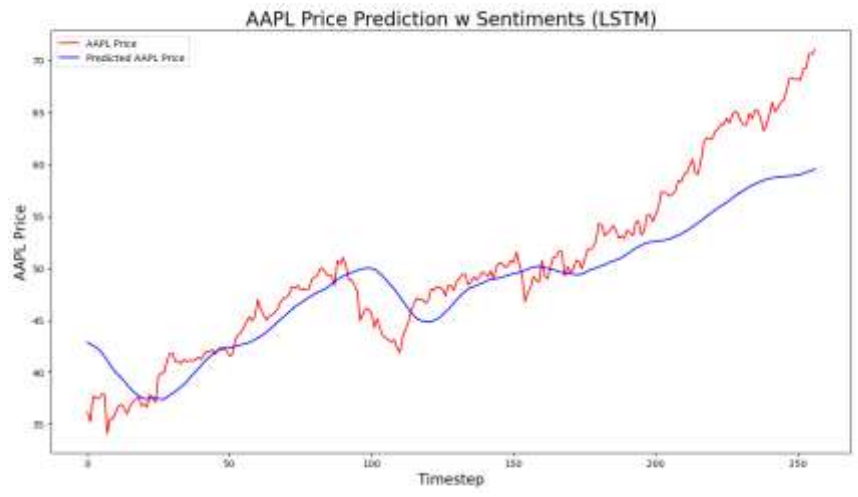


Figure 22: LSTM, window size is 20, with sentiments

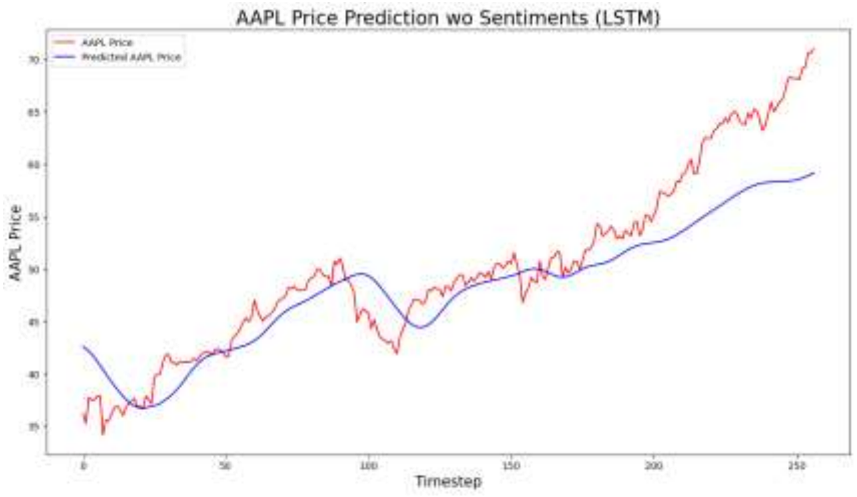


Figure 23: LSTM, window size is 20, without sentiments

Conversely, utilizing a smaller window size of 10 with sentiments showcased a lower MSE of 11.45 and a slightly higher accuracy of 54.8%. Moreover, without sentiments, the LSTM model exhibited a Mean Squared Error (MSE) of 10.85 and an accuracy of 54.5%. This smaller window size focused on more immediate historical patterns, potentially providing more relevant information for shorter-term predictions. Although it demonstrated better accuracy and a reduced MSE compared to the larger window size, the predictive capabilities might still have been affected by the complexities introduced by sentiment analysis.

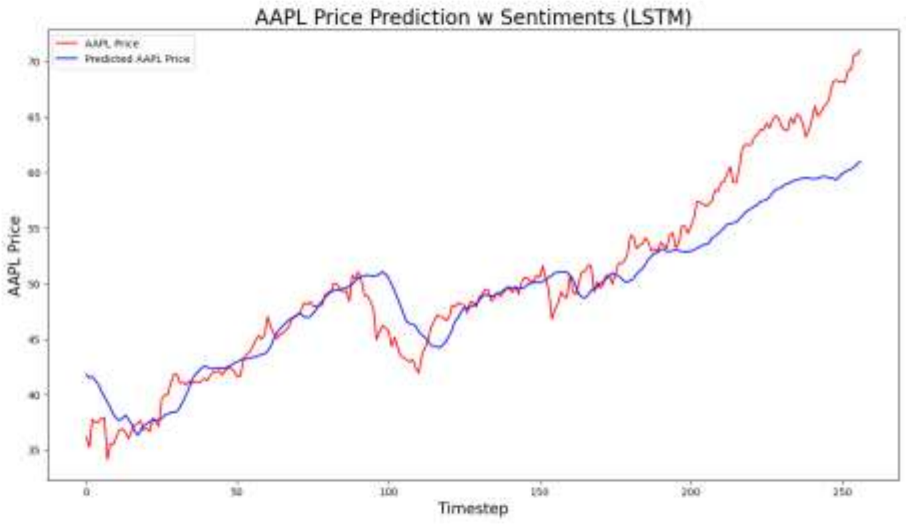


Figure 24: LSTM, window size is 10, with sentiments

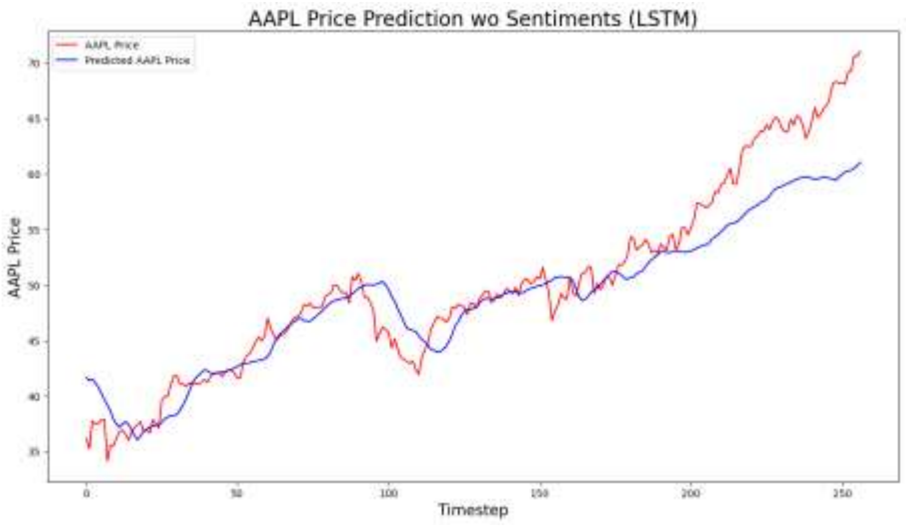


Figure 25: LSTM, window size is 10, without sentiments

6.1.2 GRU

Similarly, the Gated Recurrent Unit (GRU) model's performance analysis, considering different window sizes and sentiment integration, displayed distinctive outcomes. Utilizing a window size of 20 alongside sentiments resulted in an MSE of 2.23 and an accuracy of 50.4%. Moreover, utilizing a window size of 20 without sentiments resulted in an MSE of 1.70 and an accuracy of 50.9 %Despite the lower MSE, indicating a better fit to the data compared to the LSTM model with the same window size, the accuracy was notably lower. The larger window size attempted to capture long-term dependencies and trends in the data, thus achieving a lower MSE. However, the higher complexity introduced by sentiments might have affected its ability to make precise predictions, leading to reduced accuracy.

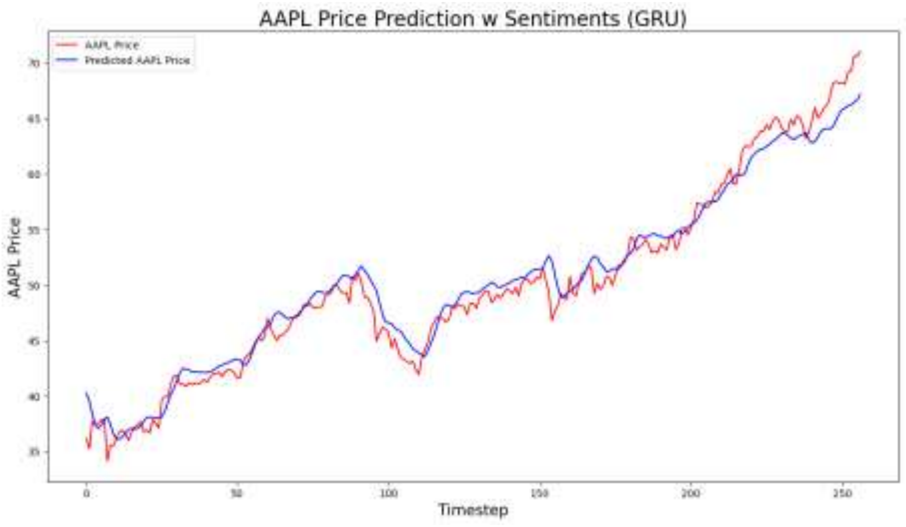


Figure 26: GRU, window size is 20, with sentiments

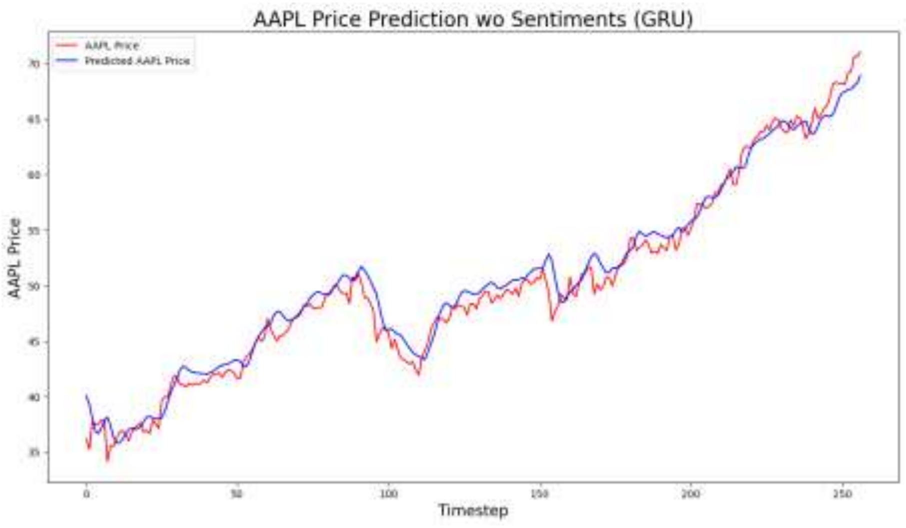


Figure 27: GRU, window size is 20, without sentiments

With a window size of 10 and sentiments, the GRU model demonstrated an MSE of 3.91 and an accuracy of 55.3%. Moreover, the GRU model demonstrated an MSE of 3.69 and an accuracy of 56.0% without sentiments. This smaller window size provided a focus on more immediate historical trends, potentially enhancing accuracy by emphasizing relevant information for shorter-term predictions. Despite a higher MSE compared to the 20-window model, the smaller window size managed to achieve slightly better accuracy, suggesting a trade-off between precision and capturing long-term dependencies.

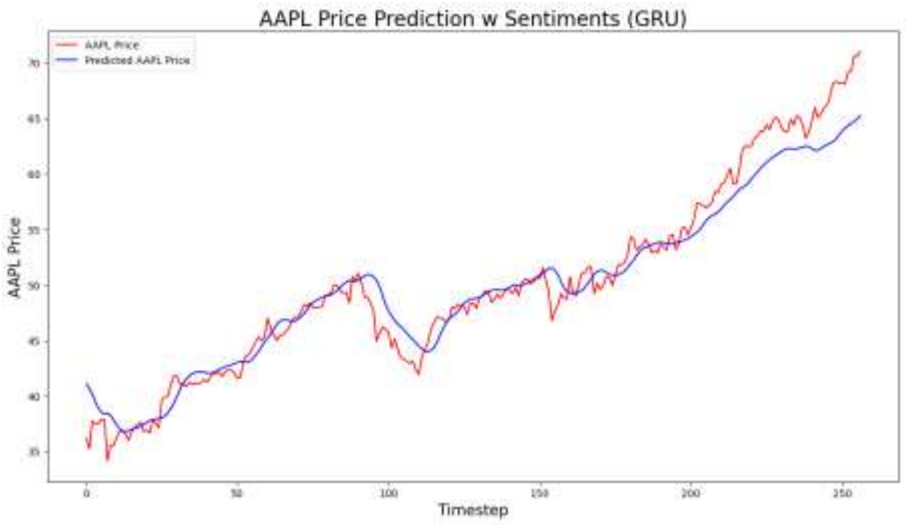


Figure 28: GRU, window size is 10, with sentiments

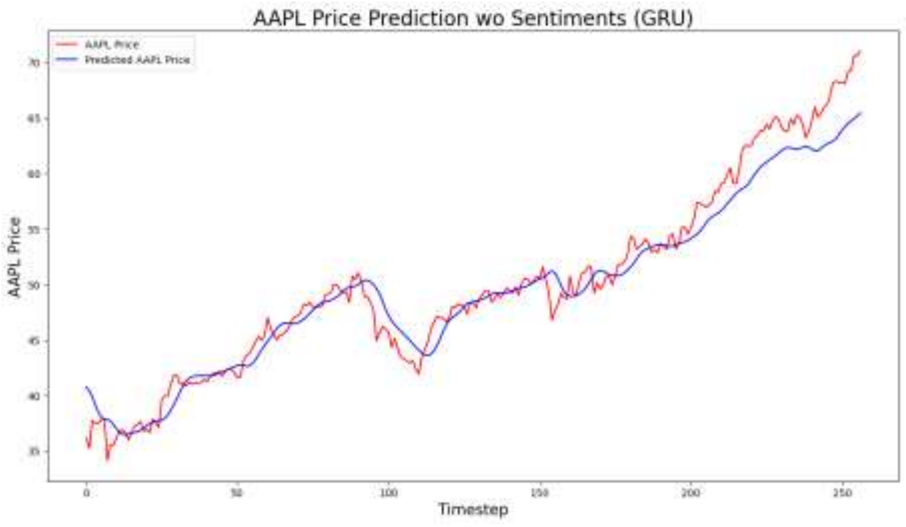


Figure 29: GRU, window size is 10, without sentiments

6.1.3 Comparison of Models with and without Sentiments

When assessing the LSTM model's performance, incorporating sentiments alongside historical data demonstrated nuanced differences. With a window size of 20, the LSTM model displayed an MSE of 16.30 and an accuracy of 56.0% when sentiments were included. Comparatively, without sentiments, utilizing the same window size yielded a lower MSE of 16.06 and a higher accuracy of 57.6%. This comparison implies that while sentiments contributed to a broader market sentiment perspective, their inclusion seemed to introduce additional unpredictability and volatility, slightly reducing the model's accuracy.

Similarly, the GRU model exhibited variations in performance concerning sentiments. With a window size of 20, integrating sentiments showed an MSE of 2.23 and an accuracy of 51.4%. In contrast, excluding sentiments resulted in a smaller MSE of 1.71 and a slightly higher accuracy of 51.0%. These findings suggest that sentiment analysis inclusion, although providing a wider market sentiment context, seemed to contribute to increased complexity and reduced accuracy in predicting stock prices.

The comparison between models with and without sentiments implies that while sentiments offered a broader market sentiment perspective, their incorporation introduced challenges related to volatility, unpredictability, and potentially increased model complexity. These findings underscore the intricate nature of integrating qualitative data such as sentiments into quantitative models for stock price prediction.

6.1.4 Overall Analysis

The integration of sentiment analysis alongside different window sizes in both the LSTM and GRU models revealed complex interplays between historical context, model complexity, and predictive accuracy. Larger window sizes aimed at capturing comprehensive historical contexts, allowing models to capture long-term trends but potentially introduced complexity and noise, impacting shorter-term prediction accuracy. Conversely, smaller window sizes focused on immediate historical trends, aiding in short-term predictions but potentially missing out on capturing long-term trends.

Incorporating sentiment analysis alongside traditional financial indicators introduced additional unpredictability and complexity in both models. While sentiment analysis provided a broader market sentiment perspective, it seemed to introduce noise, making predictions less accurate. These results underscore the intricate relationship between social sentiment and market dynamics, warranting further research and refinement to effectively integrate qualitative data into quantitative models for more accurate stock market predictions.

6.2 Ensemble Methods

6.2.1 XGBoost

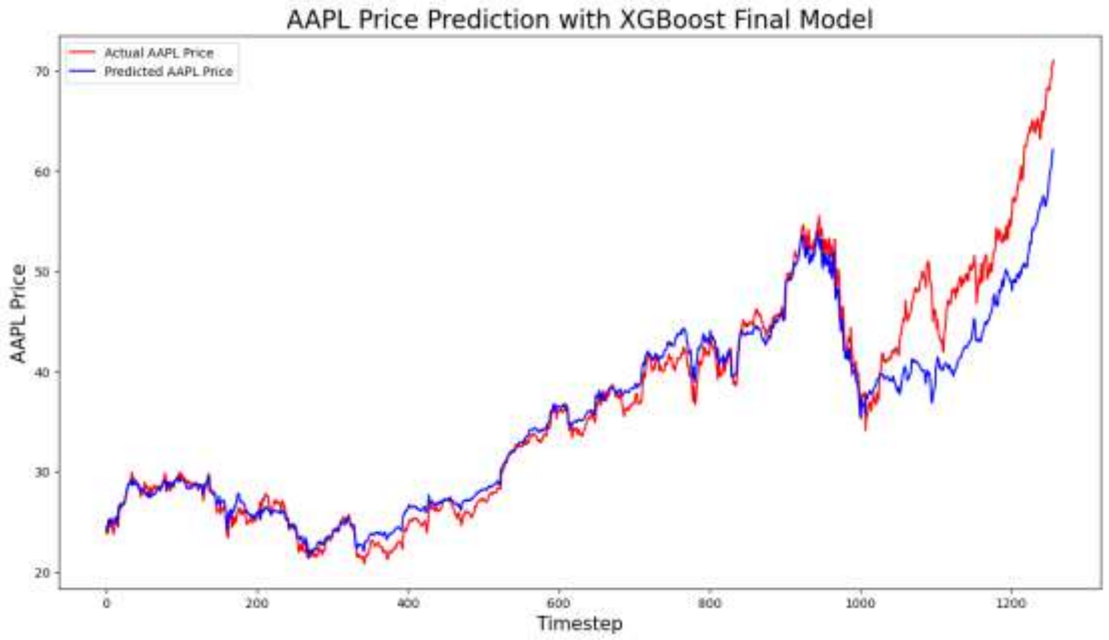


Figure 30: XgBoost Final Model

In assessing the predictive prowess of the finalized XGBoost model, we employed two statistical measures: the mean squared error (MSE) and the root mean squared error (RMSE). The model manifested an MSE of 10.58, signifying the highest mean of squared discrepancies between the predicted and actual values across all models we have developed thus far. The RMSE, offering a dimensionally consistent evaluation of the model's prediction error, was determined to be 3.25. A critical analysis of the XGBoost's output led us to recognize that tree-based algorithms exhibit certain limitations in learning sequential dependencies within time series data. This contrasted with neural network approaches, which demonstrated superior capabilities in capturing and extrapolating temporal patterns. Our findings underscore the challenges intrinsic to utilizing ensemble tree methods for time series forecasting, as compared to their neural network counterparts.

6.2.2 CatBoost

The application of the CatBoost model for predicting AAPL stock prices resulted in a Mean Squared Error (MSE) of 26.25. This relatively high error rate can be attributed to the model's overfitting to the training data and its subsequent failure to generalize to unseen test data. Essentially, while the model performed well on the data it was trained on, it struggled to accurately predict new, unseen price levels. This issue is often observed with ensemble methods like CatBoost when they are extensively tuned to the training dataset, leading to a lack of adaptability and poor performance on real-world, unencountered data. The high MSE in the test phase indicates a significant discrepancy between the model's training accuracy and its effectiveness in practical scenarios. Since the price difference prediction target also failed in the XGBoost model, it was decided to continue with a different model.

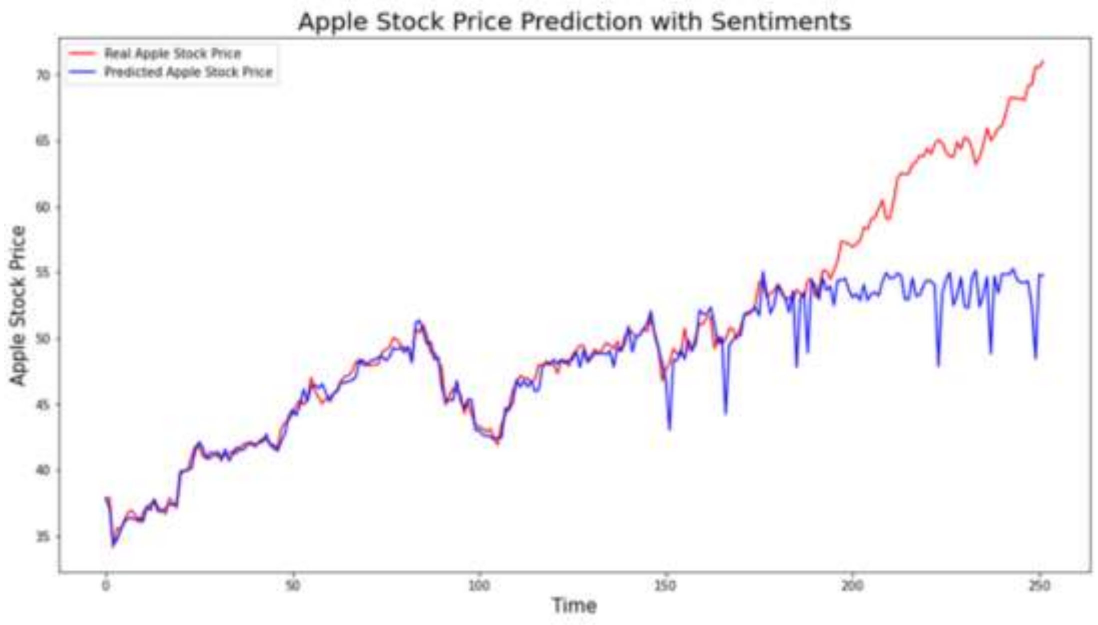


Figure 31: CatBoost Model on Test Data

6.3 Support Vector Regressor

The SVR model was employed to predict the closing prices of AAPL stock, utilizing a dataset with 1,252 days. This dataset included historical price data obtained from yfinance. When the model incorporated both tweet sentiments and historical data, it achieved a mean squared error

(MSE) of 3.197 and an accuracy of 53.7% in predicting the direction of the next day's price movement. In contrast, the model's performance without sentiment analysis, relying solely on historical price data, yielded a lower MSE of 2.45 and a higher accuracy of 58.56%. These results highlight the complexity of integrating sentiment analysis with traditional financial indicators. While incorporating sentiments from tweets provides a broader perspective on market sentiment, it appears to introduce additional volatility and uncertainty in the model's predictions. This suggests that the predictive relationship between social media sentiment and stock market movements is not straightforward and can potentially introduce noise that affects the accuracy. The challenge lies in accurately quantifying the impact of public sentiment on the stock market, which is influenced by a myriad of factors. The slight decrease in prediction accuracy when including tweet sentiments underscores the intricate dynamics of stock market prediction, where adding more variables does not always equate to better forecasting.

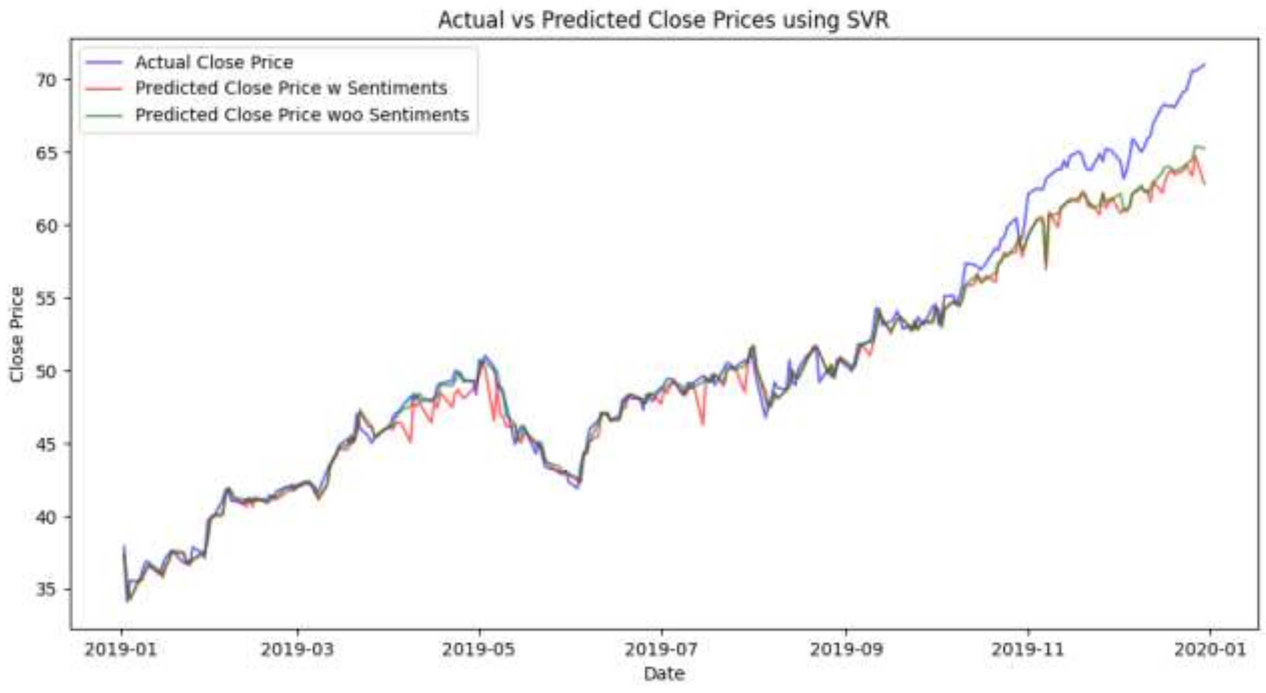
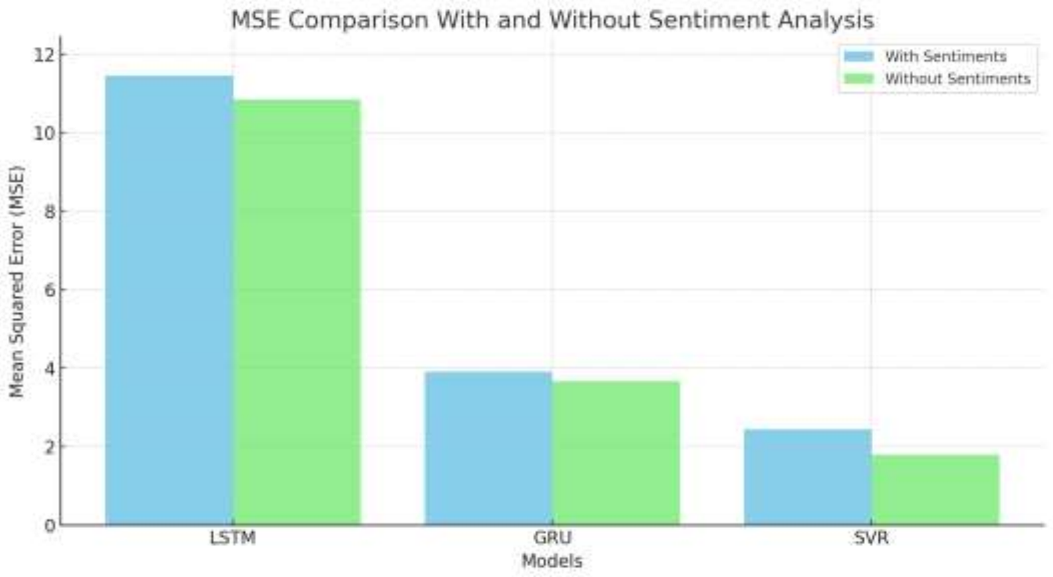


Figure 32: SVR on Data With and Without Sentiments

7.Discussion



Mean

Figure 33: Square Error Comparison

In our exploration of various models for stock price prediction, we observed distinct outcomes with LSTM, GRU, XGBoost, CatBoost, and Support Vector Regression (SVR). The LSTM model, when integrated with sentiment analysis, resulted in an MSE of 11.45 and an accuracy of 54.9%. Without sentiment analysis, the performance slightly improved in terms of MSE to 10.85 but the accuracy marginally decreased to 54.4%. This suggests that while sentiment analysis adds a layer of insight, it may also introduce complexity that doesn't significantly enhance prediction accuracy in the case of LSTM.

The GRU model showed a more pronounced effect of sentiment analysis. With sentiment data, it achieved an MSE of 3.91 and an accuracy of 55.2%, whereas without sentiment analysis, the MSE improved to 3.67 with a comparable accuracy of 54.4%. This indicates that GRU models, known for their efficiency in processing time-series data, can be slightly more effective without the inclusion of sentiment analysis, possibly due to reduced noise and complexity in the data.

Turning to ensemble methods, XGBoost and CatBoost yielded MSEs of 23.90 and 26.25, respectively. These figures are substantially higher than those of the LSTM and GRU models, suggesting that these ensemble methods may not be as adept at capturing the nuances of stock market data, particularly when compared to models that excel in handling sequential data.

SVR, a model well-regarded for its regression capabilities, showed the most promising results. With sentiment analysis, it recorded an MSE of 2.44 and an accuracy of 53.7%. Interestingly, removing sentiment analysis led to a lower MSE of 1.788 and an improved accuracy of 58.2%. This outcome reinforces the notion that sentiment analysis, while providing valuable market insights, can complicate the predictive modeling process, potentially leading to less accurate predictions in certain models.

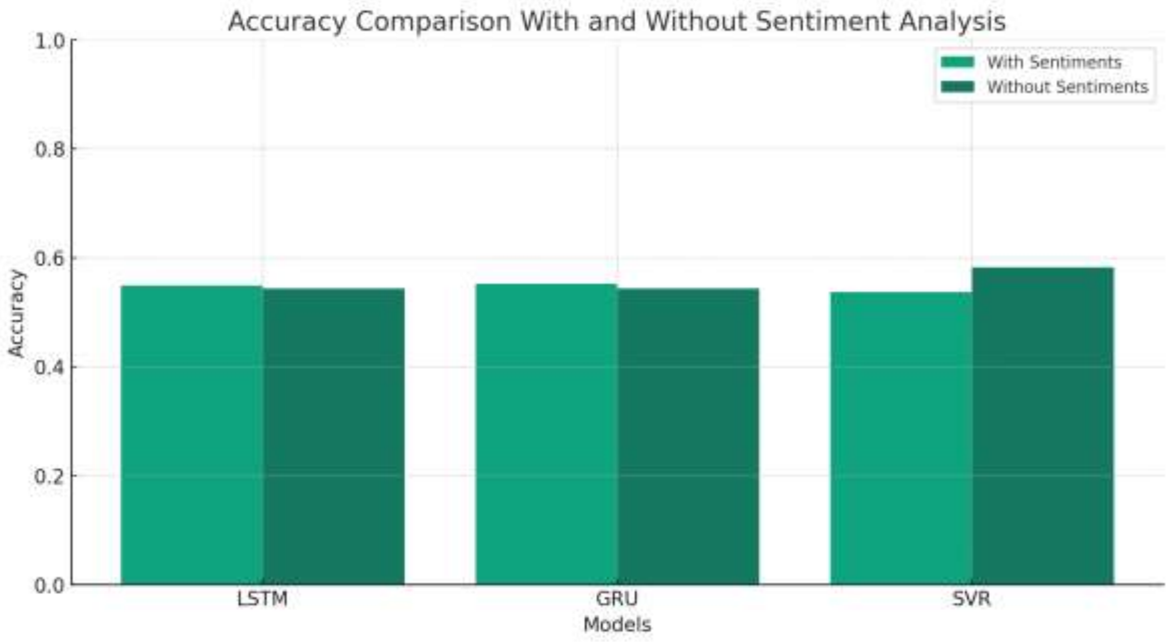


Figure 34: Accuracy Comparison

Overall, our analysis highlights the intricate balance between model choice, feature inclusion, and the nature of financial data. While sentiment analysis offers a broader perspective on market conditions, its integration into predictive models does not always guarantee improved accuracy. Instead, it can sometimes introduce additional variability that may not align with the more quantitative aspects of stock price movements. The performance differences between LSTM, GRU, ensemble methods, and SVR underline the importance of model selection and feature consideration in the complex realm of stock price prediction.

8.Conclusion

Reflecting on our stock price prediction project, we encountered surprising challenges, particularly with sentiment analysis, which unexpectedly reduced model accuracy. After testing various models like LSTM, GRU, XGBoost, CatBoost, and SVR, we found that SVR outperformed others, especially without sentiment analysis, achieving the lowest MSE and highest accuracy.

Looking ahead, our focus will be on refining data processing and model optimization. Despite SVR's success, we plan to further explore how sentiment analysis can be more effectively integrated. This project has highlighted the dynamic nature of financial modeling and the importance of adaptability and continuous innovation in this field.

9.Appendix

Work Contribution

Y.Kaan Baycan: Implementation of CatBoost and SVR models, focusing on their improvements and parameter tuning.Alse handled SVR and CatBoost related parts in the report in addition to the evaluation part.

Kimya Ghasemlou: BlobText and Bert models implementation and comparison

Uğur Sorar: Implementation of the XGBoost model, improvement of the training and prediction algorithm and hyperparameter tuning. Evaluation of the results.

Yusuf Doğan: Experimenting sentiment analysis with different models.

Deniz Aydemir: Implementation of the LSTM and the GRU models and their improvements.

10.References

[1] "Tweets about the Top Companies from 2015 to 2020", Kaggle, (2020). [Online]. Available: https://www.kaggle.com/datasets/omermetinn/tweets-about-the-top-companies-from-2015-to-2020

[2] “Simplified text processing¶,” TextBlob, https://textblob.readthedocs.io/en/dev/ (accessed Nov. 25, 2023).

[3]O. G. Yalçın, “Sentiment analysis in 10 minutes with bert and hugging face,” Medium, https://towardsdatascience.com/sentiment-analysis-in-10-minutes-with-bert-and-hugging-face-294e8a04b671 (accessed Dec. 25, 2023).

[4] *yfinance*. (2023c, October 4). PyPI. https://pypi.org/project/yfinance/

[5] “Project jupyter,” Project Jupyter, https://jupyter.org/ (accessed Nov. 25, 2023).

[6] “Colab.google,” colab.google, https://colab.google/ (accessed Nov. 25, 2023).