REPORT ON
STOCK SENTIMENT ANALYSIS USING MACHINE LEARNING

Prisha Sinha
Enrollment No.: 23124027, BSBE, IIT Roorkee

What is Sentiment Analysis?

Sentiment analysis is a natural language processing (NLP) technique that involves determining the sentiment expressed in a piece of text. It is widely used in various applications to understand the opinions, emotions, and attitudes expressed by users towards a particular subject or topic.

In context of the stock market, this project aims to utilise news related to a particular stock, major geopolitical events and industry specific developments to predict bullish or bearish trends.

Overview

Under this project, a machine learning model was built that resorts to news articles to understand the overall sentiment related to a particular stock. In addition, some technical parameters such as open, high and low prices of the different days were also used as features to train the model.

Flow of project

- Web Scraping news articles: Using the beautiful Soup python library, various news websites were scraped.
    - Headlines and sub headings were collected from web sites including the economic times, money control, yahoo finance, live mint, msn etc.
    - The model was trained on news of the past 3-5 months for the following stocks;
        - HDFC BANK
        - Adani Enterprise
        - ITC
- Obtaining technical data: Data regarding the OHLC prices for the past 3 months for each of the three stocks was obtained using the yfinance python library.
- Calculating sentiment scores: The data was scraped from the web and analysed using VADER to determine positivity, negativity, neutrality, and compound scores. Additionally, subjectivity and polarity were also assessed. Due to the predominantly factual nature of the stock price-related data, detecting nuanced sentiments like irony or sarcasm posed minimal challenges.

- VADER calculates sentiment scores by looking up words in a sentiment lexicon, assigning sentiment scores to each word, summing these scores for the entire text, and then normalizing the sum to produce a final sentiment score.
- Lexicon Lookup: VADER uses a pre-built lexicon (dictionary) which contains words that are rated on a scale from -4 to +4, where:
    - **-4**: Extremely negative
    - **-1**: Negative
    - **0**: Neutral
    - **+1**: Positive
    - **+4**: Extremely positive

Each word in the lexicon is associated with a polarity score (positive or negative) and an intensity score.

Polarity measures how positive or negative a text is. It's a float value within the range [-1.0, 1.0], where -1.0 indicates highly negative sentiment, 0.0 indicates neutral sentiment, and 1.0 indicates highly positive sentiment.

Subjectivity measures how subjective or opinionated the text is. It's also a float value within the range [0.0, 1.0], where 0.0 is very objective (factual) and 1.0 is very subjective (opinionated).

A sample of the dataset created:

Ticker	Date	Open	High	Low	Close	label	headlines	Positivity	Negativity	Neutrality	Compound Score	Polarity	Subjectivity
ADANIENT.NS	2024-04-01	3230.199951	3291.800049	3207.850098	3252.100098	1	10:1 Split Ratio Soon: FMCG ITC A High Convict...	0.191	0.0	0.809	0.9831	0.110404	0.542626
	2024-04-02	3258.949951	3285.000000	3240.000000	3268.750000	1	10:1 Split Ratio Soon: FMCG ITC A High Convict...	0.191	0.0	0.809	0.9831	0.110404	0.542626
	2024-04-03	3250.000000	3260.149902	3222.000000	3233.449951	0	10:1 Split Ratio Soon: FMCG ITC A High Convict...	0.191	0.0	0.809	0.9831	0.110404	0.542626
	2024-04-04	3250.000000	3273.000000	3201.699951	3210.800049	0	10:1 Split Ratio Soon: FMCG ITC A High Convict...	0.191	0.0	0.809	0.9831	0.110404	0.542626

Figure 1. Data Set Sample

- Model training: Each stock was trained and tested with different algorithms (LDA, Random Forest, Support Vector Machine and Deep Neural Networks). The model with best accuracy and precision was considered. Moreover, the are under the ROC curve was also studied and the model giving the maximum area for both train and test sets was finally used to further train the ensemble model.
- The precision of each of these models was also given due importance while selecting an appropriate algorithm for the ensemble model , given the fact that false positives could potentially cause losses for the investor.

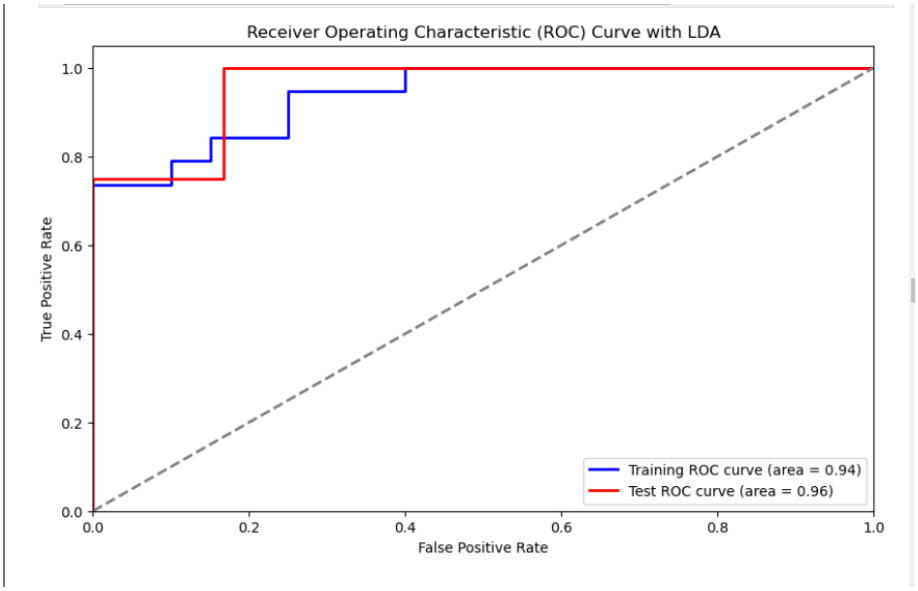

Figure 2. AUC-ROC

- Training the ensemble model: After getting the best model for each, (in this project a Linear discriminant Analyser and 2 Random Forest classifiers) the three models were used to train the ensemble model.
- Since slight overfitting could be seen with the models, the best ensemble method to address this issue would be a Voting classifier.
- A voting classifier in ensemble learning combines multiple individual models (classifiers or regressors) and aggregates their predictions to make a final prediction. This aggregation process reduces the chances of overfitting.
- A hard voting classifier was used and it achieved accuracy of 0.9.

```
LDA (model_1) Accuracy: 0.90
Random Forest (model_2) Accuracy: 0.80
Random Forest (model_3) Accuracy: 0.80
Voting Classifier Accuracy: 0.90
```

Figure 3.Accuracy of ensemble model

- Testing the model: After training the ensemble model, we used the model to predict on new testing data. Since the accuracy of the model highly depends on the amount of relevant news articles one feeds into it, it is advised that prior to making decisions based on market sentiment using this model, one should have enough news websites relevant to the stock(industry related news, geopolitical news- as they highly impact macroeconomic factors and therefore stock prices stock specific announcements-such as those related to dividends, bonuses ,rights issues or stock split/consolidation).
- Finally, the model was tested by feeding news articles and technical parameters related to a new stock into the ensemble model.
- Following are the parameters upon which the model was tested:
    - **Sharpe Ratio**
      It measures the return of an investment compared to its risk.
      A Sharpe ratio >1 is considered good.

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}$$

      Where:

        - R_p= The expected return of the portfolio (investment),
        - R_f = The risk-free rate of return,
        - $R_p - R_f$ = Portfolio's excess return
        - σ_p = The standard deviation of the portfolio's excess return

    - **Maximum Drawdown**
      Represents the maximum loss from the peak values of an investment to its lowest point before a new peak is reached.

$$MDD\ \% = \frac{peak\ value - trough\ value}{peak\ value} * 100$$

    - **Number of trades executed**

      It is the total number of buy and sell transactions of different stocks made within a specific time period. It helps traders understand volatility and market trends better.

    - **Win Ratio**
      It is the proportion of profitable trades or investments compared to the total number of trades or investments made over a period of time

$$win\ ratio = \frac{Total\ number\ of\ winning\ trades}{Total\ number\ of\ trades} * 100$$

These parameters can easily be calculated using the yfinance library by mentioning the ticker symbol, start date and end date and by defining functions for the ratio's (note that there would be a slight change in these functions for short positions).

Results

The model was tested on new stocks:

- **Hindustan Unilever**
  we can clearly see that the stock prices of HUL from 7th to 14th June 2024 fell significantly;

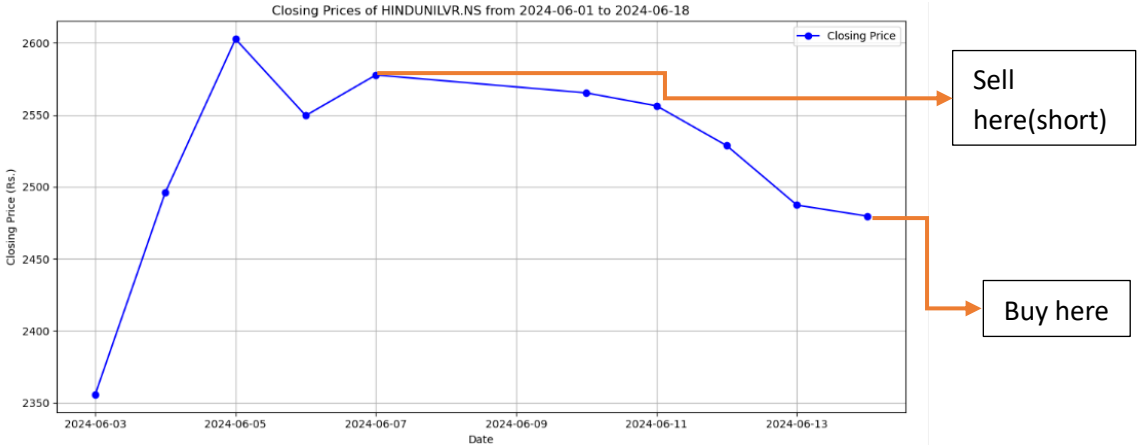

Figure 4.Identifying buy and sell positions

```
y_predict
```

```
array([0, 0, 0, 0, 0])
```

```
y_actual =df_1['label']
print(y_actual)
```

```
Ticker         Date
HINDUNILVR.NS  2024-06-10    0
               2024-06-11    0
               2024-06-12    0
               2024-06-13    0
               2024-06-14    1
Name: label, dtype: int32
```

```
accuracy = accuracy_score(y_actual,y_predict)
print(f'Accuracy: {accuracy:.2f}')
```

```
Accuracy: 0.80
```

Figure 5.The models predictions and its accuracy

- And as clearly predicted by the ensemble model, y_predict array has all values as 0 (which means that this model predicts that for the given time period, the closing prices each day were lower than the opening prices).
- Thus, going short on the stock would be advisable.
- Note that the accuracy in this case is 80 % which means the models predictions are almost same as what would actually happen.
- If we went short on HUL on 7th June and bought it on 14th then our returns would be as follows:

```
[**********************100%%**********************]  1 of 1 completed
Sharpe Ratio: 22.4225
Maximum Drawdown: 0.0269
Number of Trades Executed: 4
Win Ratio: 100.00%
```

Figure 6. Returns on model's predictions for HUL

Excellent results with a perfect win ratio can be seen.

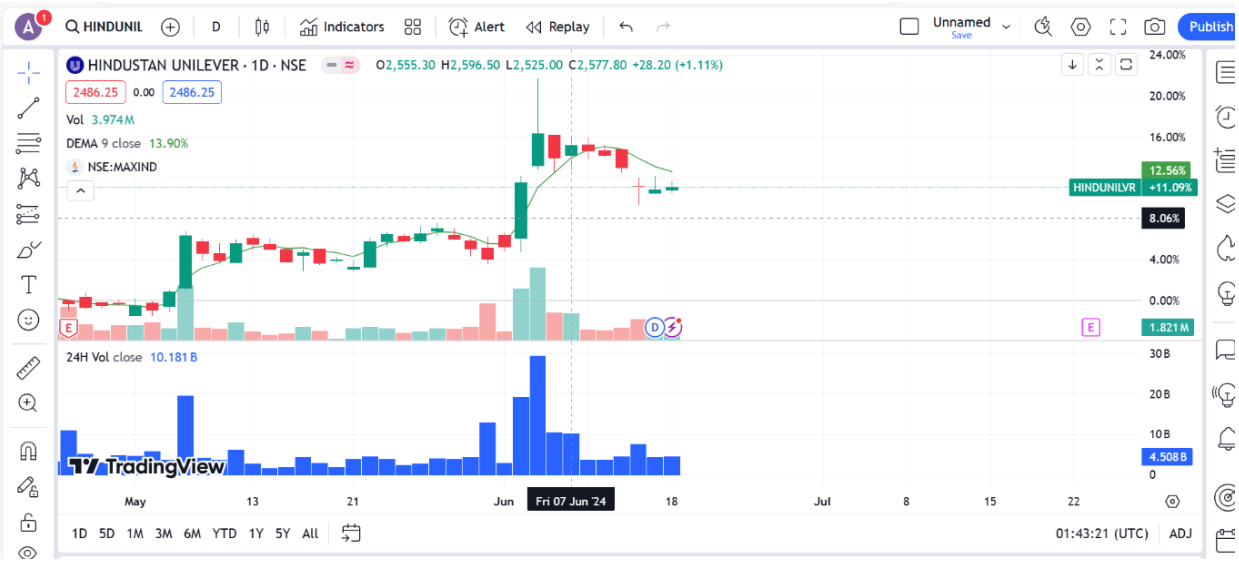

Figure 7. Snapshot of actual trend from trading view (the model accurately predicted the bearish sentiment).

- The investor should buy and hold HUL as a trend reversal can be seen-indicating a positive sentiment 14th June onwards.

- **Nvidia**

  Nvidia stock prices rose from 21st may to 18th June 2024, especially after the stock split announced on 7th June, thus our model should be predicting a bullish trend.

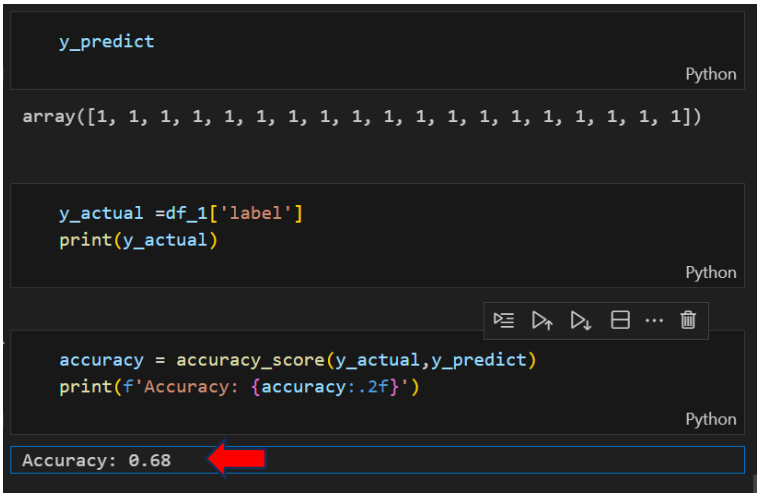

Figure 8.Model's predictions and its accuracy

  - As it can be clearly seen, the model predicts '1' for each day in the given time frame, indicating that one should go long on Nvidia.
  - The accuracy is 68 % so it is very likely that our model's predictions are correct.

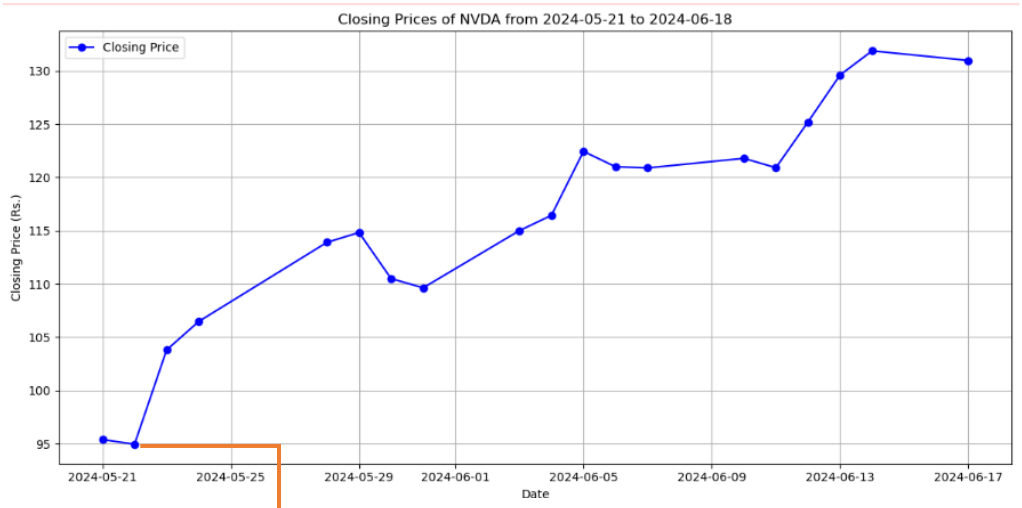

Figure 9.Identifying buy and sell positions

- From the above graph, we can see that the model correctly predicted the bullish trend of the stock
- The returns are as follows:

```
0]                                                                    Python

[**********100%%**********]  1 of 1 completed
Sharpe Ratio: 9.79
Maximum Drawdown: 5.44%
Number of Trades Executed: 17
Win Ratio: 64.71%
```

Figure 10.Returns on model's predictions for Nvidia

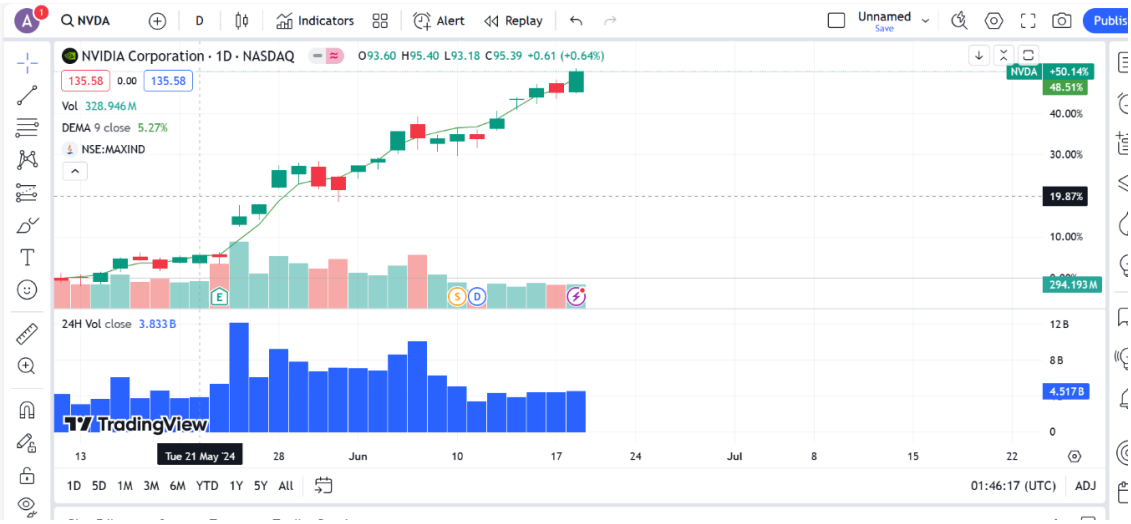

Figure 11.Snapshot of the actual trend from trading view

Trading Strategy

- Higher the accuracy for smaller time periods, the better the chances are for you to gain profits if you act according to the model's predictions.
- If accuracy > 0.6 for the stock that you have predicted the labels for, then you should rely on the model's predictions.
- The more relevant the news articles are with respect to the time period and industry specific news, the better would be the model's accuracy (for instance, news regarding elections significantly impacted stock prices of Adani enterprise in the 2024 general elections).
- Greater number of 0's in the y_predict array indicate bearish trend, hence going short on the stock is recommended
- Greater number of 1's in the y_predict array indicate bullish trend, hence going long on the stock is recommended

Challenges faced

- While training, unique events such as stock price fall on the 4th of June for Adani Enterprise was difficult for the model to learn as it was an outlier, thus, the model was trained again by optimising the class weights to account for the minority classes (e.g.: fall in stock prices due to sudden loss in investor confidence).

- Finding relevant news articles for web scraping proved to be challenging as a wide variety of factors affect stock prices. In addition, news related to a stock online is often limited to significant announcements and events, which makes the data collection process time-consuming.

Way forward

- Moving forward, continuous monitoring and optimization of the models is essential to maintain accuracy and relevance in dynamic market conditions. Additionally, integrating user feedback and employing robust evaluation methods will contribute to enhancing the system's performance and usability over time.
- Ultimately, a well-executed stock sentiment analysis project not only facilitates better understanding of market sentiment but also empowers stakeholders with actionable insights to navigate the complexities of financial markets effectively.

References

- TradingView — Track All Markets
- How to Improve Class Imbalance using Class Weights in ML? (analyticsvidhya.com)
- Linear Discriminant Analysis in Machine Learning - GeeksforGeeks
- Voting Classifier - GeeksforGeeks