Natural Language Processing for Stock Market Indicators

AUTHORS	STUDENT ID
Zhuoer Feng	1004971
Ge Ziyu	1004880
Muhammad Abid Firas	1006017
Rio Chan Yu hoe	1005975
Chin Wei Ming	1006264

Singapore University of Technology and Design

50.038: Computational Data Science

Prof. Dorien Herremans, Prof. Soujanya Poria

April 8, 2023

1. Introduction

Accurately predicting stock prices' rise and fall has long been a dream of traders and researchers alike. The instantaneous nature of Twitter means that people's sentiments may be felt directly on stock prices, which created a new prediction layer. Elon Musk's announcement of Tesla's privatization and Kylie Jenner's comment on Snapchat both had dramatically impacted the stock prices of the two companies. In recent years, with the advancement in NLP, such as FinBERT, people have tried to incorporate public and/or professional sentiments into stock price prediction.

For instance, Mehtab & Sen (2019) attempted to predict the movement of NIFTY 50 index 1 week into the future based on Twitter sentiments using an LSTM model, while Sonkiya et al (2021) used GAN and BERT to predict the price of Apple Inc. (ticker symbol: AAPL) 5, 15, and 30 days into the future. Besides Twitter, which reflects the views of the public and influencers, some such as Puh & Babac (2023) also used professional views from the Wall Street Journal to predict the stock market.

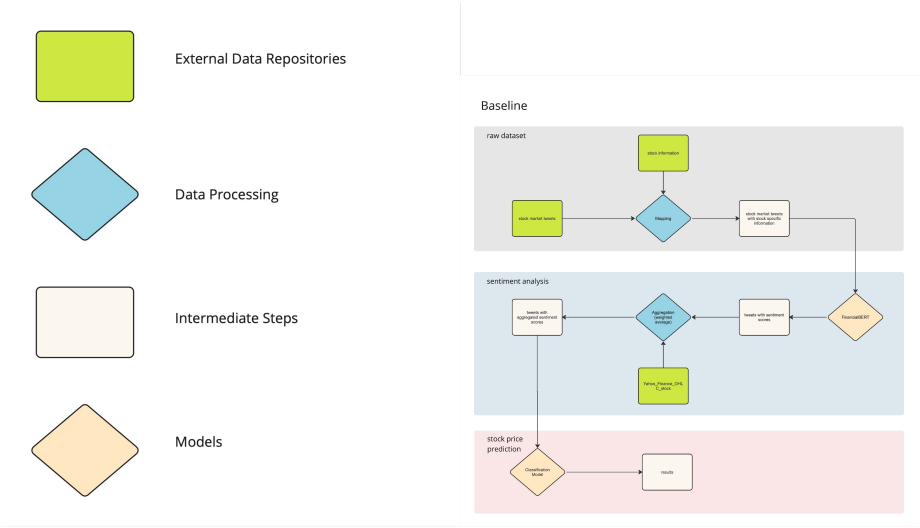
However, we found that existing literature has largely focused on the movement of stock prices multiple days into the future, which is at odds with the common trading practice of day trading at financial institutions. Hence, our project would like to focus on developing a model to predict the movement of stocks within the trading day based on public sentiments i.e., tweets and market information.

2. Methodology

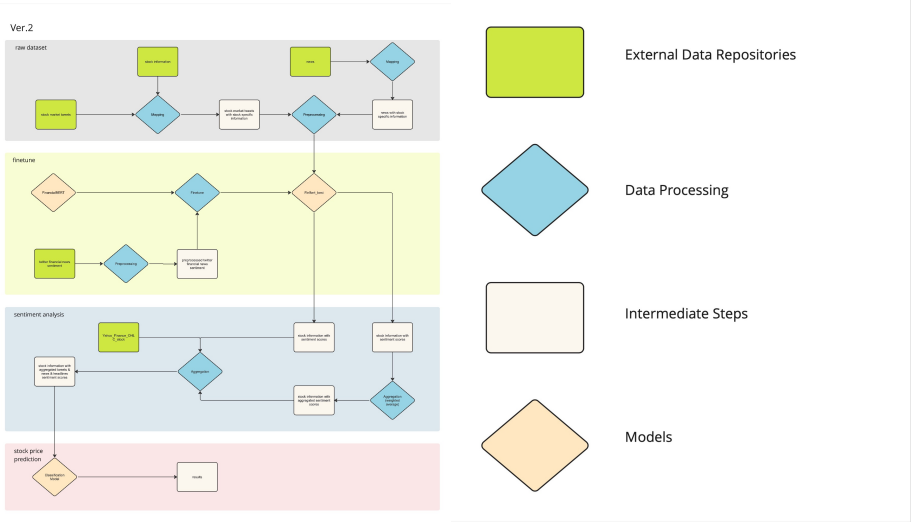
2.1. Workflow

To explore the impact of sentiment analysis on stock price prediction, we developed two workflows: a baseline version and an enhanced version, which includes additional data.

As shown in the figure below, the baseline workflow has three main components: a dataset composed of tweets, sentiment analysis of these tweets, and a classification model to predict stock price movements.



To broaden our exploration of stock indicators, we've expanded our dataset to include news articles and fine-tuned the FinancialBERT (FinBERT) Model. These improvements aim to increase the accuracy of our sentiment analysis within the updated workflow, as displayed in the diagram below.



2.2. Dataset

2.2.1. Dataset Collection

As outlined in the workflow diagram presented in the previous section, we utilize five distinct sub-datasets. Each dataset has a specific source and purpose as detailed below.

1. stock_market_tweets
   Source: "mjw/stock_market_tweets" from HuggingFace
   Description: This dataset consists of tweets specifically related to stocks, including text content and additional details such as the number of retweets, likes, and comments.

2. stock_information
   Source: Scraped from Nasdaq
   Description: Information about stocks including details about their country, sector, and industry.

3. news
   Source: "ashraq/financial-news" from HuggingFace
   Description: A collection of financial news headlines.

4. twitter_financial_news_sentiment
   Source: "zeroshot/twitter-financial-news-sentiment" from HuggingFace
   Description: An English corpus of finance-related tweets annotated for sentiment, used to fine-tune the FinBert model.

5. Yahoo_Finance_OHLC_stock
   Source: Extracted using Yahoo Python API
   Description: Stock price information, specifically Open, High, Low, and Close (OHLC) data.

2.2.2. Dataset Manipulation

The datasets 'stock_market_tweets' and 'stock_information' are merged through an inner join operation on the stock symbol.

In the case of the 'news' dataset, which lacks a specific column identifying the relevant stock, we first compile a list of distinct company names from 'stock_information'. We then attempt to extract patterns of these company names from the news headlines.

After predicting and aggregating sentiment scores, the datasets 'stock_market_tweets', 'news', and 'Yahoo_Finance_OHLC_stock' are merged together based on the stock name and date information. Further details about the sentiment aggregation process will be discussed in the following section.

2.3. Sentiment Analysis

We selected the pretrained FinBERT model, 'ahmedrachid/FinancialBERT-Sentiment-Analysis,' to serve both as the tokenizer and the prediction model. This transformer-based model outperforms traditional BERT models in handling financial texts. Optimized specifically for financial contexts, the sentiment labels are designated as follows: the positive label suggests a 'buy' action, zero indicates 'do nothing,' and the negative label advises to 'sell'.

2.3.1. Fine-tuning

In the baseline architecture, the model was not fine-tuned. However, for the enhanced version, we fine-tuned the model using 11,931 labeled financial tweets from the 'twitter_financial_news_sentiment' dataset. The training parameters employed were as follows:

learning_rate=2e-5,

per_device_train_batch_size=16,

per_device_eval_batch_size=16,

num_train_epochs=20,

weight_decay=0.01,

evaluation_strategy="epoch",

save_strategy="epoch",

metric_for_best_model='accuracy'

In addition, we quantified the effects of preprocessing steps, which included removing repeated punctuations, cleaning hashtags, and replacing URLs with the '[URL]' token. As shown in the table, the sentiment prediction accuracy on the test dataset remains relatively consistent, both before and after fine-tuning. This indicates the tokenizer's effectiveness in handling messy text.

	Before Fine-tuning	After Fine-tuning
With Preprocessing	0.7428810720268006	0.8454773869346733
Without Preprocessing	0.7462311557788944	0.8408710217755444

2.2.2. Sentiment Aggregation

2.2.2.1. Tweets Given the large volume of tweets related to each stock on any given day, it becomes necessary to aggregate the respective sentiment scores to obtain a coherent overall sentiment. Furthermore, to gauge the social impact of specific tweets, we also consider engagement metrics to measure the public influence exerted by each tweet.

The sentiment score, however, originally a categorical variable with three labels (0 for negative, 1 for neutral, and 2 for positive), is not directly suitable for multiplication with quantitative metrics like retweet counts, like counts, or comment counts, as this operation lacks intuitive sense with categorical labels.

To address this, we convert the sentiment score into a numerical value by assigning specific weights to each category, allowing for the calculation of a weighted sentiment score. This method involves mapping the sentiment categories to numerical values:

sentiment_weights = {0: -1, 1: 0, 2: 1}

- 0 represents negative sentiment (assigned a weight of -1)
- 1 represents neutral sentiment (assigned a weight of 0)
- 2 represents positive sentiment (assigned a weight of 1)

The formula for weighted average score calculation:

$$\text{Weighted Sentiment Score} = \frac{\sum_i (sentiment\_score_i \times (retweet\_num_i + like\_num_i + comment\_num_i))}{\sum_i (retweet\_num_i + like\_num_i + comment\_num_i)}$$

This formula calculates the weighted average of sentiment scores, taking into account the weights provided by the retweet counts, like counts, and comment counts for each tweet.

2.2.2.2. News Headlines For news items, we calculate the average sentiment score of all headlines related to a specific stock on the same day. Additionally, we include an average sentiment score for news related to the stock's industry, thus incorporating an industry sentiment indicator. This dual-layered approach helps provide a broader perspective on the sentiment influencing both the specific stock and its wider industry.

2.2.2.3. Integrated Dataset for Predicting Stock Price Movements The aggregated sentiment scores from tweets and news are combined with the OHLC (Open, High, Low, Close) pricing data based on stock name and tickers. The resulting dataset, prepared for stock price prediction, includes the following columns:

- post_date
- ticker_symbol
- Country
- IPO Year
- Sector
- Industry
- Open
- High
- Low
- Close
- Adj Close
- Volume
- Shifted_Open

- Shifted_Adj Close
- Shifted_Close
- prev_day_Open
- prev_day_Open_AdjClose
- prev_day_Close
- news_sentiment
- industry_sentiment
- weighted_tweet_sentiment

2.4. Data Pre-processing

2.4.1. Motivation

The data preprocessing consists of five major tasks, i.e., data cleaning, reduction, scaling, transformation and partitioning (Xiao and Fan, 2014; Fan et al., 2015a; Fan et al., 2015b)

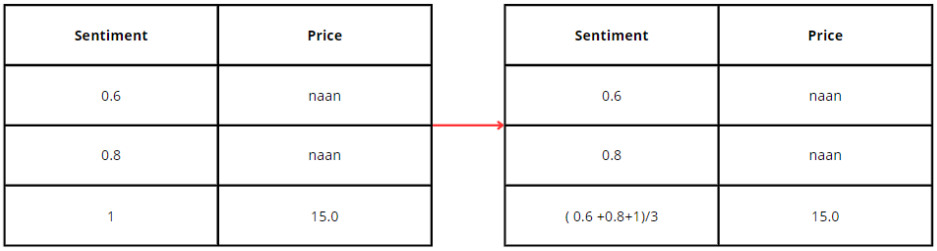
. These techniques effectively address inconsistencies, redundancies, and irrelevant information within the data, ultimately enhancing data quality and model accuracy (Gandomi and Haider, 2015)

2.4.2. Methodologies

The techniques we used to preprocess the dataset, ensures its compatibility with various machine learning models. Through a series of strategies, the raw data underwent preprocessing to enhance its usability and effectiveness in subsequent modeling tasks.

2.4.2.1. Data cleaning We clean the data to handle missing data effectively. Initially, missing values were addressed using the forward fill method (method='ffill') to propagate non-null values forward along the specified axis. Subsequently, rows with missing values in critical columns such as 'Open', 'High', 'Low', 'Close', and 'Adj Close' were removed using the 'dropna' method. This ensured that the dataset was cleansed of incomplete or erroneous records, facilitating robust analysis and a viable dataset for training the models.

In addition, a rolling accumulation of sentiment scores was performed to augment the dataset's richness. This process involved carrying forward and averaging sentiment scores in rows where price columns contained missing values. By aligning sentiment scores with price data through this rolling accumulation, the dataset's temporal coherence was maintained, facilitating a more comprehensive analysis of market sentiment trends.



2.4.2.2. Data transformation The dataset was restructured into numerical and categorical to ensure that different types of features are appropriately handled during preprocessing. Categorical features : ['ticker_symbol', 'Country', 'IPO Year', 'Sector', 'Industry'], within the dataset were transformed into a numerical representation to make them compatible with machine learning algorithms. This transformation was achieved through label encoding, wherein each categorical feature's values were converted into numerical labels using the LabelEncoder module from the scikit-learn library. By encoding categorical variables into numerical form, the dataset was prepared for subsequent analysis and model training.

A new binary target variable, denoted as 'y', was generated to facilitate binary classification. This variable was derived from the 'Close' and 'Open' price columns, where 'y' was assigned a value of 1 if the 'Close' price was greater than the 'Open' price, indicating a positive price change. Conversely, 'y' was assigned a value of 0 if the 'Close' price was less than or equal to the 'Open' price, indicating a negative or neutral price change. This labeling process enhanced the dataset's suitability for binary classification tasks.

2.4.2.3. Data scaling To ensure consistency in feature scales and facilitate convergence during model training, numerical features underwent scaling. The Min-Max scaling technique was employed, which rescales each feature to a specified range (typically 0 to 1) using the MinMaxScaler module from scikit-learn. By scaling numerical features, the dataset's features were normalized, preventing features with larger scales from dominating the learning process.

2.5. Model

2.5.1. Objective

We follow a typical day trading scenario, in which all positions are closed before the trading day to minimize risks and price change after the market closes. The decision to long, in anticipation of an increase in the stock price, or short, in anticipation of an decrease, is made at the opening of a day's market, with the stock being sold or bought at the closing to close the position.

Therefore, our algorithm attempts at answering the classification problem: will the stock's closing price be higher than the opening price, given historical market

data and recent tweets & news about the company and/or industry?

2.5.2. Evaluation Methodology

In our evaluation methodology, we will utilize different metrics depending on the target variable type. For continuous, we employ Mean Squared Error (MSE) (Hyndman & Athanasopoulos, 2014). MSE measures the average squared difference between predicted and actual values, with lower MSE indicating better model performance for continuous prediction tasks.

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

For discrete (classification tasks), we use a combination of metrics. Firstly , we look towards accuracy measurement, the ratio of correctly classified instances (Japkowicz & Stephen, 2016), as a common starting point. We define correctly classified instances as actual labeled y inputs to be the same as predicted y inputs. Additionally, we will employ Binary Cross-Entropy (BCE) loss (Kingma & Ba, 2017). BCE loss measures the difference between predicted probabilities and actual class labels, and it's often used during model training as a loss function to minimize classification error.

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

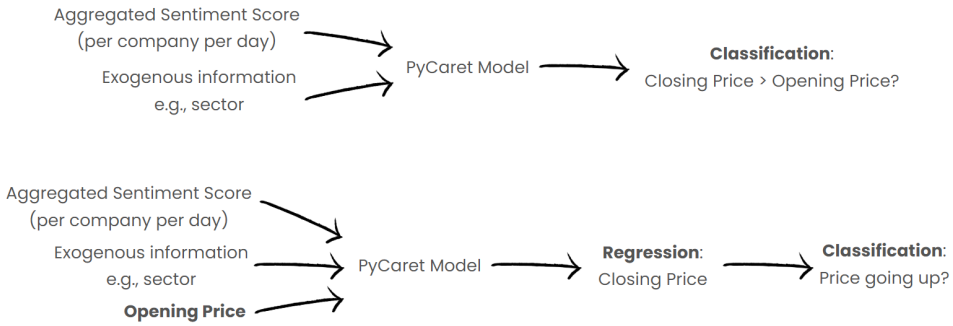
We will further explore classifier scores (precision, recall, F1-score) for imbalanced classes (Sokolova et al., 2006) and consider confusion matrices (Powers, 2020) to gain deeper insights into our random forest model's performance.

2.5.3. Naive Model

2.5.3.1. Framework Two naive models were used to help us get a taste of the problem. For the first naive model, we framed the question as a pure classification problem, combining the NLP result (the stock's aggregated sentiment score) and the market information (exogenous information e.g., sector) to predict whether the closing price will be higher than the opening price.

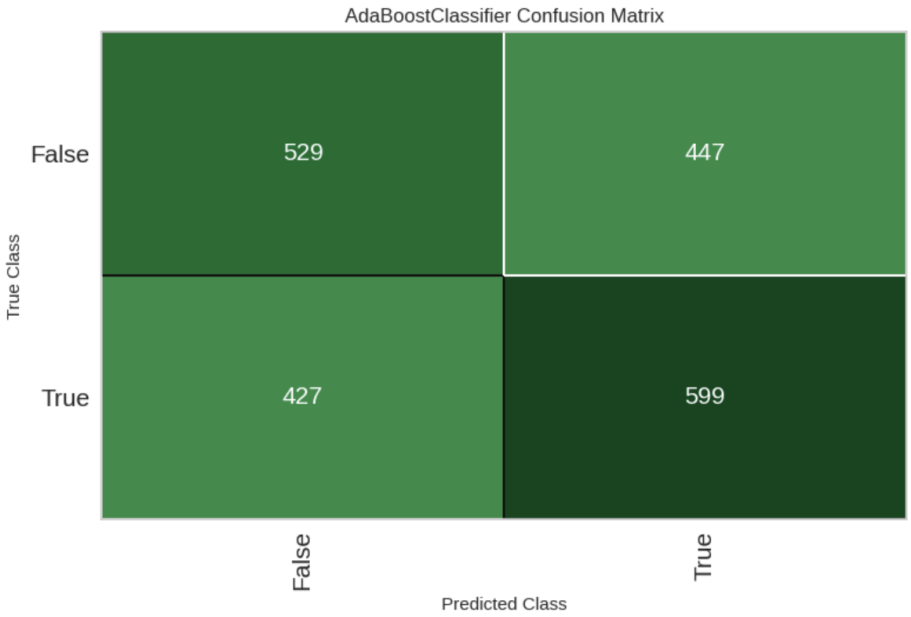
We also tried to first frame the question as a regression problem, where we use the NLP result, market information, and the opening price to predict the exact expected closing price of the stock. The predicted closing price will then be compared with the opening price to answer whether the price is expected to go up during the day. We believe by framing the problem in two ways, we will be able to test both classification and regression algorithms and make a more

informed decision on how to approach the problem for subsequent modeling. A simple illustration of the logic flow of our naive model is shown below.



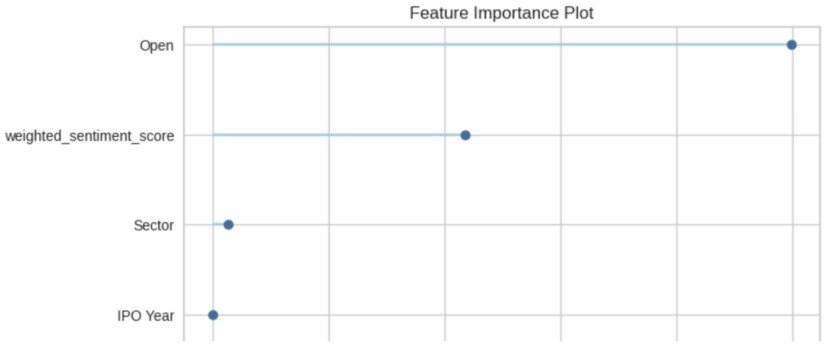
2.5.3.2. Implementation We used PyCaret as our naive model implementation, which trains and compares across different machine learning models, and selects the one with the best accuracy after 10-fold validation.

For the classification approach, the highest accuracy of 57.01% was obtained by an AdaBoost Classifier model. In comparison, the dummy model returned a 51.25% accuracy. The confusion matrix and the top 3 performing models are given below.



Model	Accuracy
Ada Boost Classifier	0.5701
Ridge Classifier	0.5637
Logistic Regression	0.5629

For the regression approach, the highest accuracy obtained was slightly worse at 55.79%. However, we were able to confirm the importance of twitter sentiment on stock price movement. As shown in the feature importance plot below, the variable weighted_sentiment_score is the second most important feature among all, only falling behind the opening price but vastly exceeds the importance of other exogenous variables like the sector and the companies' IPO year. This confirms the role twitter sentiment plays in dictating stock price movement and prompted us to fine-tune the sentiment evaluation and improve our model.



2.5.4. Models Tested

2.5.4.1. FBProphet FBProphet is a time series forecasting tool by Facebook Open Source. Besides the time series of closing price itself, FBProphet also allows us to use other non-temporal regressors to perform a multivariate prediction. In our model, we tried to predict the next day's closing price, based on the historical time series of: the closing price, the trading day's opening price, stock-specific twitter sentiment, news sentiment, and industry sentiment. We also acknowledge that there may be a "delay" in a tweet's impact on the stock price, for instance when the tweet was posted between the closure of the previous day's market and the opening of the present day's market. Hence, for all sentiment-related variables, we included both sentiment scores on the day itself and the previous day. As an example, for Twitter sentiment, we use the following two variables as our regressors:

- weighted_sentiment_score: Tweets' weighted aggregate sentiment on the trading day
- weighted_sentiment_score_-1: Tweets' weighted aggregate sen-

timent from the previous trading day.

The results were obtained iteratively, for which the FBProphet model is retrained everyday, with new information added. To ensure sufficient training data initially, we start training the time series model on the 366th trading day, which is equal to around 30% of all data. However, we did not witness a significant accuracy increase for later days as we accumulate a longer time series. We were able to obtain an overall testing accuracy of around 58.61% from our FBProphet model.

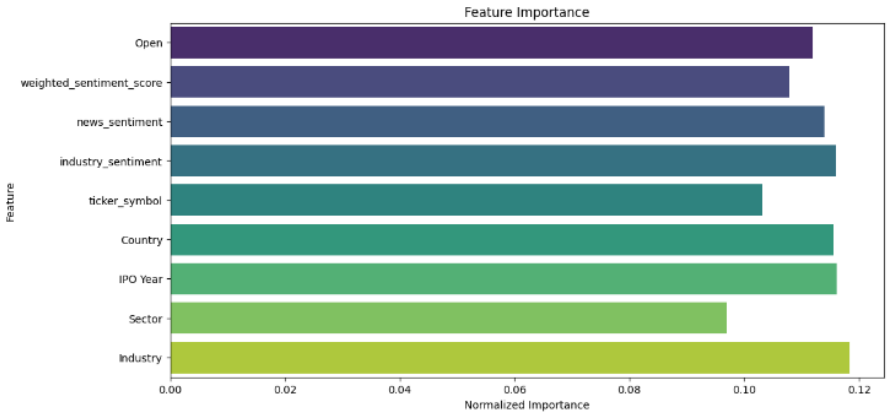
2.5.4.2. Recurrent Neural Networks (RNNs) Our dataset inherently embodies a time series prediction framework, characterized by timestamps for each entry. In light of this temporal structure, Recurrent Neural Networks (RNNs) were selected for their capability to leverage sequential data. Various RNN architectures, including Gated Recurrent Units (GRU) with attention mechanisms and Long Short-Term Memory (LSTM) networks, were implemented to capture the intricate patterns and features embedded within the dataset, ultimately aiming to predict asset prices.

Our initial approach involved amalgamating data from all ticker symbols into a unified dataset. This merged dataset facilitated the collective analysis of categorical and numerical features. The training and testing datasets were then partitioned, with the training set sequenced to preserve temporal dependencies. Evaluation of the model on the test data revealed an accuracy of 55.68%.

Subsequently, we pursued a more nuanced analysis by segregating the dataset based on individual ticker symbols. This approach unveiled distinct performance variations across different datasets. Notably, the model exhibited higher accuracy (61.62%) for the Microsoft (MSFT) ticker symbol, while yielding lower accuracy (51.14%) for Tesla (TSLA). These discrepancies underscored the model's propensity to excel in specific contexts, hinting at potential limitations in its robustness and adaptability.

To further elucidate the underlying dynamics, we explored alternative RNN architectures, including GRU and attention models. Despite variations in model complexity, the results remained largely consistent. This convergence suggested that performance trends were less influenced by architectural nuances and more by shared factors such as hyperparameters, initializations, and loss functions. Our emphasis remained on understanding the fundamental characteristics of the base model, with minimal fine-tuning deemed necessary.

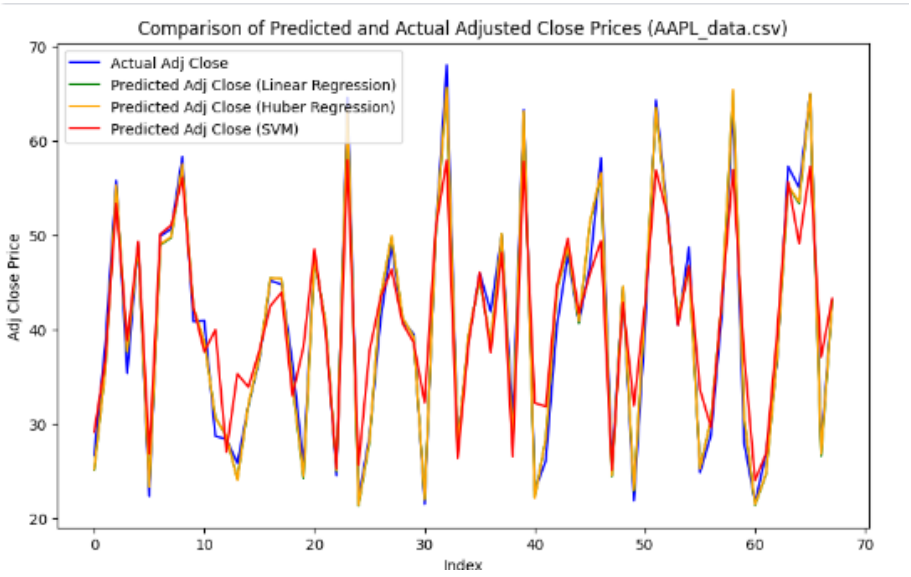
Incorporating sentiment accumulation strategies into the dataset yielded promising results, with each model experiencing a modest increase in accuracy ranging from 1% to 2%. However, a compelling discovery emerged during feature importance analysis. Contrary to expectations, sentiment scores exhibited uniform weightage alongside other inputs, implying that they were not predominant factors driving model predictions. This revelation prompted a reevaluation of our approach, signaling the need for alternative methodologies to enhance model accuracy and efficacy.



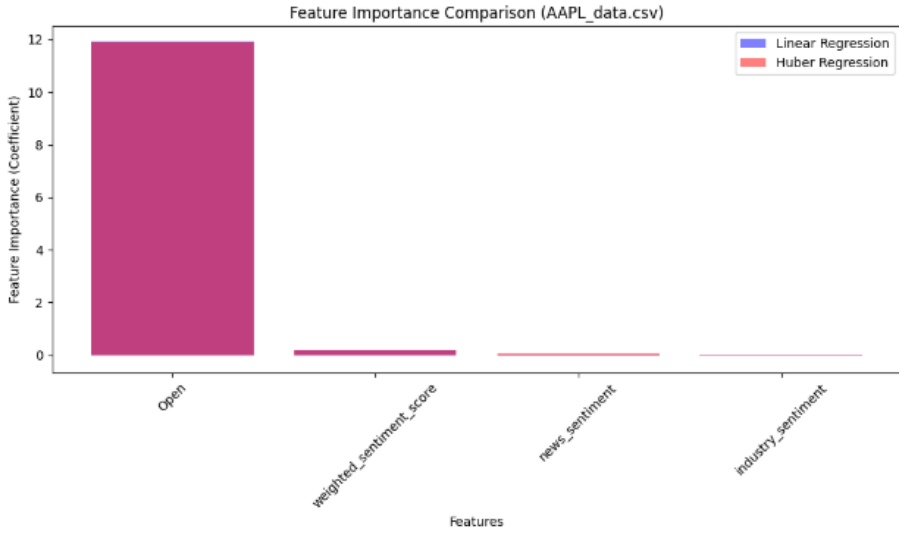
2.5.4.3. Linear Regression In our pursuit of modeling temporal data, Linear Regression emerged as a candidate worthy of exploration. Employing various methodologies, including Support Vector Machines (SVM) and Huber Regression, we sought to evaluate their predictive accuracy.

Initially, we assessed the performance of these models on a merged dataset, where all ticker symbols were consolidated. The Mean Squared Error (MSE) for Linear Regression was 0.713, for Huber Regression it was 0.686, and for SVM it was notably higher at 78.24. These results provided an initial glimpse into the models' efficacy in handling the combined dataset.

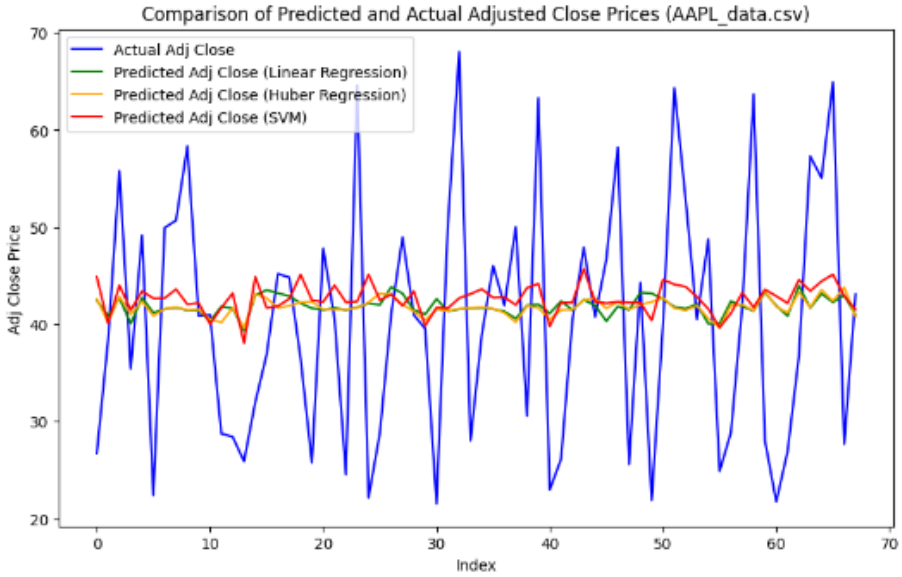
Subsequently, we conducted a more granular analysis by splitting the dataset based on individual ticker symbols. For instance, for the AAPL_data.csv, the MSE for Linear Regression was 2.373, for Huber Regression it was 2.371, and for SVM it was 25.05. Despite achieving high accuracies ranging from 88.06% to 98.50%, it became evident that the models' performance varied across different datasets.



Further analysis revealed the dominance of the 'Open' price feature in influencing predictions, indicating a misalignment with our goal of leveraging sentiment scores for price prediction.



Attempts to de-emphasize the 'Open' price and prioritize sentiment scores led to a drastic increase in MSE and a decline in accuracy. For instance, excluding the 'Open' price feature resulted in significantly higher MSE values (Linear Regression: 163.64, Huber Regression: 162.11, SVM: 158.94) and reduced accuracies (Linear Regression: 50.75%, Huber Regression: 47.76%, SVM: 52.24%).



Model	Merged Dataset (MSE)	Split Dataset (MSE)	Split Dataset without 'Open' (MSE)
Linear Regression	0.713	2.373	163.64
Huber	0.686	2.371	162.11
SVM	78.24	25.05	158.94

From the above table, the findings underscore the challenge of reconciling model objectives with feature importance. While Linear Regression demonstrated high accuracy, its reliance on 'Open' price undermined the incorporation of sentiment scores. This highlights the need for tailored approaches that strike a balance between feature relevance and predictive performance, paving the way for more nuanced modeling strategies.

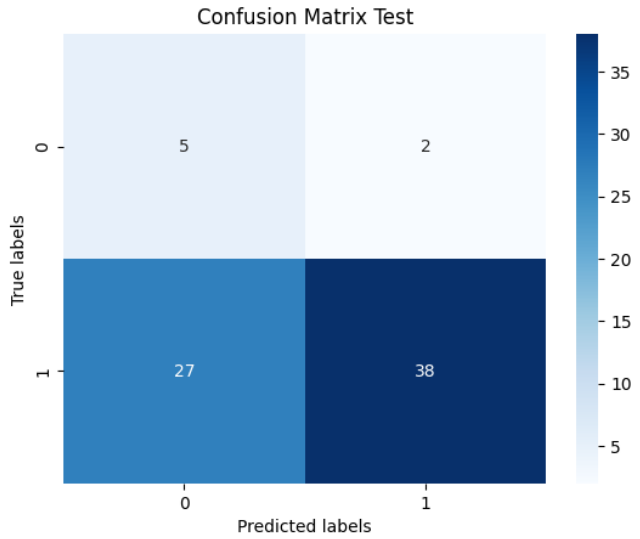
2.5.4.4. Random Forest The choice of utilizing Random Forest as our model was inspired by a reference project on a GitHub repository by Anubhav Anand on his stock market prediction using sentiment analysis project. In his project, by utilizing Random Forest, he successfully achieve the accuracy of 91.96% on United Airlines stock.

We utilized Apple stock for this experimentation. Firstly, we ran the base model of random forest taking in account weighted_sentiment_score, Open, Close, Low, High, news_sentiment, industry_sentiment as features and the output will be an indicator of 1 or 0. An indicator of 1 will signal that the closing price is greater than the opening price, and an indicator of 0 will signal otherwise.

Our test result for the base random forest model is as shown:

```
Accuracy: 0.5972222222222222

Classification Report:
              precision    recall  f1-score   support

           0       0.16      0.71      0.26         7
           1       0.95      0.58      0.72        65
```

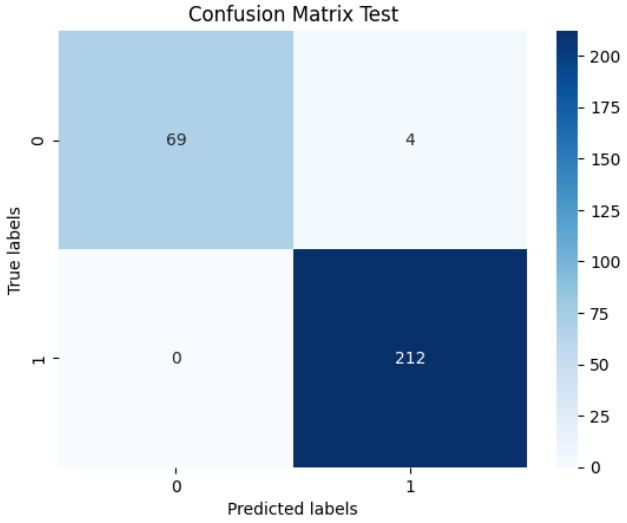
As well as the confusion matrix:



We also experimented with the trained random forest model on the original train dataset. The results is as follows:

```
Accuracy: 0.9859649122807017

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        73
           1       0.98      1.00      0.99       212

    accuracy                           0.99       285
   macro avg       0.99      0.97      0.98       285
weighted avg       0.99      0.99      0.99       285
```

As well as the confusion matrix:



Analyzing the results of how well the model performed with the train data and test data, we hypothesize that the model is overfitting on the train data. The huge difference in accuracy of train data (98.60%) and test data (59.72%) shows that the model does not perform well with unseen data.

Following this observation, we looked further into the parameters of the Random Forest model. We found an article that talks about hyperparameter tuning with Random Forest and attempted to follow its approach in finding the best parameters for our model.

We will be trying to adjust the following hyperparameters:

- N_estimators - This parameter allow us to specify the number of trees in the Random Forest
- Max_depth - Controlling the maximum depth of each tree in the Forest
- Min_samples_split - Specifying the minimum number of samples required to split an internal node.
- Max_features - Determines the number of features to consider.
- Min_samples_leaf - Setting the minimum number of samples required to be at a leaf node.
- Bootstrap - Determine whether bootstrap samples are used when constructing trees.

Using a GridSearchCV, we create a parameter grid and perform 5-fold cross validation on each combination of the parameters we defined to search through. The 5-fold cross validation will give us the most extensive evaluation of the performance of each combination of parameters. It will split the data to 5 sets,

and it will fit the model 5 times and take a different set as a test set to evaluate the performance for each 5 times.

Following the search, we found the best performing parameters:

Best Hyperparameters:

{'bootstrap' : True ,

'max_depth' : None ,

'max_features' : 'sqrt' ,

'min_sample_leaf' : 1 ,

'min_sample_split': 2 ,

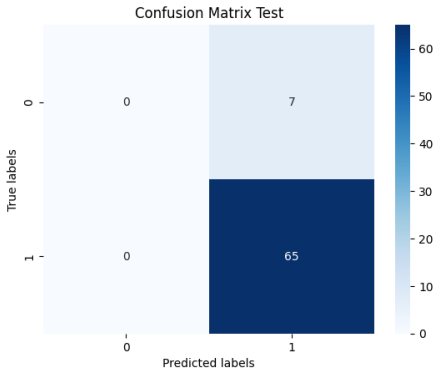
'n_estimators': 100}

Evaluating the performance of the model under these hyperparameters, we can see an improvement of robustness of the model.

Performance of best hyperparameter model on train dataset:

```
Accuracy: 0.9027777777777778

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         7
           1       0.90      1.00      0.95        65

    accuracy                           0.90        72
   macro avg       0.45      0.50      0.47        72
weighted avg       0.82      0.90      0.86        72
```

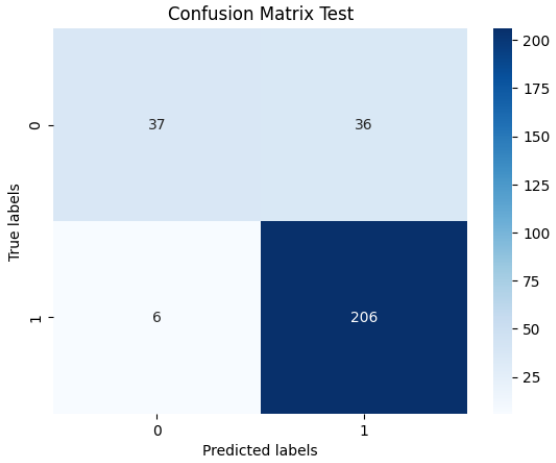
As well as its confusion matrix:



Performance of best hyperparameter model on test dataset:

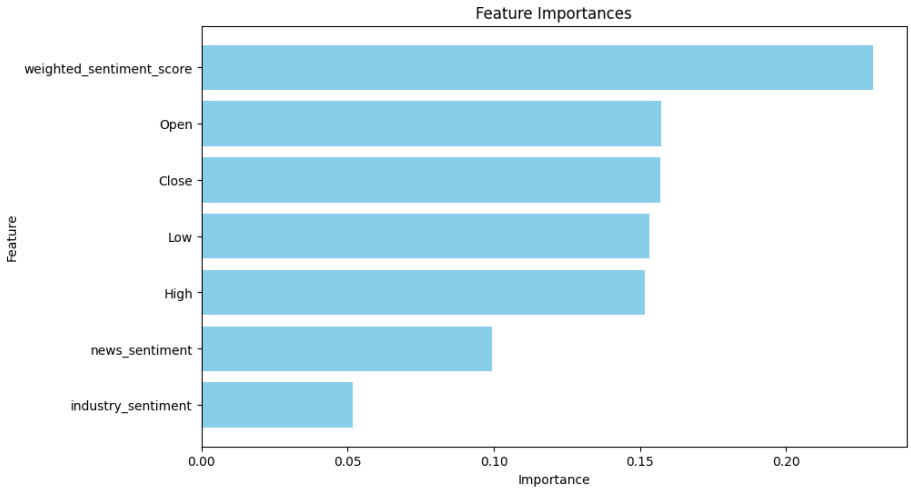
```
Accuracy: 0.8526315789473684

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.51      0.64        73
           1       0.85      0.97      0.91       212

    accuracy                           0.85       285
   macro avg       0.86      0.74      0.77       285
weighted avg       0.85      0.85      0.84       285
```

As well as its confusion matrix:



We see a significant improvement in the robustness of the model after performing hyperparameter tuning for our Random Forest Model. With the a smaller difference between the accuracy of test and train data as compared with the base model without hyperparameter tuning. Furthermore, we see a significant increase in the performance of the model from 59.72% to 90.28%.

Next we evaluate the feature importance of the model:

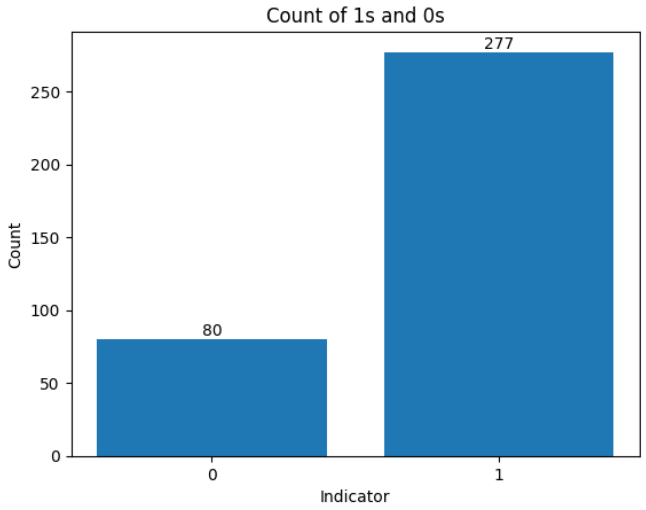


As seen in the chart above, we gladly see the weighted_sentiment_score has the biggest importance, which supports our initial hypothesis that twitter sentiment does play a part in the stock market.

3. Results and Discussion

In this paper, we experimented with multiple machine learning models and our 2 highest performing models are Linear Regression and Random Forest. But looking at the feature importance chart of the Linear Regression model, twitter sentiment does not play a big part when predicting the stock movement, whereas for the Random Forest model, we see a significant improvement in both robustness and performance after hyperparameter tuning. And we can see that twitter sentiment does play a part in predicting stock movement in the Random Forest model.

Acknowledging a limitation that we have in our dataset is there is an imbalance in the 1s and 0s as we can see in the bar chart below for Apple Stock:



Although our best performing Random Forest model has the potential of predicting stock prices with a relatively high accuracy, this limitation would hugely influence the credibility of our model.

We would like to acknowledge that there are plenty of approaches when tackling this problem and our approach may not be the best way. To further evaluate and create a model that is able to predict stock market movement with a credible accuracy, we would have to experiment with multiple other approaches in tackling this problem as well as other machine learning models which we did not manage to explore given our time constraint.

But in this paper, we can conclude that twitter sentiment does play a part in predicting stock market movement as we can observe in the feature importance chart.

References

Mehtab, S., Sen, J. (2019). A Robust Predictive Model for Stock Price Prediction Using Deep Learning and Natural Language Processing. *arXiv:1912.07700 [q-fin.ST]*.
https://doi.org/10.48550/arXiv.1912.07700

Puh, K., & Bagić Babac, M. (2023). Predicting stock market using natural language processing. *American Journal of Business*, *38*(2), 41-61.
https://doi.org/10.1108/AJB-08-2022-0124

Sonkiya, P., et al. (2021). Stock price prediction using BERT and GAN. *arXiv:2107.09055 [q-fin.ST]*.
https://doi.org/10.48550/arXiv.2107.09055

(PDF) machine learning: A review on Binary Classification. (n.d.). https://www.researchgate.net/publication/313779520_Machine_Learning_A_Review_on_Binary_Classification

Sabaren, L. N., Mascheroni, M. A., Greiner, C. L., & Irrazábal, E. (1970, January 1). A systematic literature review in cross-browser testing. *Journal of Computer Science and Technology*. http://www.redalyc.org/journal/6380/638067784001/html/

Text mining in medicine – knowledge discovery and intelligent systems – KDIS – University of Córdoba. Knowledge Discovery and Intelligent Systems KDIS University of Crdoba. (n.d.). http://www.uco.es/kdis/textminingmedicine/

Forecasting: Principles and practice (3rd ed). OTexts. (n.d.). https://otexts.com/fpp3/

Fan, C., Chen, M., Wang, X., Wang, J., & Huang, B. (2021, February 15). A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data. *Frontiers*. https://www.frontiersin.org/articles/10.3389/fenrg.2021.652801/full

https://github.com/anubhavanand12qw/STOCK-PRICE-PREDICTION-USING-TWITTER-SENTIMENT-ANALYSIS

Koehrsen, W. (2018, January 10). *Hyperparameter tuning the random forest in python*. Medium. https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74#:~:text=In%20the%20case%20of%20a,each%20node%20learned%20during%20training