Predicting Stock Prices with Naive Bayes and K-Means

Sabastian Mugazambi, Simon Orlovsky

Abstract

In this paper, we explore the relationship between tweets from high profile users, namely Donald Trump, and the effect on the stock market. We are interested in seeing if there is data that we can extract to predict stock prices. To solve this problem we created a model utilizing a version of the Naive Bayes Classifier for sentiment analysis implemented by the Natural Language Toolkit. Given the sentiment classification we also use a modified KNN algorithm to predict the actual percentage change in the stock price due to the tweet.

INTRODUCTION

According to Bollen et al's exploration of Twitter mood predicting the stock market, emotions on Twitter profoundly affect individual behavior and decision-making. With the rise of powerful political figures on Twitter we wanted to see if these Tweets could be useful for predicting future stock prices. We created a model and used historical tweets and historical stock data to test the model's performance. Before we discuss the model's performance in detail and the results we got from running the model, we wanted to discuss the design decisions we made at each step of the model creation and how it would potentially affect our results starting with data acquisition and wrangling.

Data Wrangling

One of the biggest issues with this project is that the data was not handed to us in neat format. We had to scrape two sets of data from different servers and do all data cleanup by ourselves to put it into the format we needed it in.

STEP 1 : Twitter Dumper & Financial Data Gathering

We wrote a script that utilises a library called Tweepy. The python script uses Twitter developer credentials to scrape the web and dump all tweets of a given userID into a CSV file. We chose Tweepy because its framework is perfectly designed for this scraping and the library understands the structure of Twitter data on the Twitter servers. After getting all the dates, ids and texts of all of Donald Trump's tweets since December 2016 we selected the 26 tweets in which he mentioned companies to make a separate dataset used latter part of our algorithm for percentage change prediction. We then used Quandl, a financial and economic database with a public API, to pull down end of day (EOD) stock prices for the tweets with companies mentioned in them.

STEP 2 : Training Tweets Creation

A major issue we also faced was that there was no existing data sets of Donald Trump's tweets already classified with labels of sentiments. After acquiring thousands of tweets which chose some of the most recent and older tweets and we manually read and classified the tweets as negative or positive. By doing this we created our training tweets which already had labels and then we could train our NLTK Naive Bayes model on this training set.

SENTIMENT ANALYSIS

After creating our training set with positive and negative tweets we had to do something with the texts of the tweets so that our model could extract feature words from the tweets for further Naive Bayes calculations given a new tweet.

STEP 3 : Tokenize the training tweets
The tokenization process takes the training tweets set. For each tweet, we normalize the tweet by lower casing every letter and remove stop words from the tweet. This process involves identifying stop words and having a data set that represents stop words. The big key feature of tokenizing is to be able to identifying when a word is adding valuable context to the tweet and when it's acting as a stop word and thus needs to be removed. We felt like the algorithm did this well using the NLTK methods.

STEP 4 : Feature Extraction
Next we extract the word features from the tweets and order them by frequency of appearance. Thus, we basically create a frequency distribution of the words appearing in the tweets after tokenizing. Words become the features of the tweet. Once we have all of our features we need to extracter to determine which features are containing in a given tweet. We can use the word features along with the extracted features from our dataset to train our Naive Bayes classifier. The classifier uses the knowledge of the frequency of the features and their respective labels to predict the sentiment of a given new tweet.

STEP 5 : Classifying New tweet
To classify a new tweet we will need to firstly calculate what we call the prior probability of each label (Negative or Positive). The prior probability of each label is basically the ratio the label appears in the training data set.

If we have 20 total tweets and,

10 positive then Prior Probability for 'Positive' is (10/20) = 0.5

10 negative then Prior Probability for 'Negative' is (10/20) = 0.5

Given the prior probability we calculate the initial log probability of each label as follows,

'Negative' log$_2$(0.5) = -1.0 & 'Positive' log$_2$(0.5) = -1.0

Equipped with these initial likelihoods we can use our conditional likelihoods for the tokens/features in our new tweet to calculate a final 'Positive' and final 'Negative' likelihood that we will compare in the final step. For each feature in the tokenized new tweet we calculate its probability conditioned on 'Positive' tweets or 'Negative' tweets in the training dataset. More clearly we will use an example;

If we encounter the word 'illegal' as the first in our new tweet, we analyse its conditionals in the training data set;

If the word 'illegal' appears 4 out of 5 in negative tweets then the probability of seeing the word 'illegal' given a negative tweet is ⅘

- *running_Negative = -1.0 * ⅘*

If the word 'illegal' appears 1 out of 5 in positive tweets then the probability of seeing the word 'illegal' given a positive tweet is ⅕

- *running_Positive = -1.0 * ⅕*

We keep on doing this for each word until in the end we have two final values of 'negative and positive likelihood. We then take the max of these two values and assign the label of that max value.

```
If max{running_Negative, running_Positive} == running_Negative:
    tweet_Label = 'Negative'
esle:
    tweet_label = 'Positive'
```

PREDICTING SHIFTS

STEP 6 : Get predicted percentage price change

To predict shift in stock prices we used a KNN approach. KNN is a pattern recognition algorithm for classification and regression. The algorithm finds the k (parameter given by user) nearest neighbors using a particular distance metric. In our case we used Euclidean distance. We want to compare the final [*running_Negative, running_Positive*] of the new tweet and compare it with likelihood values of other company tweets. By picking the k-nearest neighbors we average the percentage changes of all the k-nearest neighbors to give us our new tweet's predicted percentage change. We chose KNN because generally if the tweet has the same words mentioned with our new tweet then it's most likely to cause the same magnitude of price change.

STEP 7 : Calculating the predicted End of Day Stock Price

The final calculation is a straightforward one. We basically take the previous day's end of day stock price for the company in question and we apply the percentage change to it to get our predicted EOD stock price for the company. Whether we subtract or add this change is determined by our tweeter sentiment classification we did earlier for that tweet. This solves our two main problems we had at hand and concludes the work done by the entire model.

RESULTS ANALYSIS

Our results can be analysed on two levels. We can analyse the model's performance on Twitter sentiment classification and we can analyse the model's performance on price prediction since we can get the historical data of the stock market trends for the companies mentioned by Donald Trump.

Sentiment Classification Results

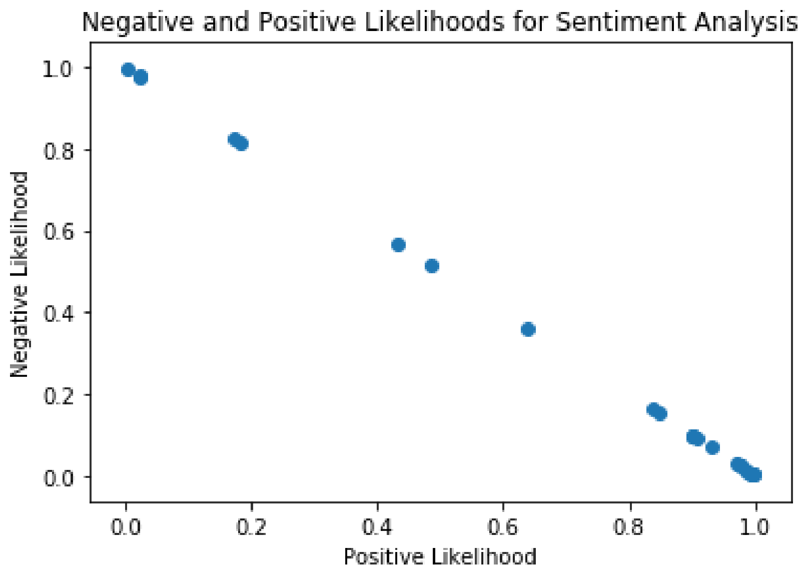
We found out that Twitter sentiment analysis his heavily reliant on the size of the training tweet data size. Firstly, since the training tweets are manually classified, the training tweets become very subjective to the user who classified them. If the user classifies them incorrectly the model is thrown off. Secondly the size of the training data set is of great importance, the more more the training tweets the better the classification model performs as seen the graph below. This is expected since the more your know about how Trump speaks and the words he uses, the more one can easily tell if his sentiment is negative or positive. Potentially we could have thousands of training tweets but then we would have to manually classify each and every one of them. With our current training data set size we reached a maximum *accuracy rate* of *98.42%* .

Another minor thing we noticed in the training data set was that we had to maintain a balance between negative and positive tweets in the training data set to achieve high performance. Having more tweets of one label than another tilts the values of the Prior probabilities and the log probabilities.

Price Prediction Results

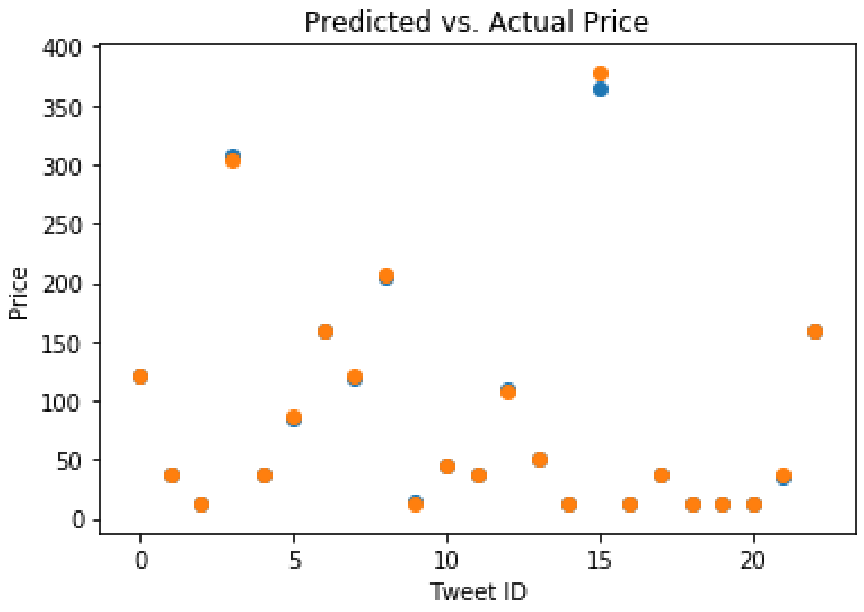
Before we dive into analysis of the price prediction model we have to find a way to gain confidence in our sentiment analysis Naive Bayes model. While the accuracy rates for classification are good, we had to first analyse how well our model conformed to the rules of the regular Naive Bayes model. One way to do this is to compare the final positive label likelihood and the final negative label likelihood of each tweet with a company name in it. The graph below is a plot of the two likelihoods. Notice that the equation of the line of best fit is almost accurately

'Negative' Likelihood = 1.0 - Positive Likelihood



This proves that our version of the Naive Bayes implementation via the NLTK algorithm for sentiment analysis is consistent with the Naive Bayes model we studied in class. Given this we can safely analyse the performance of our price predictor model as whole.

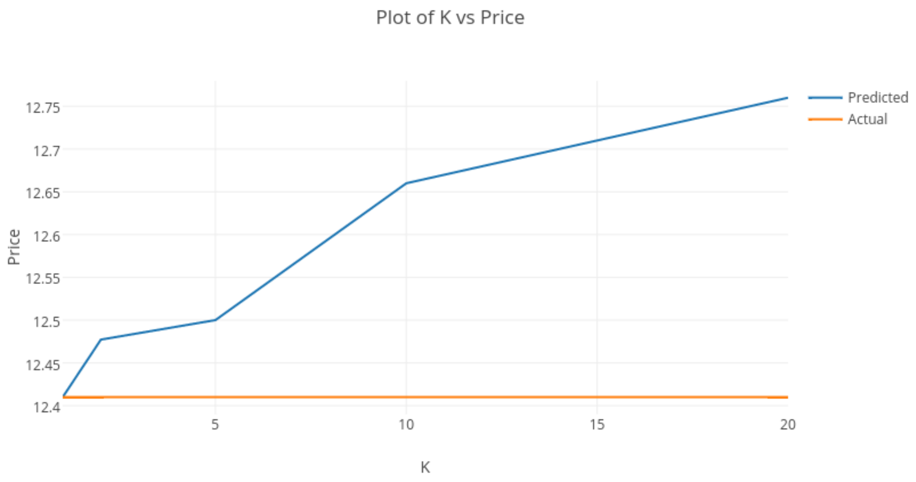
With some level of confidence in our sentiment analysis classification model we can run end of day price prediction for all the companies mentioned by Donald Trump and compare our predicted prices to the actual end of day prices for the days the tweets were made. The graphs below shows the plots of the predicted (Orange) and actual (Blue) prices. After we run leave one cross validation, one can easily see that somehow the model performed well and did a good job of predicting the end of day stock price after the tweet is made.



Accuracy numbers do not make sense with price prediction since this is continuous data but we can still use leave one cross validation and plot the predicted prices vs actual prices of each company tweeted by Trump.

Analysis of K

We noticed that the prediction model performed badly as we increased our k. We did not expect this so we did further investigation. It turns out that using a relatively small training dataset (26 company tweets) for our KNN algorithm meant that we were limited to a relatively small k. In the graph below, one can see that as k increases the SSE increases as well. In this case we could choose the optimal k to be either 1 or 2. While all price predictions are still in a 0.3 deviation margin we see the increase in error as k increases.



FInal Note

While combining the Naive Bayes model implementation and the KNN prediction implementation has its own drawbacks, it seemed to work relatively well in the case of stock data prediction. Accuracy numbers for sentiment analysis were relatively high but could be improved with a better tokenizing function. Feature extraction is always a challenge as the model has to decide what words are stopwords and not extract them as feature words. For the sake of development the model could be adjusted for real time stock prediction since the market does not wait until the end of day to react to Donald Trump's tweets. Given the individual we are dealing with, multiple tweets about the same company could undermine the validity of our model. However, in general, we are happy with the performance of our model.

Citations

Roesslein Joshua, Tweepy, Sphinx http://tweepy.readthedocs.io/en/v3.5.0/ , 2009
Quandl Inc, DMCA, https://www.quandl.com/tools/python ,2017
Laurent Luce, http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/, 2012
Johan Bollen, https://arxiv.org/pdf/1010.3003.pdf, 2010