Predicting Stock Market Movements from News Headlines

Jesse Liang
CSE 158
University of California, San Diego
j4liang@ucsd.edu

ABSTRACT

This paper will detail efforts of applying the machine learning methods taught in CSE 158/258 in predicting stock market movements. This paper examines the applications of regression models, classifier models, and recommender systems in predicting stock market movements.

1 DATASET SELECTION

In order to predict stock market movements from headlines, I had to find two datasets: a news headline dataset and a stock market price dataset.

The stock market dataset could be easily found and could also be acquired from sources like Yahoo Finance. I decided on using a dataset on the prices of S&P 500 companies from Kaggle[1].

The news article dataset was harder to find, as many of financial news datasets have been taken down due to copyright issues. I ended up having to experiment on two different news datasets because I was unsure which would produce the better model. There was also another set on Kaggle on headlines spanning from 2008 to 2016, but it did not meet the dataset size requirement[2].

Both news datasets provided news headlines and their publishing dates, on top of other data that I did not utilize. The first news dataset I found was a financial news article dataset on Kaggle[3]. It was a large dataset of articles, but I decided against using this dataset because of its short span of articles, which I will explain later.

The second news dataset was a HuffPost news headline dataset, which produced a better model, even though HuffPost is not a financial news site[4]. As a result, it is possible that the dataset has many random and unrelated articles that could cause our models to overfit on irrelevant data

1.1 NEWS DATASETS

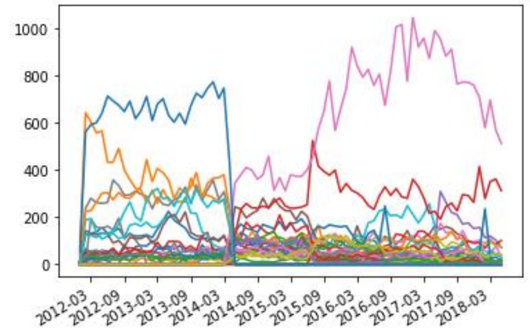
Table 1. Basic info about news datasets

	HuffPost	Financial
# of Articles	200853	306125
# of sites	1	14
Range of dates	~ 5 years	5 months
Years	2012-2018	2018

Seeing the basic information about both datasets in **Table 1** above, we can see possible reasons why the HuffPost dataset would perform better. One example would be the range of dates for the financial dataset. Because the data was limited to only 5 months of news articles, it is quite easy to overfit on a small period. This period would not provide as much information about significant market adjustments that might happen every few years. For example, the presidential election in 2016 could have caused a significant market movement, but the financial dataset could not track that. Furthermore, a 5-month period could not cover things like large market adjustments, bull markets, etc. In the end, I found that the HuffPost dataset produced a better performing model.

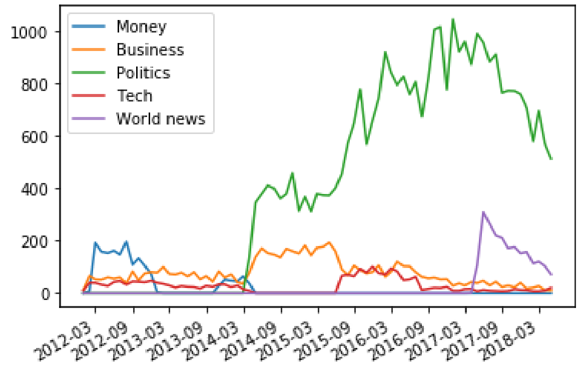
[1] Nugent, Cam. (2018). S&P 500 stock data, Version 4.
[2] Sun, J. (2016, August). Daily News for Stock Market Prediction, Version 1.
[3] Jeet.J. (2018). US Financial News Articles.
[4] Misra, Rishabh. (2018). News Category Dataset.

Figure 1. HuffPost News Category Freq.



Analyzing the HuffPost dataset, we can see some of its shortcomings. As stated before, HuffPost is not a primarily financial news provider. Therefore, it is quite possible we train our models on irrelevant news. We see the frequency of articles per category over the course of 5 years in **Figure 1** above. I also isolated the 5 news categories that I believed were most relevant to the finance and the stock market in **Figure 2** below. As we see, categories like money and business are relatively infrequent compared to others.

Figure 2. 5 Finance-related Categories



1.2 STOCK MARKET DATASET

I had chosen to use S&P 500 for this experiment because it is a good indication of the general stock market growth and movement. My dataset provided the stock price of each individual S&P 500 company in February 2018. This means that even if the company was not part of S&P 500 prior to February 2018, this dataset would still track its stock prices over the 5 years. I used the aggregate closing price of S&P 500. This means at any given date, the price would be equal to the sum of the closing prices of all S&P 500 companies. **Figure 3** shows the aggregate price.

Figure 3. Aggregate S&P 500 Price

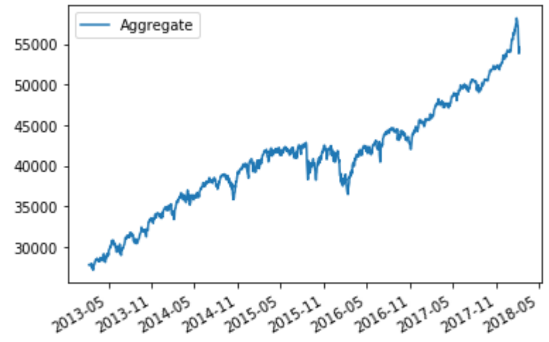


Table 2. Basic info about aggregate price

Starting price	$27783.48
Ending price	$54419.91
Total growth	$26636.43
Total Change (%)	+ 95.87%
Avg. Change Per Day	+ 0.03898%
Avg. Change Per Day*	+ 0.05655%

* Average change per trading day (Excludes weekends, holidays, etc.)

2 PREDICTIVE TASKS

The objective of using these datasets is to help predict stock market movements for stock trading. Therefore, we want a model that can predict how much the stock market will grow given a certain article (positive or negative). However, it would still be useful to only know if the market was going up or down.

Therefore, the model that would most fit our needs would be a **regression model**, which could predict the price change of the general stock market. Furthermore, we can explore other models that can perform the same task, such as a **recommender model** that predicts price changes. Lastly, we can also see how it performs against a **classifier model**, which can only say if the stock market goes up or down.

2.1 POTENTIAL MODELS

For my regression models, I used **Linear Regression** and **Ridge**. These models are effective for predicting a value given input, which would be perfect for predicting the price change for a given day.

For my classifier models, I **used Logistic Regression** and **Gaussian Naïve Bayes**. These models are effective for classification. In this case, we would classify articles as positive price change and negative price change (stocks going up or down).

For my recommender, I used **Cosine Similarity** and predicted the price change by finding the most similar articles. Essentially, articles are the "user" and words are the "items". We are doing user to user similarity.

2.2 FEATURES & DATA

The purpose of this experiment is to take news article headlines and predict something about the stock market. Therefore, we will use NLP methods to break down headlines into features. Then we will train our models on these features to produce our desired output.

I tried a combination of different feature representations to find the most optimal model. I transformed headlines into features with **Bag-of-words Model** and **TF-IDF**. Furthermore, I used different **Unigram** and **Bigram** models. For both bag of words and TF-IDF, I will only use the 1,000 most popular words, meaning my input data is a list of 1,000 counts or TF-IDF scores.

The S&P 500 dataset contained the pricing information for all 500 companies throughout 5 years (Ex: Pricing of Apple stock for 1259 trading days). I simplified this by calculating the aggregate pricing at each date (the sum of all stock prices at each date). For my training output, I used the price change from the previous aggregate price. This is because training on the total aggregate price would not make sense, as I wanted to create a model that would predict how the market would react to news. Moreover, using price delta data would not make sense, as that value is dependent on the stock price at that date. Therefore, it was obvious to use the **price change percentage** for my training output.

$$\text{price change} = \frac{\text{new price - old price}}{\text{old price}}$$

Furthermore, I had to fill in holidays and weekends, as trading stops on those days. I experimented with using the next trading day for holidays (ex: Using Monday's price change for Saturday and Sunday). However, I found that filling the change as 0% change created more accurate models.

For the usage of this data, I pair each article headline with the price change (percentage) of its publish date. Because the dates of the articles and stock market data did not perfectly line up, the length of final dataset was 159,735.

2.3 METRICS & BASELINE

For the regressor models, I will use **Mean Squared Error** as my metric, with the baseline being the average percent-changed per day. This means my baseline prediction would be 0.03898% for every article.

For the classification models, I will use an **accuracy score** (correct/all), with a baseline of random guessing.

3 PROPOSED MODEL

The model I propose to perform this predictive task is the **Ridge Regression**. This model would produce an output that tells us both market direction (up or down) and magnitude (how much it changed). It would be most useful to use this model.

I chose Ridge over Linear Regression because it tries to minimize overfitting the model. Because we have a lot of data that might be irrelevant, such as gossip news, we would want to use Ridge to minimize the influence of these data points.

Other models I considered were Logistic Regression, Naïve Bayes, and a recommender model. I decided against Logistic Regression and Naïve Bayes because they are classification models and do not provide an output as useful as our regression models. My recommender model can provide the same outputs, but it performed much worse, as I will show in the results. Therefore, it was obvious to me that Ridge would be the model that I want.

3.1 OPTIMIZATION & SCALABILITY

For both Ridge and Logistic Regression, I created pipelines to run the models on different regularization parameters. For Ridge, I found that

I also split up my data into training, validation, and test sets. 75% training, 25% validation, and 25% test.

I found that as I scaled up, I achieved a better score. I tested with 30,000 datapoints, then 100,000, and finally the full set of 159,735.

3.2 UNSUCCESSFUL ATTEMPTS

This dataset is a compilation of news articles for a 5-year period. Therefore, each article is directly tied to a publishing date. My first attempt was to *not* randomize my data because that would result in duplicate articles in the training and test data. For example, if two articles were written about Miley Cyrus on Thanksgiving, they could be separated into training and test sets when randomizing. **This would essentially be using "future" data to predict the future.**

However, I thought that it could interesting to run the model on a randomized set. Interestingly, while randomizing could train the model to predict the "future" using future data, it performed slightly worse than not randomizing.

Therefore, it would be in my best interest to not randomize the data, as chronology was important and it produced a better model.

3.2 STRENGTHS & WEAKNESSES

The strengths of Ridge over Linear Regression are clear, as it reduces double-counting in our model. Ridge also beats classification models like Logistic Regression and Naïve Bayes because its output can provide more information.

The weaknesses of our regression models will be shown when we finally compare our models in the end.

4 LITERATURE

The news dataset I used, "News Category Dataset" was intended to train models to categorize news articles by their headlines. Therefore, the studies on this dataset would be unrelated to this.

However, there are many other studies that attempt to predict stock market movements from news articles or news headlines. Of the studies I read, I found two that had success in their findings.

The first study, called "Stock Trend Prediction Using News Sentiment Analysis," applies some of the same NLP methods as I did in my project, such as TF-IDF and Bag of Words[5]. The differences are that this study tries to take one step further in their NLP analysis. Where my models use NLP to correlate characteristics of text to stock price, this study uses it to correlate text with sentiment. This means their NLP attempts to further understand the meaning behind the article (determining whether it is positive or negative), rather than simply looking for similarities in headlines. They then used this sentiment analysis model to correlate with stock price. Their classifier models (Naïve Bayes, SVM, Random Forest) for sentiment analysis have a performance of over 83% accuracy, with SVM performing a 92% accuracy.

The second study, "Stock Trend Prediction Using News Articles," also had similar results[6]. Using an SVM model to predict the positive or negative effects of a news article, they were able to choose an accuracy rate of 83%. Interestingly, they also discussed the "Efficient Markets Hypothesis," which states that the market reacts to new information only. This means that my assumption to pair market movements to an article's publish date was somewhat backed by financial market theory.

It was interesting that the studies on this topic tend to lean towards a classifier model

[5] Joshi, Kalyani & N, Bharathi & Rao, Jyothi. (2016). Stock Trend Prediction Using News Sentiment Analysis.

[6] Falinouss, P. (2007). Stock trend prediction using news articles : a text mining approach.

rather than a regression model, specifically SVM. As you will see in part 5, my results match that conclusions that others have made, albeit not as successfully.

5 RESULTS & CONCLUSIONS

A regression model would give us exactly what we desire, but we will see in our results that the classification models do much better. While classification cannot be a replacement for regression, it will be clear that our classification models provide more utility than our regression models.

But first, let us compare and analyze our non-classification models. Our recommender model acts like a regression model. It uses the price changes of the most similar articles to predict its own price change. Therefore, we will compare it directly with the regression models.

Table 3. Rec. & Regression Model MSE's

	Ridge	Linear	Rec.
uni, tfidf	4.128e-5	4.198e-5	N/A
uni, BW	4.127e-5	4.198e-5	N/A
bi, tfidf	4.127e-5	2.58e+20	4.50e-5*
bi, BW	4.127e-5	2.05e+20	N/A

Baseline MSE: 4.128e-5; BW = Bag of Words
* Tested on only 1000 articles (1.5 hours)

Clearly, these models failed because the same result could be achieved by simply using the average price change. Recall that our baseline was simply using the mean price change for each output.

Although the MSE's look small, they represent the mean *squared* error. Because we are dealing with decimals, the actual error is orders of magnitude greater. Furthermore, in the context of the average price change of 0.03898% or 0.0003898, the actual error would be greater than the value itself. This means it would be common to predict a negative change while it is actually positive, or vice versa. **These models are essentially ineffective**, as we can get the same results by getting a line of best fit. This renders the model useless because we want to predict the price changes based on news articles, not the stock price given a date.

Table 4. Classification Model Scores

	Logistic	Naïve Bayes	Base
uni, tfidf	0.55717	0.51791	0.50261
uni, BW	0.55347	0.51791	0.50261
bi, tfidf	0.60383	0.39759	0.50261
bi, BW	0.60231	0.39810	0.50261

BW = Bag of Words

Table 4 shows that logistic regression comes out on top, using bigrams and TF-IDF scores. While the model is not incredibly accurate, you are still more likely to get a correct answer than not.

Our desired model can predict if the market direction and magnitude. We have seen that our regression models and recommender models failed to do so. While these classification models cannot tell us how much the market changes, it at least provides some information. Therefore, the Logistic Regression provides much more utility than our other models, despite not producing the output we wanted.

5.1 FEATURES AND PARAMETERS

To train each of my models, I used a combination of **Unigrams** and **Bigrams** with **Bag of Words** and **TF-IDF**. For Logistic Regression and Ridge Regression, we see that bigrams performed better than unigrams, and that TF-IDF did slightly better than bag of words. For Linear and Naïve Bayes, we see the opposite.

As for the parameters of the models, I created a pipeline to train Ridge and Logistic with different regularization parameters. I saw that Ridge favored higher regularization parameters, while Logistic favored a range of different regularization parameters.

5.2 FAILURES

Looking at our results, we see that our model obviously failed, as well as other regression models. However, the greatest failure would probably be the recommender model because it performed poorly and had a extremely slow runtime. This model predicts the price change by finding the dot product of similarity to other articles and their price changes. It would do this by looping though training data and finding its cosine similarity. This resulted in each prediction of a single article to take about 5.3 seconds on my relatively powerful personal computer. Running the model on 1,000 test articles took about 1.5 hours. To add that on top of its poor performance, it became unreasonable to further optimize or test this model. I will acknowledge that this model may actually be the most accurate and give us the exact output we desire; however, the further optimization and testing was not performed.

5.3 FINAL THOUGHTS

The poor performance on all our models is most likely due to the naïve implementation of NLP, as well as our irrelevant datapoints.

As seen in the other implementations of stock prediction, researchers use NLP to perform sentiment analysis and correlate that with stock performance. Our model is much more simplistic because we tried to correlate the articles directly to stock market changes. In our model, two articles can say totally opposite things, but have the highest similarity. For example, if one article says that Amazon is bad, while another says that it is good, our model may do very little to differentiate between them.

The dataset also contains many articles on celebrity gossip and other irrelevant news that may have skewed our model.

Finally, our models may perform better on specific stocks, rather than a S&P 500 stock index. For example, we could have run our models to predict the growth of Google stock specifically. This way, we can add weights to an article if it mentions Google or a Google product. Although I cannot verify this would be effective, it seems to have good potential.

Overall, this project was an interesting experiment in using news headlines for this purpose. Even when our dataset was suboptimal, we achieved some level of success.

6 REFERENCES

[1] Nugent, Cam. (2018). S&P 500 stock data, Version 4. Retrieved December 5, 2020 from https://www.kaggle.com/camnugent/sandp500.

[2] Sun, J. (2016, August). Daily News for Stock Market Prediction, Version 1. Retrieved December 5, 2020 from https://www.kaggle.com/aaron7sun/stocknews.

[3] Jeet.J. (2018). US Financial News Articles. Retrieved December 5, 2020 from https://www.kaggle.com/jeet2016/us-financial-news-articles

[4] Misra, Rishabh. (2018). News Category Dataset. 10.13140/RG.2.2.20331.18729.

[5] Joshi, Kalyani & N, Bharathi & Rao, Jyothi. (2016). Stock Trend Prediction Using News Sentiment Analysis. International Journal of Computer Science and Information Technology. 8. 67-76. 10.5121/ijcsit.2016.8306.

[5] Falinouss, P. (2007). Stock trend prediction using news articles : a text mining approach. Retrieved December 5, 2020 from http://www.diva-portal.org/smash/get/diva2:1019373/FULLTEXT01.pdf.