ORIE 4741 Final Report

Ziming Zeng (zz494) & Tiansheng Liu (tl752)

1 . Why we do this project

The most popular hypothesis about financial market is the efficient market hypothesis, which states that the current price of an asset has reflected all of the available information. So, under this assumption, what drives the price? The answer is news. In other words, price of an asset reacts to new information. Therefore, analyzing news is essentially the core of modern finance. There are 2 approaches to analyze the news – monitoring news 24 hours a day, or design an algorithm to analyze the sentiment of news automatically. The problem with the first approach is that market nowadays react to news so rapidly that human cannot possibly catch up with its speed. For example, the VIX index skyrocketed in a few seconds after Trump's tweet about "Fire and fury". That leaves us with only the second approach and we believe this is the future of finance.

2 . Data Sources

2.1 News data: Thompson Reuters historical real-time news

The reason we choose Thompson Reuters is that they are famous for the speed of reporting news and the industry is actually using Thompson Reuters' service. Comparing to Thompson Reuters, other sources like Bloomberg, Wall Street Journal and Financial Times are too slow. For our project, we are interested in implementing trading strategies at a relatively high frequency. Therefore, speed means everything for us.

2.2 Intraday millisecond stock price data: WRDS TAQ Database

NYSE Trade and Quote (TAQ) is a database created by NYSE. It contains intraday transactions (trades and quotes) data for all securities listed on New York Stock Exchange and American Stock Exchange. As the transactions data are recorded under millisecond level, datasets in TAQ are extremely big (20 million observations for one day's data). Therefore, to handle TAQ datasets efficiently, we used SAS and a particular algorithm called Dow Loops to access TAQ data remotely on WRDS server.

2.3 Daily index data & stock price data: WRDS CRSP Database

The Center for Research in Securities Price (CRSP) is a database contains end-of-day and month-end prices on primary listings for NYSE, NASDAQ, and Arca exchanges, along with basic market indices. We use CRSP mainly for estimation of the beta in CAPM model.

2.4 Fama French Database

Fama French Database contains risk free rate and market risk premium data. We use this database also mainly for the estimation of beta.

3 . Answers to possible questions

(1) Is your data corrupted: No. This is the advantage of using high-quality commercial news data from Thompson Reuters, TAQ and CRSP.

(2) How many features do you use: We use a dictionary (we will explain in great details below) that has 2337 negative words and 353 positive words. Therefore, the total number of features we use is 2690.

(3) How do you solve overfitting: The weights we design will handle overfitting. Most of trivial words will have weights close to 0 assigned to them. Actually, in our problem setting, we don't care about overfitting. We care about whether this strategy makes money. We will discuss this further in subsequent sections.

(4) How will you test the effectiveness of your model: Cross validation. Specifically, we do not use R squares as our criterion, we use portfolio returns. Again, we only care about whether this strategy makes money.

(5) Why not include more sources of data: Including other sources may cause duplicities. For example: Thompson Reuters reported fire and fury in 12:00 am and WSJ reported it again at 12:05 am. In our model, we need to tag return for the news, and if we don't have consistent time for the same news we will be in trouble.

(6) Why do you think market reacts to sentiment of news: You will be convinced after seeing our results.

(7) How do you trade assets that are mentioned in the news: We designed an algorithm on our own to extract company names from news! And we only trade liquid stocks that are mentioned in the news to avoid market impact.

(8) Techniques from classes that you use: We use linear regression, n-fold cross validation, Lasso regression and Ridge regression.

(9) **Sentiment analysis has been studied for 10 years, what is the unique point about your project:** Although sentiment analysis has been studied for 10 years, no study has been published about its implementation in trading at a frequency of 1 min to 5 min. This totally makes sense. No one will publish a successful strategy.

4 . Our Idea

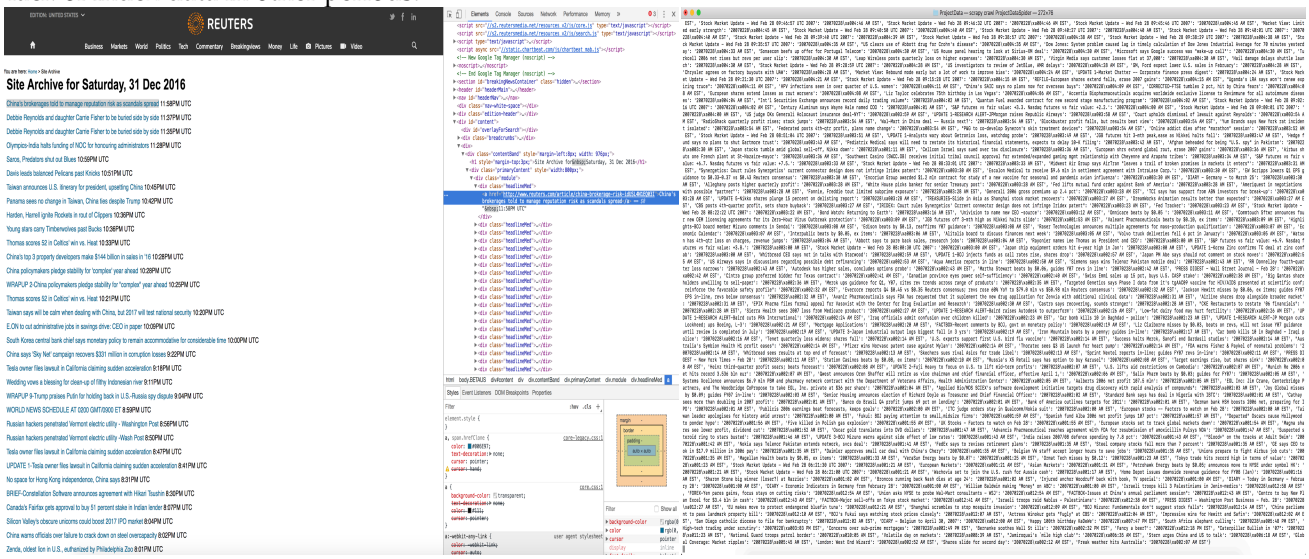
Our idea can be divided into the following 2 steps. Details about each step can be found in following sections.

1. Given a news, analyze the sentiment of the news
2. Trade the company that is mentioned in the news according to the sentiment of the news and hedge the trade using S&P 500 futures.

5 . Step 1: Analyze the sentiment of news

5.1 Web Scrapping

We designed a web scraper to scratch the Thompson Reuters historical real-time news headline from 2007 – 2017. By locating XPath of the html code, we used Python and Scrapy to download news headlines from Thompson Reuters and stored them in the JSON format. This gave us 1GB size data. However, we only used data from 2011 – 2014 in the end due to the lack of Index data in other periods.



5.2 News Data Cleaning

The raw news data were too messy for Natural Language Processing. Therefore, we did the following steps to clean the data:

- Deleted stop words from news. Stop words are some most common words in a language that does not convey any meaning. A few examples of stop words from Python NLTK packages are: 'his', 'the', 'an'.
- Deleted numbers from news.
- Delete all punctuations.
- Converted all words into lower case
- Deleted words that are not English. Some of the news in Thompson Reuters were in Japanese and Chinese. And we deleted them using utf-8 encode and ASCII decode.
- Only kept news data from 9:30 AM EST to 4:00 PM EST. **The effects of news reported outside this time horizon will be reflected in the opening auction.** Therefore, we only consider the news happening during regular trading hours.

5.3 Build Features

To build our features, we used a famous financial dictionary created by Loughran and McDonald. The reason we use this dictionary is that the sentiments of words in financial news can be quite different from our usual language. For example, "liability" may have a negative sentiment in our normal conversation, but in finance it is just a concept in financial report with no particular sentiment. Loughran and McDonald created this list of positive and negative words using 10-K and 10-Q reports from SEC and it has been widely used by the industry. This gives us 2337 negative words and 353 positive words. Therefore, the total number of features we use is 2690.

5.4 Our Model

Given a sentence of news headlines, we tokenized the headlines into single word and matched each word in the sentence with the Loughran and McDonald dictionary.

We used a method put forward by Jegadeesh and Wu (2013) to generate weights of words. The intuition behind this is to assign weights for each word according to the market reaction to each headline that contains those words.

Mathematically, the above idea can be expressed as follows:

$$Sentiment_i = \sum_{j=1}^{J} (w_j F_{i,j}) \frac{1}{a_i}$$

where J is the total number of positive and negative words in our lexicon, w_j is the weight of word j, $F_{i,j}$ is the number of occurrences of word j in headline i, and a_i is the total number of words in headline i. The only unknown is the above equation is w_j and we can estimate w_j using the following regression model:

$$r_i = a + b \sum_{j=1}^{J} (w_j F_{i,j}) \frac{1}{a_i} + \varepsilon_i$$

Since b is a constant, combining b and w_j together and write it as B_j gives us the regression model:

$$r_i = a + \sum_{j=1}^{J} (B_j F_{i,j}) \frac{1}{a_i} + \varepsilon_i$$

where r_i is the abnormal return of stock market caused by headline i. After getting $B_j^{\sim}$, we can get $w_j^{\sim}$ using standardization.

Why do we choose regression instead of classification? We have 10 years news data. It is impossible for us to tag them manually as 'positive' or 'negative'. Therefore, we have to tag them with returns. But if we only tag them as positive or negative according to returns, we fail to capture the magnitude of returns. Therefore, we choose to use multivariate regression to capture this magnitude.

5.5 Identify company names in news

The importance of trading the correct companies mentioned in the news is evident. And we design an algorithm to identify company names in news.

5.5.1 Data Cleaning

For company names data, we downloaded a list of all tradable companies in NYSE, NASDAQ and AMEX. And we did the following data cleaning procedures:

- Convert all words into lower case
- Delete punctuations
- Delete words like 'inc', 'corporation', 'corp'. This is particularly important because companies usually will be mentioned in its full names. For example, Apple Inc. may be called Apple in the news.
- Delete some companies that uses common words as its name. Specifically, we delete companies with following tickers: INCR, BSTI, QTWO, TIME, NWS, LN, CA, DNOW, TGT, GPI, BOX, USCR. BSTI, for example, is the ticker for Best Inc. After deleting 'inc', we are only left with 'best', which appears everywhere in the news.
- Delete companies that have market capital below 2 billion dollars. This is also important for high frequency trading. For small companies, buying and selling at big amounts will affect the prices greatly. Therefore, we only trade liquid big companies in order to avoid market impact.

5.5.2 Algorithm

Our algorithm can be easily illustrated using an example. Suppose we want to identify the company in following news: "Morgan Stanley is doing a great job this year." We test whether each word of the company names is in the news. If we are testing whether this news is talking about Goldman Sachs, we can see neither of "goldman" or "sachs" appears in the news. Therefore, the match score is 0/2 = 0, where 2 is the length of company name. If we are testing whether this news is talking about Morgan Freeman (Let's just pretend Morgan Freeman really started a company and named it using his own name). Then we can see that "morgan" appears in news but "freeman" does not. So the match score is 0.5. Now let's test Morgan Stanley, both "morgan" and "stanley" appears in the news so the match scores will be 1.

Our algorithm only keeps observations with the match score greater than 0.8 in order to make sure that we are trading the right companies. By random sampling from our match result and checking manually, we found that this algorithm gave us an excellent result. The overall accuracy was above 90%. For technical details, please refer to the python program we have uploaded - "Get_company_names_from_news.py".

5.6 Abnormal Return Computation

5.6.1 Beta Estimation

The computation of abnormal return is based on the Capital Asset Pricing Model (CAPM), which is the foundation of asset pricing. CAPM states that:

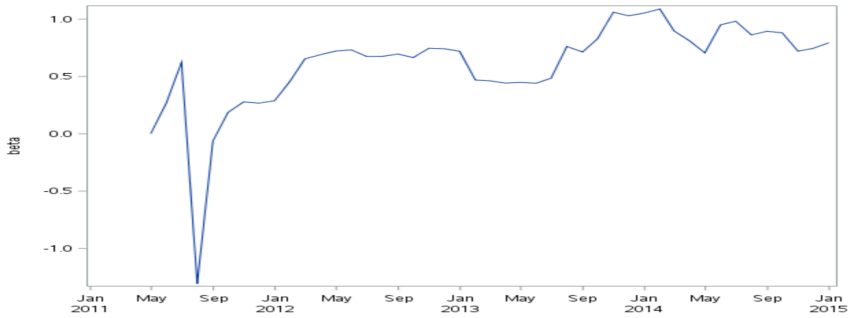
$$E[R_i] - R_f = \beta(E[R_m] - R_f)$$

where R_i is the return of the stock i, R_f is the risk-free rate and R_m is the return of the overall market. In plain language, it states that if you buy 1 share of stock i and sell β shares of market index, your expected payoff will be 0.

Please note that this is the core of our trading strategy. For example, we want to short 1 share of Apple but afraid of the risk of overall market goes up. We can hedge the trade by longing β shares of market index. Our net expected payoff should be zero according to CAPM model. Therefore, we only care about the relative performance of Apple comparing to the market. If a negative news comes out about Apple, which makes Apple perform worse than the overall market, we are in business.

This is also the reason why we don't care about the accuracy of our sentiment analysis regression. We are doing a hedge trade here. Even if our prediction is wrong, we will not lose a lot because it is a hedged trade and the expected payoff is 0 (if we do not consider transaction costs). If we know nothing and trade based on the result of a flip of coin, we have a 50% probability of being right. This strategy works like a casino. As long as we have a slightly higher probability of winning, even if our sentiment scores only give us 51% probability of being right, we will win in the long run.

One problem we had was that the coefficients β could change over time. For example, the graph bellow shows the beta of Apple over time. We can see that the β of Apple was around -1.2 on August 2012 but it was around 0.6 on Sep 2014.



Therefore, we used a 24-months rolling regression approach to estimate beta of stocks at different dates. The idea of rolling regression is simple. Suppose we are considering the beta of Apple on 2014.4.7. Then we use the return data from 2012.4.7 to 2014.4.7 to estimate beta on 2014.4.7 for Apple. Now suppose we are considering the beta of Microsoft on 2012.9.8, Then we use the return data from 2010.9.8 to 2012.9.8 to estimate beta on 2012.9.8 for Microsoft.

Return data were from CRSP and the risk-free data were from Fama French Database. For technical details, please refer to our attached SAS program "Rolling_regression.sas".

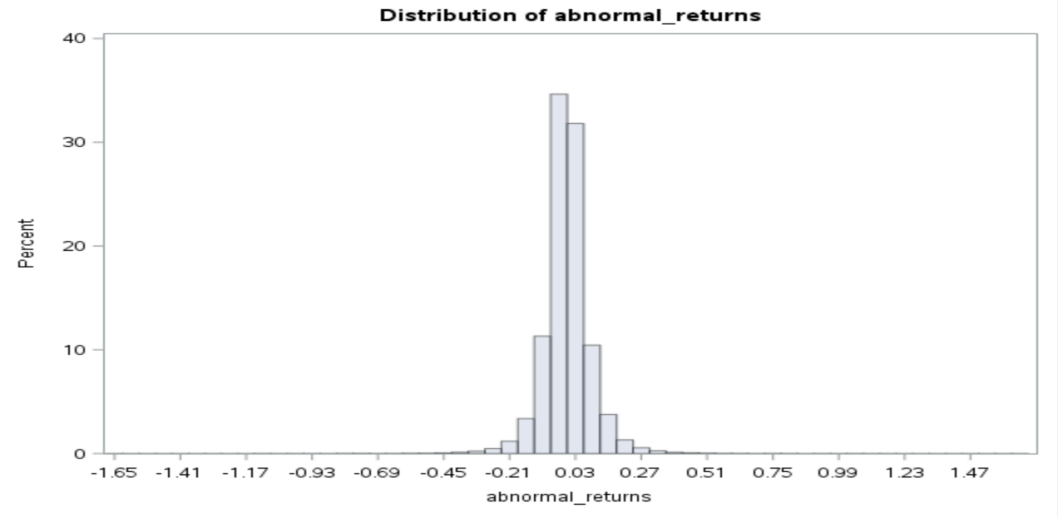
5.6.2 Stock Returns and Abnormal Stock Returns

After getting beta from rolling regression, we used TAQ database in order to get intraday stock returns data. And we used a special algorithm called Dow Loop in SAS to remotely handle TAQ data efficiently. For intraday index return data, we use CRSP Intraday Total Market Index as our data.

We tested 2 frequencies here: 5-minute frequency and 1-minute frequency. And our abnormal returns are calculated using the following formulas:

$$r_{abnormal,t,t+5} = r_{stock,t,t+5} - \beta r_{index,t,t+5}$$
$$r_{abnormal,t,t+1} = r_{stock,t,t+1} - \beta r_{index,t,t+1}$$

Given the low interest rate after the financial crisis in US, we can ignore the risk-free rate. Therefore, according to CAPM, the abnormal returns should have an expected value of 0. From the data, we can see that this is roughly true.



6 . Step 2: Simulate Trading

After getting abnormal returns from section 5.6, we can finally start to run regression to assign sentiment scores based on our model described in section 5.4.

6.1 Sparse Matrix Construction

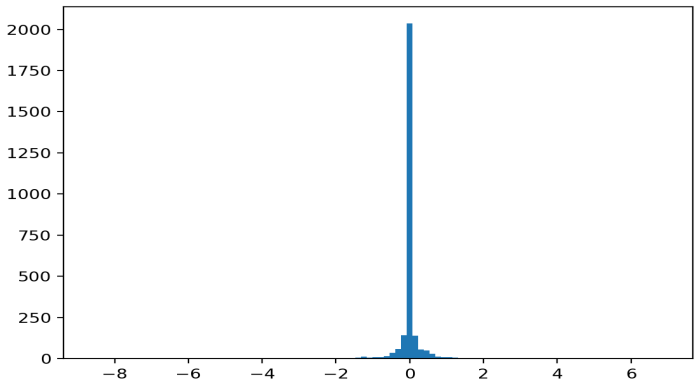
In our current setting, we have 2690 features and 4 years' data. We only consider this period because lack of TAQ data from 2015 to 2017 and the lack of data of CRSP intraday index data from 2007 to 2010. It seems that using a sparse matrix is not necessary. However, if you want to extend this model and build a lexicon of your own, you may have 1 million features and you cannot possibly store the data as a matrix because of its size. In this case, sparse matrix is definitely going to be helpful.

We used the Consolidated Sparse Row(CSR) format to store our data. The CSR sparse matrix stores a sparse $m \times n$ matrix M using 3 arrays X, Y, and Z. It will greatly reduce the amount of space needed to store the data and can be passed to the Sklearn package in Python as inputs. We manually constructed the 3 vectors of CSR matrix and used it directly in our regression. For technical details, please refer to "Sparse.py".

6.2 Linear Regression

6.2.1 Generate Weights and Document Scores

We put 80% our datasets into training set 20% of our data into test set. After running the regression described in section 5.4. We get the following distribution of result of our weights. Remember that we have has 2337 negative words and 353 positive words, we should expect the most of weights to be negative. And that is true according to the graph. The majority of the words have slightly negative weights.



Using 3-fold cross validation, the average of R-square of our regression is disappointing, we have, on average, a 0.017 Adjusted R-square on the training set and a -0.008 R-square on the test set. However, again, this doesn't matter at all. We care about the profit but not the accuracy of our model.

After getting the weights for single words, we generate the news scores for each news using the following formula in section 5.4:

$$Document\ Scores_i = \sum_{j=1}^{J} (w_j F_{i,j}) \frac{1}{a_i}$$

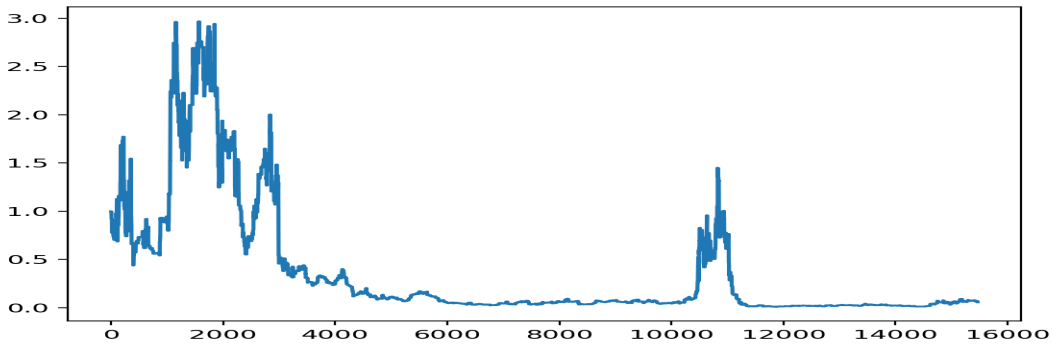
Then, using the document scores in **training data**, we computed percentiles for positive and negative document scores respectively. They will be used as trading signals.

6.2.2 Simulate Trading

Our trading strategy, as I have mentioned in section 5.6.1, is simply a hedge trade.
Given a news and its sentiment score,

- If sentiment score > 0: we long 1 share the company that is mentioned in the news and hedge the trade by shorting β shares of market index.
- If sentiment score < 0: we short 1 share the company that is mentioned in the news and hedge the trade by longing β shares of market index.

And we get the following result on the test set:



The x-axis is the number of trades and the Y-axis is the portfolio value, which starts at 1. We can see from the graph that we are basically blown up and end up with a portfolio value close to 0.

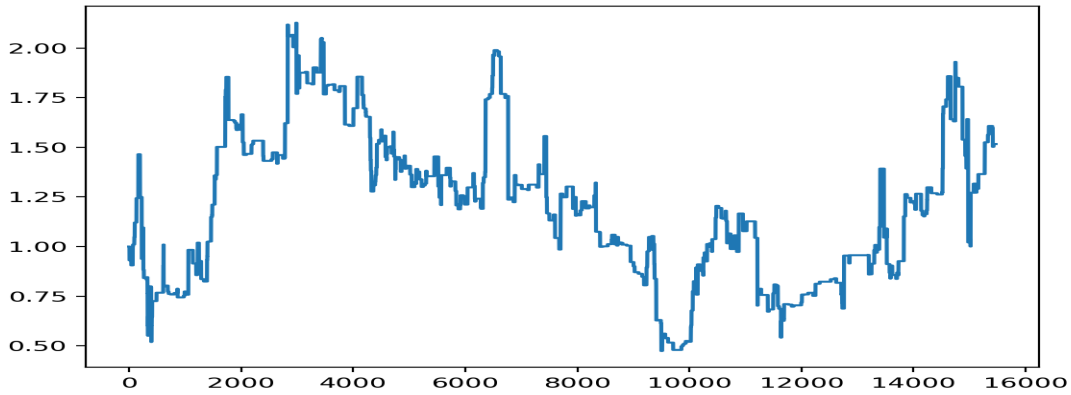
The reason we are blown up is simple. We trade too many weak sentiment news and make too many mistakes. So, can we be smarter?

Remember that at the end of section 5.7.2.1, we computed percentiles for positive and negative document scores respectively. Denote the "buy signal" to be the 80 percentile of positive document scores and "sell signal" to be the 20 percentile of negative document scores (note that the most negative scores have lower percentile).

Given a news and its sentiment score,

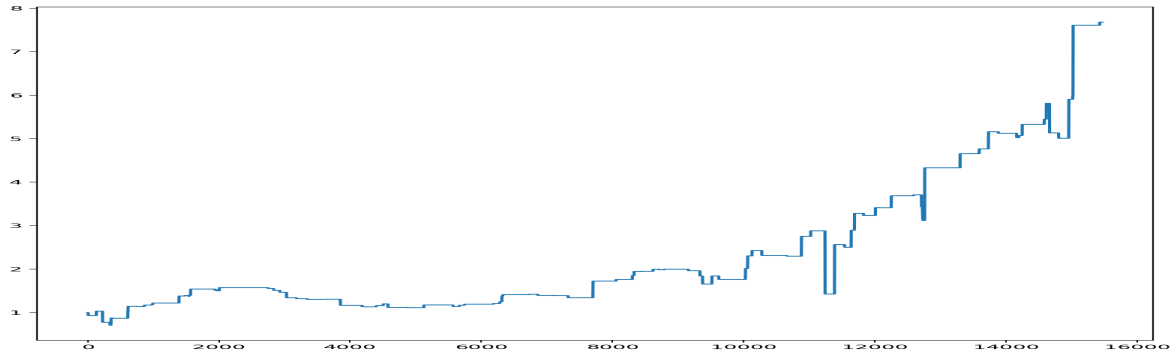
- If sentiment score >buy signal: we long 1 share the company that is mentioned in the news and hedge the trade by shorting β shares of market index.
- If sentiment score < sell signal: we short 1 share the company that is mentioned in the news and hedge the trade by longing β shares of market index.

We can get the following result on the test set:



Our portfolio values grow by 51% in less than 8 months. The only thing different here is that we only trade those news that have stronger sentiments (big in absolute values).

Now here comes the interesting part, we notice that the loss from the above trading strategy is mainly caused by a few false trades. If we let "buy signal" = 90 percentile "sell signal" =10 percentile, which means stronger constraint, we will get the following on the test set:



768% growth in 8 months! I guess this is the reason why high-frequency trading is so attractive. From the above result, we can see that:

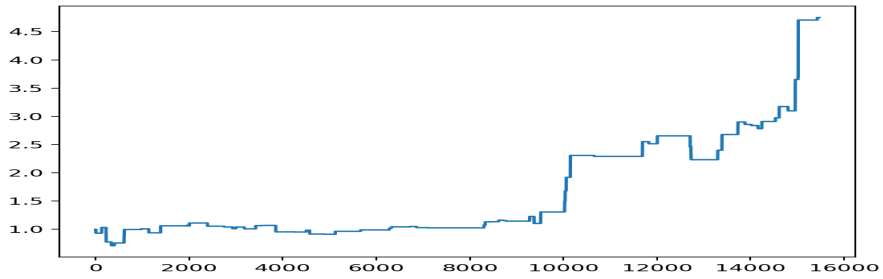
- We have a lot more "flat" region here, this is because the majorities of news have weak sentiments and we do not trade those news.
- The result becomes more stable. This is also because we only trade those news we are sure about and there are fewer mistakes.
- A big proportion of our profit is generated by a few "jumps". This totally makes sense. News that causes big moves in the market usually has a high sentiment score (in absolute value sense). And our strategy is designed to trade those news with strong sentimente.

By cross validating on different training and test set, we are unable to always replicate this 768% return. But a result at **500% level is ALWAYS achievable.**

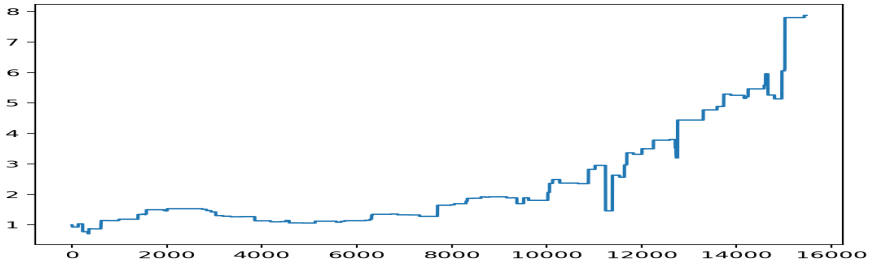
6.3 Lasso Regression and Ridge Regression

The reason we want to test Lasso and Ridge is that they tend to penalize the large coefficients. As you have seen in our previous discussion, the majority of our losses are caused by a few large false trades. If we have a more stable prediction, the probability that we make false decision should be lower.

By cross validating different alphas based on the profits, we find that Lasso with alpha = 0.000001 produces the best result under the case of "buy signal" = 90 percentile, and "sell signal" = 10 percentile:



By cross validating different alphas based on the profits, we find that Ridge with alpha = 0.001 produces the best result under the case of "buy signal" = 90 percentile, and "sell signal" = 10 percentile:



Indeed, Ridge improves our result to 787%, which is what we have expected. However, the Lasso does no perform well. We think this is mainly because Lasso tends to produce sparse solutions. Sparsity means the sentiment of the news is based on fewer words because a lot of words have weights 0 assigned to them under Lasso. Therefore, it will increase the chances of making mistakes.

7 . Feasibility of this trading strategy

Of course, we are not likely to get the 500% result in reality. But given the high magnitude of our return, we are still expected to generate a substantial amount of profits in reality. We will address some of your concerns first and discuss some of the drawbacks of our model.

1. Bid-ask spreads
   We are using the ACTUAL transaction prices from TAQ database. We believe this is a good estimation of our cost and returns. If you are concerned about the bid-ask spread, simply trade those companies with both high volumes and high market capital. For those companies, the bid-ask spread impact could be lower.
2. Transaction costs
   Note that in our strategy, we get a great and stable result in the case where "buy signal" = 90 percentile and "sell signal" = 10 percentile. We do not need to trade a lot because of the strong constraint. Therefore, our transaction costs will not be too high in this case. But for the case where "buy signal" = 90 percentile and "sell signal" = 100 percentile case, yes transaction costs will be a big concern.
3. Market impact
   Notice that we only trades companies with market capital more than 2 billion dollars. Assume we are poor (which is definitely true for me) and we only have a small amount of capital like 100 thousand dollars. Our trades will not have a big impact on the market. But if you are working as an institutional investor, then yes you should consider about the market impact.
4. Competitions
   Yes, we will face competitions from Citadel and Renaissance Tech. But the interesting thing is that, if we adjust our holding period from 5 minutes to 1 minute, our returns become worse. Why? Because we need to wait for

human traders to bid up the price. Again, we are trading large companies. In US stock market, the majority of the investors are still traditional investors. Even a small amount of quant investors like Citadel can place orders faster than us, they cannot lift the price a lot for big companies. Even if they can, they will usually not do it because institutional investors won't want to put too many risks on a single trade.

Despite the great result, there are also some drawbacks of our model:

1. We implicitly assume that we can trade fractions of shares.
2. The volatility is high. We are assuming that we put all of our money into a single hedge trade, wait for 5 minutes, and get out. Evidently this will cause a lot of volatilities.
3. We do not consider the case of margin call. In the case of shorting, we might be forced to close our positions.

8 . Future Works

As we have discussed in the mid-term report, we want to explore the following topics:

- Can we generate lexicon of our own?
- Can we improve our result by using bi gram?
- Can we use Google N-Gram to speed up the process of generating sentiment scores?
- Can we optimize our portfolio by introducing more assets like bonds into consideration?
- Can we design some different trading strategies like volatility trading strategies?

9 . References

[1] Jegadeesh, Narasimhan, and Di Wu. "Word power: A new approach for content analysis." *Journal of Financial Economics* 110.3 (2013): 712-729.

[2] Treynor, Jack, and Kay Mazuy. "Can mutual funds outguess the market." *Harvard business review* 44.4 (1966): 131-136.