Estimation of Tesla Stock Price after Elon Musk Tweets.

CS7CS4-202223 Project by Group 41

December 2 2022

James Lunt
18323467

Niall Groves
18324201

Joel Jose Abraham
22303884

1 Introduction

Elon Musk is one of the world's richest and most well-known people. He owns Twitter, and people eagerly wait for his tweets. The project attempts to predict Tesla's stock price based on the words used by Elon Musk in his tweets. This project shows that people can manipulate stock prices with popularity and social media.

A sentiment score is predicted given a tweet's content. Then, the stock price is predicted using the sentiment score and the date. Figure 1 shows the plot of stock sentiment and date.

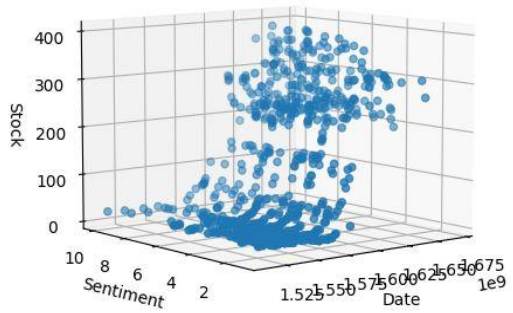


Figure 1: Plot of stock, sentiment and date

2 Dataset and Features

Data Retrieval

The dataset includes tweets from Elon Musk and the stock price of Tesla from Yahoo Finance from 17-07-2017 to 15-11-2022. A python script was used to remove the extra columns and calculate the average share price using opening and closing values. A new CSV file was created, and the date and estimated average share price were included. Using the Python tool SnScraper, Elon Musk's tweets, likes, and retweets were scraped from Twitter. A CSV file was created and saved with the Twitter data. Elon Musk's tweets, likes, and retweets were used to produce the sentiment values for the training set. The final model used date, sentiment score, and average share price as input features. Figure 2 shows the stock and date plot, which is similar to Figure 3, which shows the stock price in Yahoo Finance. They look the same.

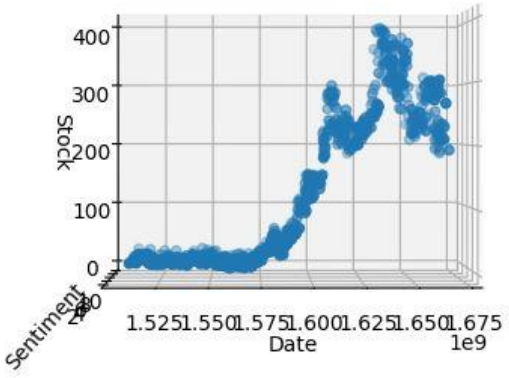


Figure 2: Plot of stock and date

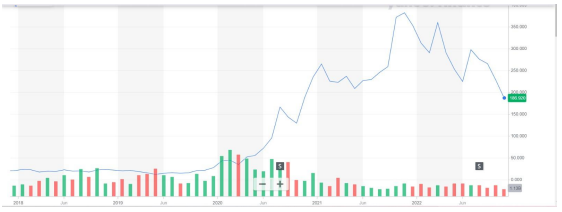


Figure 3: Stock prices in Yahoo Finance

Data limitations

There were some issues with the data that had to be considered. Firstly, the max amount of data points was restricted by a combination of facts such as, Elon Musk did not tweet every day and that Tesla's stock return was not reported on the weekends or holidays, therefore, there were gaps in the training set due to Elon not tweeting and the stock return for that day being useless or the stock return for that day being reported, but Elon didn't tweet. Furthermore, there is one stock return value for each day, so for the model training vectors to be of the same size, average daily sentiment had to be calculated rather than individual tweet sentiments. The raw data includes 2,757 tweets and 1,259 days of stock return. After data processing, this was reduced to 767 data points.

3 Methods

Sentiment score

To create a model to predict the sentiment score, the inputs were the words from Elon Musk's tweets and a combination of likes and retweets. First, each retweet was multiplied by a weight depending on how many likes the tweet that was retweeted had. It was decided to weigh retweets based on likes as a retweet could be considered positive or negative and it was decided the amount of likes would be a good indicator to which. This number was found by using a selection of interquantile ranges. Next, when the retweet had been weighted correctly, it was added to the likes to create a sentiment score. This sentiment score was then scaled to a number from 1 to n+1 depending on how many interquantile slices were taken (n being the number of interquantile slices), in the end 10 was used as seen in figure 4.

The CountVectorizer command was used to pre-process the text, tokenize the text and remove stop words. It builds a dictionary of features and turns the tweets into feature vectors. In our code we used stop words = 'english' to remove unnecessary words such as 'the' and 'in'.

Once this dictionary was created, tf-idf(Term Frequency Times Inverse Document Frequency) downscaling was applied. This downscales the weights for words that frequently appear in many tweets. Since these words appear frequently, they have less information associated with them and distract from the true sentiment value of the tweet.

Once the bag of words and the sentiment values had been assigned, they were put through a Logistic regression model to predict a sentiment value for a tweet. Several models were tested. The penalty type and the weight of the penalty were varied to find the optimum value. Knn and dummy classifiers were also tested. The number of quantile slices used to decide how many classes of sentiment value there were was also varied. For example, figure 4 shows that Elon Musk had a lower sentiment value initially because he had fewer followers.

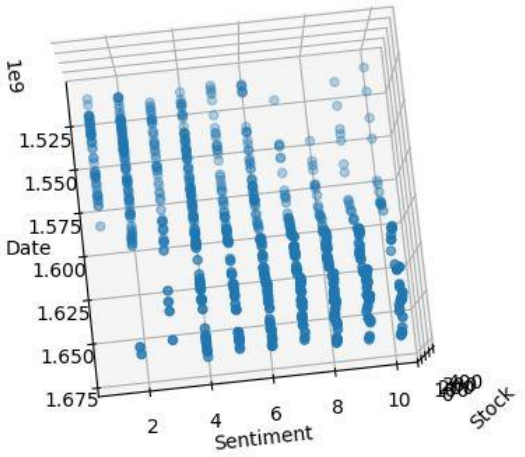


Figure 4: Plot of sentiment and date

3.1 Logistic regression

Several Logistic Regression models were trained for the sentiment predictor. The models use no penalty, a l1 penalty and a l2 penalty to see which performed the best.

L1 penalty: prefers models with fewer zero-weights.

L2 penalty: used for solutions with small feature weights.

An L2 penalty responded best to the data and was chosen. All models were trained on the same set of weights and a weight of 30 was chosen. These results were found via cross validation as can be seen from the figure 5 below.

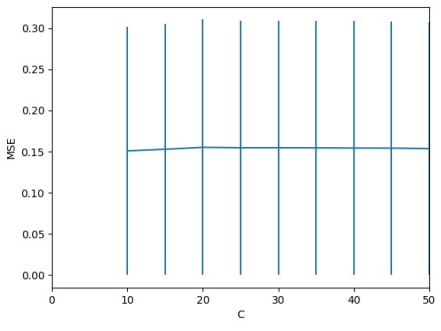


Figure 5: cross validation graph for weights vs mean squared error

3.2 kNN

A kNN regression model was trained. The model takes date and sentiment score as inputs and the stock return as output. In order for the date feature to be continuous it is converted into its time-stamp as seen on the axis in previous figures(figure4). Cross-validation was performed on the model to compare different 'k' hyperparameter values. The 'k' hyperparameter is the number of neighbours used in the model for each training point. The 'k' hyperparameter that produced the smallest mean squared error is used. Different weighting strategies were used to determine the value of each nearest neighbour. The first weighed every neighbour equally in every direction, and the second weighed points by the inverse of their distance, where closer neighbours of a query point will have a more significant influence than neighbours who are further away.

3.3 Lasso and Ridge Regression

Ridge & Lasso linear regression models were trained. Cross-validation was performed on both models to compare the effectiveness of polynomial features and penalty weights. The polynomial range was[2,3,4,5,8,10] and the penalty weight range was [0.001,0.01,0.1,1]. Both features (Date and sentiment) were processed into polynomial features, and the cross-validation was done in a 5 k-fold training data split. The model is then trained with the polynomial and penalty weight hyperparameter combination that produces the smallest mean squared error.

3.4 Baseline Regression Predictor

The 'Dummy Regressor' from the 'Sklearn' package was used as a baseline predictor. The model predicts every input as the median value among the training data.

4 Experiments and Results

Sentiment results

A Logistic Regression model with an 'l2' penalty and a weight of 30 was chosen as our model. The sentiment value and bag of words were passed through this. The results for the sentiment score varied. The baseline predictor will predict (1)/n, where n is the number of quantile slices used. If n is 10, the baseline predictor will be right 10% of the time. When n was set to 10 and used to set the model's input features, the accuracy was 15.7%, a 5.7% increase on the baseline. While this is not a massive improvement on the baseline, by looking at two above and two below predicted, the accuracy went up to 58.4%. While the model was not accurate to a point, it was close to the correct result 58.4% of the time. This was deemed acceptable to pass on to the next model predicting stock returns.

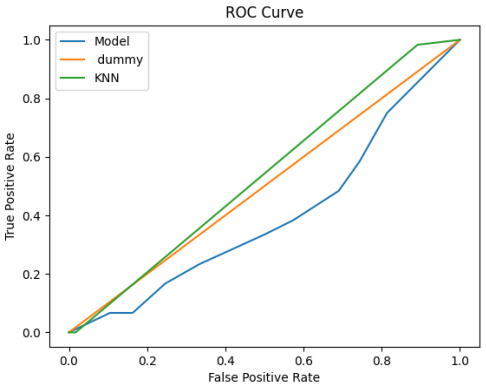


Figure 6: ROC curve for model, dummy and KNN

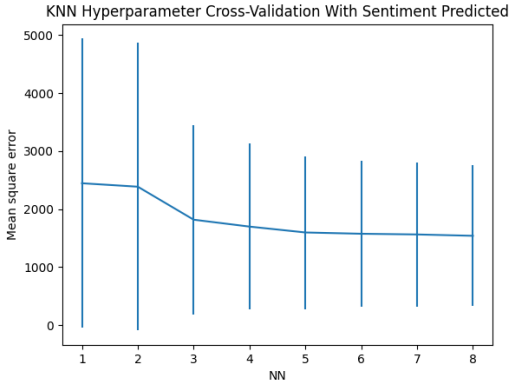
As can be seen, since the model is trying to place the tweet into the correct 1 of 10 categories, it often gets a lot of false positives. However as stated earlier, looking within two of the predicted values yielded a better result.

Lasso & Ridge results

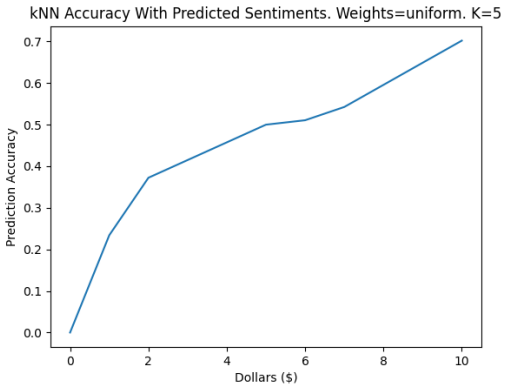
Due to the nature of the data, both lasso and ridge regression were ineffective at capturing the trend in the data. They did not prove useful for either creating sentiment scores or calculating stock returns.

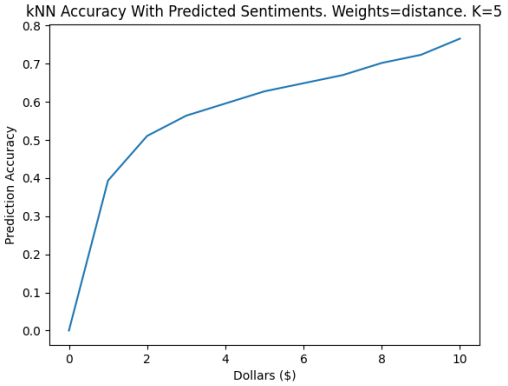
kNN results

Our kNN model produced the best results. It was found that even with a reduced training set we were able to achieve decent accuracy levels. It was decided that it would be very difficult to predict an exact stock return so the accuracy of the model is determined within range, e.g. the prediction is accurate within 5 dollars. Cross validation for the number of neighbours to be used was undertaken.



From this graph, the value 5 was chosen to be the amount of nearest neighbours as it is the least complex without over or underfitting and has one of the smaller mean-squared-errors and standard error bars. The model was trained with ‘uniform’ and ‘distance’ weighting strategies. The models were used to predict on a test set which was excluded from 25% of the training data. The following is our results:





The graphs can be interpreted where the x-axis displays the dollar range on either side of the correct stock return and the y-axis is the accuracy to which the regression model can predict the stock return. For example where X is $2 in the second graph Y is approximately 0.44 or 44%. If the true value of the stock return is $119.3 then the predicted value is somewhere between $121.3 and $117.3. To further support our predictor we also calculate whether the stock return decreased or increased. The uniform weighting model predicted a correct increase or decrease in stock with 33.8% accuracy and the distance weighting model predicted correctly with 76% accuracy. This makes sense as kNN when placing value on closer neighbours will better show the sign of the stock difference. The differences are calculated by subtracting the current day's stock return by the previous day's stock return for both the predicted target values and the real target values and then comparing the two to show an accuracy.

5 Summary

Elon Musk’s tweets were taken via the python library ‘snscraper’. Tesla’s stock return was taken from Yahoo finance. Tweet likes and retweets count were combined to create a sentiment score for every tweet and every tweet was parsed into a ‘bag of words’. A logistic regression model was trained with the ‘bag of words’ as input and sentiment scores as outputs. This model was then used to predict test/future tweets sentiment scores. The model was then used to predict sentiment

scores of a test set of tweets. The predicted sentiments are then combined with dates and stock return, any tweets that land on the same day have their sentiment score homogenised and the score is allocated to that day as a daily average sentiment. Next a kNN model is trained given the new dataset with predicted sentiments. The model can then predict stock return, within range, for a given date and sentiment value.

6 Contribution

Niall: Developed the sentiment predictor.
Joel: Attempted Ridge and Lasso. Collected Data
James: Developed the stock return prediction.
Everyone did a bit of everything.
Everyone contributed equally.

7 Conclusions

The machine learning models combined produced an accurate score in a range. While not great at predicting the exact return, the returned value was accurate to within six dollars 60% of the time. This shows that there is a correlation between what Elon Musk Tweets and a change in his Tesla stock price thus proving our hypothesis that an influential person can have a massive impact on the stock market and other people's money just from their words on social media.

8 Pivots

The model for predicting sentiment score was only slightly better then a baseline predictor so there would need to be more time spent on developing this aspect of the project. This could have been fixed possibly by more data, different/more feature engineering, or a different model approach.
Lasso and Ridge could also have been looked into more provided we had had more time.

7 Code Repository

james-lunt/Elon-Musk-Stock-Predictor-Using-Tweets: Group Project for TCD Module CS7CS4: Machine Learning. (github.com)