Machine Learning Engineer Nanodegree

Capstone Project Report

Prakhar Choudhary
May 23, 2017

I. Definition

Project Overview

The stock market has always been a goldmine of adventures for mathematicians and statisticians.
They keep trying to find patterns such that the behaviour of stock market can be predicted, however the huge amount of data and buy/sell decisions carried out everyday makes it almost impossible to be analysed manually. This is where computers come in. Early research on stock market prediction was based on random walk theory and the Efficient Market Hypothesis (EMH)[1], which ironically is not so efficient. However in the recent few years the emergence of new organised data, high computational power and machine learning algorithms has given us the ability to use computer for predicting the behaviour of stock market.

Problem Statement

The goal of this project is to apply sentiment analysis and machine learning principles to find the correlation between ”public sentiment” and ”market sentiment”[2] and use it to predict stock market movement. Herein, the public sentiment refers to the sentiment values found out via classifying tweets as positive or negative while the market sentiment is merely a collection of open, close, high and low prices of the DJIA each day. Furthermore the project will learn from new data every day from the previous day's data(using Yahoo Finance API).
Since the final aim of the project is prepare a model that suggest us whether the next day's closing price will be greater or less than today's closing price, hence, the final model will actually be a classification model which will determine the next day's closing price status as increasing or decreasing, up or down, more or less.
The project is inspired from a research paper by Anshul Mittal and Arpit Goel on “Stock Prediction Using Twitter Sentiment Analysis”.[2]

Metrics

The performance was evaluated on the basis of accuracy of each of the six algorithms used, when tested on the Test dataset. The reason for choosing accuracy is that the purpose of this project is to be as efficient as possible when predicting whether the next day's closing price will be moreor less than current day's closing price. Hence, metrics like r2_score which determine how close the prediction is to the real value which will not serve the purpose necessarily, since a predicted value which is very close to the true value but somehow less than the previous day's closing price while the true value is more, completely defeats the purpose as the predicted value will give a 0 while true value means a 1 in terms of increase.

The test dataset is the last 20% data of entire dataset.

In case of classification approach, depending on given labels a values out of 1 or 0 is predicted wherein 1 means increase while 0 means decrease. This value is matched against the true value and hence the accuracy is tested.

In case of regression approach, depending on given labels a new value is predicted(i.e. Next day's closing price) which is then compared with the current day's closing price and if found less, the prediction is coined 0 else 1. This series of 0s and 1s are the prediction values which are matched against the true testing targets to calculate accuracy. Once again the usage of regression metrics is not necessary as the purpose of our model is to find how correctly it classifies the next day's closing price status and has nothing to do with the closeness of predicted value to the true value.

To carry out the comparison and accuracy calculation, in classification approach, I used the score function of all the classification methods given in sklearn, whereas for the regression approach, I used the accuracy_score function of scikit-learn(sklearn.metrics.)

II. Analysis

Datasets and Inputs

Two main datasets will be used in the project which are:

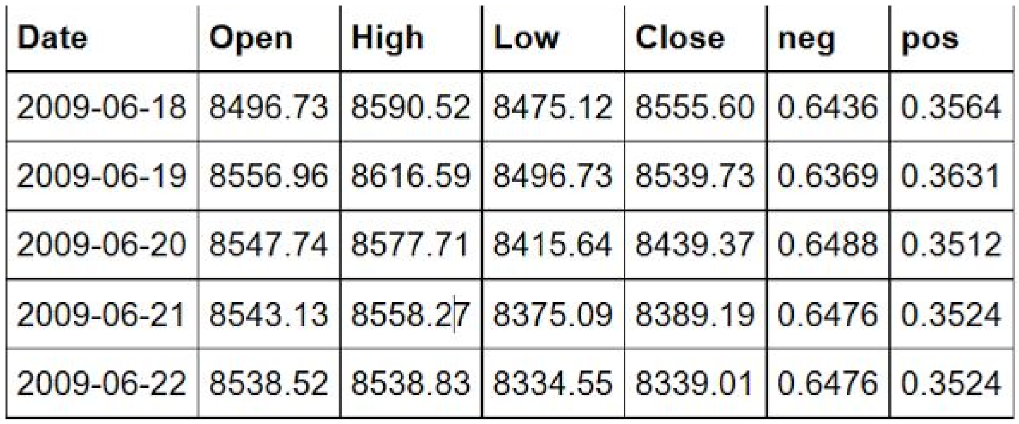
1. Dow Jones Industrial Average (DJIA) values from June 2009 to December 2009. The data was obtained using Yahoo! Finance and includes the open, close, high and low values for a given day.

2. Publicly available Twitter data.
   This data was was collected in the following manner:

a. A web scraper automatically scraped about 2000 tweets for each day. These tweets contained words like ‘feel’, ‘feels’, or ‘I’m feeling’, to ensure the tweets conveys a sentiment.
b. The prepared dataset of tweets was then labeled either 0 or 1, on the basis, whether the tweet is predicted to be positive or negative by the Sentiment Analysis classifier developed by me.
c. Later, once all the tweets were tagged with a binary sentiment scheme, each day's positive and negative index was calculated by finding the ratio of count of one kind of tweets over the count of all tweets for a day.

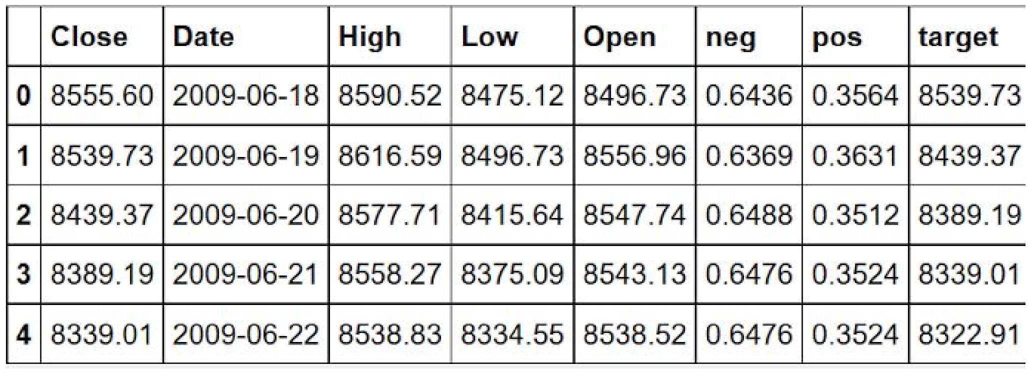
3. Final Dataset

The final data set is collection of data points/values for the categories, Close, Open, High, Low, positive sentiment index and negative sentiment index of each day from 18 June 2009 to 16 December 2009. A sample is as follows:



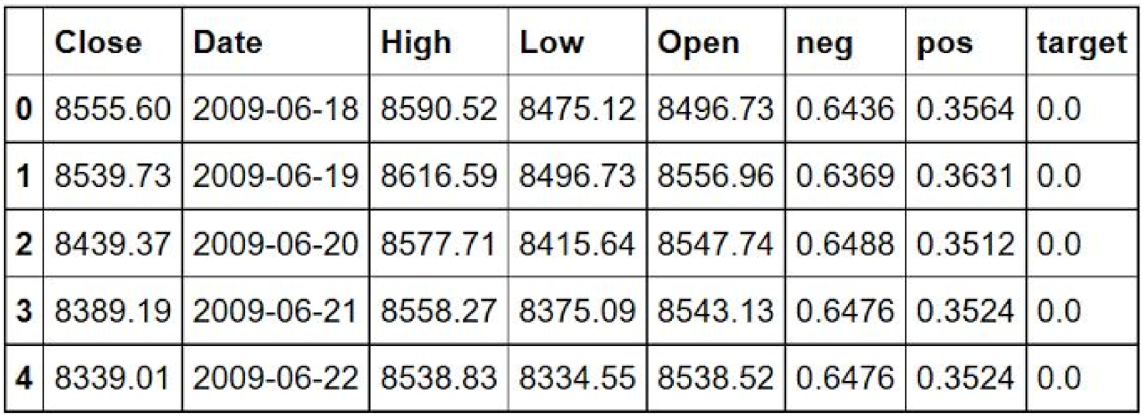
Date	Open	High	Low	Close	neg	pos
2009-06-18	8496.73	8590.52	8475.12	8555.60	0.6436	0.3564
2009-06-19	8556.96	8616.59	8496.73	8539.73	0.6369	0.3631
2009-06-20	8547.74	8577.71	8415.64	8439.37	0.6488	0.3512
2009-06-21	8543.13	8558.27	8375.09	8389.19	0.6476	0.3524
2009-06-22	8538.52	8538.83	8334.55	8339.01	0.6476	0.3524

It still lacks targets which are actually the next day’s closing price. Hence the new dataset will be:



	Close	Date	High	Low	Open	neg	pos	target
0	8555.60	2009-06-18	8590.52	8475.12	8496.73	0.6436	0.3564	8539.73
1	8539.73	2009-06-19	8616.59	8496.73	8556.96	0.6369	0.3631	8439.37
2	8439.37	2009-06-20	8577.71	8415.64	8547.74	0.6488	0.3512	8389.19
3	8389.19	2009-06-21	8558.27	8375.09	8543.13	0.6476	0.3524	8339.01
4	8339.01	2009-06-22	8538.83	8334.55	8538.52	0.6476	0.3524	8322.91

Since we are only concerned with the fact whether next day’s price will be higher or lower than the current day’s closing price we can transform the targets into 1 or 0. We shall achieve this by comparing targets with the closing price and if found greater, it will be coined as 1 else 0. This ensures our problem can be solved well using a classification model.



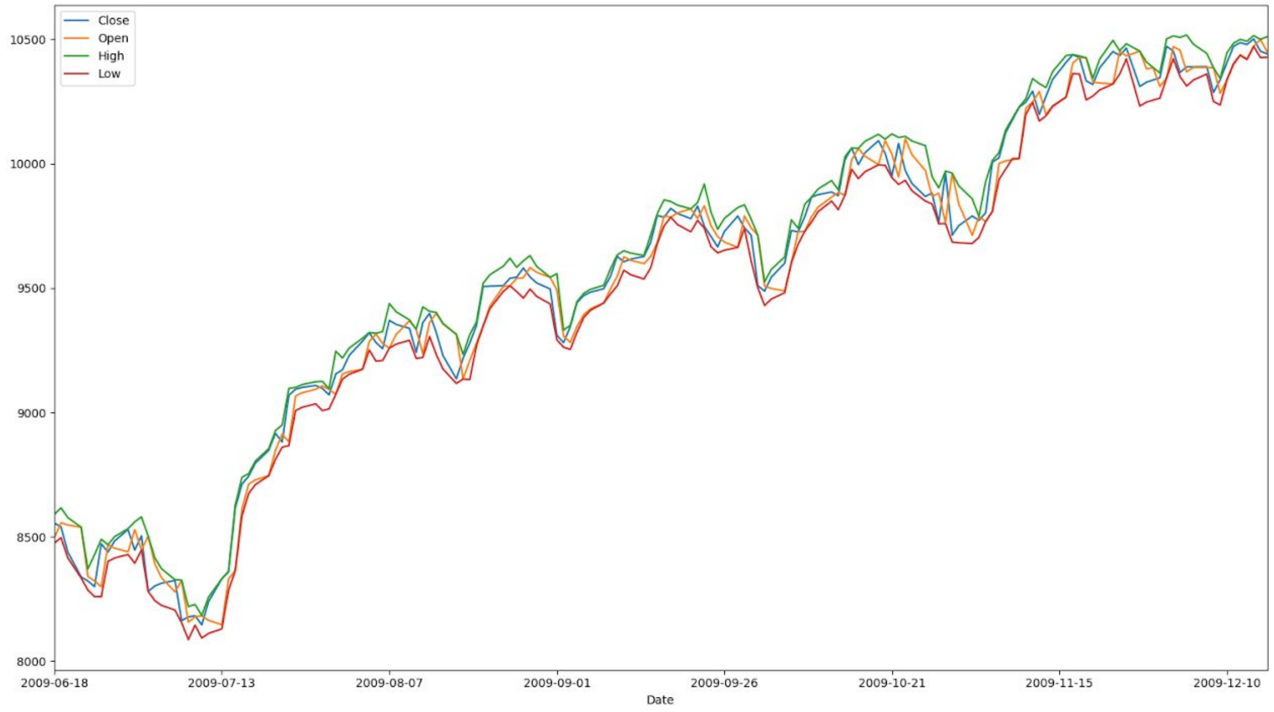
	Close	Date	High	Low	Open	neg	pos	target
0	8555.60	2009-06-18	8590.52	8475.12	8496.73	0.6436	0.3564	0.0
1	8539.73	2009-06-19	8616.59	8496.73	8556.96	0.6369	0.3631	0.0
2	8439.37	2009-06-20	8577.71	8415.64	8547.74	0.6488	0.3512	0.0
3	8389.19	2009-06-21	8558.27	8375.09	8543.13	0.6476	0.3524	0.0
4	8339.01	2009-06-22	8538.83	8334.55	8538.52	0.6476	0.3524	0.0

The features which will be used to train the model are Close, High, Open, Low, negative sentiment index(neg) and positive sentiment index(pos) for each day.

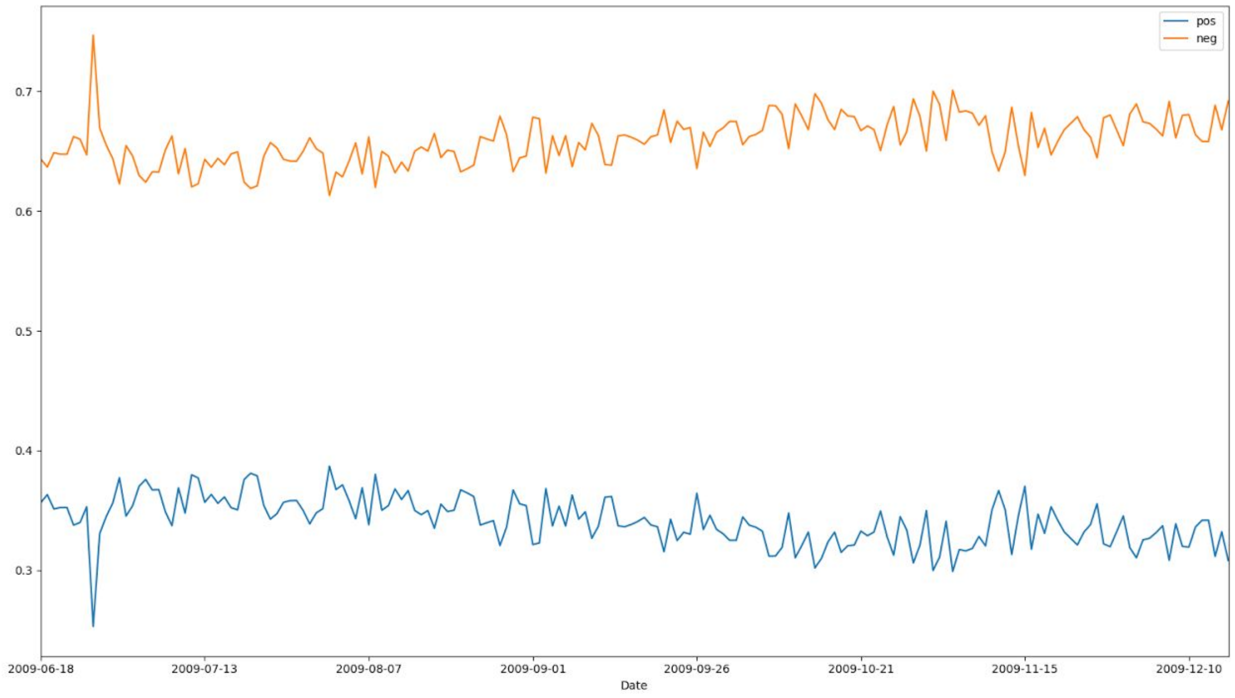
Furthermore, the DJIA dataset was missing data for a few days which was filled in using a concave function as mentioned in Implementation section.

Exploratory Visualizations

Given below is a visualization of DJIA price indices over the span of 6 months:



And the following visualization shows the trend of positive and negative index of the day based on the tweets for each day:



Algorithms and Techniques

Logistic Regression:
The problem is more or less approaches as a regression analysis in which the dependent variable(s) (majorly closing price, pos and neg) despite being distinct numeric values are still treated as dichotomous in nature when trained upon by the model as herein we are only concerned whether they rise or fall. Hence, for this approach Logistic Regression was one of the best options. Logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

The other two classification algorithms, i.e., SVM and SGDClassifier were tried as mix of trial-error approach and a little reference from the official sklearn cheatsheet[4].

For the regression algorithms, I merely tried them arbitrarily as as expected the results were not pleasing.

Benchmark

The work is based on Bollen et al’s strategy [1] and Anshul Mittal’s Paper[2] on it using their own validation metric and sentiment analysis mechanism. They also attempted to predict the behavior of the stock market by measuring the mood of people on Twitter. While Bollen et al’s work which classified public sentiment into 6 categories, namely, Calm, Alert, Sure, Vital, Kind and Happy, achieved a remarkable accuracy of about 87% in predicting the up and down changes in the closing values of Dow Jones Industrial Index (DJIA), Anshul Mittal’s approach led to a **75.56%** accuracy. The results of a naive classifier however lie in the range of **44%-50%** since by using probability theory in this case more or less results in an equal chance of going either way, i.e. increase or decrease.

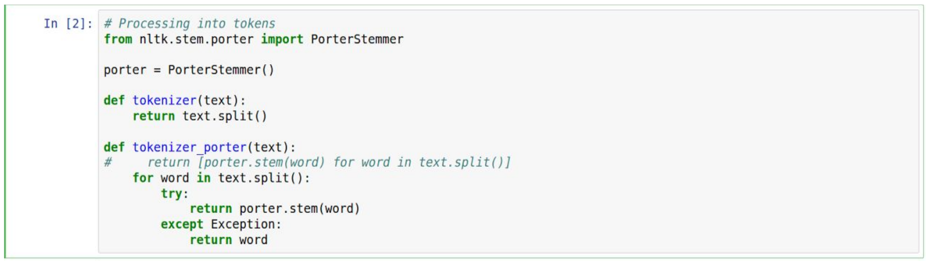
III. Methodology

Data Preprocessing

- *Filling the missing data for the DJIA dataset*
  The DJIA dataset had values missing for several days. It must be because the DJIA is not operational on Saturdays, Sundays and national Holidays.
  Hence, to find values for such dates, a concave function was used.The missing data is approximated using a simple technique by Goel [2]. Stock data usually follows a concave function. So, if the stock value on a day is x and the next value present is y with some missing in between. The first missing value is approximated to be (y+x)/2 and the same method is followed to fill all the gaps.

- *Cleaning the tweets*
  Tweets consists of many acronyms, emoticons and unnecessary data like pictures and URL’s. So tweets are preprocessed to represent correct emotions of public. For preprocessing of tweets we employed three stages[3] of filtering: Tokenization, Stop words removal and regex matching for removing special characters.

  1) Tokenization: Tweets are split into individual words based on the space and irrelevant symbols like emoticons are removed. We form a list of individual words for each tweet.



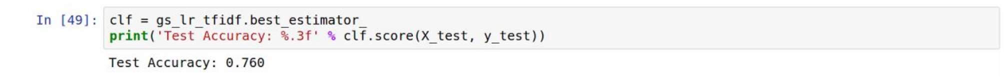
```
In [2]: # Processing into tokens
        from nltk.stem.porter import PorterStemmer

        porter = PorterStemmer()

        def tokenizer(text):
            return text.split()

        def tokenizer_porter(text):
        #     return [porter.stem(word) for word in text.split()]
            for word in text.split():
                try:
                    return porter.stem(word)
                except Exception:
                    return word
```

  2) Stopword Removal: Words that do not express any emotion are called Stopwords. After splitting a tweet, words like a,is, the, with etc. are removed from the list of words.

3) Regex Matching for special character Removal: Regex matching in Python is performed to match URLs and are replaced by the term URL. Often tweets consists of hashtags(#) and @ addressing other users. They are also replaced suitably. For example, #Microsoft is replaced with Microsoft and @Billgates is replaced with USER. Prolonged word showing intense emotions like coooooooool! is replaced with cool! After these stages the tweets are ready for sentiment classification. Also all the emoticons were removed from their positions in the tweets and appended at last.

Implementation

- *Sentiment Analysis:*
    - I had originally proposed a POMS model based approach which would help me find the emotional indices for a day in form of a fractional value each(between 0 and 1) for 4 different sentiments: Calm, Happy, Alert, and Kind.
    - However, I realised that not only was the task quite laborious and difficult to automate since not only did we have to prepare an entire dictionary of words corresponding to each sentiment, but also prepare an daily score by calculating the aggregate count of words for each sentiment divided by the total number of words.
    - Hence, I followed a different approach in project. I downloaded a dataset of Tweets tagged with a binary sentiment scheme, i.e, 1 for positive and 0 for negative and trained a Logistic Regression Classifier on it. To obtain the best result I used the estimator along with GridSearchCV.
    - The above approach returned an accuracy of 76.0% on the testing set. Now in the paper [3] it is mentioned that even human accuracy lies between 70-79% so i deemed this accuracy good enough.



```
In [49]: clf = gs_lr_tfidf.best_estimator_
         print('Test Accuracy: %.3f' % clf.score(X_test, y_test))

         Test Accuracy: 0.760
```

    - Then I used this classifier to tag all the tweets as either positive(1) or negative(0).
    - Since, each day had about 2000 tweets and aggregate sentiment had to be calculated for finding correlation between twitter sentiment and DJIA price index.
    - Hence I found the positive and negative index of thee day by finding the ratio of number of tweets of one kind over the total number of tweets for the day.
    - This however posed another problem. All days had almost a 66% negative index and 33% positive index and hence I could no longer clearly state whether the day was positive one or a negative one.
    - Hence in the final dataset i added both positive and negative index of the day as a feature so that the slightly varying values of the will still be meaningful.

- *Model Training and Prediction:*

  - Firstly, I created the targets for my dataset. These targets are actually the next day's closing price for everyday's record.
  - Since the dataset is a time series I could not simply shuffle the dataset hence, I simply split the initial 80% data into training set and the remaining 20% data in testing set.
  - Now the problem can be treated in two ways:

    Classification
    - Herein, the targets are were updated to either 1 or 0 depending on the comparison whether they were larger or smaller than the previous day's closing price. This means if the target value is greater than the day's closing price the target would be set as 1 else 0.
    - This ensures we have a dataset on which we can now carryout binary classification.
    - This dataset was then trained upon by three classification algorithms:
      a) Logistic Regression
      b) SVM(kernel: rbf)
      c) SGDClassifier
    - All the above three algorithms gave the same accuracy of 63.89%.

    Regression
    - For a numerical dataset, like ours, another approach is to train a supervised regressor on the time series data and then predict for each record it to see if the value is above the previous day's closing price or not. This gives us a prediction label of 1 or 0 which is then compared to the testing targets for calculating accuracy.
    - We used 3 different regression algorithms:
      a) Lasso
      b) ElasticNet
      c) RandomForest
    - While the first two returned 55.56% accuracy while the RandomForestRegressor returned 36.11% accuracy.

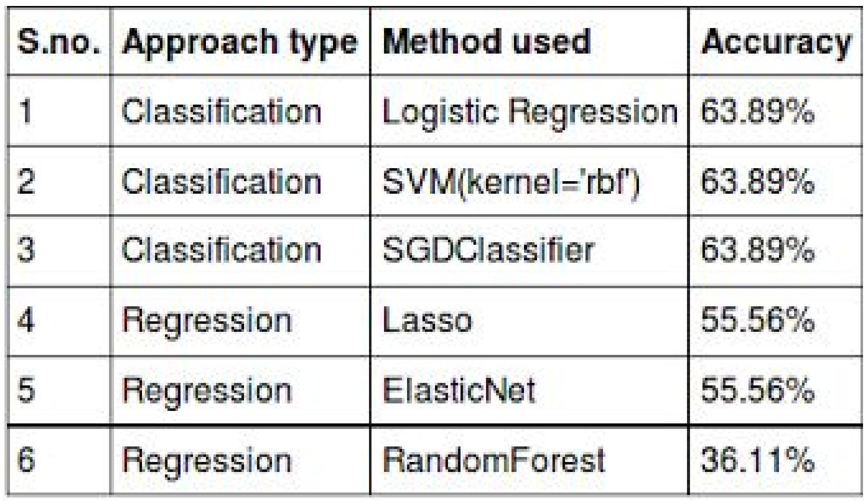
Refinement

The algorithms returned almost the same results even with varying parameters. In Logistic Regression I changed the maximum iterations to both more and less than previous values by substantial amount, it still had made no difference.
The loss parameter was set to ‘perceptron’, in case of SGDClassifier, however, as the default ‘hinge’ loss returned a worse accuracy. This change though did nothing to contribute to the maximum accuracy as it still remained 63.89%.
The SVMs however had an entirely different case. Tinkering with parameter led to no change unless i varied the kernel. So, I tried different kernels and found out that for any set of values for other parameters ‘rbf’ returned the best results.
The regression algorithms as mentioned earlier were chosen arbitrarily and thus as expected, performed poorly on the dataset and no amount of parameter adjustment helped their case.

IV. Result

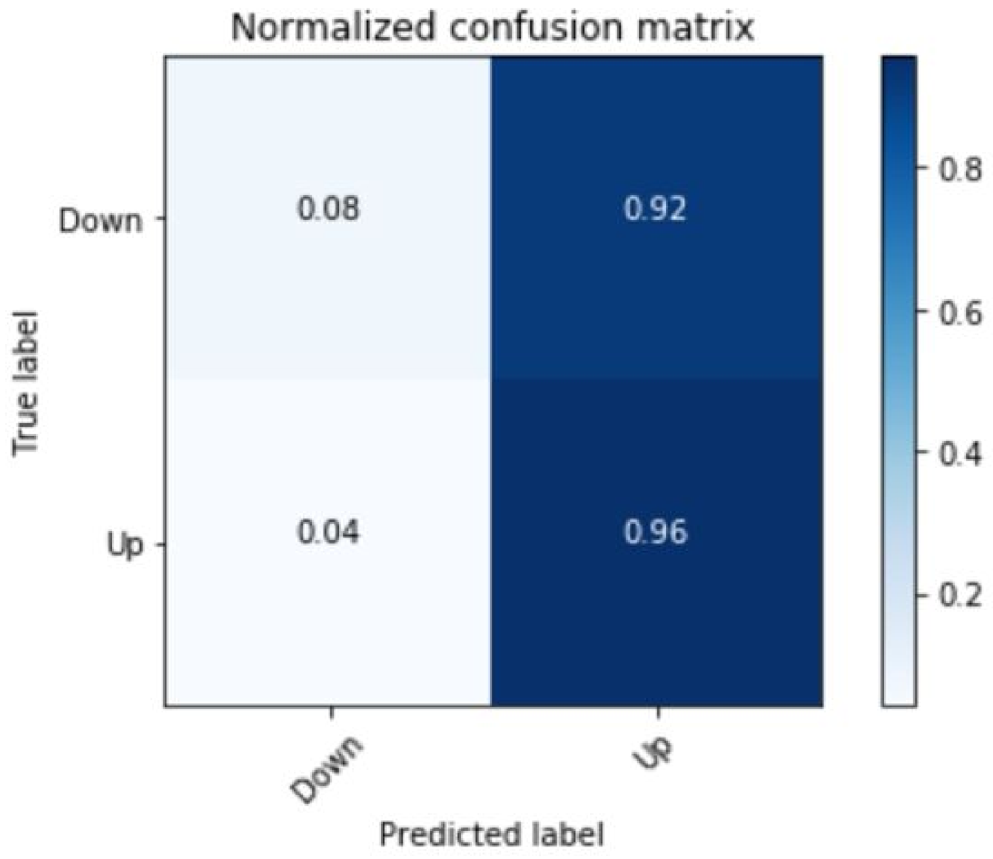
Model Evaluation and Validation

The finals results can be tabularized as:



S.no.	Approach type	Method used	Accuracy
1	Classification	Logistic Regression	63.89%
2	Classification	SVM(kernel='rbf')	63.89%
3	Classification	SGDClassifier	63.89%
4	Regression	Lasso	55.56%
5	Regression	ElasticNet	55.56%
6	Regression	RandomForest	36.11%

The above data clearly states that should the problem be approached via classification, we achieve the maximum accuracy. Hence, either of the three classification based models will serve the purpose.
With the accuracies in hand, it is time to check out for robustness. I shall approach it using a confusion matrix to see how well it predicts either of the classes. Given below is the normalized confusion matrix:



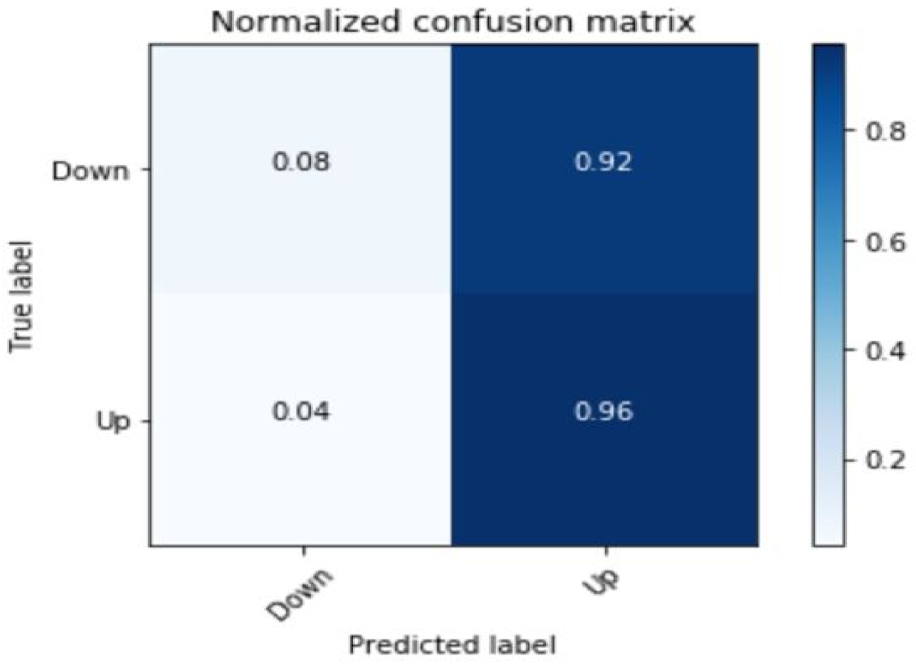
This leads to the realization that the model has generalized for a class and hence is not robust. This is something to work upon later on and hence can be included in the improvment section.

Justification

As per the benchmark section, I had hoped to achieve or surpass the accuracy of 75.56%. However, the vast difference in dataset made me realize that it was was a very tough task. Hence, my goal shifted to that of surpassing the conventional human prediction on basis of trends which is about 45-50% accurate. My results of 63.89% accuracy very well lie above this standard and hence, I can justify the progress and final output of my approach. However, I realized it is not robust and that the project has a long way to go and can be made better in several different ways.

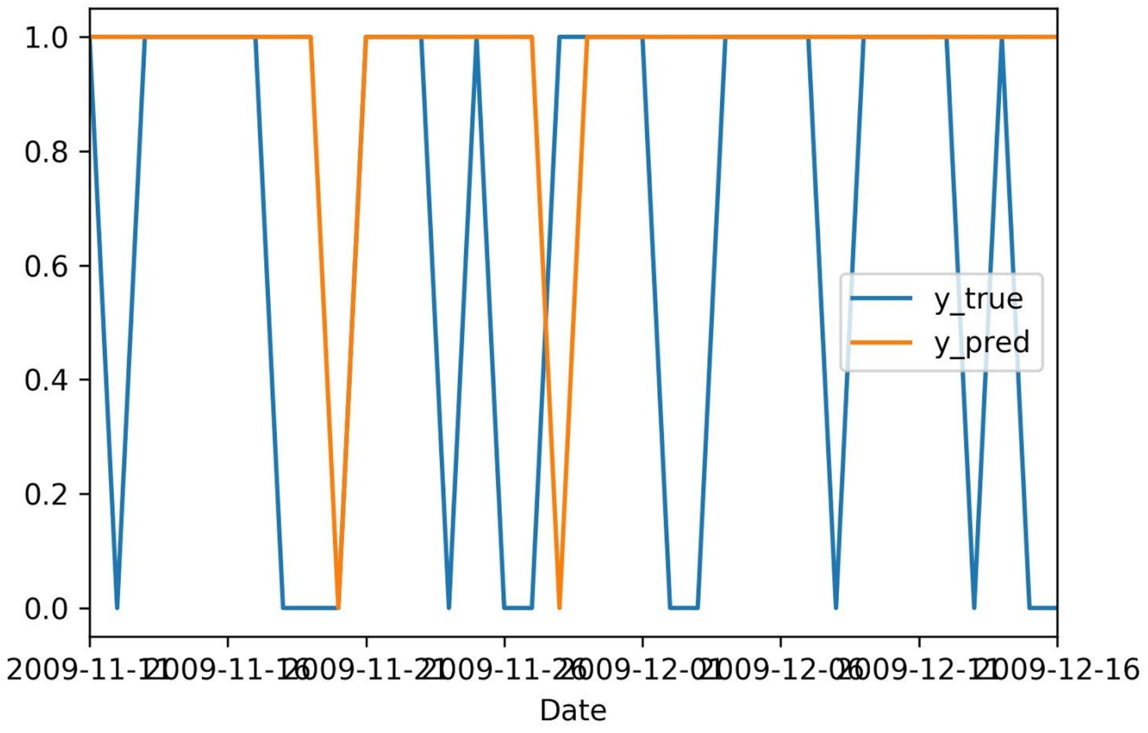
V. Conclusion

Free-Form Visualization



As mentioned above, all three classifier seem to return the same confusion matrix. Furthermore, the problem seems to be that it has generalized over the majority class ‘UP’(i.e it almost always predicts the price will go higher than the previous day), whereas the ‘DOWN’ seems to be left out.

Another helpful visualization is the correct vs predicted graph.



This visualization very well describes how the model eventually generalizes to learn only the majority class, however since we lack data to train it on any further and the accuracy seems fine as well, we can wrap it up.

Reflection

I originally began the project with an aspiration to achieve at least the same accuracy as the in the referred paper[2]. However, the first trouble I encountered was with the dataset itself. The DJIA prices dataset had data missing for several days as prices for off day's were not available. Luckily, the solution to this problem was simple and clearly mentioned in the research paper itself. But, twitter dataset was still the the biggest roadblock. Not only did I have to scrape a lot of tweets, it had to be done without the help of any available twitter apis as they only return data for upto past 7 days, whereas the data required by me was almost a decade old.
Hence, I had to resort to conventional scraping from the new Twitter Advanced Search. However scraping all tweets of each day for six months would have been extremely time taxing and carrying out any form of learning on such a huge dataset in my machine would be nearly impossible. So I sufficed with scraping about 2000 tweets for each day in the six month range and that compiled about 350,000 tweets.
This dataset though was quite small in comparison to the one mentioned in the paper[2] and hence I knew before hand that achieving the aspired accuracy was almost impossible.

Next was the task of processing this data and tagging it. Once again the the POMS model based approach for 4-class based classification of the tweets seemed quite vexing as well as demanding in nature. Not only did it require me to create a dictionary of synonyms for each of more than 65 words, it was supposed to look for these words in each tweet maintain an index for each words and then needed me to come up with a way to determine how to use count of these 65+ words to achieve the 4-class based classification.

So, i rather chose to go with a much better and reliable approach. I decided to train a classifier which can successfully classify the sentiment of a sentence as either positive or negative. This was done by training a Logistic Regression classifier(a probability distribution based method) on pre-tagged twitter dataset from the a previous kaggle challenge. This classifier was then pickled and used to tag the scrapped tweets.

Now each day had about 2000 tweets and each tweet had a sentiment. However, I required a net sentiment for each day such than it be used as a feature along with the 4 prices of DJIA data, for any given date. Hence, I counted the fraction of positive and negative tweets each day and believed that for any day whichever of the two values is higher, that sentiment dominates the day. However, once I got data, a new challenge appeared. For every day, about 64% tweets were negative and 36% were positive(with about +-1% variation). The only difference was in the float values which varied each day, but when rounded, remained the same. So rather determining whether the day was positive or negative I chose to use these fractional values as two different features for everyday.

With the final dataset ready, I prepared the targets as mentioned in the Implementation section and decided to carry out training and comparing results. The training included trying different learning approaches or algorithms, tuning parameters sometimes for specific reason while sometimes arbitrarily and then eventually collecting them all together to visualize and evaluate the model, its accuracy and its robustness.

Improvement

The origin paper [1] was successful in achieving an accuracy of 87% while the one by Anshul and Arpit[2] achieved an accuracy of about 75%. Hence, even after submission and acceptance of the project I will continue to work on it. The major changes required in my opinion are:

1. Larger dataset.(Both the above paper mentioned usage of a dataset of 400+ million tweets while mine was merely 350K tweets, hence a much larger dataset is required to obtain better results.
2. The sentiment tagging was quite primitive and hence will be improved. Not only will I use multi class tagging but will also look for better and latest researches.
3. I did not use the neural networks based learning which served as a crucial agent in improving the accuracy in referred paper[2]. Hence, I will look into neural nets based learning to improve the accuracy.
4. Look into the matter of robustness as the model eventually seems to generalize towards the majority class.

Libraries and Packages

1. Scikit-learn: http://scikit-learn.org/stable/user_guide.html
2. Nltk: http://www.nltk.org/
3. Scraping twitter: https://github.com/prakharchoudhary/TwitterAdvSearch

Acknowledgement

[1] J. Bollen and H. Mao. “Twitter mood as a stock market predictor”. IEEE Computer, 44(10):91-94
[2] Mittal Anshul, and Arpit Goel. ”Stock prediction using twitter sentiment analysis.” Stanford University, CS229(2011 http://cs229.stanford.edu/proj2011/GoelMittalStockMarketPrediction-UsingTwitterSentimentAnalysis.pdf) (2012).
[3] Venkata Sasank Pagolu, Kamal Nayan Reddy Challa, Ganapati Panda, Babita Majhi. “Sentiment Analysis of Twitter Data for Predicting Stock Market Movements”.

arXiv:1610.09225v1 [cs.IR] 28 Oct, 2016
[4]http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html