Tesla Stock Prediction using Twitter and Historic Data

Nic Wiggins

Project URL:
https://github.com/nicwigs/482/tree/master/project

ABSTRACT

Stock price predictions are a notoriously difficult prediction problem. Exercise 5 showed how using 30 years of McDonald's own historical stock data, the model resulted in an r^2 value of 0.0018. There had to be a way to improve upon this idea, hence this project. I formulated this prediction of Tesla stock price delta as a regression problem with historical data and twitter sentiment analysis as features. After applying linear regression to predict the deltas, it was found that Twitter data and other stocks, while playing a small role, do in fact make the model better.

1. INTRODUCTION

To many the stock market is a very abstract arena, and stock prices seem to change relatively randomly. The idea of being able to predict stock prices is a bold idea. The past four days of stock price change does almost nothing in terms of helping predict the price, as displayed in Exercise 5. While historical data is important, that doesn't capture what is going on inside the company, the consumers and the investors. Luckily, there is something that gives a hint of capturing these things that the prices do not – Twitter. Tesla stock was chosen as the stock to predict due to a personal connection in being an intern there, but also due to the fact that it is a company that many of its owners and critics talk about on Twitter.

In order to use Twitter data to predict the stock, sentiment analysis was used. Sentiment analysis is able to determine the overall mood of a sentence or even paragraphs, in this case a Tweet. With this, each tweet is able to be broken down into probabilities of being a positive or negative tweet, and all the tweets over a given day can then be used to help predict the stock for the next day.

The goal of this project is to predict the change in TSLA stock on a given day given previous days stock deltas and tweets. A goal along the way is to see if sentiment analysis from tweets does in fact perform better than a model that is only based off historical data.

The type of data collected was stock data and tweets over the course of about a year (see Section 2).

Finding the stock data was very straight forward, and once the sentiment data was mentioned in class this was no longer an issue. Wanting to use a package, TextBlob, to create the sentiment model, the large quantity of data caused my computer to freeze and crash. While this was unfortunate, a Naive Bayes was recently discussed in CSE 440, and utilizing AWS and Hadoop for word counting, a Naïve Bayes sentiment model was used. This simply compares the probability of a tweet being positive or negative based off the probabilities of each word in the tweet. Another challenge was finding about 10 days of tweets missing in the collected yearlong tweet data. Instead of replacing these values, these days were removed, and were not used in the predictor.

While it was hoped that the twitter data and other stock's historical data would largely increase the model's accuracy, they did in fact increase the accuracy, however, only slightly.

2. DATA

The three main data sources were the stock historical data [2], the labeled mood for tweets [5], and the database of tweets via the professor [1]. The historical data for each stock in Nasdaq [4] was a lot, and an online tool, [3] allowed to speed up the process to get all the historical stock data from Yahoo [2]. To lighten the load on the online tool, a script was made to break the list of all companies into a bunch of smaller lists. From here the tool was used to download all the data in 34 different CSV files. These files included date and closing price for all Nasdaq stocks. All these files were then loaded in and joined on their date to further be processed. The labeled twitter mood data was a CSV, and consisted of a label, 0 for negative, 1 for positive, and the tweet's text. Lastly, the tweet data from the CSE server included many JSON files of tweets, the format was how the tweets would be received using Twitter API, including fields like time created, user, text, etc.

The labeled twitter mood data consisted of 1.58 million labeled tweets based off mood. A few of these tweets had the new line character '\n' which made reading the file in with Pandas create some errors. Pandas read_csv() has the ability to not fail on errors, so this was chosen, and loaded tweet texts that had this character were removed. Removal was chosen to save time, and there were about 20 tweets out of the whole set that were removed due to this.

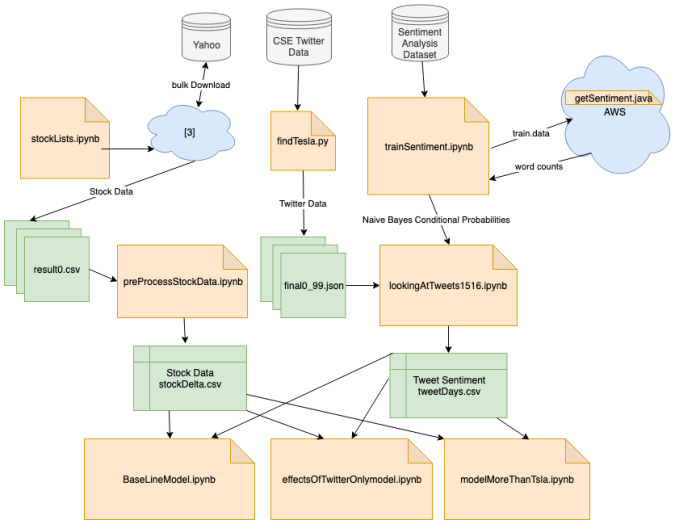
The Twitter data on the CSE server consisted of over a terabyte of files, understandable considering it had all tweets for a year. These tweets ran from 4/27/2015 to 4/06/2016, and most didn't involve Tesla. All Tweets whose text didn't include the word tesla were not considered, and of those tweets, only the creation time and text fields were needed out of all the features. Due to size and time restraints, the output of this script ran on the CSE server for different groups of files of tweets in JSON format. These files were then all downloaded locally and joined together in one table. Further down the line, it was realized there were tweets missing for dates 1/25/16 to 2/05/16. Instead of replacing these missing values, these days were simply disregarded in training the model. This was deemed acceptable since this was only a few days out of an entire year, so removing them didn't affect the prediction due to their small percentage of the sample size. The tweet's creation time was converted to a day using Pandas to_datetime(), and then the table was grouped by day and labeled sentiment. This data was then split up into test/train and the training set was output as a tab delimited file sent to AWS for Hadoop to count the word appearances for the Naïve Bayes algorithm to be set up. The results were a word, and the count of times it was in a positive tweet and number of times it was in a negative tweet, in a tab delimited format. This was then used to train the Naïve Bayes model (See Methodology).

The stock data downloaded was from June 29, 2010 until present. This was a much larger set of data than was needed since this was downloaded before knowing the range on the twitter data. The actual used stock data lied between the range of the Twitter data. The stock data had multiple missing values, and after all stock data was joined on date, stocks of the list [4] were removed if they had any NaN entries over the span of the Twitter data. This decreased the number of different companies' stocks from 3223 to 2239.

This is a regression problem with the delta stock price of TSLA as the target attribute. The features used consisted of previous days' stock and twitter data. What this looked like for the stock related attributes was columns of the 1 to 6 previous' days stock delta for certain stocks. For the twitter analysis, there was the percent of positive tweets for the previous day, the average percent positive of the tweets on that previous day, and the average percent negative of the tweets on that previous day, as well as the sentiment – 0 negative, 1 positive. In addition to the previous day's twitter data, further days in the past twitter data with the following metrics were also used (positive percent, average positive probability, average negative probability). The percent of positive tweets was to portray the confidence of the sentiment value, this is the percent of positive tweets that day. In the Naïve Bayes model, the log of the conditional probabilities of each word are added together and compared. If the probability of being positive is larger than that of the negative probability, then the tweet is marked positive. The average percent positive and average percent negative was to portray the confidence of the percent positive. These values are the average probabilities among all the tweets for a given day.

Since selecting the predictors was an iterative approach – picking which companies' stock, how many companies, how many days of historical data, how many days of historical tweets, what attributes of historical tweets – the two main data sets for all 2239 stock and each twitter analysis are 10.1 Mb and 29 kb. The size of the data sets actually used for the final model for showing the pure benefit of twitter data, and the combined benefit of twitter data and other companies resulted in data sets of sizes 62kb and 89kb, respectively.

3. METHODOLOGY



A Sklearn linear regression model was fit to the features mentioned above.

The twitter mood data was split into test and train with train_test_split() by Sklearn, with train percentage of 33% for training the Naïve Bayes model, resulting in an accuracy of 73.5%. Further in the process, cross validation was done on the linear regression to select the final model. What features that were picked was based off of this cross validation. A few nested loops looped over stocks to look at, number of days to look back on for stock as well as tweets, and what attributes to look at for the previous day's tweets. The number of folds was selected to be 7, since with this and a training size of 70%, this would have each fold be about 24 days, which seemed like a fair amount of days to validate the training data models with.

The following is a summary of the code written for the project:

- twitterSearch.ipynb: This is the Jupyter notebook file that I wrote to view what the Twitter data looks like and look at API.
- stockLists.ipynb: Created multiple lists of stock names to then upload to [3] to get all stock data in bulk.
- loadProfData.ipynb: This is the Jupyter notebook file to help write and debug findTesla.py
- findTesla.py: This is a python script ran on the CSE server that dumps tweets that have the word or hashtag tesla. Could not loop over all ~7000 files at once, so initial for loop was changed to read a select number of files at once. Output is JSON file of tweets involving tesla and the needed fields– text and time created.
- getSentiment.java: Java file ran on Hadoop on AWS. Input is tab separated, training Naïve Bayes data, in the form of sentiment \t tweet text. The output is a list of common words, frequency above 10, with number of appearances in positive tweets, and negative tweets.
- Sent.sh: Script to run getSentiment.java on AWS.
- trainSentiment.ipynb: Splits twitter mood data into test and train set. Test set is output as tab delimited, sent to AWS. Then, it reads output from getSentiment.java and used Naïve Bayes to calculate conditional probability of each word given the class. Output are two JSON files, each a dictionary of conditional probabilities for words in tweets.
- lookingAtTweets1516.ipynb: Loads all tweets that had keyword Tesla. Groups data frame by day of tweet and calculates probability of sentiment for each tweet and finds total for the day. Output is a csv that has percent of positive/negative tweets for that day, average positive probability among the tweets, average negative probability of the tweets, and sentiment for each day.
- preProcessStockData.ipynb: Loads all stock data, joins on day. Selects data around the twitter data dates, removes companies that have NaN values. Outputs CSV of each day as a row, columns are companies, and values are the stock price change on that given day.
- BaseLineModel.ipynb: Creates a baseline model for predicting TSLA stock based off the previous 4 days stock change. Multiple models were used linear regression, ridgeCV, and SVM among them. A linear regression model was used for simplicity as the baseline.
- effectsOfTwitterOnlymodel.ipynb: Creates a predictor that uses the same previous 4 days stock deltas, but also includes twitter sentiment analysis data. Shows an improvement over the base.

- modelMoreThanTsla.ipynb: Creates a predictor that searches over multiple combinations of stocks, days to look back and twitter days to look back.

4. EXPERIMENTAL EVALUATION

4.1 Experimental Setup

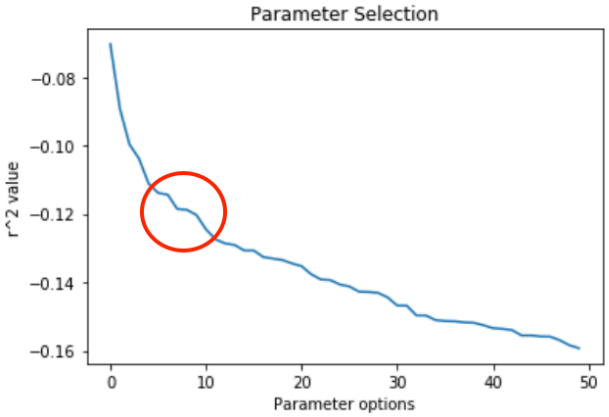
All but getSentiment.java was run on my local machine, MacOS. Due to the large amount of mood data, 1.58 million, this was ran on AWS cluster of 3 nodes.

The baseline method was using the previous four days' stock changes to predict the next day for TSLA. Because TSLA stock has a lot of variation, using the mean in r^2 is not the best. Thus, root mean squared was also used to judge the models. While the main point of the project was to predict the actual price delta, accuracy in terms of classifying if the stock price was going to increase or decrease was also looked at when comparing models.

4.2 Experimental Results

	Baseline	Baseline + Twitter	Multiple Stocks + Twitter
RMS	6.9081	5.7512	5.7516
R^2	-0.0474	-0.0950	-0.0779
Accuracy	0.5694	0.5820	0.6029

With all possible descriptors available, loops were implemented to test multiple combinations of other company stocks, number of days in history to look at stock delta, and number of days in history to look at twitter data. Each iteration used cross_val_score() to validate it, and the scores were then averaged. These different options and their averaged scores were plotted, and looking for knees in the plot (See below), parameters were chosen and then finally the test set was applied. While there is the opportunity to have over 2000 different companies' stock as predictors, 10 different stock were chosen by finding the least correlated stock to Tesla and from there, the least correlated stock the that stock, and so on until the limit of 10 was reached. The idea was to not have a bunch of correlated stock since this would overfit the training data and perform poorly on the test set.



TSLA_d-1	0.006463
TSLA_d-2	0.041912
TSLA_d-3	-0.025997
TSLA_d-4	-0.154065
TSLA_d-5	-0.080570
GENC_d-1	-4.953115
GENC_d-2	2.555521
GENC_d-3	-1.582309
GENC_d-4	-1.951668
GENC_d-5	3.803804
MARK_d-1	3.137745
MARK_d-2	0.519822
MARK_d-3	1.871722
MARK_d-4	8.043606
MARK_d-5	9.535392
d-1_perP	-22.148617
d-1_app	0.112135
d-1_apn	-0.341313
d-1_sentiment	-2.572983
d-2_perP	7.991345
d-3_perP	-12.746637

To the left are the coefficients of the final model. It's interesting to see that the model depends the least on the previous days changes in Tesla stock. Both the uncorrelated stock, GENC and MARK have somewhat equal weighting on the model. The model also highly weights the percent of positive tweets the previous day, which intuitively makes sense. We see large weights on the other positive percentages for 2 and 3 days before the prediction.

The project was successful. The root mean squared error decreased and accuracy increased. However, the improvement is relatively small. Poor performance is most likely from either poor classification of sentiment within the tweets or selecting the wrong stocks to be predictors. This may be due to the fact that a Naïve Bayes model with 75% test accuracy was used to classify the twitter data. The other thing to note is there are over 2000 stocks to choose from to be predictors. These two were chosen since GENC is closest to 0 in the correlation matrix for TSLA, and MARK is the closest to 0 for the correlation matric for GENC. Again, this was done to avoid correlating predictors that would tend to overfit the model.

5. CONCLUSIONS

Predicting the change in stock price for Tesla is possible using linear regression. While historic data does work, this project found that other stock historical data and twitter sentiment analysis on the company can increase the performance of the model. Future work to improve this method would be further looking into the best way to select different stock to use as predictors instead of how it was done here.

6. REFERENCES

[1] /user/research/ptan/data/Twitter/

[2] https://finance.yahoo.com/

[3] http://finance.jasonstrimpel.com/bulk-stock-download/

[4] https://www.nasdaq.com/screening/company-list.aspx

[5] http://thinknook.com/wp-content/uploads/2012/09/Sentiment-Analysis-Dataset.zip