Predicting Stock Market Returns Using a Twitter Sentiment Analysis

Vidur Arun, Abdulla Kerimov

Introduction

Across the years, multiple instances have shown that certain tweets can affect their targeted stock's returns, like with Elon Musk in 2018 (whose tweet caused Tesla's stock price to increase by 6% [1]) and Donald Trump in 2017 (whose tweet caused Amazon's stock market valuation to drop by USD$5bn [2]). It is instances like these that show us that stock prices are not solely influenced by data like financial statements (such as a firm's profits, costs and revenue) or economic indicators (such as a firm's growth), but investor sentiments as well.

Social media platforms such as Twitter serve as a real-time measure of the mood of the public, and utilizing Machine Learning to predict stock prices using a twitter sentiment analysis allows for data analysis to find the correlation between these variables as well as providing useful insights into market inefficiencies, offering new opportunities for investment and trading tactics.

Multiple studies have been conducted in the past to provide valuable insights about predicting stock returns using a twitter sentiment analysis, and their correlation. A paper by **Christian Palomo** [3] focused on developing an NLP model to predict certain stock prices by directly analyzing the sentiment of a tweet using a transformer based neural network, a method that differs greatly from traditional Machine Learning and Natural Language Processing techniques. Palomo's work utilized the dataset ***Twitter-Financial News Sentiment*** from the **HuggingFace** website, containing 12,424 entries of finance-related tweets. Each entry had been split into three categories based on whether the tweet corresponds to an increase in stock price, a decrease in stock price, or neither. Multiple models for specific months were created and tested. The model developed in this study aimed to predict the stock movements of Tesla from August 2022 through December 2022. The model's results from November 2022 had an accuracy of 82%, with its predictions matching the stock movements of 19 out of the 22 days the market was open for.

Moreover, **Anshul Mital** and **Aprit Goel** [4] had conducted a similar study in which they aimed to find the correlation between public and market sentiments. Mital and Goel targeted values from the Dow Jones Industrial Average (DJIA) from June 2009 to December 2009, obtaining their data from ***Yahoo! Finance*** (containing the opening prices, closing prices, as well as the highest and lowest prices) and publicly available Twitter data (containing more than 476 million tweets including the timestamp, username and tweet text). Mital and Goel preprocessed their data by utilizing a concave function ($(y + x)/2$: where x is the DJIA value on a given day, and y is the

next available data point with n number of days between x and y), adjusting stock values by shifting prices up/down for jumps/falls with a large magnitude, and removing periods of significant volatility to prune their dataset. Next, a sentiment analysis of tweets was conducted, classifying tweets as either positive or negative. However, Mital and Goel also used a much lesser-known form of classification: multi-class classification, in which they used four mood classes — Calm, Happy, Alert, and Kind. Mital and Goel then finished preprocessing their data by generating a word list (based on the Profile of Mood States), filtering tweets, computing a daily score for every POMS word on a given day, and mapping those scores. To train and test models, Mital and Goel used four different algorithms: Linear Regression, Logistic Regression, SVMs and Self-Organizing Fuzzy Neural Networks (SOFNNs). The accuracies of these models were derived using *k-fold sequential cross validation* where k was 5. The results documented in this study show that the SOFNNs performed the best out of the four algorithms with an accuracy of around 75.56%.

The objective of this study is to understand the effect of tweets on their targeted stock's price. Additionally, we investigate how long, if any, the effect of the tweet lasts and/or impacts the stock price into the near future.

Dataset

Tweet Sentiment's Impact on Stock Returns is a dataset that can be accessed from Kaggle [5]. It contains 862,231 labelled tweets and their respective stock returns. Each tweet had their date of the tweet extracted as well as the company the stock is aimed at. Furthermore, all labelled tweets have been assigned a polarity (the tone of the tweet) that will aid the user in their analysis of the data using machine learning.

Tweet Sentiment's Impact on Stock Returns contains several columns affiliated with tweets and their targeted stock's returns or losses. Every entry in this dataset contains the **TWEET** (The text of the tweet), the **STOCK** (stock mentioned in the tweet), and the **DATE** (The date at which the tweet was posted). Additionally, the dataset also contains the **LAST_PRICE** (The targeted stock's price at the time of tweeting), **PX_VOLUME** (The volume of shares traded at the time of tweeting), **VOLATILITY_10D** and **VOLATILITY_30D** (The targeted stock's volatility across a 10 and 30-day window), and **1_DAY_RETURN, 2_DAY_RETURN, 3_DAY_RETURN**, and

7_DAY_RETURN (The returns of losses of the targeted stock 1, 2, 3, and 7 days after the tweet was posted). The sentiment of each tweet can be found in 2 columns: **LSTM_POLARITY** (The polarity of the tweet obtained by using a LSTM Neural-Network), and **TEXTBLOB_POLARITY** (The sentiment of each tweet obtained using TextBlob). Finally, **MENTION** shows users the number of times a company was mentioned in a labelled tweet.

Methods and Models

Input Features and Target Output

We filtered the dataset to tweets related to Apple. This was done to ensure that any noise was left out, allowing for a more in-depth analysis of the relationship between tweets and a certain company's stock returns.

We created multiple models using **1_DAY**_RETURN, 2**_DAY_RETURN**, **3_DAY_RETURN**, **7_DAY_RETURN**, and **LSTM_POLARITY**, as shown in Table 1. We started off by giving the X-column 1 feature: **LSTM_POLARITY** and giving the Y-column the **7_DAY_RETURN** feature. Then, I created another model by adding **1_DAY_RETURN** to the X-column. The process was repeated for **7_DAY_RETUTN** until the features had the **LSTM_POLARITY** feature and the *X_DAY_RETURN* of the days less than the *X_DAY_RETURN* in the Y-column. This was repeated for **1_DAY_RETURN**, **2_DAY_RETURN**, and **3_DAY_RETURN**. This ensured that all possibilities were accounted for, allowing for a better analysis of results.

Table 1: Input features and target outputs of each model

Model No.	Input Features	Output Target
1	LSTM_POLARITY	1_DAY_RETURN
2	LSTM_POLARITY	2_DAY_RETURN
3	LSTM_POLARITY, 1_DAY_RETURN	2_DAY_RETURN
4	LSTM_POLARITY	3_DAY_RETURN

5	LSTM_POLARITY, 1_DAY_RETURN	3_DAY_RETURN
6	LSTM_POLARITY, 1_DAY_RETURN, 2_DAY_RETURN	3_DAY_RETURN
7	LSTM_POLARITY	7_DAY_RETURN
8	LSTM_POLARITY, 1_DAY_RETURN	7_DAY_RETURN
9	LSTM_POLARITY, 1_DAY_RETURN, 2_DAY_RETURN	7_DAY_RETURN
10	LSTM_POLARITY, 1_DAY_RETURN, 2_DAY_RETURN, 3_DAY_RETURN	7_DAY_RETURN

Train-Test Split

Developing any type of machine learning model involves splitting the data into training and testing data. The model developed in this study utilized a time-series train-test split instead of the more conventional random train-test split [6]. Initially, we had split the data using the train-test split function. However, after further inspection of the dataset, it became apparent that there were multiple tweets posted on the same day. This became a problem as it provided the model with some of the values it needed to predict in during its training process, resulting in testing accuracies increasing without the model being able to fully learn about the relationships between variables. On the other hand, using a time-series split allows the data to be separated according to the date each tweet was posted. This means that tweets posted on the same day will be grouped together into either the training or testing sets, preventing the model from coming across information that would prevent it from fully learning about the patterns in the dataset.

Models and Hyperparameter Tuning

After splitting the model, we implemented linear regression and Random Forest baseline models. Random Forest is a supervised-learning algorithm that utilizes an ensemble learning method for regression. The tool obtains its results by creating several decision trees (each tree makes its own predictions) and creates an output by using either the mode of the classes, or the mean prediction

for classification and linear regression respectively [7]. Random forest has several parameters that can be fine-tuned to ensure that the model outputs the highest possible score. We chose to tune ***max_features***, ***min_samples_split***, ***n_estimators*** and ***max_depth***. ***max_features*** regulates the maximum number of features a given decision tree is allowed to use in order to make a prediction. ***n_estimators*** is the number of decision trees used in the random forest model. ***min_samples_split*** is used to define the minimum number of samples required to split a node in a random forest. ***max_depth*** defines the depth of each decision tree in the forest [8].

We used R^2 and mean-squared error (MSE) as our evaluation metrics for this study, and we compared the training scores to the testing scores of the evaluation metrics to determine whether the models were being overfitted. R^2 is used to determine the proportion of variance in the target output, measure how well it can predict outcomes in the future and reflect how well a model fits the dataset on a scale of 0 to 1, with larger values of R^2 indicating a better fit and smaller values indicating a worse fit. The value of R^2 can be obtained using the formula: $R^2 = 1 - \left(\frac{SSE}{SST}\right)$ [9], where SSE is the sum of squares of errors (sum of squared differences between training and testing values) and SST is the total sum of squares (sum of squared differences between individual training values and the mean of the training values).

MSE is a loss function and measures the average of squared differences between the model's predictions and the actual values of those predictions. A lower MSE indicates that the model's predictions are getting close to the actual values and vice versa for a higher MSE. Values for this loss function can be obtained using the formula: $MSE = \left(\frac{1}{n}\right)\sum_{i=1}^{n}\left(Y_i - \hat{Y}^i\right)^2$ [10], where n is the number of data points, i is the index value of each data point, Y_i is the actual value of the ith data point, and $\hat{Y}_i$ is the model's prediction of the ith data point. It is important to note that MSE places a bigger weight on larger differences because it takes its square, making the function sensitive to outliers in the dataset.

After developing a baseline random forest model, we tuned it using a *GridSearch* approach — a process where we alter the values of the chosen parameters to get the best possible result. *GridSearch* is an optimization function that tries every combination using the provided parameter values to find the best model [11]. We had specified the ***scoring*** to be R^2, ***cv*** (cross-validation strategy) to be the time-series split. The value of the ***n_estimators*** parameter remained constant

for all models: 500. The value of ***max_features*** ranged from 1 to the number of features minus 1 with an increment of 1. The value of ***min_samples_split*** varied from 10 to 200 with an increment of 5.

SHAP Analysis

In this study, a SHAP analysis will be conducted to determine the impact that the dataset's features have on our model's predictions of stock returns after a certain number of days the tweet was posted. Using a SHAP analysis is an integral part in finding out how much each feature contributes to the model's predictions. SHAP is based on Shapley Values, a game theory framework developed by Lloyd Shapley, and can be used to interpret any type of machine learning model [12]. Features with a positive SHAP value will positively affect the model's prediction, and vice versa for features with a negative SHAP value. Every feature in the SHAP analysis is given an importance value which represents that feature's contribution to the model's output, doing so allows for the calculation of the extra contributions made by each feature.

Conducting a SHAP analysis of features will allow us to visualize the contributions made by features. This is extremely important, especially when the model's results are unexpected. A SHAP analysis will help us find the features that made the most contributions to a model's prediction, and features that made the least contributions: this analysis may help us identify whether **LSTM_POLARITY** has an impact on the model's predicted stock returns.

Results and Discussion

Model Results: Best Parameters, Train and Test MSE and R^2 Values

In this section, we will be discussing the models developed in this study, their results, best parameters, and train and test R^2 scores. We developed 10 models to take all possibilities into account, and we obtained their train and test R^2 scores as well as MSE values. After that, we tuned the models and used GridSearch to find the best parameters. These will help us determine whether the models are able to make predictions using the provided input features.

Table 2 shows that models predicting a certain X_DAY_RETURN (Models 1, 2, 4, and 7) using LSTM_POLARITY as its only input feature have extremely low training and testing R^2 scores. The model predicting 2_DAY_RETURN using LSTM_POLARITY as its only input feature had R^2 scores of 7.68×10^{-6} and 0.000166 for the training and testing set respectively. Low R^2 scores in models like these indicate that the model performs poorly when it comes to its predicting power, and they also indicate that these models were unable to explain the majority of variances within the dataset.

On the other hand, models that used multiple features such as model 10 (where LSTM_POLARITY, 1_DAY_RETURN, 2_DAY_RETURN, and 3_DAY_RETURN were used to predict 7_DAY_RETURN) performed marginally better compared to models 1, 2, 4, and 7. Model 10 obtained a training R^2 score of 0.999988 and a testing R^2 score of 0.996942. These R^2 scores indicate that model 10 was able to explain the variances within the dataset and make accurate predictions, while also having minimal overfitting and extremely low MSE values.

A similar pattern can be seen in model 9 (where LSTM_POLARITY, 1_DAY_RETURN, and 2_DAY_RETURN were used to predict 7_DAY_RETURN). This model had a training R^2 score of 0.956834 and a testing R^2 score of 0.819039. These scores indicate that model 9 was able to make a significant number of accurate predictions during both training and testing. However, some overfitting was present, evident in the 14% decrease in R^2 scores from training to testing. Nevertheless, the model still had considerably low MSE values during training and testing, indicating that the model can make good generalizations about the dataset.

While most models are able to make good generalizations about the dataset, some models still experience overfitting. One such model is model 8 (where LSTM_POLARITY, and 1_DAY_RETURN were used to predict 7_DAY_RETURN), which obtained a training R^2 score of 0.876224 and a testing R^2 score of 0.628146. The variance of these scores indicates some overfitting and shows that model 8, while still being able to make some generalizations, was unable to make generalizations to the same degree of accuracy as it did during training, suggesting that it may have learnt the patterns in the training set a little too well, causing it to perform worse during testing.

Table 2: Best parameters, train and test R^2 and MSE for each model

Model No.	Input Features	Target Output	Best Parameters	R^2 (Train)	R^2 (Test)	MSE (Train)	MSE (Test)
1	LSTM_POLARITY	1_DAY_RETURN	max_features: 1, min_samples_split: 80, n_estimators: 500	0.001099	0.000166	-0.000142	0.000224
2	LSTM_POLARITY	2_DAY_RETURN	max_features: 1, min_samples_split: 190, n_estimators: 500	0.0000077	0.000397	-0.054058	0.000173
3	LSTM_POLARITY, 1_DAY-RETURN	2_DAY_RETURN	max_features: 1, min_samples_split: 10, n_estimators: 500	0.970368	0.942775	0.000012	0.0000094
4	LSTM_POLARITY	3_DAY_RETURN	max_features: 1, min_samples_split: 74, n_estimators: 500	0.000055	-0.039843	0.000562	0.000433
5	LSTM_POLARITY, 1_DAY-RETURN	3_DAY_RETURN	max_features: 1, min_samples_split: 50, n_estimators: 500	0.944464	0.87072	0.000031	0.000054
6	LSTM_POLARITY, 1_DAY-RETURN, 2_DAY_RETURN	3_DAY_RETURN	max_features: 2, min_samples_split: 80, n_estimators: 500	0.986152	0.980537	0.0000078	0.0000081
7	LSTM_POLARITY	7_DAY_RETURN	max_features: 1, min_samples_split: 50, n_estimators: 500	0.017493	-0.723959	0.00135	0.001036
8	LSTM_POLARITY, 1_DAY-RETURN	7_DAY_RETURN	max_features: 1, min_samples_split: 5, n_estimators: 500	0.876224	0.628146	0.00017	0.000223

9	LSTM_POLARITY, 1_DAY-RETURN, 2_DAY_RETURN	7_DAY_RETURN	max_features: 2, min_samples_split: 10, n_estimators: 500	0.956834	0.819039	0.000059	0.000109
10	LSTM_POLARITY, 1_DAY-RETURN, 2_DAY_RETURN, 3_DAY_RETURN	7_DAY_RETURN	max_features: 2, min_samples_split: 70, n_estimators: 500	0.999988	0.996942	0.000000016	0.0000018

SHAP Analysis

This section will analyze the SHAP plots for the models that predict 7_DAY_RETURN (models 7-10). We decided to choose these 4 models because the models predicting 7_DAY_RETURN were the ones that performed relatively better compared to the other models. Using the SHAP plots will allow us to understand the marginal contribution of the features on the models' predictions.

Figure 1 displays the SHAP plots of model 7 (where the input feature is LSTM_POLARITY, and the target output is 7_DAY_RETURN). The summary plot shows us that higher LSTM_POLARITY values (red) are pushing the model's predictions towards negative values, and lower LSTM_POLARITY values (blue) are pushing the model's predictions towards positive values. The bar plot shows us that the average SHAP value for LSTM_POLARITY is slightly greater than 0, indicating that the feature has a small but positive effect on the target output. Figure 1 revealed that LSTM_POLARITY does not have a major impact on the model's predictions.

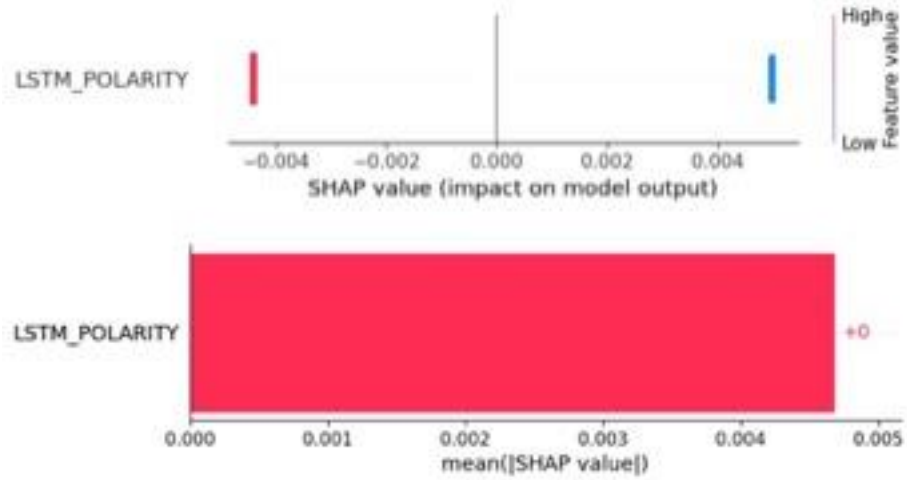

Figure 1. SHAP plots of model 7

Figure 2 displays the SHAP plots for model 8 (input features for this model are LSTM_POLARITY and 1_DAY_RETURN, and the target output is 7_DAY_RETURN). The summary plot shows us that higher LSTM_POLARITY values slightly push the model's predictions to the left, and lower LSTM_POLARITY values slightly push the model's predictions towards the right, indicating that LSTM_POLARITY has a small impact on the model's predictions. Moreover, the plot also shows us that some higher and lower 1_DAY_RETURN values can push the model's predictions to the right, and some can also push the model's predictions towards the left. The bar plot shows us that 1_DAY_RETURN has a much larger impact on the model's predictions compared to LSTM_POLARITY. It shows us that 1_DAY_RETURN increases the model's predictions by 0.02 on average, whereas LSTM_POLARITY's contribution is almost negligible.

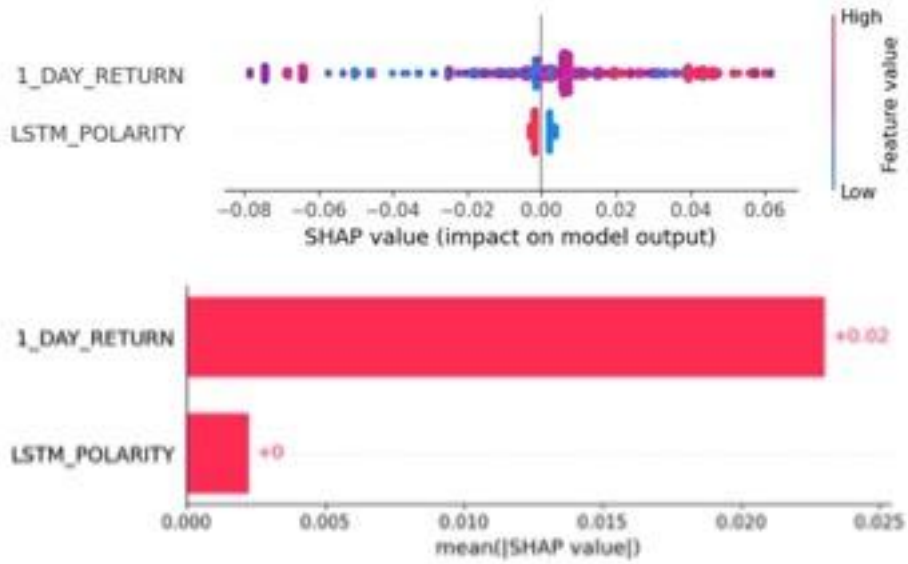

Figure 2. SHAP plots for model 8

Figure 3 displays the SHAP plots for model 9 (where the input features are LSTM_POLARITY, 1_DAY_RETURN and 2_DAY_RETURN, and the target output is 7_DAY_RETURN). The summary plot shows us that LSTM_POLARITY's contribution to the model's predictions is almost negligible, and that compared to Figure 2, 1_DAY_RETURN's contributions seem to have decreased. Nevertheless, both high and low values of this feature push the model's predictions to

the left, and some push the model's predictions to the right. The summary plot also shows us that 2_DAY_RETURN has the biggest impact on the model's predictions, and that higher values for this feature positively impact the model's predictions, and vice versa for lower values. The bar plot shows us that 2_DAY_RETURN increases the model's predictions by 0.02 on average, 1_DAY_RETURN increases the model's predictions by 0.01 on average (lower than the value displayed in Figure 1), and LSTM_POLARITY has a near negligible effect on the model's predictions.

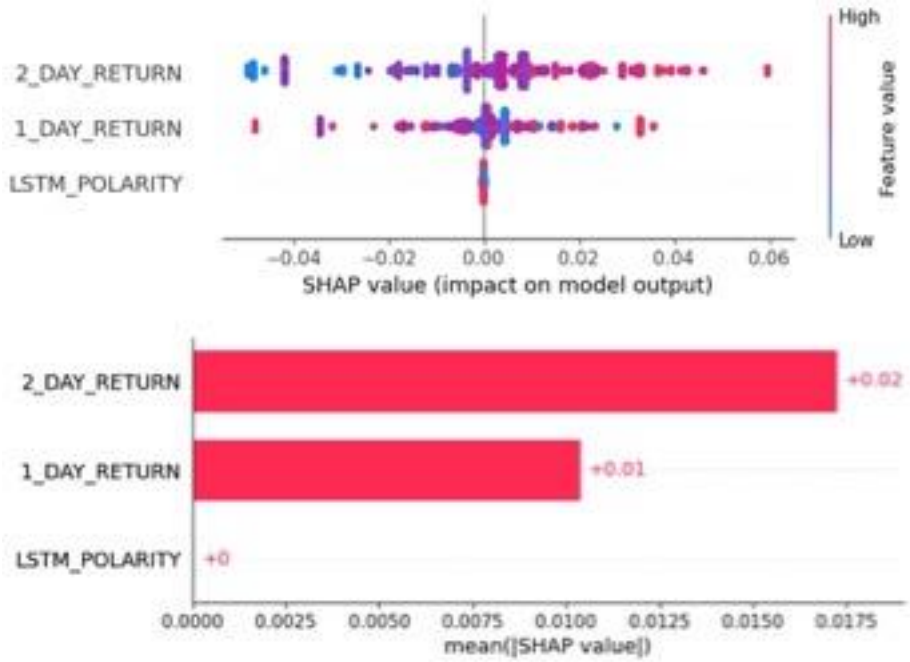

Figure 3. SHAP plots for model 9

Figure 4 displays the SHAP plots for model 10 (where the input features are LSTM_POLARITY, 1_DAY_RETURN, 2_DAY_RETURN and 3_DAY_RETURN, and the target output is 7_DAY_RETURN). The summary plot shows us that 3_DAY_RETURN had the largest contribution to the model's predictions, with higher values of this feature positively impacting its predictions, and vice versa for lower values. 2_DAY_RETURN has similar results to that of 3_DAY_RETURN. However, for 1_DAY_RETURN, it appears as if higher values of this feature are influencing the model's predictions to a higher degree compared to lower values of this feature, and LSTM_POLARITY seems to have the smallest contribution out of the four input features. The bar plot shows us that 3_DAY_RETURN increases the model's predictions by 0.02 on average (a

noticeable trend is that the greatest X_DAY_RETURN seems to increase the model's predictions by 0.02 on average). It also shows us that 1_DAY_RETURN and 2_DAY_RETURN increase the model's predictions by 0.01 each, and that LSTM_POLARITY has a negligible contribution.

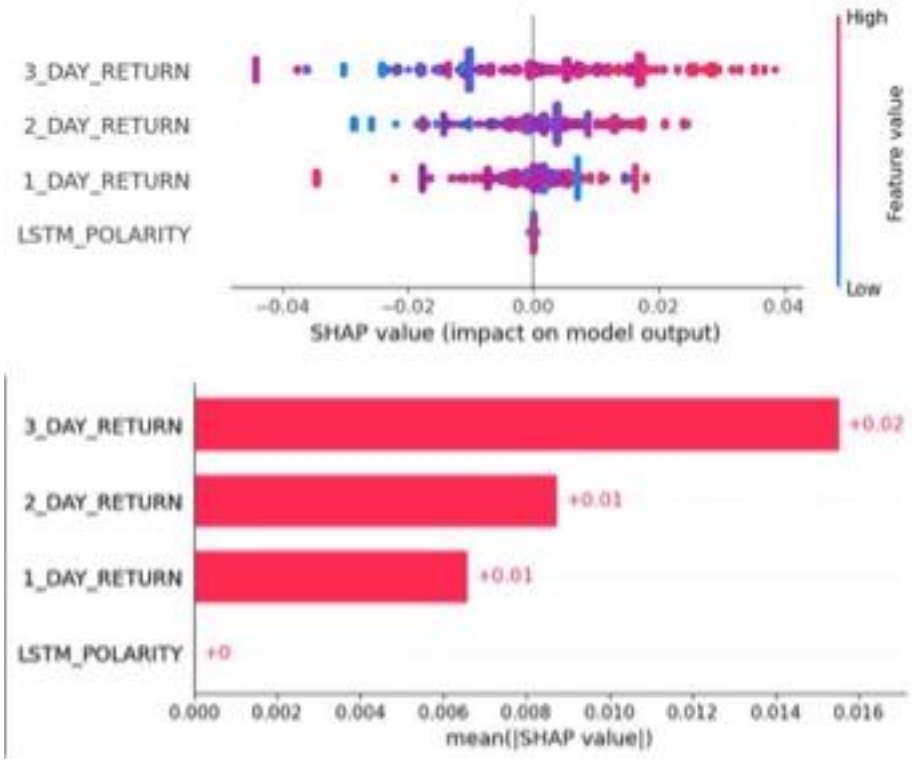

Figure 3. SHAP plots for model 10

Conclusion

This study aimed to understand the effects of tweets on their targeted stock's returns, and how long these effects lasted for. We developed various machine learning models to explore this relationship, and we used the features provided in the dataset (LSTM_POLARITY, 1_DAY_RETURN, 2_DAY_RETURN, 3_DAY_RETURN, and 7_DAY_RETURN). We also filtered our dataset to specifically target tweets related to Apple to ensure that our analysis would be as accurate as possible.

Using the previously mentioned scores, values and parameters, we concluded that model 10 (input features were LSTM_POLARITY, 1_DAY_RETURN, 2_DAY_RETURN, and 3_DAY_RETURN, and the target output was 7_DAY_RETURN) had performed the best (see

table 2) and had obtained R^2 scores of 0.999 and 0.996 for training and testing respectively. We believe that this may have occurred because this model had the greatest amount of input features, thus feeding the model more information to help it find patterns and make generalizations about the dataset.

Finally, we conducted a SHAP analysis of models 7-10 to understand the marginal contributions of the input features on the model's predictions. These plots showed us that LSTM_POLARITY had a negligible effect on the model's predictions, with the other input features having a much more significant contribution. This is also evident in the models' results, as models with LSTM_POLARITY as their only input feature had obtained R^2 scores that were close to 0. We also realized that adding features such as 1, 2 and 3 day returns significantly improved prediction accuracies.

References

[1] McCormick, E. (2023) "Tesla trial: did Musk's tweet affect the firm's stock price? Experts weigh in," *The guardian*, 29 January. Available at: https://www.theguardian.com/technology/2023/jan/28/tesla-trial-elon-musk-what-you-need-to-know-explainer (Accessed: October 16, 2024).

[2] *The Guardian* (2017) "Amazon stock market value falls by $5bn after critical Trump tweet," 16 August. Available at: https://www.theguardian.com/us-news/2017/aug/16/trump-amazon-taxes-tweet (Accessed: October 16, 2024).

[3] Christian Palomo. (2023). *Tweet Sentiment Analysis to Predict Stock Market*. CS224N, Stanford University. Available at: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1234/final-reports/final-report-170049613.pdf (Accessed: 31 August 2024).

[4] Goel, R. and Mittal, A. (2011) *Stock market prediction using Twitter sentiment analysis*. CS229, Stanford University. Available at: https://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf (Accessed: 31 August 2024).

[5] *Tweet sentiment's impact on stock returns* (no date) *Kaggle*. Available at: https://www.kaggle.com/datasets/thedevastator/tweet-sentiment-s-impact-on-stock-returns/data (Accessed: 30 May 2024).

[6] En-nasiry, M. (2024, June 21). *Time series splitting techniques: Ensuring accurate model validation*. Medium. https://medium.com/@mouadenna/time-series-splitting-techniques-ensuring-accurate-model-validation-5a3146db3088#:~:text=TimeSeriesSplit&text=It%20divides%20your%20data%20into,in%20tscv.split(X)%3A

[7] Chakure, A. (2023b, April 27). *Random Forest regression in Python explained*. Built In. https://builtin.com/data-science/random-forest-python

[8] Fraj, M. B. (2017, December 21). *In depth: Parameter tuning for Random Forest*. Medium. https://medium.com/all-things-ai/in-depth-parameter-tuning-for-random-forest-d67bb7e920d

[9] Onose, E. (2023, August 8). *R squared: Understanding the coefficient of determination*. Arize AI. https://arize.com/blog-course/r-squared-understanding-the-coefficient-of-determination/#:~:text=The%20R%2Dsquared%20metric%20%E2%80%94%20R%C2%B2,be%20explained%20by%20your%20model.

[10] *Mean square error (MSE): Machine learning glossary: Encord*. Encord. (n.d.). https://encord.com/glossary/mean-square-error-mse/

[11] *Grid search*. Dremio. (2024, July 16). https://www.dremio.com/wiki/grid-search/

[12] Awan, A. A. (2023, June 28). *An introduction to shap values and machine learning interpretability*. DataCamp. https://www.datacamp.com/tutorial/introduction-to-shap-values-machine-learning-interpretability