The Impact of Elon Musk's Tweets on the S&P 500, Tesla, and Dogecoin

George Kim

Summary

Elon Musk's tweeting behavior seems to have a tangible impact on the stock market. In this experiment, I use various machine learning models to predict the daily percent change of certain underlying assets (the S&P 500, DOGECOIN, and Tesla) using sentiment analysis on Elon Musk's tweets. Accurately predicting the exact change proved to be near impossible due to the simple fact that the input data (twitter history) cannot encapsulate everything that influences the market (news about the underlying, government policies, world news, etc.). However, directionality i.e. whether the stock goes up or down is still important and extremely useful. Assuming that random buy and sell orders would net somewhere around 50% accuracy, we can see from the results that Elon's tweets do in fact give valuable information about stock directionality. Notably, the best model for DOGE had a 52% accuracy and the best models for Tesla and the S&P 500 had a 58% accuracy, well above the 50% benchmark.

Introduction

The stock market is notoriously hard to predict, with many studies showing that random buying and selling can produce returns that are comparable to and occasionally better than calculated orders. With random buy and sell behavior, it would be reasonable to assume that 50% of those decisions would be profitable. Despite the myriad of potential predictors of the market, Twitter is certainly one of the more intriguing ones. The viability of twitter as a market predictor has really gained traction in the last few years, the most notable being JP Morgan creating the Volfefe Index to measure market volatility as a direct result of Donald Trump's twitter activity. Any information that helps to predict the market clearly has immediate monetary impacts, since knowing which direction a stock price moves is the only information required to make a profitable buy/sell decision.

Related Work

Source 1:

https://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf'

This paper performed sentiment analysis on tweets to predict "public mood" and used this public mood and the previous day's DJIA price to predict today's DJIA price. In terms of preprocessing data, they selected a subset of tweets that were more expressive/emotional: they determined this by looking for "I'm feeling", "this makes me", "feels", and other key words in the tweets. They

then used the POMS lexicon (well-known words associated with moods like calm, happy, alert, kind) and found the proportion of total tweets in a day including those words for each mood and then used Granger causality analysis to figure out how impactful each mood is on price prediction. Importantly, calm and happy were the best predictors with correlation 0.87
They explored a variety of machine learning models: linear regression, logistic regression, SVM, and neural networks with neural networks performing the best due to stock market trends typically being nonlinear. Performance was measured using MAPE (mean absolute percentage error) between the predicted price and normalized stock value in addition to “direction accuracy” indicating the proportion of correct trend predictions (up/down). The best directional accuracy was 71.11%, with most models being in the high 50s to low 60s.

Source 2:

https://towardsdatascience.com/can-we-beat-the-stock-market-using-twitter-ef8465fd12e2

This paper is very similar to the one above but uses a different variation of sentiment analysis on the tweets: they use the VADER lexicon to assign each tweet scores for positive, negative, neutral, and compound (summary statistic for the other three). Their data preprocessing involved removing entirely neutral tweets (neutral = 1.0) and incorporated the impact of a twitter user by multiplying the compound score by the number of followers. The machine learning models they tested were SVM, KNN (for direction prediction), decision tree, random forest, logistic regression, and neural network, with neural network once again performing the best
They measured performance by taking two metrics: direction accuracy (up/down) and the magnitude of price change, so essentially the same thing as the first paper. The stocks observed included EA, telecom, and healthcare companies, with the best models having 72% directional accuracy and decent performing models having high 50s to mid 60s accuracy.

Work Done

My solution involved looking at directionality analysis of DOGECOIN, Tesla, and SPY. One thing that was immediately different from the sources I read was the fact that Elon is one person as opposed to the other studies using sentiment analysis on the general public. As such, there were many gaps/days where there weren't any tweets from Elon or only a few insignificant ones. As such, it was immediately obvious that outright price prediction would be futile since Elon's tweets could not possibly account for market movements from outside factors like general political effects or other world news. This differs from tweets from the general public, which would definitely talk about any and all sorts of topics and news. This is why the other papers served well on price prediction and as will be shown, had better directionality accuracy as well. Since it is implicitly futile to do accurate price prediction, I decided to shift focus to directionality analysis.

In terms of input variables, I first had to preprocess the tweets to get rid of user mentions, links, and images which provide no value to sentiment analysis. I then converted emojis to their literal meanings. Finally, I used the VADER lexicon (used by nearly every source I saw online for sentiment analysis) to assign sentiment scores to each tweet. The scores consisted of a neg, neu, and pos component, which were then used as the inputs to the machine learning models. Another input is the daily stock prices, which of course was missing weekend and holiday data. I used pandas interpolation to fill in the missing gaps as just a linear approximation of the endpoint dates as this would allow me to use Elon's weekend/holiday tweets in prediction. I merged the sentiment data and stock data by aggregating the stock data and taking the mean of the sentiment values to represent a given day. The final input is the lag data, which basically considers the last N days' percent change as input variables. In terms of output variables, I used daily percentage change, which is the percent difference between the open and the closing price on a day. This is the metric chosen by most other similar work and

also removes the fact that stock prices have grown exponentially since 2010 so daily price movements naturally have bigger magnitudes as time went on. Therefore, percent change was essentially a standardizer for this.

After that, it was a matter of training machine learning models using the inputs and calculating the accuracy of the outputs. The code for the project uses regressors since the original scope was to do both directionality analysis and price prediction, but after figuring out that price prediction is unfeasible due to the data's lack of coverage of outside factors, I could just use the sign of the prediction (positive/negative daily percent change) to derive the direction. I tried out a slew of machine learning models: Lasso, Ridge, Bagging, Boosting, and Neural Networks. After hyperparameter tuning, these are the accuracy results:

Output	Model	Hyperparameters	Accuracy
SPY	Lasso	Lag: 3, Alpha: 0.01	0.540107
SPY	Ridge	Lag: 1, Alpha: 0	0.524064
SPY	Bagging	Lag: 2, Estimators: 100, Depth: None	0.574866
SPY	Boosting	Lag: 1, Estimators: 10, Learning Rate: 0.1	0.540107
SPY	Neural Networks	Lag: 1, Hidden Layer: (100, 100, 100), Activation: relu, Alpha: 0.001	0.569519
Tesla	Lasso	Lag: 1, Alpha: 0.1	0.52139
Tesla	Ridge	Lag: 2, Alpha: 0	0.505348
Tesla	Bagging	Lag: 2, Estimators: 100, Depth: None	0.574866
Tesla	Boosting	Lag: 2, Estimators: 200, Learning Rate: 0.1	0.548128
Tesla	Neural Networks	Lag: 2, Hidden Layer: (100, 100, 100), Activation: relu, Alpha: 0.1	0.550802
DOGE	Lasso	Lag: 1, Alpha: 1	0.484076
DOGE	Ridge	Lag: 2, Alpha: 0	0.44586
DOGE	Bagging	Lag: 4, Estimators: 10, Depth: 5	0.515924
DOGE	Boosting	Lag: 1, Estimators: 10, Learning Rate: 0	0.484076

DOGE	Neural Networks	Lag: 1, Hidden Layer: (100, 100, 100), Activation: logistic, Alpha: 0.0001	0.522293

Suggestions for Future Work

I think the biggest difference maker in this project would've been the time granularity of stock data. That is, I have a strong sense that a tweet's effects would be better felt on an hourly basis rather than a daily basis. After all, that infamous tweet by Elon stating that Tesla stock is overvalued tanked the stock by 20% within an hour, most of which was recuperated by the end of the day. Dealing on an hourly basis would also remove the need to aggregate all the tweet data in a day and allow for further isolated usage of a tweet's sentiment more effectively. That is, by aggregating and using the mean, we are definitely losing the hourly/immediate short-term effect of a tweet. Furthermore, perhaps it'd be interesting to use overall general twitter sentiment as an additional input. This would close up the gaps between Elon's tweeting history and cover a good portion of the missing outside information (company news, politics, etc.). However, this would be tricky because a lot of general twitter sentiment would come from responding/reacting to what Elon tweeted about, which invokes collinearity between these two inputs. It's a tricky idea but otherwise, I see no feasible way to do straight up price prediction.

Relative Effort

I would say the vast majority of effort in this project went into data preprocessing. Obtaining the data was pretty simple due to the ease of access to Elon's tweets on Kaggle and daily historical stock data on Yahoo finance. Preprocessing was tough because I had to deal with preprocessing both the tweets themselves using sentiment analysis (a new skill) and the stock data (which I was already aware beforehand from the literature that I would need to account for

gaps in the time series). I was not familiar with sentiment analysis whatsoever and so I was unfamiliar with what was required in terms of preprocessing messy tweets. Thanks to the preexisting literature I was aware that the VADER sentiment lexicon was the overwhelming favorite for twitter sentiment analysis since the training data is specifically from social media. I initially just ran the VADER sentiment analysis on raw tweets and got strange results: notably, neutral sentences like “Absolutely” would be skewed towards positive/negative emotion due to a twitter username being in the tweet, and https clearly did not provide meaningful value. It took a while to come up with the right regex expressions to remove @userX, https, etc. Next up was the stock data, which was missing values for weekends and holidays. I initially tried just moving weekend/holiday tweets to the next available date, but after some preliminary plotting and data exploration I realized that this wasn’t a great idea since I would be aggregating tweets by date and as such three days’ worth of tweets (each day varying wildly in terms of sentiment) would just cancel each other out and come out as close to net neutral sentiment, thus hindering those tweets useless. I fixed this by instead interpolating the missing values using a linear approximation whereby for a missing date, I would assign the stock values for that date to be the average of the previous date and the next date.

After handling the preprocessing above, running the models provided another challenge: handling lagged data as input. The thing about stock data and most time series data is that predictions are massively improved by considering the output data from the previous N days as inputs to predicting the current day’s output. We hadn’t really covered this in class before so it was really on me to figure out how to do this part. Luckily there was a good amount of literature out there on time series forecasting and I was able to find a handy python package called skforecast which did most of the coding work for me. Essentially, all I needed to do was to pass a machine learning model (e.g. RandomForestRegressor), a lag value (e.g. 2 days), and exogenous variables (the sentiment scores) to skforecast, which would then be able to predict outputs when given test data. Once I had this working, I was able to utilize techniques from the

previous assignments in class for hyperparameter tuning. All in all, I was able to test out several models, some of which like the Neural Networks took a good amount of time to run.

Bibliography

- https://towardsdatascience.com/can-we-beat-the-stock-market-using-twitter-ef8465fd12e2
- https://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf'
- https://joaquinamatrodrigo.github.io/skforecast/0.4.3/notebooks/direct-multi-step-forecasting.html
- https://www.kaggle.com/datasets/ayhmrba/elon-musk-tweets-2010-2021
- https://www.kaggle.com/code/redwankarimsony/nlp-101-tweet-sentiment-analysis-preprocessing/notebook
- https://pypi.org/project/emoji/