Analyzing Stock Price Based On Twitter Sentiment

Fynn Reckhorn

Project URL: https://github.com/freckhorn/twittersentiment

**ABSTRACT**

This project sought to use twitter sentiment on a keyword (in this example " tesla " to predict stock price change. More specifically whether the price would go up or down. If this worked Even a little bit, it could make people a lot of money. I captured tweets from the professors Twitter data folder, applied sentiment analysis on it and then compared these data points to stock price changes. The algorithm I used was a simple tree regression approach. In the end I also used some other data points, like stock price change between open and the highest point in that day, as well as volume traded that day. It turns out that only the stock price change from open to high was used for the best accuracy.

# INTRODUCTION

For introduction, you need to include the following information:

1. Since the beginning of time humans have been striving to achieve financial gain/ increase their wealth with the least amount of work or effort required of them. Probably the easiest way to do that, is to buy a commodity for low, and sell it for higher. In today's day and age we have seen many cases of individuals investing into a certain stock when a company first goes public, and being able to sell this stock for a much higher price after some time has passed. This is usually only possible, because of a lucky guess on what company would prosper and increase their stock prices. Recently, tech companies have been able to increase their stocks worth, just by swaying public opinion in their favor. Most famously, Tesla's CEO, Elon Musk, has done a terrific job of this, being able to sway prices, just by taking to Twitter and making bold claims. Tesla has not even been a particularly profitable company, yet their price has often sky rocketed. This made me think, that if I am able to gauge how enthusiastic people are about Tesla, I might have a good shot at predicting general stock price fluctuations for the company. I took a look at a tweet that contains the keyword " Tesla ", counted the positive words, and negative words, and classified it based on which it contained more of.
2. I think that I will be able to predict whether a stock will go up or down in value, based on Twitters sentiment towards the company. I will do this by grabbing tweets about it, running sentiment analysis on it, and then running a regression algorithm on stock price changes and the sentiment for a particular day.
3. I collected tweets containing the keyword " tesla " from a collection of roughly 9000 files, each containing millions of tweets. This data was provided by Professor Tang.
4. I encountered problems along every step of the way. The first one being finding data. I wanted to just pull my own data from twitter, but discovered that I would not have enough days to collect from. So after I finished the entire code to grab tweets and analyze them, I had to rebase the entire project, to instead comb through the twitter data. The next challenge was how immense the dataset actually turned out to be. I expected to download the files and do all the work locally, but ended up having to write a python script that would navigate to the proper directory, comb through all the files in it, find only the tweets that were relevant, and store them. This was the first time I had to do something like this so even just this took a lot of learning on my part. After I did this, I realized that I had saved the tweets incorrectly (json got messed up due to characters like '{' in the tweets). So I spent many days rewriting my script, until I found out about pickle format. Basically every tweet I stored only the relevant parts into a dictionary. Then I made a list of all the dictionaries, in a file and then pickled them (which means that they are saved in a binary format). Once I ran this, I kept timing out after some hundred files were combed through, and lost all of my progress. So I had to rewrite my script to save the tweets to file every time one was found, so disconnecting wouldn't lose me all my progress. Once I had all this done, the actual preprocessing was a nightmare.
5. I was not able to fully achieve my hypothesis. Once I had all my data in place and ran the model on it, the highest accuracy tree turned out not to use the sentiment data. But using historical data I was able to reach a pretty good accuracy.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*CSE881-2015*, Month 1–2, 2004, City, State, Country.

Copyright 2004 ACM 1-58113-000-0/00/0004…$5.00.

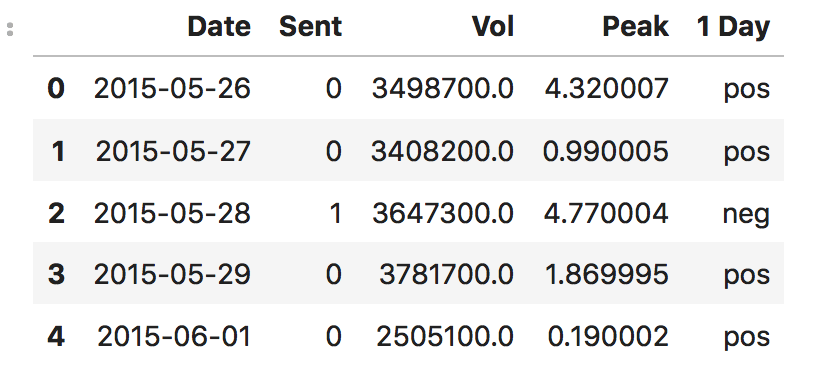
# DATA

At first I grabbed some Twitter data using the method Professor Tang showed us this semester (rest api and tweepy). I then realized that this data would not suffice and switched over to using the twitter data that had been collected in the past. I combed through those thousands of files containing millions of tweets, looking at the content of every tweets and only recording the ones that containing my keyword. These came from Unix executables, were stored as a list of dictionaries, which were the pickled and saved in a binary .p file. I also used yahoo finance data on the actual stock prices from that date range as shown in the following link(https://finance.yahoo.com/quote/TSLA/history?period1=1432440000&period2=1443412800&interval=1d&filter=history&frequency=1d). I wrote my own crawler for going through the twitter data. It is a python file that I executed on my arctic account, that then navigated to the data folder and looked through the files. I also wrote the tedious logic for joining together the tweets, with their sentiment and then after that the sentiment with data and stock price data. I did this using numerous lists and dictionaries. I also saved some of the median steps to csv files, in order to save my progress on the way. The final file which I ran the actual regression on was a csv file.

The actual features I collected along the way included, date, time, userid, tweet content, index, positive, negative, or neutral sentiment, and a lot of data about the stock. The final data points I compared were date, sentiment, stock volume, stock high- open price to predict up or down trends.

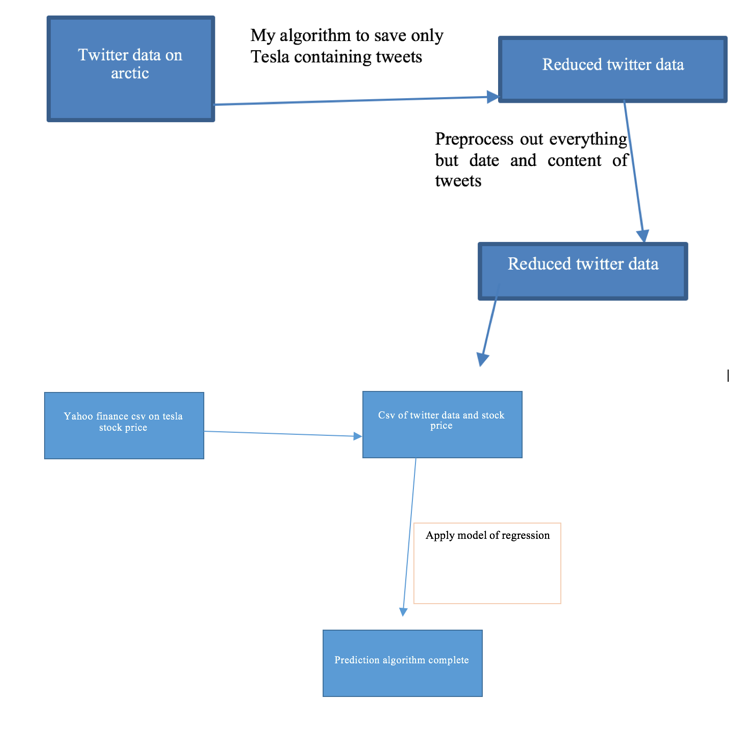
The twitter data was very feature rich. Many of these attributes I never even looked at. From the beginning I never needed location of sender, or the amount of likes or anything like that. The only characteristics I needed were the text and date. I grabbed tweets from the entirety of the twitter data which turned out to be a timeframe from May to September of 2015.

Some issues I ran into were parsing the twitter data and saving it in a way where I could pull what I needed efficiently. As I mentioned before, I did a lot of preprocessing, and resaving objects from the pickle file into lists and dictionaries and csvs in order to merge the data, as well as eliminate columns I didn't need and perform the content analysis on it. The classes that I was predicting were price increase and price decrease. The predictors were sentiment, volume, and open to max price.



Here is a screen shot of my final data frame. It has a total of 88 rows and 5 columns. The csv is only 4kb in size.

# METHODOLOGY



I use decision tree classifier to perform my classification from Python Surprise toolkit to do the prediction.

The hardest part was the preprocessing and merging of the data.

Finally, give a brief summary of the the code you have written for this project. For example, you can summarize it as follows:

* finalprojscript.py: this is the python script file that I wrote to collect data from arctic. It also does some preprocessing. It eliminates all columns not needed from the twitter json files and saves them to a pickle file called tes\_data.p
* tes\_data.p: This is the binary file that saved all the tweets
* sentiondata.ipynb: this is the Jupyter notebook file to preprocess some more, and merge in the csv yahoo finance data and perform the classification task of the project.
* TSLA.csv: the yahoo finance file that contains the stock prices
* combined.csv the data of merged twitter and yahoo that the regression is run on

All additional packages are ones that are imported into python in my code.

# EXPERIMENTAL EVALUATION

This section describes the experimental setup and results you obtain.

## Experimental Setup

This section should include:

1. Computing platform (what operating system and hardware you use to do the experiment). Are you using AWS cluster? If so, how many nodes?

I used the arctic server, which I ran my finalprojscript.py on as well as my own machine which has python and jupyter installed. That is about all one should need.

1. What baseline methods did you to compare against your approach? Make sure you report the results of your method as well as the baseline methods. This is important to demonstrate whether your project was successful.

I ran decision tree classifier algorithm. It tried different approaches and patterns and ended up chosing one where it only used one predictor attribute for a 69.01% accuracy.

1. What evaluation metric did you use to report the results (e.g., accuracy, root-mean-square-error, F1-measure, etc).

I used accuracy. The accuracy is calculated by number of correct predictions over all the predictions.

## Experimental Results

I performed the decision tree analysis. This means that I ran a bunch of different weighted tests to see which combination of the data in my data frame would yield the best prediction schema. It turns out that the sentiment analysis data doesn't help increase the accuracy very much and is left out in the final model.

The fact that the sentiment didn't do very much for the model was surprising to me. Ultimately my hypothesis was wrong, but I was still able to achieve high accuracy using the historic stock data. I think that the sentiment would have worked better if I had a stronger dataset. I should have looked at more keywords (variations of Tesla) and I should have used tweepy for the duration of the project and captured my own data. This would have enabled me to average the sentiment and have stronger data points. As it is I only had a couple per day or sometimes none.

# CONCLUSIONS

In summary my project was not successful in proving my hypothesis. In fact, when I used regression the best model was one that didn't even consider the sentiment data. I did find that I got a pretty good accuracy just using the historical data. How I would improve the project is very easy. I should have grabbed about 1000 tweets every day for a month (or longer) and then had stronger sentiment analysis data points. I also think that I should have done a price prediction instead regression. The biggest thing is that I only had few data points, because there were only so many Tesla tweets in the data set. If I redid this I would look up more keywords than just " tesla ", but maybe also variations on the word as well as relevant terms like "Elon Musk", "model 3", "model s" or "electric car". Having more data would definitely increase my accuracy, but this was a great intro to this type of work and what all you have to deal with and anticipate.

# REFERENCES (at least 3 references)

1. Yahoo Finance. *https://finance.yahoo.com/quote/TSLA/history?p=TSLA*.
2. Pang Ning Tang. *CSE 482 Big Data Course*, Michigan State University.
3. Geeks for Greeks. Twitter Sentiment Analysis using Python. https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/

Grading criteria

Note that the project accounts for 10% of your final grade. The project will be graded based on the following criteria:

1. Presentation - structure/organization and clarity of writing (including tables and figures).
2. Technical - Correctness and thoroughness of the analysis performed. What are the challenges faced and how well did you address them? How do you evaluate the performance of the method you'd applied to the data? How much detailed discussion you provide to explain the results you'd obtained (e.g., discussion about why the method works or didn't work on the data)?
3. Difficulty level - How large is the dataset used? How much effort you had to spend to collect, integrate, preprocess, and analyze the data? Are you implementing the project on a cluster or a single machine? What tools did you use (do you have to implement them or are you simply using existing libraries)?