UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Sentimental Analysis on StockTwits

Gwendoline Hays-Valentin, Hugo Michel, Thomas Rigou, Charaf Zguiouar

Professor: Thomas Renault
Applied Big Data Analytics
-
M2 Finance Technology Data - Sorbonne School Of Economics

Université Paris 1 Panthéon-Sorbonne
Paris, France, December 2023

Code Github: Here

Contents

List of Tables

1 Introduction

Social media, particularly X (ex Twitter), has become a focal point for studying public sentiment, offering insights into stock market movements. Previous research suggests a correlation between aggregated Twitter sentiments and the Dow Jones Industrial Average Index (DJIA), sparking interest in stock market prediction based on public mood. For instance, Pagolu (2017) shown that a strong correlation exists between the rise and falls in stock prices with the public sentiments in tweets. In fact, positive news and tweets in social media about a company would definitely encourage people to invest in the stocks of that company and as a result the stock price of that company would increase.

Several approaches for predicting stock prices have been presented over the years. In general, they are divided into four groups. The first is basic analysis, which is based on publicly available financial information. The second kind is technical analysis, which involves making recommendations based on previous data and pricing. The third includes the use of Machine Learning (ML) and Data Mining (DM) massive volumes of data gathered from various sources. The last is Sentiment Analysis (SA), which makes predictions based on previously published news, articles, or blogs. Kordonis (2016) employ various machine learning techniques such as Naive Bayes Bernoulli classification and Support Vector Machine (SVM), for providing a positive or negative sentiment on the tweet corpus. He found correlation between sentiment of tweets and stock prices. Gupta (2020) investigate the impact of sentiment expressed through StockTwits on stock price prediction using machine learning algorithm and demonstrate the proposed work on stock price prediction is demonstrated through experiments on five companies (Apple, Amazon, General Electric, Microsoft, and Target) using nine-month StockTwits data and daily stock data. Batra (2018) performed sentiment analysis on tweets related to Apple products, which are extracted from StockTwits from 2010 to 2017 and show that there is positive relation between people opinion and market data and proposed work has an accuracy of 76.65% in stock prediction.

Nowadays, the arrival of deep learning models such as transformer based model reshuffle the cards of the sentiment analysis. For instance, Aysun Bozanta (2022) compare the performances of various traditional, deep learning, and state-of-art pre-trained transformer models for text classification of tweets related to the stock market. The numerical study showed that RoBERTa outperformed traditional classifiers and deep learning algorithms in terms of average F1-scores

Through this research, we seek to bridge the gap between sentiment analysis and investment decision-making, emphasizing the pivotal role sentiments play in market perceptions and behaviors. By exploring various methodologies in terms of preprocessing and employing different machine learning model, the study aims to provide investors with better tools to distill insights from financial tweets.

2 Data Collection

The data collection process involved gathering a substantial volume of messages from the social media platform **StockTwits**, focusing on discussions related to the ticker symbol **AAPL** representing Apple Inc. To collect this data, we utilized the StockTwits API, which allowed us to retrieve tweets along with their associated sentiments. Given the nature of our supervised machine learning approach, where the sentiment of each tweet serves as the true label for our dataset, it was imperative to filter the tweets to retain only those for which sentiment data was available. These sentiments, categorized as **bullish** or **bearish**, provide the label for training our model and evaluating its performance.

Upon collection, we structured our dataset into two distinct sets: one balanced dataset comprising 500,000 positive messages and 500 000 negative messages, and another imbalanced dataset with 800,000 positive messages and 200 000 negative messages. This deliberate variation in dataset composition allows for a comparative analysis of model performance under different data distributions, offering insights into the impact of dataset balance on sentiment prediction accuracy.

To efficiently manage and access the collected data, we leveraged the MongoDB Cloud database. MongoDB is a document-oriented database, meaning that each document corresponds to a collected tweet. In our database, each tweet is represented as a document containing key-value pairs encapsulating essential attributes such as the tweet's unique identifier, timestamp, content, and true sentiment label. For example, a tweet's document might include fields such as *"_id"* for the unique identifier, *"date"* for the timestamp, *"content"* for the tweet text, and *"true_sentiment"* for the sentiment label (see figure .1). This document-centric approach to data storage provides flexibility and scalability, facilitating efficient retrieval and analysis of tweet data for subsequent model training and evaluation.

3 Benchmark Model

3.1 Size impact

In this section, we establish a benchmark model using the Naive Bayes Classifier with 10-fold cross-validation and unigram text features, without any pre-processing. Hence, the raw data is utilized for model training and evaluation. The primary evaluation metrics used are *Accuracy (AC)* and the *Matthews correlation coefficient (MCC)*, considering various dataset sizes for both balanced and unbalanced datasets without any pre-processing

The choice of the Naive Bayes Classifier with unigram text features serves as a baseline for comparison due to its simplicity and effectiveness in text classification tasks. We opt for the *CountVectorizer()* function to create the *Bag of Words (BoW)*

representation (i.e. numerical representation of words), which is commonly used in text analysis.

As expected, the table .1 shows that the accuracy of the classification demonstrates a strong positive correlation with the size of the dataset. For the balanced dataset, the accuracy increases progressively from 55.60% for a dataset of 500 messages to 73.63% for a dataset of 1 000 000 messages. However, the marginal improvement diminishes as the dataset size grows. Doubling the dataset size from 500 000 messages to one million only results in a 0.6 percentage point increase in accuracy, highlighting diminishing returns in accuracy gains with larger datasets. This suggests that while having more data generally improves classification accuracy, a satisfactory level of accuracy can be achieved with a dataset size ranging from 100 000 to 250 000 messages.

Similarly, on the unbalanced dataset, accuracy increases from 78.20% for a dataset of 1000 messages to 82.48% for a dataset of one million messages. This reaffirms the notion that larger datasets tend to yield higher accuracy levels. However, the rate of improvement in accuracy diminishes as the dataset size increases, indicating the diminishing marginal returns associated with dataset expansion. This underscores the importance of striking a balance between dataset size and computational resources, with dataset sizes ranging from 100 000 to 250 000 messages offering a viable compromise between accuracy and resource efficiency.

To provide a way to compare the results between the balanced dataset and unbalanced dataset we created two additional tables of results. The first one considers the Balanced Accuracy score instead without any further changes, while the second one further inversely weights the class labels during training such that losses from negative tweets would count 4 times more than the positive ones. We can see that the balanced accuracy of the first table is way lower, starting at only 50% which is no better than a coin flip, and going up to 67.4%, or 6% less than for its balanced counterpart. Though, when we adjust class weights we can see that the accuracy almost matches the score of its balanced counterpart, indicating that unbalancedness, when dealt with inverse frequency scaling, do not significantly affect the result given that the size of the dataset is large enough.

3.2 Ngrams impact

In this section, we delve into the impact of considering n-grams, contiguous sequences of words, as text features in the benchmark model, which is the Naive Bayes Classifier.

To conduct this analysis, we utilize a balanced dataset comprising 250 000 tweets. We evaluate the accuracy of the classification when features consist of unigrams, unigrams and bigrams (sequences of two words), trigrams, and four-grams.

The table .2 indicates that incorporating bigrams significantly enhances the accuracy of the classification. As highlighted by Mahmoudi (2018), the classification accuracy (MCC) increases from 72.11% when features are solely composed of unigrams to 74.98% when both unigrams and bigrams are considered as text features. This improvement can be attributed to bigrams capturing more context around each word, thereby facilitating a better understanding of word sequences, particularly those preceded by negating words like *"not"*. However, including trigrams and four-grams does not yield significant improvements in classification precision. This finding aligns with the observations of Wang (2012), suggesting that beyond bigrams, the marginal benefits of considering longer n-grams diminish. This findings underscores the importance of striking a balance between the computational complexity introduced by longer n-grams and the potential improvements in classification accuracy they offer.

4 Data Processing

The pre-processing step of textual data plays a crucial role in enhancing the performance of machine learning classifiers. Different pre-processing strategies can significantly impact the predictive accuracy of models. In this section, we explore various pre-processing techniques and evaluate their effectiveness in improving sentiment analysis on financial tweets. Our objective is to identify the optimal pre-processing method that should be implemented before running a classifier. We present the results in the table .3.

1. **Removing Stopwords using NLTK Corpus**: We begin by eliminating stopwords, common words like "the," "is," and "and," using the NLTK (Natural Language Toolkit) corpus. However, we find that this strategy significantly decreases the accuracy of the classification. This unexpected result is attributed to the inclusion of stopwords relevant to sentiment analysis in finance, such as "up," "down," "below," or "above." Saif (2014) support this finding, suggesting that using pre-existing lists of stopwords negatively impacts sentiment classification performance for short messages on social media.

2. **Stemming Text using PorterStemmer**: We apply stemming to reduce words to their base or root form using the PorterStemmer algorithm. However, stemming also decreases the accuracy of classification and performs worst overall with an accuracy of only 62.64%. This is consistent with previous findings, such as those documented in Renault (2018), indicating that stemming may not improve accuracy, especially when dealing with large datasets. For example, words like "short," "shorts," "shorted," and "shorter" convey distinct sentiments in finance, and stemming them to a common root may lead to a loss of information.

3. **POS Tagging using Penn Treebank Part-of-Speech Tagset**: We perform Part-of-Speech (POS) tagging using the Penn Treebank tagset to categorize words into their respective parts of speech. However, this pre-processing step also results in a slight decrease in accuracy for our classification tasks.

4. **Inclusion of Punctuation and Signs**: We include specific punctuation marks and signs like *"!", "?", "%", "+", "-", "=", ":", ";", ")", "(", and "]"* in the text data. We find that including punctuation improves the precision of classification by 0.28 percentage points. This finding aligns with previous research by Renault (2018), indicating that punctuation can enhance sentiment analysis performance.

5. **Include Emojis**: We enrich the text data by adding emojis. Incorporating emojis into the dataset leads to an increase in precision by 0.5 percentage points. This result is consistent with Mahmoudi (2018), highlighting the significance of emojis in sentiment analysis, especially in social media contexts.

6. **Combination of Emojis and Punctuation**: This approach yields an accuracy of 77.34%, outperforming both Punctuation and Emojis when taken separately. Emojis and punctuations are prevalent in social media discourse and including them as features can improve sentiment analysis performance.

7. **TweetTokenizer**: This approach yields the highest score with an accuracy of 78.18%, outperforming all other preprocessing methods. Indeed, while correctly incorporating Emojis and punctuation, the TweetTokenizer further stems repetition of characters up to a certain length, for exemple, with the default settings the token "ahhhhhhhhh" would become "ahhh", which reduces the size of the number of unique tokens while preserving the initial meanings of such writing style, inherent to social media contents.

8. **DicoSlang** : As we are on a social network, we notice that users use a lot of abbreviations. We decided to use a dictionary that allows abbreviated words to be exchanged with full words to see if this improved accuracy and MCC. The accuracy of 72.19% results in a slight decrease in accuracy for our classification tasks. Replacing slang with formal terms in sentiment analysis can dilute emotional nuances and affect model accuracy due to increased data sparsity and inconsistencies in text interpretation. Such replacements also alter the vocabulary, impacting the effectiveness of models like CountVectorizer, and removing informal tone indicators, crucial for interpreting sentiments in social media texts.

In summary, our analysis reveals that while some pre-processing techniques like removing stopwords and stemming may seem intuitive, they can actually degrade classification accuracy, particularly in the context of financial sentiment analysis on tweets. Incorporating emojis and punctuation, on the other hand, proves to be highly beneficial for enhancing model precision, emphasizing the importance of considering diverse pre-processing strategies in text analysis tasks. Overall we would advice for choosing preprocessing methods that are tailored to the problem at hand whenever available or feasible, as can be seen from the increase and decrease in performance accuracy when using standardize methods (PorterStemmer, StopWords, DicoSlang...) versus specific ones (Punctuation, Emojis, TweetTokenizer).

5 Model Comparison

In this section, we compare several machine learning algorithms to predict the sentiment (bearish or bullish) of sentiment tweets. The objective is to investigate whether more complex machine learning methods consistently outperform simpler techniques.

With Grid Search

The machine learning models considered for comparison are logistic regression, support vector machine, random forest, and multilayer perceptron and Vader algorithm. To find the optimal combination of hyperparameters for each model, we employ *GridSearchCV* with threefold cross-validation. Our pre-processing pipeline includes the inclusion of emojis and punctuation, and we impose a minimum word frequency of 0.0001% (or 25 occurrences) to remove very infrequent words and reduce computation time for *GridSearchCV*. This is achieved by setting the *min_df* parameter of the *CountVectorizer*.

For training and evaluation, each model is trained using threefold cross-validation on a dataset comprising 250,000 messages, evenly split between positive and negative sentiments. We utilize unigram and bigram as text features, along with the TweetTokenizer whom yielded the highest score among preprocessing methods. We report the accuracy of the final classifier on the test data as well as the time needed to fit the model that yielded the highest result during Cross-Validation.

The table .5 illustrates that more complex algorithms such as Vader, Random Forest and Multilayer Perceptron do not demonstrate significant improvement in classification precision compared to simpler methods like Logistic Regression or Support Vector Machine. Despite the potential of complex models to capture intricate patterns in the data, the associated costs of hyperparameter optimization (time, complexity, lack of transparency, and computing power) outweigh the marginal gains in accuracy.

We conclude that for social media sentiment analysis tasks, simple classifiers such as Naive Bayes, Support Vector Machine, or Logistic Regression are often sufficient. This finding aligns with previous research by Antweiler (2004) and Sprenger (2014), indicating that simpler models can effectively capture sentiment trends in social media data without the need for complex algorithms.

With Optuna

We decided to perform the performance of various machine learning algorithms after hyperparameter tuning with Optuna. Optuna's efficient optimization algorithm employs a Bayesian approach to search the hyperparameter space and is capable of handling a wide range of parameter types. The Support Vector Machine (SVM) shows significant improvement in accuracy with a finely tuned regularization parameter C. The Multi-Layer Perceptron (MLP), with its optimized α for regularization and learning rate initialization, demonstrates the highest accuracy among the models, suggesting a well-tuned balance between bias and variance in the network's learning process. Lastly, the Random Forest (RF) classifier's performance underperform the Grid Search's tuning. These results underscore the effectiveness of Optuna in enhancing model performance through meticulous hyperparameter tuning.

6 Investor sentiment and stock returns

6.1 Sentiment indicators

In the absence of a methodology for calculating the average daily sentiment in the paper, we have developed an approach to compute an average daily score. This score is based on the ratio of positively predicted tweets to the total number of tweets sent between T-1 after 4 PM and T until 4 PM Eastern Time each day. We calculate this average using a benchmark method (Naive Bayes with no preprocessing and only unigrams) and a more advanced Naive Bayes model that includes preprocessing, emoji interpretation, and bigrams. Initially, we considered averaging out the predicted probabilities for each tweet on a daily basis, but ultimately, we settled on the ratio approach due to its reproducibility. We have graphed both measures against Apple's (AAPL) returns and price data since the year 2018. For further details, please refer to the appendix.

6.2 Correlations

In the correlation matrix presented (please refer to the appendix), the *Return* variable is correlated with both *Positive_ratio_bigram* and *Positive Ratio Unigram*. The correlation between *Return* and *Positive_ratio_bigram* is highlighted as 0.39, and between *Return* and *Positive Ratio Unigram* as 0.34. The correlation coefficients are stronger when using the bigram method compared to the unigram. This suggests that including bigram analysis provides a more sensitive or relevant measure of sentiment in relation to stock returns than just analyzing single words (unigram), potentially capturing more complex expressions of sentiment that are more predictive of stock movement. This is consistent with the findings of the paper.

6.3 Regressions

6.3.1 Forecasting Sentiment

$$\text{Sentiment (t)} = \beta_0 + \beta_1 \text{Sentiment(t-1)} + \beta_2 \text{Return(t-1)} + \epsilon \tag{6.1}$$

We explored predicting the next period's sentiment using the last average sentiment and last return, we did this from 2018 to 2024 as an extension of the paper's approach.

The bigram method exhibits superior model performance compared to the unigram approach, the higher R-squared value (0.409 compared to 0.361). This indicates that the bigram model, which accounts for pairs of words, explains a greater proportion of the variance in positive sentiment ratios. Similarly, the adjusted R-squared values, which account for the number of predictors used, affirm the bigram model's enhanced fit.

Both models display statistically significant constants, although the constant in the bigram model is slightly lower, suggesting a different baseline sentiment when considering word pairs. The coefficients for lagged sentiment (*Positive_ratio_lag1* and *Positive_ratio_bigram_lag1*) are both positive and significant, indicating a continuity in sentiment over time. Notably, lagged returns show a significant negative relationship with sentiment in both models, with the effect being more pronounced in the bigram model. This could imply that previous returns might inversely impact sentiment more noticeably when analyzed through the lens of bigrams.

6.3.2 Forecasting Returns

$$\text{Return (t)} = \beta_0 + \beta_1 \text{Sentiment(t-1)} + \beta_2 \text{Return(t-1)} + \epsilon \tag{6.2}$$

Consistently with the paper's results and with the literature, forecasting future returns with past sentiment yields no significant results whether in the bigram technique's case or the unigram technique's case. The R-squared values for both models are 0.000. Neither unigrams nor bigrams effectively capture the variance in stock returns based on past sentiment and returns. The negative adjusted R-squared values further emphasize the models' inability to account for stock returns, highlighting their overall inadequacy for this task. Both models show low F-statistics and high probabilities (well above the typical significance threshold of 0.05), indicating that the models do not fit the data well and the predictors are not statistically significant.

NB: Standard errors were computed using White's heteroskedasticity robust standard errors

7 Conclusion

Sentiment analysis are widely used and goes beyond price movements, providing a holistic view that considers qualitative factors. By analyzing sentiments in financial tweets, investors can identify trends, catalysts, and anomalies, empowering them to make informed decisions.

In this paper, we offer guidance for finance researchers on deriving sentiment indicators from social media text. We prioritize dataset size over pre-processing and algorithm choice. Utilizing pre-labeled data from platforms like StockTwits can provide researchers with large datasets, avoiding manual annotation. We show the importance of text pre-processing, particularly the positive impact of including emojis and punctuation. Unlike traditional media articles, social media messages require tailored pre-processing methods. We show that complex algorithms may not enhance accuracy, advocating for simpler models like Naive Bayes and Logistic Regression. Despite the belief in sophisticated models, simplicity often suffices for sentiment analysis. In exploring sentiment's link with stock returns, we find no empirical evidence for its predictive power on large-cap stocks at a daily frequency. Previous literature relying on smaller datasets and machine learning methods may need reassessment in light of our findings.

References

Antweiler, W. (2004). Is all that talk just noise? the information content of internet stock message boards.

Aysun Bozanta (2022). Sentiment analysis of stocktwits using transformer models.

Batra, R. (2018). Integrating stocktwits with sentiment analysis for better prediction of stock price movement.

Gupta, R. (2020). Sentiment analysis for stock price prediction.

Kordonis, J. (2016). Stock price forecasting via sentiment analysis on twitter.

Mahmoudi, N. (2018). Deep neural networks understand investors better decision support systems.

Pagolu, V. S. (2017). Sentiment analysis of twitter data for predicting stock market movements.

Renault (2018). Intraday online investor sentiment and return patterns in the us stock market.

Saif (2014). On stopwords, fltering and data sparsity for sentiment analysis of twitter.

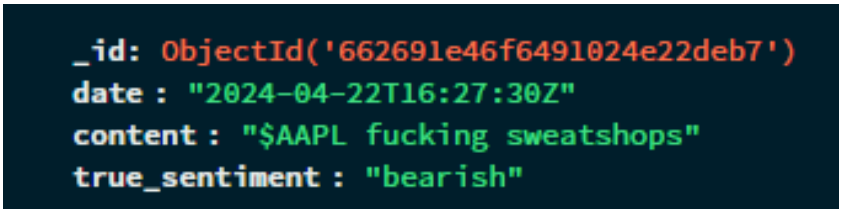
Sprenger, T. (2014). News or noise? using twitter to identify and understand company-specifc news fow.

Wang, S. (2012). Baselines and bigrams: Simple, good sentiment and topic classifcation, in 'proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2.

Appendix

.1 Example of tweet stored in the MongoDB Cloud database

Figure .1: Tweet stored in JSON format



.2 Impact of size dataset

Table .1: Size of the dataset and Classification Accuracy

Panel A: Balanced Dataset		
Dataset Size	**Accuracy (%)**	**MCC**
500	0.556000	0.113326
1000	0.599000	0.197996
2500	0.614400	0.229231
5000	0.632800	0.265649
10000	0.650000	0.300162
25000	0.685480	0.371189
50000	0.702760	0.406013
100000	0.710000	0.420605
250000	0.721776	0.444175
500000	0.729682	0.460252
1000000	0.736317	0.473638

Panel B: Unbalanced Dataset		
Dataset Size	**Accuracy (%)**	**MCC**
500	0.782000	0.003534
1000	0.800000	0.009626
2500	0.807600	0.066034
5000	0.791600	0.124845
10000	0.797800	0.190525
25000	0.798640	0.231514
50000	0.799720	0.269425
100000	0.807190	0.317422
250000	0.814072	0.355603
500000	0.819240	0.378025
1000000	0.824828	0.395147

Panel C: Unbalanced Dataset with Balanced Accuracy		
Dataset Size	**Balanced Accuracy (%)**	**MCC**
500	50.9492	0.054983
1000	51.3714	0.071349
2500	51.3989	0.073226
5000	52.3516	0.103883
10000	55.0270	0.168528
25000	57.9864	0.219437
50000	61.3960	0.283372
100000	63.4930	0.316829
250000	65.6096	0.356054
500000	66.5809	0.377616
1000000	67.4082	0.397544

Panel D: Unbalanced Dataset with Balanced Accuracy and Class Weight Training		
Dataset Size	**Balanced Accuracy (%)**	**MCC**
500	56.3314	0.137182
1000	58.0096	0.147212
2500	57.5080	0.134901
5000	59.0621	0.159166
10000	62.4286	0.216252
25000	65.5217	0.264185
50000	67.8060	0.299521
100000	69.3292	0.322809
250000	71.2222	0.352542
500000	72.3626	0.370008
1000000	73.2466	0.383842

.3 Impact of N-grams

Table .2: Impact of N-grams on Classification Accuracy

N-grams	Accuracy (%)	MCC
Unigrams	72.1128	0.443133
Unigrams + Bigrams	74.9872	0.501025
Unigrams + Bigrams + Trigrams	75.0364	0.502947
Unigrams + Bigrams + Trigrams + 4-gram	74.7816	0.499229

.4 Impact of pre-processing

Table .3: Pre-processing Performance Comparison

Pre-processing Method	Accuracy (%)	MCC
Benchmark	76.7256	0.535504
Punctuation	77.0060	0.541255
Stemmer	62.4724	0.261860
Emojis	77.2316	0.545332
StopWords	74.8520	0.498224
Emojis + Punctuation	77.3428	0.547655
PosTagging	76.6048	0.534285
Tweet Tokenizer	78.1828	0.564614
DicoSlang	72.19	0.4446

.5 Comparison of Machine Learning Algorithms

.5.1 With Grid Search

Table .4: Comparison of Machine Learning Algorithms

Algorithm	Accuracy (%)	Time
Multinomial Naive Bayes	78.38	2 s
Logistic Regression	77.7	6 s
Support Vector Machine	77.03	25 s (no parallelization)
Random Forest	77.8	22 s
Multilayer Perceptron	78.37	2m 40 s
Vader algorithm	45	15 s

.5.2 With Optuna

Table .5: Comparison of Machine Learning Algorithms After Hyperparameter Tuning

Algorithm	Best Parameters	Accuracy (%)
Support Vector Machine (SVM)	$C = 0.01587$	77.738
Multi-Layer Perceptron (MLP)	$\alpha = 0.01103$, learning rate init = 0.000213	78.984
Random Forest (RF)	n_estimators = 178, max_depth = 42	73.072

.6 SVM Model: Classification Report and Confusion Matrix

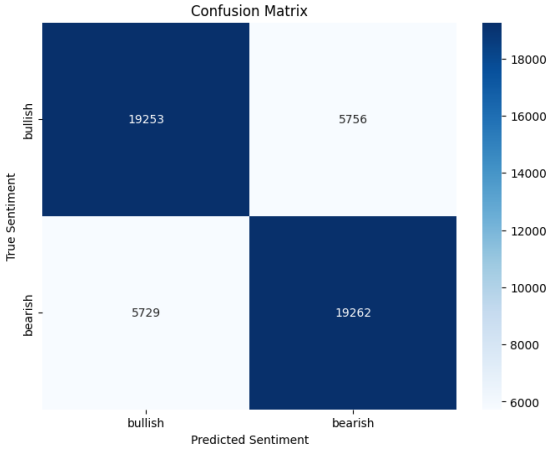


Figure .2: Confusion Matrix: SVM

```
######## Classification Report ########
              precision    recall  f1-score   support

         0.0       0.77      0.77      0.77     25009
         1.0       0.77      0.77      0.77     24991

    accuracy                           0.77     50000
   macro avg       0.77      0.77      0.77     50000
weighted avg       0.77      0.77      0.77     50000

######## Accuracy Score ########
0.77
```

Figure .3: Classification Report: SVM

.7 Random Forest Model: Classification Report and Confusion Matrix

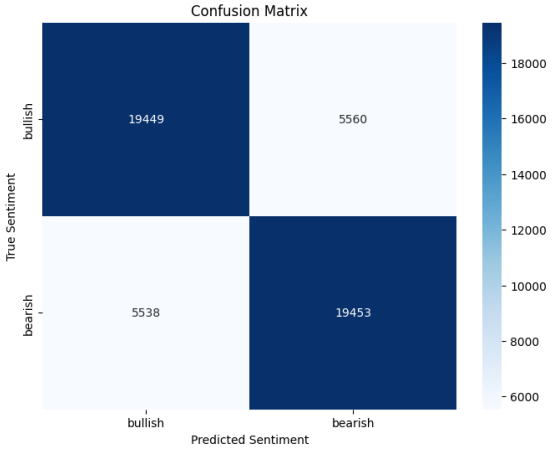


Figure .4: Confusion Matrix: Random Forest

```
######## Classification Report ########
              precision    recall  f1-score   support

         0.0       0.78      0.78      0.78     25009
         1.0       0.78      0.78      0.78     24991

    accuracy                           0.78     50000
   macro avg       0.78      0.78      0.78     50000
weighted avg       0.78      0.78      0.78     50000

######## Accuracy Score ########
0.78
```

Figure .5: Classification Report: Random Forest

.8 Logistic Regression Model: Classification Report and Confusion Matrix

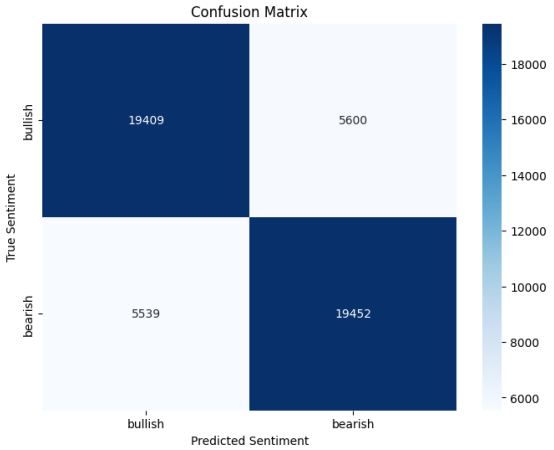


Figure .6: Confusion Matrix: Logistic Regression

```
######## Classification Report ########
              precision    recall  f1-score   support

         0.0       0.78      0.78      0.78     25009
         1.0       0.78      0.78      0.78     24991

    accuracy                           0.78     50000
   macro avg       0.78      0.78      0.78     50000
weighted avg       0.78      0.78      0.78     50000

######## Accuracy Score ########
0.78
```

Figure .7: Classification Report: Logistic Regression

.9 MultiLayer Perceptron: Classification Report and Confusion Matrix

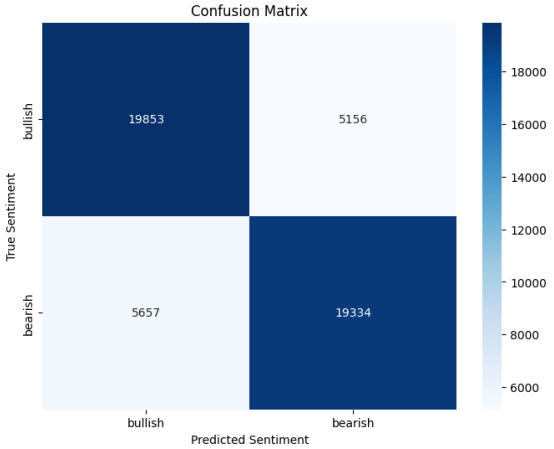


Figure .8: Confusion Matrix: MultiLayer Perceptron

```
######## Classification Report ########
              precision    recall  f1-score   support

         0.0       0.78      0.79      0.79     25009
         1.0       0.79      0.77      0.78     24991

    accuracy                           0.78     50000
   macro avg       0.78      0.78      0.78     50000
weighted avg       0.78      0.78      0.78     50000

######## Accuracy Score ########
0.78
```

Figure .9: Classification Report: MultiLayer Perceptron

.10 Vader Algorithm : Classification Report and Confusion Matrix

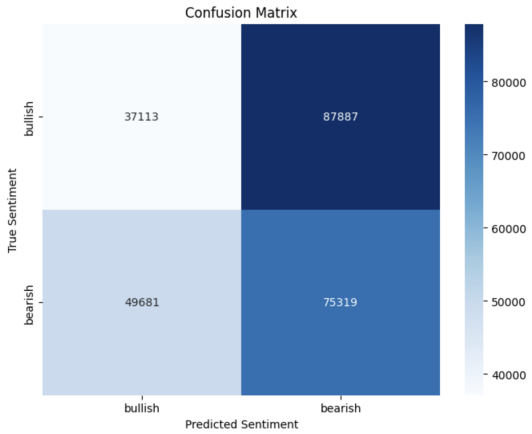


Figure .10: Confusion Matrix: Vader Algorithm

```
######## Classification Report ########
              precision    recall  f1-score   support

         0.0       0.43      0.30      0.35    125000
         1.0       0.46      0.60      0.52    125000

    accuracy                           0.45    250000
   macro avg       0.44      0.45      0.44    250000
weighted avg       0.44      0.45      0.44    250000

######## Accuracy Score ########
0.45
```

Figure .11: Classification Report: Vader Algorithm

A Investor Sentiment

A.1 Graphs

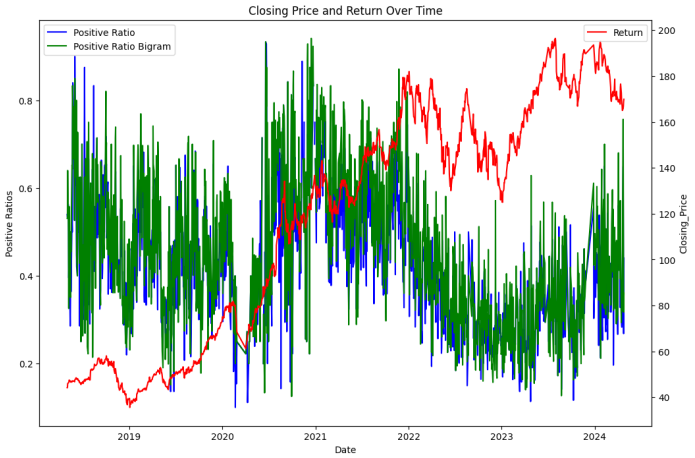


Figure A.1: Daily price and daily sentiment calculated using both a unigram-based and bigram-based techniques

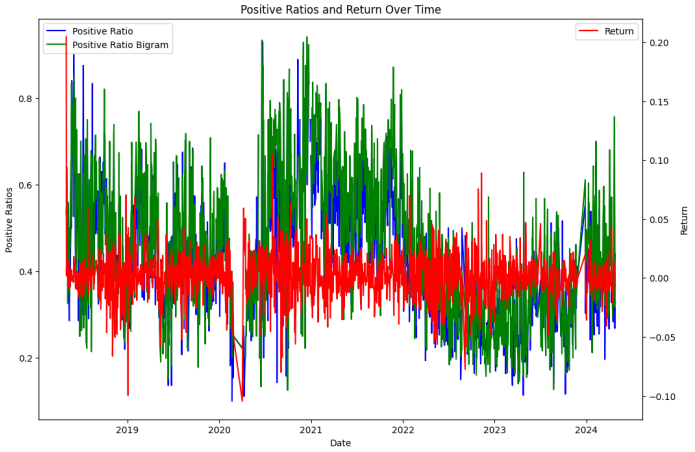


Figure A.2: Daily returns and daily sentiment calculated using both a unigram-based and bigram-based techniques

A.2 Tables

A.2.1 Correlations

	Positive_ratio_bigram	Positive Ratio Unigram	Return	Closing Price
Positive_ratio_bigram	1.000000	0.836012	**0.387585**	0.351200
Positive Ratio Unigram	0.836012	1.000000	**0.344269**	0.387706
Return	**0.387585**	**0.344269**	1.000000	0.145902
Closing Price	0.351200	0.387706	0.145902	1.000000

A.3 Regressions :

article booktabs

Table A.1: Comparison of OLS Regression Results for Unigram and Bigram Methods

Statistic	**Unigram**	**Bigram**
Dep. Variable	Postive_ratio	Postive_ratio_bigram
R-squared	0.361	0.409
Adj. R-squared	0.360	0.408
F-statistic	338.3	424.7
Prob (F-statistic)	1.89e-120	1.09e-144
Log-Likelihood	1048.4	909.52
AIC	-2091	-1813
BIC	-2075	-1797
Coefficient: Const	0.1655	0.1561
Std Err: Const	0.010	0.011
P>|z|: Const	0.000	0.000
Coefficient: Lag1	0.6126	0.6622
Std Err: Lag1	0.024	0.023
P>|z|: Lag1	0.000	0.000
Coefficient: Return_Lag1	-0.4771	-0.8681
Std Err: Return_Lag1	0.155	0.179
P>|z|: Return_Lag1	0.002	0.000

Table A.2: Comparison of OLS Regression Results for Forecasting Returns using Unigram and Bigram Techniques

Statistic	**Unigram**	**Bigram**
Dep. Variable	Return	Return
R-squared	0.000	0.000
Adj. R-squared	-0.001	-0.001
F-statistic	0.1490	0.1790
Prob (F-statistic)	0.862	0.836
Log-Likelihood	3561.7	3561.8
AIC	-7117	-7118
BIC	-7102	-7102
Coefficient: Const	0.0011	0.0014
Std Err: Const	0.002	0.002
P>|z|: Const	0.513	0.369
Coefficient: Positive Ratio Lag1	-0.0002	-0.0008
Std Err: Positive Ratio Lag1	0.004	0.003
P>|z|: Positive Ratio Lag1	0.949	0.784
Coefficient: Return Lag1	-0.0170	-0.0154
Std Err: Return Lag1	0.033	0.033
P>|z|: Return Lag1	0.602	0.635