Predictive Analysis of Stock Market Volume: *A Comparative Study of Machine Learning Models*

Bryan Lu   
Masters of Science in Data Analytics  
University of Central Oviedo, US  
br532303@ucf.edu  
  
Maria Greenfield  
Masters of Science in Data Analytics  
University of Central Oviedo, US   
ma411084@ucf.edu  
  
Brandon Diaz Lopez  
Masters of Science in Data Analytics  
University of Central Oviedo, US  
br673118@ucf.edu  
  
Jeremy Bouhadana  
Masters of Science in Data Analytics  
University of Central Oviedo, US  
je829493@ucf.edu

# Introduction

In this rising age of technology, everyday people around the world are able to access information on stocks and financial investing. As a result, they also have the potential to influence the market in a multitude of ways. In previous years, the driving factors of the stock market included the company’s performance and future prospects. However, with the rise of the internet comes increased ability to gain and share information almost instantaneously. Highly influential people can now share information, such as opinions on the fluctuations of the market and what they are investing in, with everyday people and inspire them to invest on their own through a platform like Twitter. This trend has inspired numerous analyses on the topic, and the goal of our project will be to determine whether or not media buzz on twitter has a real influence on a stock’s performance.

The initial plan was to collect live data from Twitter on a few select companies and perform our analysis on the number of mentions and sentiments gathered, similar to another analysis performed my Matthew Wallach [6, 7]. However, Twitter's updated policies made it difficult to gain software developer access to the tweets from which we would collect our data, which lead to a shift in approach. Instead, we used an alternative dataset from Kaggle [1], which had available tweet counts and sentiments as well as stock volume (of shares) and price information relating to Netflix stock data from January 2018 to November 2022.

# The Dataset and Cleaning

This dataset included sentiments categorized as positive, negative, or neutral (0), derived from a large number of tweets analyzed using the roBERTa-base model [3]. The Kaggle dataset featured sentiment scores represented by P\_mean (average sentiment for the day) and P\_sum (sum of sentiment scores for the day). The scraped Twitter data required data cleaning, involving steps like lowercasing, removing links, hashtags, symbols, and numerical values, translating emojis, tokenizing sentences, removing stop words and punctuation, and lemmatization. Stock data (volume of shares and price) extracted from Yahoo Finance is already a cleaned dataset.

The data cleansing we performed on the twitter data involved dropping rows associated with missing values and binarizing P\_mean into P\_mean\_bin, where a 0 indicates negative sentiment and 1 indicates positive sentiment. Additionally, we standardized the data to have a mean of zero and a standard deviation of one, as is commonly required for many machine learning algorithms. The overall dataset included the following feature variables to be used as predictors:

1. The Open: The opening price of the company's stock at the beginning of the trading day. It represents the first transaction of the day and sets the initial value for the stock.
2. High: The highest price at which the company's stock was traded during the day. It indicates the peak value the stock reached during trading hours.
3. Low: The lowest price at which the company's stock was traded during the day. It signifies the lowest point the stock reached during trading hours.
4. Close: The closing price of the company's stock at the end of the trading day. It represents the final transaction for the day and is crucial for determining the day's overall performance.
5. Adj Close: The adjusted closing price of the company's stock. It considers factors such as dividends and stock splits, providing a more accurate representation of the stock's value over time.
6. P\_mean\_bin: Average sentiment on twitter regarding tweets about Netflix for a given day. The average was then binarized into either a 0 or 1 to indicate a positive or negative sentiment.
7. P\_sum: Sum of the overall sentiment on twitter regarding tweets about Netflix for a given day. A positive value indicates an overall more positive sentiment and a negative value indicates negative sentiment.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Open | High | Low | Close | Adj Close |
| Count | 1137 | 1137 | 1137 | 1137 | 1137 |
| Mean | 402.212181 | 408.380897 | 395.473237 | 402.128012 | 402.128012 |
| Std | 116.443388 | 117.223957 | 115.429014 | 116.216659 | 116.216659 |
| Min | 163.960007 | 172.059998 | 162.710007 | 166.369995 | 166.369995 |
| 25% | 319.880005 | 325.369995 | 313.5 | 319.959991 | 319.959991 |
| 59% | 370.26001 | 375.899994 | 363.329987 | 369.609985 | 369.609985 |
| 75% | 502.339996 | 508.549988 | 495 | 502.109985 | 502.109985 |
| max | 692.349976 | 700.98999 | 686.090027 | 691.690002 | 691.690002 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Volume | P\_mean | P\_sum | twt\_count | P\_mean\_bin |
| Count | 1137 | 1137 | 1137 | 1137 | 1137 |
| Mean | 7906694 | -0.070298 | -40.118734 | 491.086192 | 0.062445 |
| Std | 6770185 | 0.05014 | 124.198738 | 564.227246 | 0.242069 |
| Min | 1144000 | -0.258065 | -3034 | 156 | 0 |
| 25% | 4277600 | -0.100877 | -41 | 304 | 0 |
| 59% | 6179900 | -0.067485 | -24 | 387 | 0 |
| 75% | 9623100 | -0.035608 | -12 | 503 | 0 |
| max | 133387500 | 0.07563 | 110 | 12404 | 1 |

## The target variable in the dataset is volume, which refers to the volume of shares purchased of Netflix stock on a given day. P\_mean\_bin is considered a categorical variable, while all other predictors are numerical. The summary statistics of the cleaned dataset with the binarized variable P\_mean\_bin (excluding the date variable) are displayed below

# EDA

## Correlation Matrix

## The visual EDA’s we ran included a correlation matrix as well as scatterplots of each of the target variable against each of the predictors and histograms to display the distribution of each variable (in terms of density). The correlation matrix (**Figure 1**) showed that twt\_count and volume had the highest correlation with the target (0.7728), which is relatively strong. P\_sum had the second highest correlation with the target (-0.6770), which is moderate. The remaining predictors fell in the range of [-0.4, -0.25], which is a weak correlation. Additionally, the stock prices all had a perfect correlation with each other, indicating collinearity. Initially, we believed that P\_mean and P\_sum would have a strong correlation, indicating a collinear relationship. However, they had a correlation of 0.4 (weak). P\_sum and twt\_count had a very strong negative correlation (-0.91).

## Scatterplots

## Most of the scatterplots (**Figure 2** right) formed a near-horizontal line when plotted against the target, indicating little to no correlation between the variables. P\_sum, had a negative linear correlation with volume, twt\_count had a positive linear correlation with volume, and P\_mean\_bin had a categorical distribution.

## Histograms

## Through the histograms (**Figure 2** left), we observed that Open, High, Low, Close, and Adj Close all have bimodal distributions when plotted against volume. This is most likely due to the polarity of the sentiments. P\_mean\_bin displayed bimodality as well, which makes sense given it has been binarized into a categorical variable. Most notably, twt\_count, P\_sum, and (most importantly) Volume displayed a Poisson distribution, and P\_sum showed a near normal distribution with a very slight left skew.

A chart with different colored squares

Description automatically generated

Figure : Correlation Matrix of all variables

## A diagram of a city Description automatically generatedA graph of blue dots Description automatically generated

Figure 2: Histograms of each variable and Scatterplots of predictors against the target.

## Pairwise Scatterplots

## Lastly, we plotted the pairwise relationships between all variables to help identify patterns, detect outliers, and select features for our final model. The strength of a relationship between two predictors is useful for extracting meaningful relationships associated with the target variable and the pairwise predictors and can help identify relationships that could potentially influence the model. The stock price variables all displayed linear relationships with each other, indicating strong collinearity with each other. More notably, P\_sum and twt\_count displayed a fairly linear relationship as well. The remaining pairwise scatterplots did not yield much useful information.

A graph of a graph

Description automatically generated

Figure 3: Pairwise Scatterplots

# Experimental Plan & Research Methods

Given the natural complexity and often nonlinear nature of financial data, the initial EDA was conducted with the goal of identifying underlying patterns, anomalies, structures, and potential relationships within the data. Utilizing these findings from our initial EDA, the following methods were selected:

## Generalized Linear Models (GLMs)

Thus, with a positive target variable, two initial GLM models were created in the hopes of providing a natural way to model a response variable’s mean as a function of linear predictors. The coefficients were estimated using the Iteratively Re-weighted Least Squares (IRLS) method, which was conducted to help handle the non-linear relationships between predictors and the target variable.

1. The first model excluded features ‘Volume,’ ‘date,’ ‘P\_sum,’ and ‘twt\_count,’ focusing on a clean set of variables
2. The second model plots ‘twt\_count’ as a predictor against ‘Volume,’ to explore the relationships between tweet count and stock market volume.

## Decision Trees and Ensemble Methods:

*Decision Trees (DT’s)* were used to capture more complex non-linear relationships that might exist within financial data. DT’s were utilized in the hopes of achieving an intuitive model; however, in practice, suspecting that the base DT model was overfitting, feature importance-based selection was utilized in order to minimize complexity. The results were evaluated initially using Root Mean Squared Error (RMSE), and a different subset of features were then considered.

To mitigate the risk of overfitting from individual decision trees, it was prudent to also apply a *Random Forest Regressor*. Carefully selecting for our models hyperparameters, including the number of trees (n\_estimators) and the randomness seed, we ran several iterations to improve this performance.

Finally, we considered *Gradient Boosting Methods*. *Light Gradient-Boosting Machine* was chosen for its efficiency and speed in working with large datasets. By employing GBM’s, we incrementally built an ensemble by correcting errors of weak learnings, further improving weak accuracy. We particularly focused on hyperparameter tuning methods to ensure more optimal performance for each model, facilitating our interpretation of results and data. *Extreme Gradient Boosting (XGB)* and *Gradient Boosting (GBM)* were explored to leverage ensemble learning techniques for the gradient boosting framework. By sequentially building weak learners, these methods hope to correct errors from previous iterations, focusing on reducing bias and variance.

## Regularization Techniques

#### To avoid issues of overfitting and multicollinearity, Lasso and Ridge via Elastic Net Regression were utilized. These methods helped to regularize regression models by penalizing large coefficients. Throughout this project, the balance between L1 (Lasso) and L2 (Ridge) penalties were chosen through cross-validation.

## Dimensionality Reduction and Feature Selection:

1. *Principal Component Analysis (PCA):* Dimensionality reduction techniques were utilized throughout experimentation. This was done to focus on the most critical variables, while also mitigating the “curse of dimensionality.”
2. *Feature Engineering:* new features were often generated, and relevant features were selected in order to improve our models’ predictive power

## Classification Models:

1. *K-Nearest Neighbors (KNN):* This non-parametric, instance-based learning algorithm was employed in various iterations (including grid search, random search, and Bayesian Optimization) to find optimal hyperparameters. Different distance metrics were subsequently tested and compared.
2. *Support Vector Machines (SVM’s)* were explored, to potentially capture nonlinear relationships again between features and target variables using different kernel functions.
3. Finally, though not the main focus of this project, *Hierarchical Clustering* was applied to see how well it could segment data into distinct clusters that could provide insights *into* underlying structures.

## Evaluation Metrics:

The performance of each model using metrics appropriate for regression (i.e., MSE, RMSE, MAE, and R-squared) and classification (i.e., accuracy, precision, recall, F1 score). The choices in metrics provided comprehensive insights into each model’s predictive ability and performance.

By utilizing a wide variety of statistical and machine learning methods, we hoped to use a diverse but rigorous approach in predicting stock market volume. Each model was chosen to test a different set of hypotheses regarding the underlying data structures, and their performance was meticulously evaluated to arrive at an optimal model for the given task. This approach ensures robustness and reliability in predictions and provides a deeper understanding of the complex relationships that exist within variables related to stock market volume.

## Hyperparamter Tuning, Feature Engineering, and Dimensionality Reduction

For each of the above models, hyperparameter tuning was conducted in the effort to find the optimal model parameters. The techniques employed included Grid Search, Random Search, and Bayesian Optimization. Grid Search would systematically search through a manually subset of hyperparameters, Random Search would search the hyperparameter space for optimal combinations, and Bayesian Optimization utilized a probabilistic model to predict the objective function and select hyperparameters to potentially improve.

Similarly, we applied Principal Component Analysis (PCA) for dimensionality reduction when applicable to see if any performance gains would result. By transforming a set of features into uncorrelated variables known as principal components, the number of features could be reduced, helping with the “cure of dimensionality” where too many features might lead to overfitting. Furthermore, PCA helped in nose reduction and visualization. Interpretability is lost in the use of PCA, and PCA assumes that the underlying structure is linear, so this methodology was not always appropriate to employ.

Furthermore, feature engineering was conducted to try to enhance models’ performance, particularly in KNN. By creating new features based on domain knowledge, relationships, and interactions, feature engineering was employed to expose patterns that models might be able to exploit, resulting in better performance. However, there is a risk of overfitting, and increased complexity causes a new assortment of issues.

# Experimental Procedures and Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | MSE | RMSE | MAE | R-Squared |
| KNN (k=5) | 2.7426E+12 | 1656081.501 | 829119.6491 | 0.924284357 |
| KNN- gridsearch | 1.9512E+12 | 1396838.37 | 755957.3099 | 0.946134027 |
| KNN- randomsearch | 2.7426E+12 | 1656081.501 | 829119.6491 | 0.924284357 |
| KNN- Bayesian Optimization | 8.441E+12 | 2905340.925 | 1088032.895 | 0.766967549 |
| KNN- feature engineer | 6.3242E+12 | 2514800.546 | 1320547.08 | 0.834283751 |
| LightGBM- feature engineer | 3.2012E+12 | 1789183.268 | 427117.567 | 0.916118216 |
| GLM- multiple predictors | 1.0473E+14 | 10233544.35 | 3449881.873 | -1.28682633 |
| GLM - predictor: twt\_count | 4.95E+16 | 222568058.4 | 13430363.51 | -1080.699693 |
| Decision Tree- Train | 1214138883 | 34844.49574 | 1634.433443 | 0.999974808 |
| Decision Tree- Test | 3.2152E+13 | 5670303.87 | 3442893.86 | 0.112364084 |
| Random Forest- Train | 2.067E+12 | 1437702.708 | 646233.8075 | 0.957112947 |
| Random Forest- Test | 1.5875E+13 | 3984340.368 | 1728969.013 | 0.561736741 |
| SVM | 3.935E+13 | 6272962.202 | 3737470.891 | -0.086344429 |
| SVM- hyperparameter tuning | 3.865E+13 | 6216941.9 | 3727991.567 | -0.067028004 |
| Gradient Boosting | 7.0082E+13 | 8371527.98 | 4597249.123 | -0.934780343 |
| XGBoost base | 4.8796E+13 | 6985395.83 | 2475713.123 | -0.347113665 |
| XGBoost-hyperparamter tuning | 4.631E+13 | 6805114.613 | 2365720.625 | -0.278477498 |

Figure 4:MSE, RMSE, MAE, & R2 of the models

## Linear Models

The initial GLM model operated as the baseline comparison for any model comparison. A given GLM can be expressed in the form:

*g*(*μ*) *= g()*= *η* = *β0 +* *β1X1 + . . . + βpXp* 

Applying the various predictors used for GLM:

*g*(*μ*) *= g()*= 4.009\*10-08 + 5.242\*10-10 \* Open + (2) (-5.189\*10-09) \* High + 5.099\*10-09\*Low + 7.056\*10-12 \* Close + 7.056\*10-12 \* Adj\_Close + 1.964\*10-08 \* P\_mean

For Glm – predictor: twt\_count: y = 1.355\*10-06 + (3) (-1.421\*10-09) \* tweet\_count

Using this equation and applying our data, we found extremely high RMSE, MSE, and MAE values (**Figure 4**). Significant performance gains were observed, compared to MLR. However, this result still indicated overfitting. Dropping variables and focusing on our most important features yielded even more negative R2 values, potentially indicating the impracticality of fitting linear models on data containing complex relationships between variables.

## Advanced Models

To improve upon the baseline Linear Models, significant performance gains were observed from the utilization of advanced models.

Our Decision Tree (DT) method yielded an initial train MSE of 0, and a Test MSE of 3,953,278, which is an improvement over the GLM model, but indicates severe overfitting. Utilizing the built in feature importance package in Python, we calculate the feature importance’s, and used only ‘Low’ and “P\_Sum,” our two most important features to run our Decision Tree Regressor. With a train RMSE of 34844 and a test RMSE of 5660304, we again see improvement over the prior iteration, but still are likely overfitting. Looking at the train R2 and test R2 values of 0.99 and 0.11(**Figure 4**), we confirm that DT’s are overfitting.

To remedy this, the Random Forest Regressor ensemble method was utilized, which improves our model’s performance, suggesting that we have successfully captured more complexity with this model, providing a better generalization. However, comparing the train and test performance metrics (**Figure 4**), we can see again that this model is overfitting.

Since the Random Forest Regressor model performed somewhat better, a Random Forest Model seemed most appropriate for subsequent experimentation. A train RMSE of 1437703 and a test RMSE of 3984340 were calculated. These significantly high values indicate that the performance of RF models is subpar for this data. Furthermore, the train R2 and test R2 were calculated to be 0.96, and 0.56, respectively. This different again this model has learned the training data’s noise instead of the underlying patterns of interest, affecting its performance on new, unseen data (overfitting).

Gradient Boosting Methods were then utilized to try to improve performance. Gradient Boosting and XGBoosting both performed poorly, with negative R2 values. Furthermore, the MSE, MAE, and RMSE values were extremely high (**Figure 4**). Hyperparameter tuning improved results slightly, but yielded high MSE, MAE, and RMSE values still, additionally with negative R2 values.

However, LightGBM calculated train R2 and test R2 of 0.75 and 0.91, respectively. RMSE, MSE, and MAE values were still high, but compared to other models, we still see some performance gains. Furthermore, utilizing hyperparameter tuning, we implemented cross-validation, using regularization techniques to bring these values down. This model perhaps performed better than our other boosting methods, due to LightGBM’s ability to handle categorical variables more efficiently.

Next, classification models were considered for modeling purposes. K-Nearest Neighbors produced extremely high MSE values, with similarly high values for RMSE and MAE values; however, the RMSE and MAE values are slightly lower compared to GLM or DT’s. Moreover, the R-Squared value jumped to 0.94, the highest yet observed. Furthermore, we utilized the KNN to do a binary prediction on our test variable, ‘Volume,’ for high and low classification. These metrics result in very high accuracy, precision, recall and F1 score, depicted in **Figure 5** below.

A bar chart with different colored rectangles

Description automatically generated

Figure 5: KNN classification performance (Accuracy, Precision, Recall, F1 score)

Support Vector Machines (SVM) was experimented with to capture the possible nonlinear relationships that likely exist between features and target variable, using different kernel functions. SVM had the highest MSE observed of any model: indicating poor performance and predictive ability. Furthermore, with an R2 value of -0.09, this clearly indicates the model is not suitable for this particular dataset, since a simple model that always predicts the mean of the target variable would have performed better than the current SVM model. Even with hyperparameter tuning and some improvements made to SVM, the reported R2 values are consistently negative.

## Herirachical Clustering

It is pertinent to note that this project primarily focuses on the performances of models in predicting volume traded within stocks. Since hierarchical clustering is an unsupervised learning method, compared to SVM or KNN (known supervised learning algorithms for classification and regression), SVM and KNN are not as readily interpretable, particularly SVM, with non-linear kernels, focused on creating decision boundaries or utilizing neighbors for predictions. Furthermore, since hierarchical clustering is not a predictive model, comparisons between SVM and KNN are not as appropriate. A silhouette score was calculated to be -0.321, which for a range of value from -1 to 1, our performance in this base model is not ideal. As such, Hierarchical clustering was conducted to demonstrate proof of concept (**Figure 6**), but was not used any further for analysis.

A diagram of a clustering diagram

Description automatically generated

Figure 6: Hierarchical clustering dendrogram (complete linkage)

## Correlated vs. Uncorrelated Trees: Interpretations and model selection Equations

Significant improvements were made to ensure that our individual decision trees would have low correlation. Since bootstrapping the training data and using random subsets of features for each tree helps to reduce the variance of the model, errors in individual trees are less likely to be consistent across the entire ensemble. As such, this uncorrelated approach enabled the Random Forest model to capture more complex aspects of the stock market volume.

However, unlike Random Forest, the trees within Gradient Boosting models can handle correlated trees. Having sequentially built trees, with each tree learning from the mistakes of previous iterations, the correlated nature of trees allows the model to reduce bias and adapt to more complex patterns, whilst also making the model more susceptible to overfitting.

It was found that LightGBM compared to Random Forest, performs slightly better and provides a higher accuracy. For our analysis, understanding the impact of tree correlation helps guide us in the modeling process, contributing to effectively selecting a model that captures the underlying dynamics of a market.

## Model Selection

The final model selection was based on a combination of statistical accuracy and interpretability. In considering a model with the best performance, KNN provides the most obvious performance gains in terms of both R-squared and other error metric, with the highest R2 and lowest RMSE, MAE, and MSE values. Further evaluation using Random Search, Bayesian Optimization, feature engineering, PCA, and hyperparameter tuning yielded no better results compared to KNN utilizing Grid Search. Furthermore, the model performed with near perfect classification metrics. However, it is important to note that due to unusually high values, the initial KNN model might be indicative of overfitting or target leakage. Further analysis would require validation of the model using unseen data and cross-validation to access generalization abilities.

Finally, if interpretability is crucial for interpretations, it may be important to consider an alternative model, rather than using KNN. For example, LightGBM might offer more insight into feature importance and relationships. Such a task is challenging to extract from a KNN model. As such, in weighing interpretability against performance is paramount to model selection, KNN is a better performing model, but LightGBM is the best performing interpretable model utilized.

# Conclusion

## Data and Feature selection

The complexity in capturing relationships between volatile variables in financial data should not be understated. The fundamental challenge in predicting stock market volumes is an inherently complex and multifaceted task. Through the application of basic models such as GLM, or more advanced techniques like Random Forest and KNN, a slightly deeper understanding of the underlying patterns and relationships can be elucidated. Future work might explore additional ensemble methods, deep learning techniques, and rigorous feature selection processes to further refine and improve predictive performance. Furthermore, additional work might apply time series models, since the data has a temporal component. The focus of this project ignores the usage of time series models; however, certain models, such as ARIMA or LSTM, may be relevant for continuation of this work: by providing models capable of capturing more complex relationships and nuances within data, time series models might be more appropriate for modeling financial data.

Moreover, feature selection techniques such as recursive feature elimination might be applied in finding the most important features. As one of the most critical and important aspects of this project, the data selection and feature selection methods were paramount to building a successful model. Insights derived from this project may have various applications in the finance section, which may pave the way for more sophisticated algorithmic trading strategies or risk management tools. Overall, this project represents an initial contribution to understanding stock market dynamics and offers a foundation for future investigation and applications.

##### References

1. Atef, K. (2022) *Netflex stock dataset with Twitter sentiment*, *Kaggle*. Available at: https://www.kaggle.com/datasets/kirolosatef/netflex-stock-dataset-with-twitter-sentiment (Accessed: 27 June 2023).
2. Barbieri, F. *et al.* (2020) ‘Tweeteval: Unified benchmark and comparative evaluation for Tweet Classification’, *Findings of the Association for Computational Linguistics: EMNLP 2020* [Preprint]. doi:10.18653/v1/2020.findings-emnlp.148.
3. *Cardiffnlp/twitter-roberta-base-sentiment · hugging face* (no date) *cardiffnlp/twitter-roberta-base-sentiment · Hugging Face*. Available at: https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment (Accessed: 27 June 2023).
4. Cardiffnlp (no date) *CARDIFFNLP/Tweeteval: Repository for tweeteval*, *GitHub*. Available at: https://github.com/cardiffnlp/tweeteval (Accessed: 27 June 2023).
5. KerolosAtef (no date) *KEROLOSATEF/stock-market-prediction-using-sentiment-analysis-of-twitter*, *GitHub*. Available at: https://github.com/KerolosAtef/Stock-market-prediction-using-sentiment-analysis-of-twitter (Accessed: 27 June 2023).
6. Wallach, D. (2018) *Financial tweets*, *Kaggle*. Available at: https://www.kaggle.com/datasets/davidwallach/financial-tweets (Accessed: 31 July 2023).
7. dwallach1 (no date) *Dwallach1/Stockerbot: Twitter bot to follow financial trends in publicly traded companies*, *GitHub*. Available at: https://github.com/dwallach1/StockerBot (Accessed: 31 July 2023).