Cornell University

ORIE 4740 Group Project

Stock Index Prediction over News Headlines

Supervisor:
Prof. Chen Yudong

Author:
Bowen Gao bg453
Kaifeng Gu kg484
Duo Sun ds2324

August 13, 2018

Contents

1 Introduction

Stock market prediction is extremely important to entire finance industry. Stock price is determined by the behavior of human investors, and the investors determine stock prices by using publicly available information to predict how the market will react. Thus financial news articles can play a large role in influencing the movement of a stock. Previous research has suggested that there is a relationship between news articles and stock price movement, as there is some lag between when the news is released and when the market has moved to reflect this information.

The goal of this project is to predict the stock market movement based on the daily news headlines. On each trading day, we get top news headlines and use them to predict the day's market movement direction and simulate the confidence interval for the Dow Jones Industrial Average(DJIA) Index. The dataset we use is "Daily News for Stock Market Prediction" from Kaggle, which contains 8 years(2008/8/8 - 2016/7/1) daily news headlines and daily data of DJIA Index. There are 1989 days. For each day, there are 25 top news headlines(string type), 5 market information(Open, High, Low, Close, Volume, Adjusted Close), and a market movement direction label(up or same as 1, down as 0).

In the data warehousing, we have two ways to construct or vocabulary: We first extract the relevant topic to stock market and then useful word types from each headline, after that we build our vocabulary based on number and frequency of each term. A second way is to category each day's headlines into four sentiments, and give a overall score.

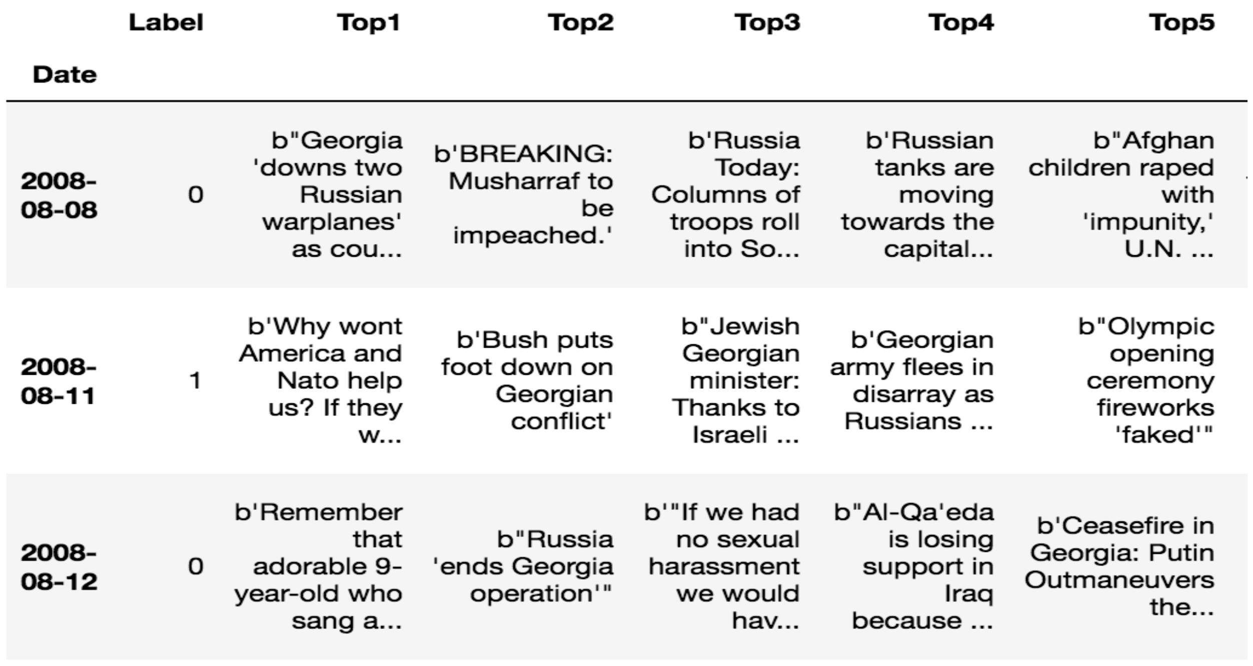
Before building our models, we observe possible clustering in order to find the specific pattern in the dataset, where SVD and TSNE are applied to reduce the dimension of the dataset.

We finally build our models, starting from benchmark models, bullish market or reversing market throughout the whole timespan. Apart from that, we tried different classification models, like KNN, Logistic Regression, Rocchio Classification, Naive Bayes Classification, Random Forest and Support Vector Machine to see the estimated test error, though the overall results are not promising and among all the models, KNN and Random forest seems more reliable than the benchmark models.

2 Data Preparation

2.1 Data Import

Use pandas to import dataset, the dataset is a 1989× 26 matrix, contains Top 25 headlines together with DJIA30 movement (1 is up, 0 is down) from 2008-08-08 to 2016-07-01. From column 2 to column 26, each entry is a complete sentence of news. Figure 1 is a screenshot of the imported original dataset.



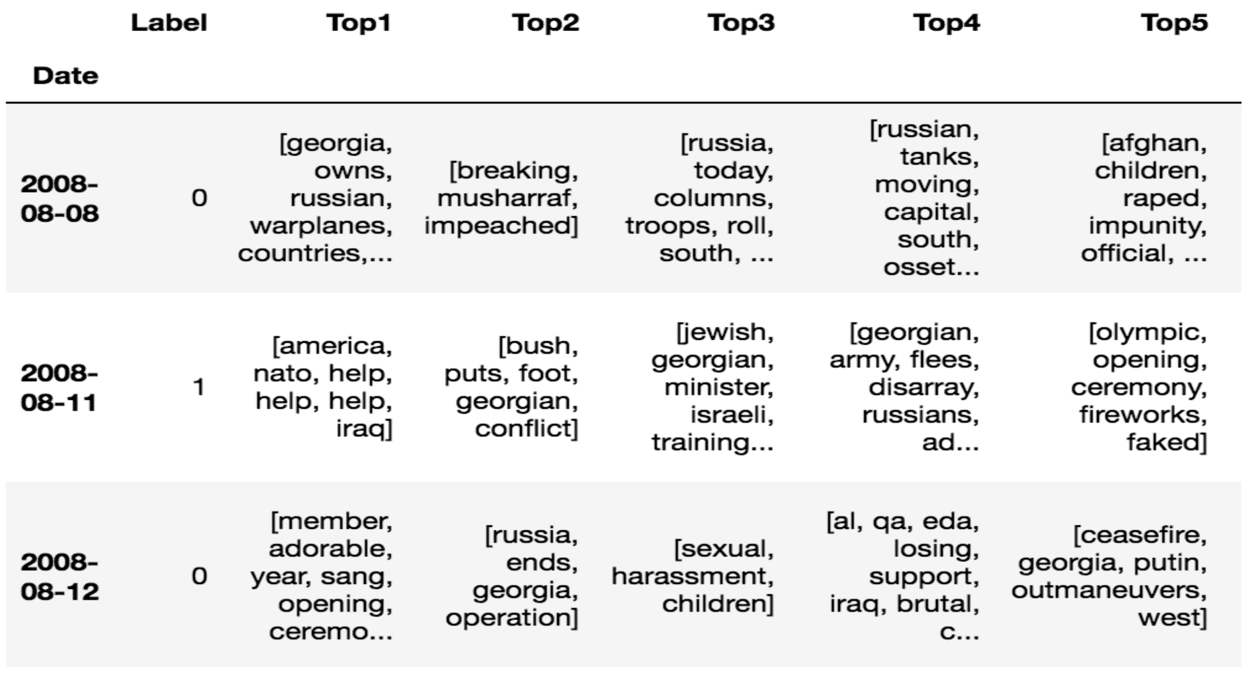
	Label	Top1	Top2	Top3	Top4	Top5
Date						
2008-08-08	0	b"Georgia 'downs two Russian warplanes' as cou...	b'BREAKING: Musharraf to be impeached.'	b'Russia Today: Columns of troops roll into So...	b'Russian tanks are moving towards the capital...	b"Afghan children raped with 'impunity,' U.N. ...
2008-08-11	1	b'Why wont America and Nato help us? If they w...	b'Bush puts foot down on Georgian conflict'	b"Jewish Georgian minister: Thanks to Israeli ...	b'Georgian army flees in disarray as Russians ...	b"Olympic opening ceremony fireworks 'faked'"
2008-08-12	0	b'Remember that adorable 9-year-old who sang a...	b"Russia 'ends Georgia operation'"	b'"If we had no sexual harassment we would hav...	b"Al-Qa'eda is losing support in Iraq because ...	b'Ceasefire in Georgia: Putin Outmaneuvers the...

Figure 1: data frame example

2.2 Tokenization

For every sentence, we first transfer the next-line to a blank, so that we can use "Stop word" [1]list to remove some useless words, for example: a, are, get, here, on, etc.

Then we can apply **Tokenizer** in the **nltk** package to transfer the string (news sentence) to a list in each entry of the dataset one column by one column, with elimination of the punctuation. Figure 2 is a screenshot of the tokenized dataset.



	Label	Top1	Top2	Top3	Top4	Top5
Date						
2008-08-08	0	[georgia, owns, russian, warplanes, countries,...	[breaking, musharraf, impeached]	[russia, today, columns, troops, roll, south, ...	[russian, tanks, moving, capital, south, osset...	[afghan, children, raped, impunity, official, ...
2008-08-11	1	[america, nato, help, help, help, iraq]	[bush, puts, foot, georgian, conflict]	[jewish, georgian, minister, israeli, training...	[georgian, army, flees, disarray, russians, ad...	[olympic, opening, ceremony, fireworks, faked]
2008-08-12	0	[member, adorable, year, sang, opening, ceremo...	[russia, ends, georgia, operation]	[sexual, harassment, children]	[al, qa, eda, losing, support, iraq, brutal, c...	[ceasefire, georgia, putin, outmaneuvers, west]

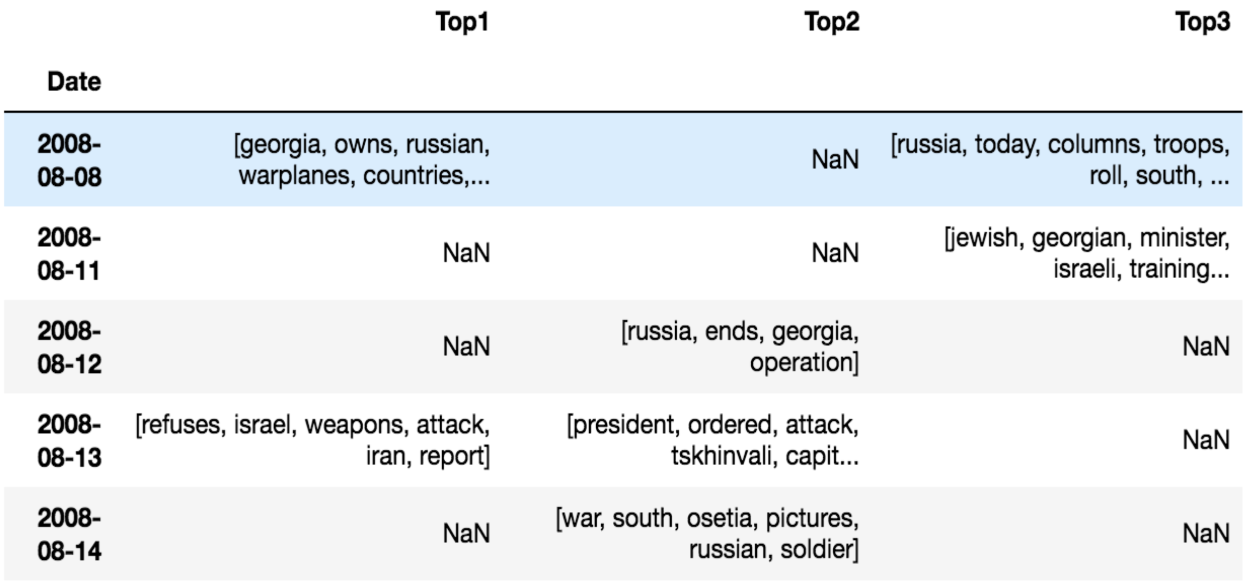
Figure 2: tokenized example

[1]From https://github.com/ahmedbesbes/How-to-mine-newsfeed-data-and-extract-interactive-insights-in-Python

3 Vocabulary Dataset

3.1 Topic filtering

Since we only care about the news relevant to the stock market movements, we can somehow discard the news of less relevant topics such as entertainment and sports. We believe that the most relevant topics are: **Finance, Money, Banking, Business, Economic, Politics, War, Terrorism, Military, Weapon, Internet, and Technology**. Hence, we specify those topics in the empath package[2] to derive a more sparse matrix with "NA" entry representing the irrelevant news. Figure 3 is a screenshot of the topic-filtered dataset.



Date	Top1	Top2	Top3
2008-08-08	[georgia, owns, russian, warplanes, countries,...	NaN	[russia, today, columns, troops, roll, south, ...
2008-08-11	NaN	NaN	[jewish, georgian, minister, israeli, training...
2008-08-12	NaN	[russia, ends, georgia, operation]	NaN
2008-08-13	[refuses, israel, weapons, attack, iran, report]	[president, ordered, attack, tskhinvali, capit...	NaN
2008-08-14	NaN	[war, south, osetia, pictures, russian, soldier]	NaN

Figure 3: filtered topic example

Let us denote this 1989 $\times$ 26 matrix as **Topic Filtering Matrix** and from now on, we will regard this matrix as our dataset.

3.2 Build Vocabulary Method 1 - Frequency Analysis

3.2.1 Training and Test Split

We split the whole topic-filtered dataset into two parts: one with the first 1700 observations and the other with the rest 289 observations. Cross Validation dataset is chosen from the 1700 Training dataset, we create 10 dynamic pair of CV-Training dataset and CV-Test dataset to estimate the test error: for each pair of CV dataset, we have 700 dataset for CV-Training and 100 dataset for CV-Test. The reason to create CV dataset is to estimate the test error of the models which will be explained in Chapter 4. The reason why to create a dynamic CV dataset is to minimize the time effect of the news, for example, stock market may have internalized the effect of finance news even before they are officially published; on the other hand, stock market may react upon on the technology news and political news only after they are officially published.

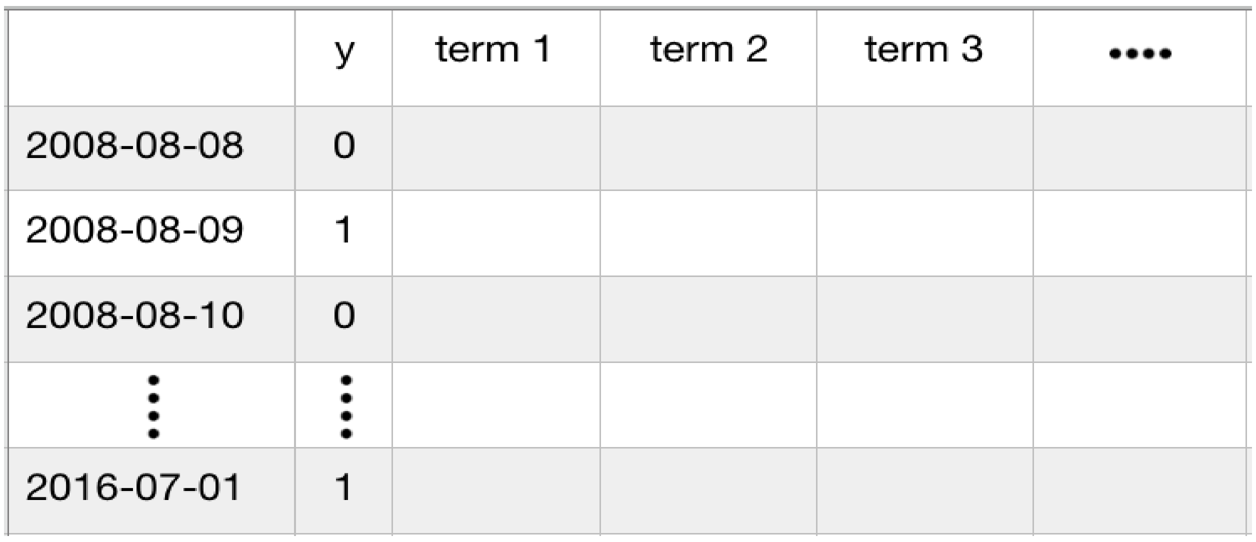
3.2.2 Word-type Selection Set

With the help of function **extract_candidate_chunks** and**buildVocabulary** [3] online, we can extract the meaningful word combination and terminology, for example: Australian Government. With the help of the function $\textbf{pos}_{\textbf{tag}}$ in the package **nltk.tag**, we can only extract the words with specific type. Here, we believe **Noun**, **Verbs**, **Adjective** and **Adverb** are more useful word types. Hence, we create a set of vocabulary whose the terms are of relevant word-types, from the previous Training dataset and CV-Training dataset, and of course, the 10 sets of CV-Training dataset are contained in Training dataset.

[2]From https://github.com/Ejhfast/empath-client
[3]http://bdewilde.github.io/blog/2014/09/23/intro-to-automatic-keyphrase-extraction/

3.2.3 Vocab Frequency Table Construction

Now we apply the word-type selection sets in Chapter 3.3 to the split datasets in Chapter 3.2. Here we use a function **TfidfVectorizer** in the **sklearn** package to transfer the topic-filtered dataset into a word frequency dataset. Row are still the days, and the columns are the terms in the Word-type Selection Set defined in Chapter 3.3. The days and terms depend on the training dataset. Figure 4 is a quick view of the frequency table.



	y	term 1	term 2	term 3	••••
2008-08-08	0				
2008-08-09	1				
2008-08-10	0				
⋮	⋮				
2016-07-01	1				

Figure 4: Frequency Table

Remark: the function **TfidfVectorizer** only takes input of a string form. However, the entry in each row of datasets (topic-filtered dataset) contains list whose elements are still lists. So before apply the word-type selection sets to the datasets, we need to convert the lists of lists into the lists of strings.

Here is a brief explanation of the function **TfidfVectorizer**

For each day i in the training dataset/test dataset, we generated a d-dimensional vector $x_i \in R^d$, where each entry of vector x_i referred to a importance score ω of the corresponding term on this day and the term is from the Word-type Selection Set. d is the size of the Word-type Selection Set. ω measures the importance of the term, based on the previous data, and ω is defined as following:

$$tf_{ij} = \frac{\text{appearances of term j on training day i}}{\text{sum of appearances of all terms on training day i}}$$

$$idf_j = \frac{\text{sum of appearances of all terms on all training days}}{\text{appearances of term j on all training days}}$$

$$\omega_{ij} = tf_{ij} \cdot idf_j$$

According to the definition of ω, a term had score zero if it did not appear on the day. A term was more important if it appeared multiple times on a day. However, a term was less important if it appeared often on all days. Hence, we have to reduce the weight of some words that are commonly used but without meaningful information, such as "the" and "a".Thus the vector now is $x_i = (\omega_{i1}, \omega_{i2}, ..., \omega_{id})$.

For each day in the test set, we would generate a d-dimensional vector as well so that we could later use our trained model to predict the new days' market movement label. To generate a new day's vector, we checked whether the day's headlines contained the terms generated from the training set. For each term j(j=1,2,..,d),idf_j was the same as before using the training data, but tf_{tj} was calculated using the new day's information:

$$tf_{tj} = \frac{\text{appearances of term j on test day t}}{\text{sum of appearances of all terms on test day t}}$$

$$\omega_{tj} = tf_{tj} \cdot idf_j$$

Remark: Since our judgment would be only based on our trained model, we would ignore any new terms in the test set to ensure that the test vectors has the same dimensions as the training vectors. For example, $x_t = (\omega_{t1}, \omega_{t2}, ..., \omega_{td})$

Now, we have finished the building vocabulary in the training datasets (Real training sets and CV-Training sets). We also have a training frequency table ready for us to build model on and a generated test frequency table ready for us to test (Real test and CV-Test)

3.3 Build Vocabulary Method 2 - Sentiment Analysis

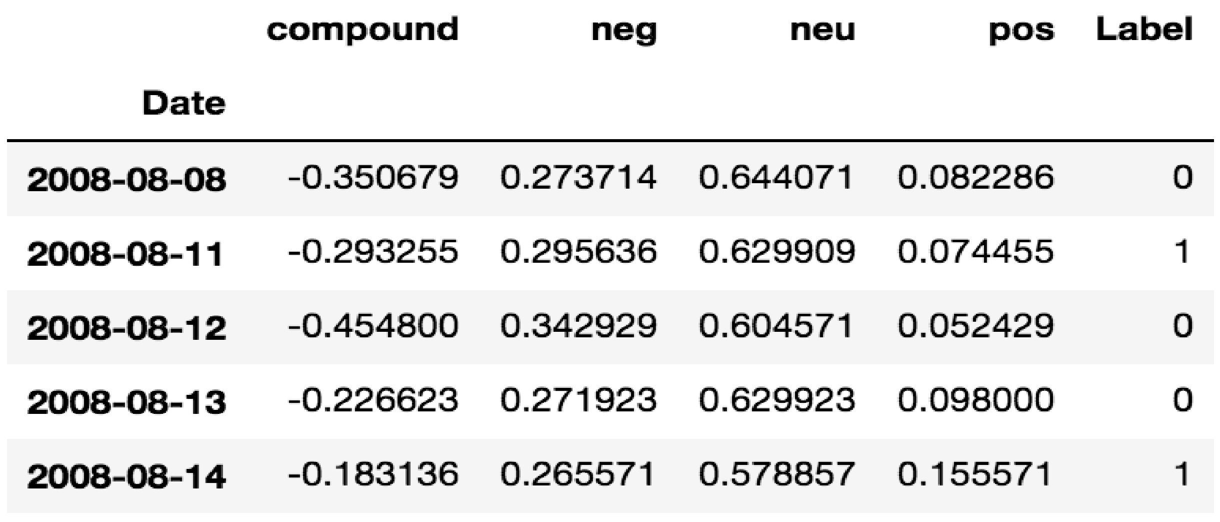
3.3.1 Sentiment Table

Sentiment analysis is simply the process of working out (statistically) whether a piece of text is positive, negative or neutral. **VADER** is a type of sentiment analysis that can give a sentiment rating to each word, where more positive words have higher positive ratings and more negative words have lower negative ratings. VADER analyses a piece of text it checks to see if any of the words in the text are present in the lexicon. Then it produces four sentiment metrics from these word ratings. The first three, positive, neutral and negative, represent the proportion of the text that falls into those categories. The final metric, the compound score, is the sum of all of the lexicon ratings, which have been standardized to range between -1 and 1.

In our project, we used function **SentimentIntensityAnalyzer** in the **nltk.sentiment.vader** package[4]. We will apply the Vader analysis on each text list of each day, and compute the average score of **compound, neg, neu, pos** for each day. Figure 6 is a screenshot of the sentiment table on the Topic Filtering Matrix. Each compound give us the sentiment value on that day.

3.3.2 Training and Test Split

Similar to the Section 3.2.1, we split the whole sentiment table into two parts: one with the first 1700 observations and the other with the rest 289 observations. Cross Validation dataset is chosen from the1700 Training dataset, we create 10 dynamic pair of CV-Training dataset and CV-Testdataset to estimate the test error: for each pair of CV dataset, we have 700 dataset forCV-Training and 100 dataset for CV-Test. Now we have the training dataset and test dataset on sentiment ready to used in Section 5 to build model.



	compound	neg	neu	pos	Label
Date					
2008-08-08	-0.350679	0.273714	0.644071	0.082286	0
2008-08-11	-0.293255	0.295636	0.629909	0.074455	1
2008-08-12	-0.454800	0.342929	0.604571	0.052429	0
2008-08-13	-0.226623	0.271923	0.629923	0.098000	0
2008-08-14	-0.183136	0.265571	0.578857	0.155571	1

Figure 5: Sentiment Table

[4]http://t-redactyl.io/blog/2017/04/using-vader-to-handle-sentiment-analysis-with-social-media-text.html

4 Visualization

In this section, we are going to apply non-supervising machine learning method to group the training data in order to find some embedded similarity in the training dataset which can help us analyze and build model easier. For the frequency training dataset with size 1700 and dimension 1129, We first use **SVD** to decompose it to 1700×20, where these 20 terms can better explain the total 1129 terms. Then we use **TSNE** to further reduce the dataset of this 1700×20 to 1700×2. In the dimension reduction, we can project the 1700 points of dimension 1129 to 1700 points of dimension 2. Then we can plot the dataset and check the possible clustering.

4.1 SVD

Singular-value decomposition (SVD) is a factorization of a real or complex matrix. It is the generalization of the eigen-decomposition of a positive semi-definite normal matrix (for example, a symmetric matrix with positive eigenvalues) to any $m \times n$ matrix via an extension of the polar decomposition.In our project, we use the function **svds** in the package [5] **scipy.sparse.linalg** on the 1700×1129 frequency dataset, The output contains 3 matrices, the $1700 \times p$ **compressed doc matrix**, the $p \times p$ diagonal **importance matrix**, and the $p \times 1129$ **compressed term matrix**. P is the number of dimension that we reduced from 1129. The advantage of this function is that in the diagonal matrix, the diagonal entry is the sorted importance. Figure 6 is the plot that shows how we choose p.

It is clear that singular value (importance) drop to a more flat value when the dimension p hits 20. When p increase to 40, the singular value stays roughly the same. Hence we choose the reduced dimension to be 20.

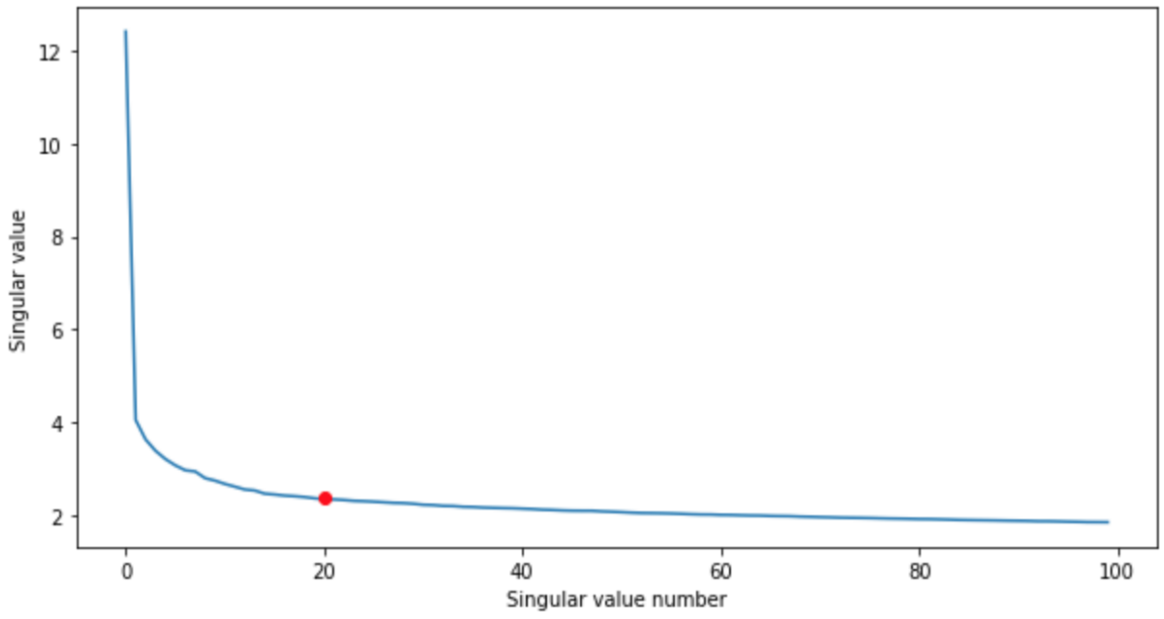


Figure 6: SVD

4.2 TSNE

t-Distributed Stochastic Neighbor Embedding (t-SNE)[6] is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. In our project, we applied function **TSNE** in the **sklearn.manifold** package [7] From 4.1 we reduced the data with 1129 dimension to 20 dimension, and we can use tsne to further reduced the data into dimension 2 (default of that function).

Figure 7 is the plot of the 1700 2-d data, red points represent the days that stock move up, while green points represent the days that stock move down. Figure 8 is the plot of the 1699

[5]https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.svds.html
[6]https://lvdmaaten.github.io/tsne/
[7]http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

2-d data, with the lag 1, that is we assign today's up/down value to yesterday's frequency data, which is reduced to dimension 2 now. It is rather clear that both the red and green dots scatter all the area without specific clustering and we cannot come to any conclusion to the similarity to the training dataset. We has to build model to make the prediction in Section 5.

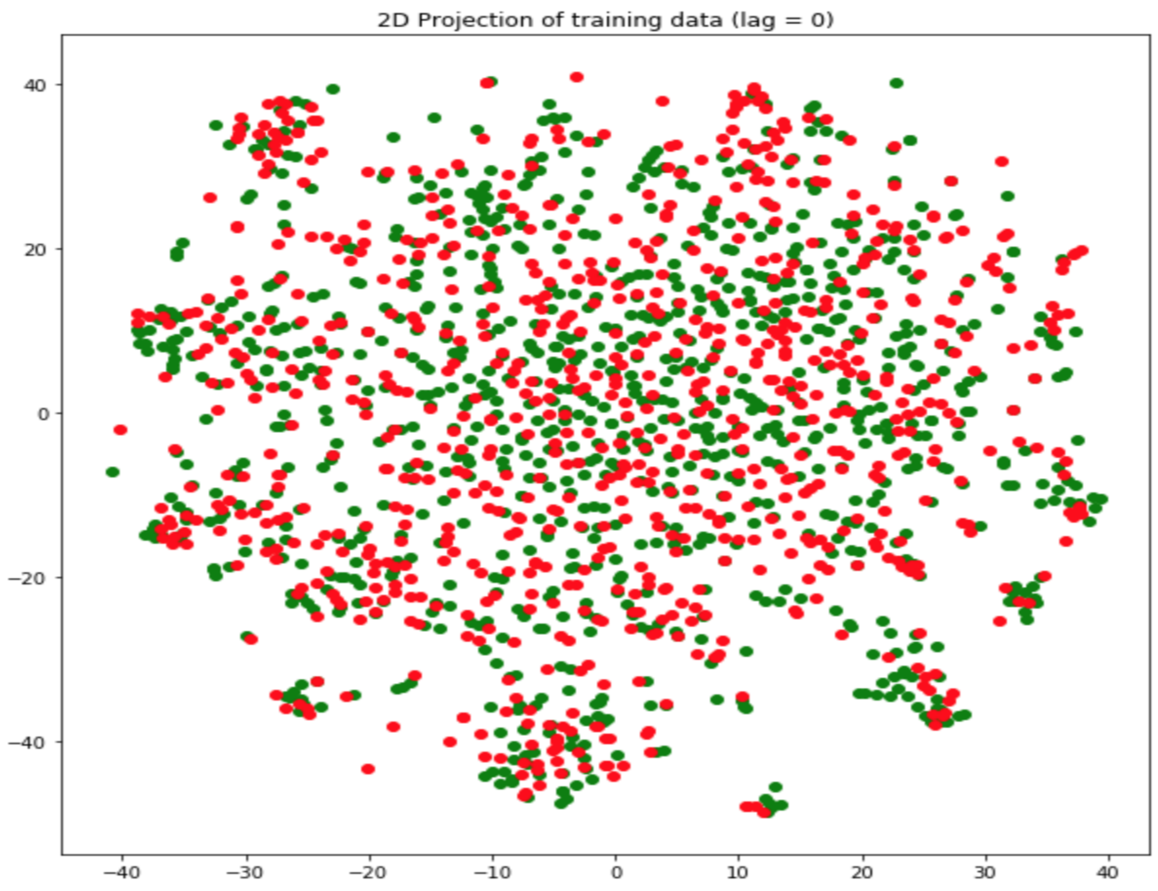


Figure 7: Clustering with No Lag

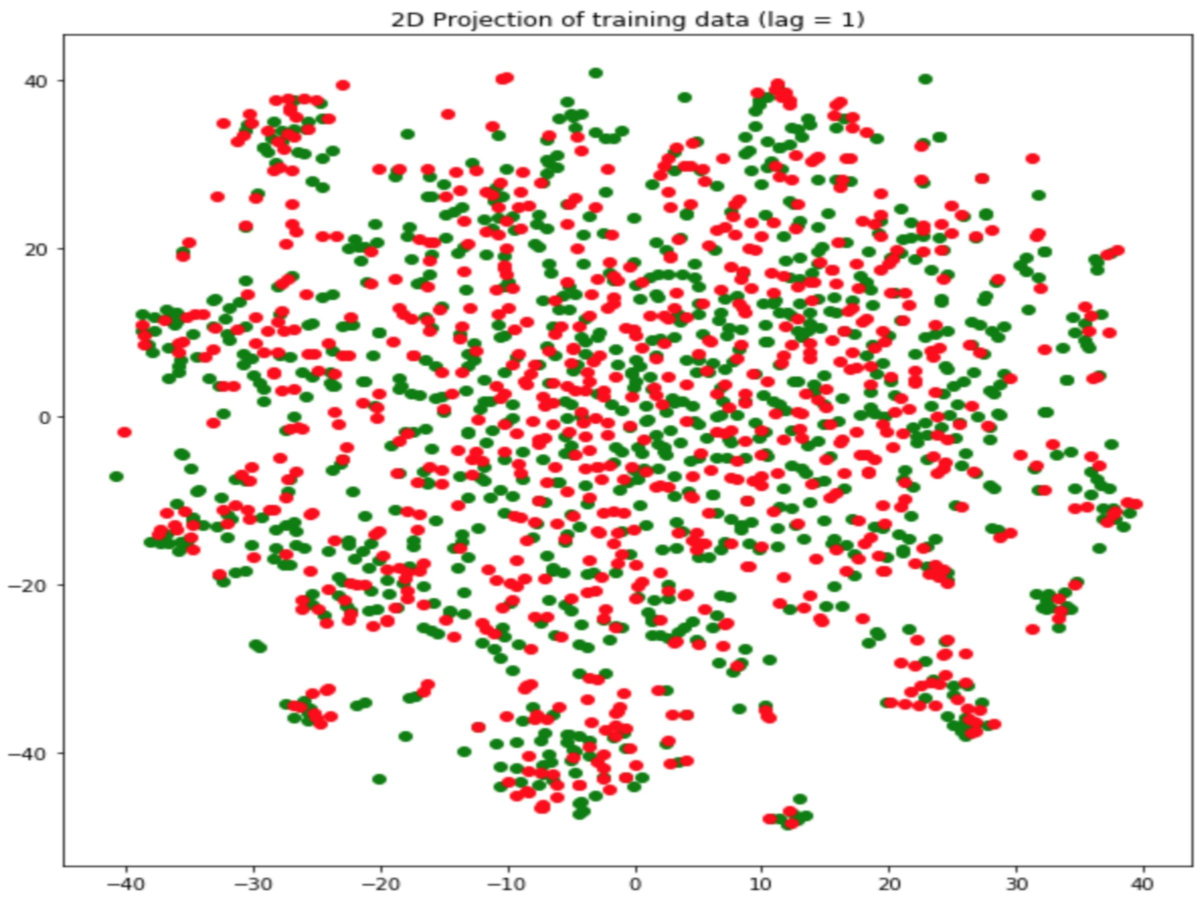


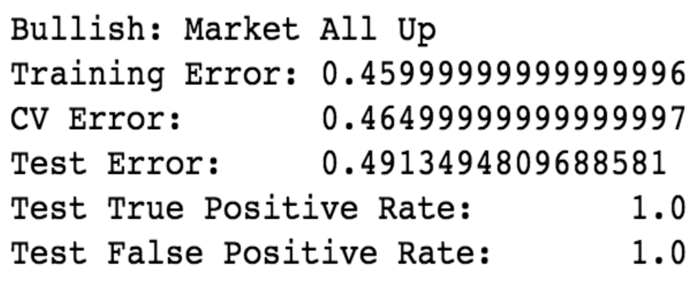
Figure 8: Clustering with lag 1

5 Classification Model

5.1 Benchmark Model

5.1.1 All Bullish

The first benchmark model is the bullish market model: we predict the market in the whole period is bullish and the DJIA will rise everyday.
From Figure 9, we have estimated test error of 0.465 and since we all the predicted value is up, and the TPR & FPR are both 1. We can see in the training dataset, there are roughly more up than downs for the DJIA and this does not happens in the real testset, since the real-test is round 0.5.

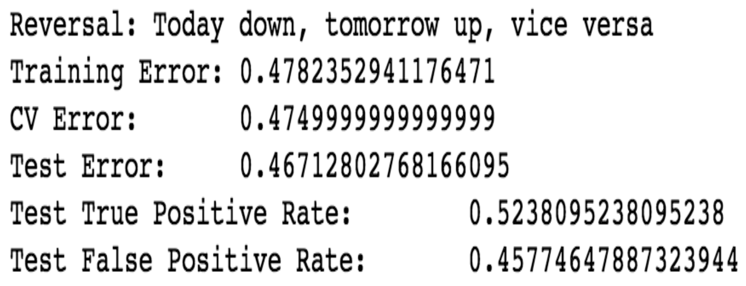


```
Bullish: Market All Up
Training Error: 0.45999999999999996
CV Error:       0.46499999999999997
Test Error:     0.4913494809688581
Test True Positive Rate:        1.0
Test False Positive Rate:       1.0
```

Figure 9: All Bullish model

5.1.2 Reverse

The second benchmark model is the reversal market model: we predict the market in day 2 to behave oppositely in day 1. For example, if in day i, the DJIA rises up, then we predict DJIA will drop in day i+1, $\forall i$ and vice versa.
From Figure 10, we have estimated test error of 0.475, slightly worse than the first benchmark. The TPR is much bigger than FPR; 52% among all the predictions that the DJIA will go up is correct, it indeed go down and 54% among all the predictions that the DJIA will go down is correct. This indicates that, when the DJIA rise today, it has a higher chance, about 54% to decrease tomorrow.



```
Reversal: Today down, tomorrow up, vice versa
Training Error: 0.4782352941176471
CV Error:       0.47499999999999999
Test Error:     0.46712802768166095
Test True Positive Rate:        0.5238095238095238
Test False Positive Rate:       0.45774647887323944
```

Figure 10: Reverse model

5.2 KNN

5.2.1 Model

The first model is K Nearest Neighbour, in building this model, for every single point of training dataset, we observe its K nearest points, check those points movement: if majority of the points show an up movement, we will predict this training data to give an outcome with up, vice versa.

The number of K is selected by cross-validation ranging from 1 to 20. Then select the K which can give the smallest Cross-Validation error. The distance between two points is measured by cosine similarity, which is from 0 to 1. The higher the cosine similarity, the nearer the two points are.

$$\text{similarity between point A and B} = cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{|\mathbf{A}||\mathbf{B}|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

5.2.2 Results

Figure 11 and 12 are the results of KNN method. From cross-validation range 1 to 20, k=5 gives the smallest estimated test error. Both vocabulary method give similar cv-error around 0.46, but in terms TPR, KNN built under the sentiment dataset is around 0.61 and beat the one under frequency dataset and at the same time, KNN built under the sentiment dataset is around 0.56, relatively high than one under frequency dataset. In general, KNN performs not good, since test errors on both vocabulary model are higher than the benchmark 2 test error.

```
Minimum CV Error: 0.4580000000000001    K = 5
KNN Model with K = 5
Training Error: 0.3100000000000005
CV Error:       0.4580000000000001
Test Error:     0.4982698961937716
Test True Positive Rate:        0.5374149659863946
Test False Positive Rate:       0.5352112676056338
```

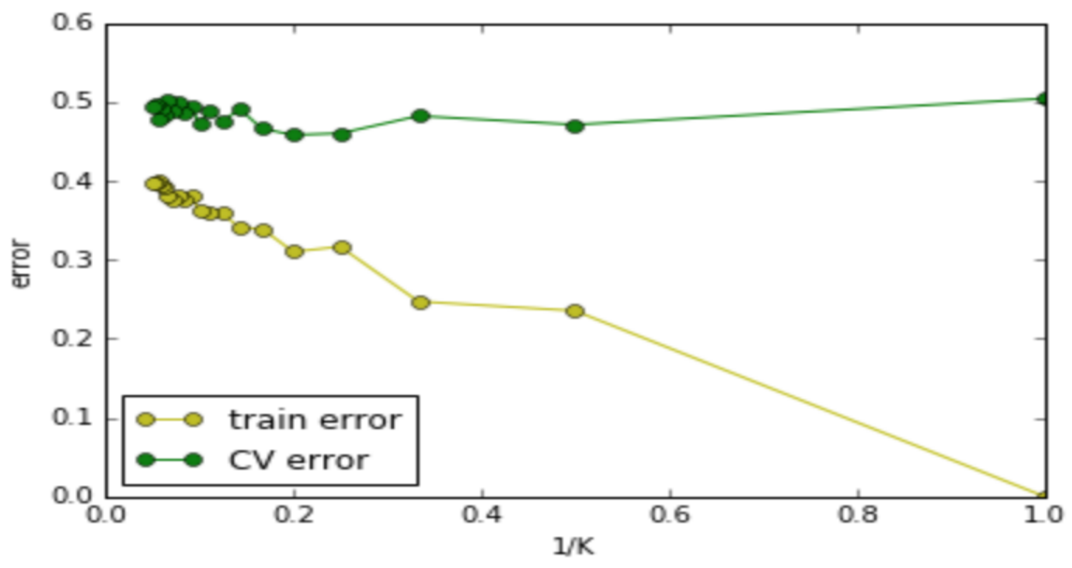


Figure 11: KNN under frequency dataset

```
Minimum CV Error: 0.46599999999999999    K = 5
KNN Model with K = 5
Training Error: 0.3076470588235294
CV Error:       0.4659999999999999
Test Error:     0.4740484429065744 4
Test True Positive Rate:        0.6122448979591837
Test False Positive Rate:       0.5633802816901409
```

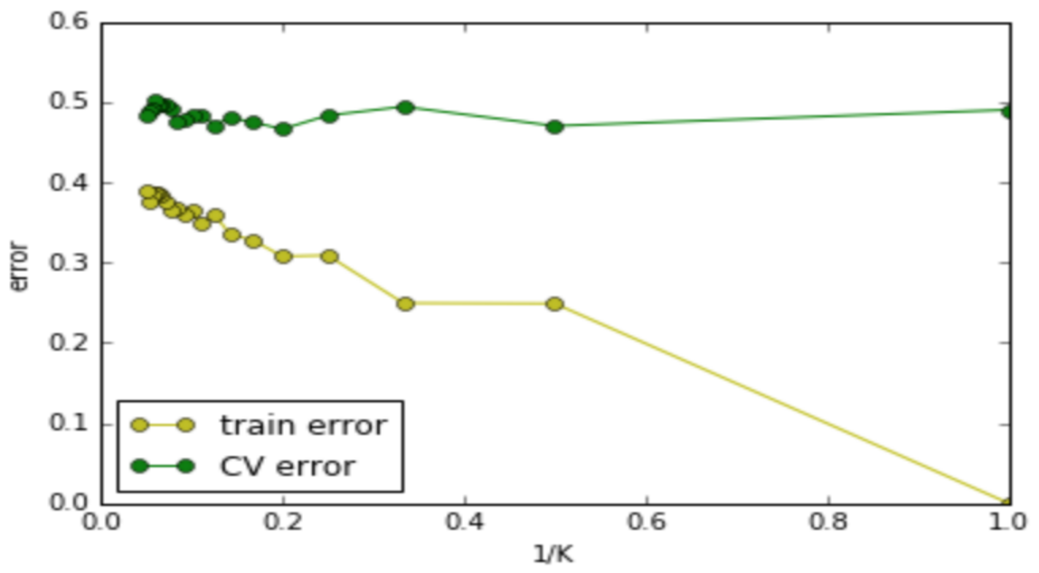


Figure 12: KNN under sentiment dataset

5.3 Logistic Regression

In the logistic regression, the probability of Y=1 is the logistic function of linear function of predictors.

$$\Pr(Y=1|\bar{X}) = \frac{\exp^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}{1+\exp^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}$$

This is equivalent to the logodds, which can be expressed as a linear combination of predictors.

$$\log(\frac{\Pr(Y=1|\bar{X})}{1-\Pr(Y=1|\bar{X})}) = \beta_0+\beta_1 x_1+\cdots+\beta_p x_p$$

In our project, we use function *LogisticRegression* from *sklearn.linear_model* to calculate the logit function.

5.3.1 Logistic Regression with Ridge Regularization

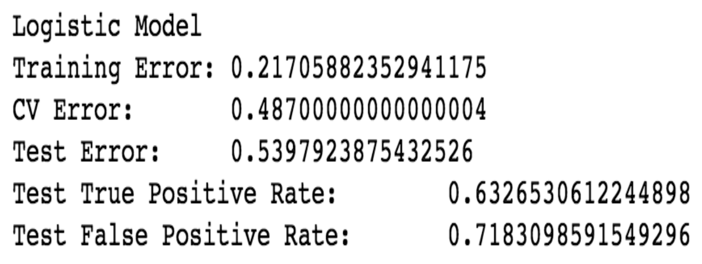
Under the least square approach to perform linear regression, we minimize the RSS:

$$\text{RSS} = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2$$

In the Ridge Regression, we will minimize RSS+$\lambda\sum_{j=1}^{p}\beta_j^2$, where λ is a tuning parameter. The higher the λ, the lower the coefficients of predictors, β_j.

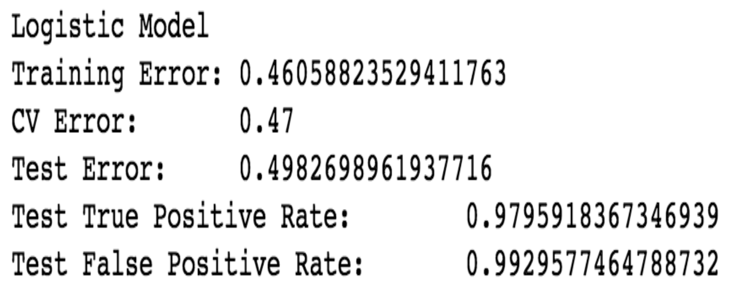
In our project, we use function [8] **LogisticRegression** from **sklearn.linear_model** to calculate the logit function. We choose the penalty

The Logistic Ridge Regression under the sentiment dataset gives a smaller estimated and real test error. However, from the TPR and FPR, we can see under the sentiment dataset, the logistic regression tend to predict the stock will rise in the next day. Considering there are only 4 predictors in the sentiment dataset, i.e. Compound, neutral, positive and negative in the sentiment dataset, one possible reason is that the model under 4 predictors give logit score greater than the threshold, and majority of the test cases are predicted as positive. Hence, in general, logistic Ridge regression is not a good model.



```
Logistic Model
Training Error: 0.21705882352941175
CV Error:       0.48700000000000004
Test Error:     0.5397923875432526
Test True Positive Rate:        0.6326530612244898
Test False Positive Rate:       0.7183098591549296
```

Figure 13: Logistic Ridge regression under frequency dataset



```
Logistic Model
Training Error: 0.46058823529411763
CV Error:       0.47
Test Error:     0.4982698961937716
Test True Positive Rate:        0.9795918367346939
Test False Positive Rate:       0.9929577464788732
```

Figure 14: Logistic Ridge regression under sentiment dataset

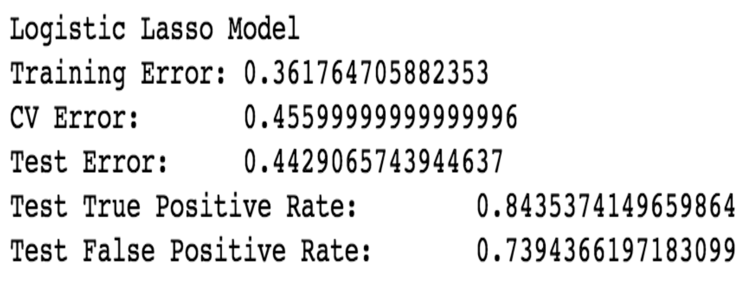
[8]From http://scikit-learn.org/stable/modules/generated/sklearn.linear$_{m}$*odel.LogisticRegression.html*

5.3.2 Logistic Regression with LASSO Regularization

In the Ridge Regression, we will minimize RSS+$\lambda \sum_{j=1}^{p} \left|\beta_j\right|$, where λ is a tuning parameter. The higher the λ, the even lower the coefficients of predictors, β_j and some β_j will be shrunk to exactly 0.

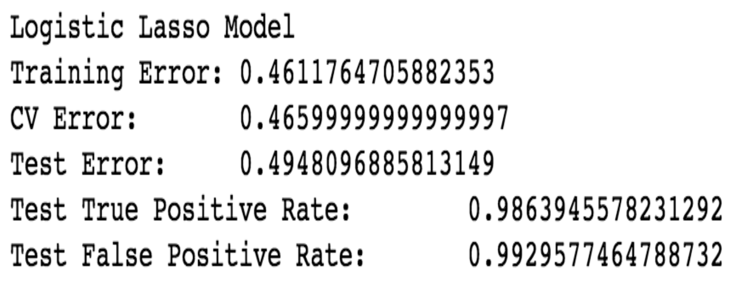
In our project, we still use the **LogisticRegression** function in python and choose the penalty

Again, LASSO gives a more extreme TPR and FPR in both sentiment dataset, though the estimated and real test error rate of the frequency dataset is slightly better than the benchmark model and the KNN. However, given the fact that it is extremely biased towards the positive (Y=1), it is not wise to choose LASSO as well.



```
Logistic Lasso Model
Training Error: 0.361764705882353
CV Error:       0.45599999999999996
Test Error:     0.4429065743944637
Test True Positive Rate:        0.8435374149659864
Test False Positive Rate:       0.7394366197183099
```

Figure 15: Logistic LASSO regression under frequency dataset



```
Logistic Lasso Model
Training Error: 0.4611764705882353
CV Error:       0.46599999999999997
Test Error:     0.4948096885813149
Test True Positive Rate:        0.9863945578231292
Test False Positive Rate:       0.9929577464788732
```

Figure 16: Logistic LASSO regression under sentiment dataset

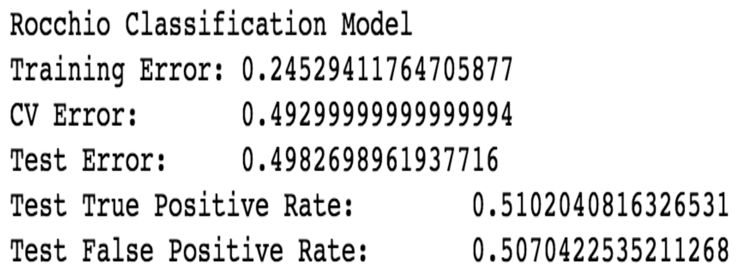
5.4 Rocchio Classification

5.4.1 Model

In building Rocchio Classification, we first calculate the centroids of the groups contains all Y=0 or Y=1. For example, for all data with Y=1, for each variable (term in frequency table or sentiment in sentiment variables), we calculate the average of that variable across the days. Hence, the centroids are the vector with the entry contains the average of variables during specific periods (controlled by the selection of training set or cv training set). Then, we calculate the Euclidean distance between the centroid and each data point. If the data point is nearer to the centroid of Y=1, then this data is assigned to Y=1 and we predict the stock will raise in this day and vice versa.

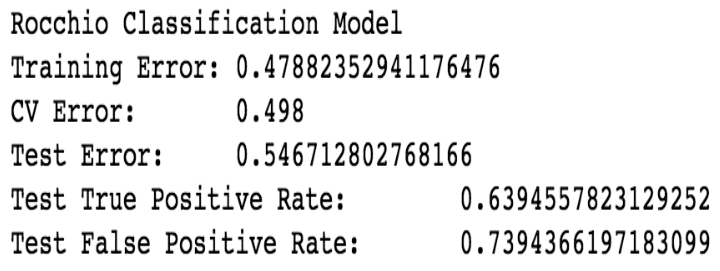
5.4.2 Results

From figure 17 and figure 18, the estimated test error is about 0.5, higher than the benchmark model. Rocchio under the frequency dataset almost has 0 discrimination power, since TPR and FPR are about 0.5. For sentiment dataset, the real test error is too high (0.54 is higher than that of the benchmark model 0.46, 0.49); also, this model is again, slightly biased towards the class Y=1, since the TPR and FPR are relatively high. So, unfortunately, Rocchino model is not ideal for our prediction.



```
Rocchio Classification Model
Training Error: 0.24529411764705877
CV Error:       0.49299999999999994
Test Error:     0.4982698961937716
Test True Positive Rate:        0.5102040816326531
Test False Positive Rate:       0.5070422535211268
```

Figure 17: Rocchio Classification under frequency dataset



```
Rocchio Classification Model
Training Error: 0.47882352941176476
CV Error:       0.498
Test Error:     0.546712802768166
Test True Positive Rate:        0.6394557823129252
Test False Positive Rate:       0.7394366197183099
```

Figure 18: Rocchio Classification under sentiment dataset

5.5 Naive Bayes

5.5.1 Model

Naive Bayes[9] is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{X} = (x_1, \cdots, x_n)$ representing some n independent variables, it assigns to this instance probabilities $p(C_k|x_1, \cdots, x_n)$ for each of K possible outcomes C_k. From Bayes' theorem,

$$p(C_k|\mathbf{x}) = \frac{p(C_k, x_1, \cdots, x_n)}{p(\mathbf{x})}$$

The denominator is constant and the numerator can be expressed as follows in chain rule:

$$\begin{aligned} p(C_k, x_1, \cdots, x_n) &= p(x_1, \cdots, x_n, C_k) \\ &= p(x_1|x_2, \cdots, x_n, C_k)p(x_2, \cdots, x_n, C_k) \\ &= p(x_1|x_2, \cdots, x_n, C_k)p(x_2|x_3, \cdots, x_n, C_k)p(x_3, \cdots, x_n, C_k) \\ &= \cdots \\ &= p(x_1|x_2, \cdots, x_n, C_k)p(x_2|x_3, \cdots, x_n, C_k)\cdots p(x_{n-1}|x_n, C_k)p(x_n|C_k)p(C_k) \end{aligned}$$

Assume that each variable x_i is conditionally independent of every other features x_j for i $\neq j, giventhecategory$C$_k$, Hence, we have $p(x_i|x_{i+1}, \cdots, x_{i+n}, C_k) = p(x_i|C_k)$.

Overall, the joint model can be expressed as

$$\begin{aligned} p(C_k|x_1, \cdots, x_n) &= \frac{1}{Z}p(C_k, x_1, \cdots, x_n) \\ &= \frac{1}{Z}p(C_k)p(x_1|C_k)p(x_2|C_k)\cdots \\ &= \frac{1}{Z}p(C_k)\Pi_{i=1}^{n}p(x_i|C_k) \end{aligned}$$

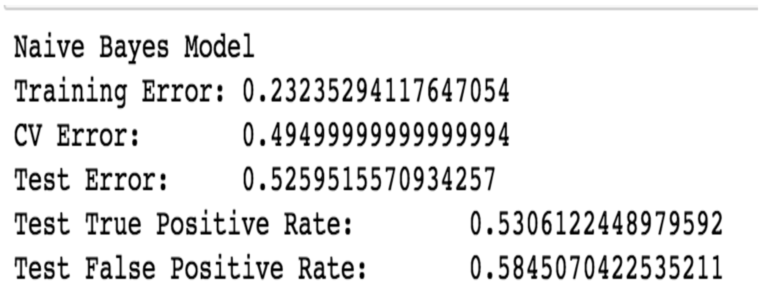
Where Z = $p(\mathbf{X}) =_k p(C_k)p(\mathbf{x}|C_k)$ is a constant. The naive Bayes classifier combines the model with a decision rule which is to pick the hypothesis that is most possible. Hence overall, a Bayes classifier is a function that assigns a class label $\hat{y} = C_k$ for some k as follows:

$$\hat{y} = \underset{k \in 1, \cdots, K}{\text{argmax}} S(t_i)p(C_k)\Pi_{i=1}np(x_i|C_k)$$

[9]From https://en.wikipedia.org/wiki/Naive$_{B}ayes_{c}lassifier$

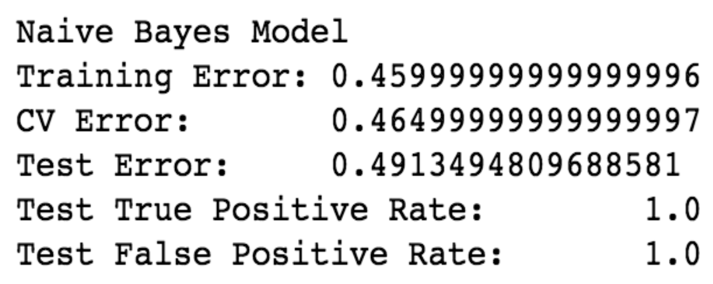
In our project, we use the function [10] **BernoulliNB** in

The most interesting part of the results is for the sentiment dataset, the TPR and FPR are both 1. Although Naive Bayes Classifier under the frequency data give a quite low training error, 0.232, it still has relatively high estimated test error and real test error, which is around 0.5. Again, the Naive Bayes classifier is not a good model.



```
Naive Bayes Model
Training Error: 0.23235294117647054
CV Error:       0.49499999999999994
Test Error:     0.5259515570934257
Test True Positive Rate:       0.5306122448979592
Test False Positive Rate:      0.5845070422535211
```

Figure 19: Naive Bayes Classification under frequency dataset



```
Naive Bayes Model
Training Error: 0.45999999999999996
CV Error:       0.46499999999999997
Test Error:     0.4913494809688581
Test True Positive Rate:        1.0
Test False Positive Rate:       1.0
```

Figure 20: Naive Bayes Classification under sentiment dataset

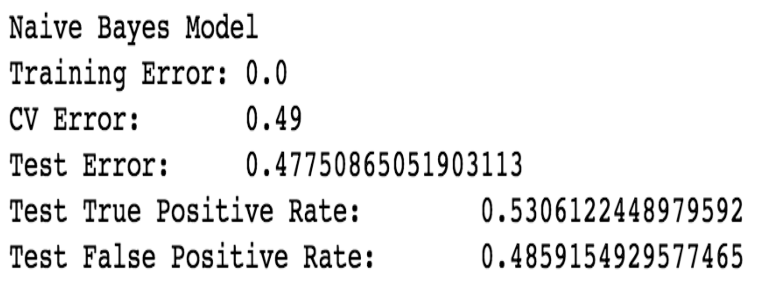
5.6 Decision Tree

5.6.1 Model

Decision tree partition the predictor space into J non-overlapping regions $R_1, R_2, \cdots, R_J$. If an observation $(x_1, \cdots, x_p)$ falls into R_j, then $\hat{y}$ =most common class of observations in R_j. The algorithm is to grow a large tree by recursive binary splitting and stop when a region has too few observations. In our project, we use the function[11] **tree** in **sklearn** package, and the function **tree.DecisionTreeClassifier** is the default setting.

5.6.2 Results

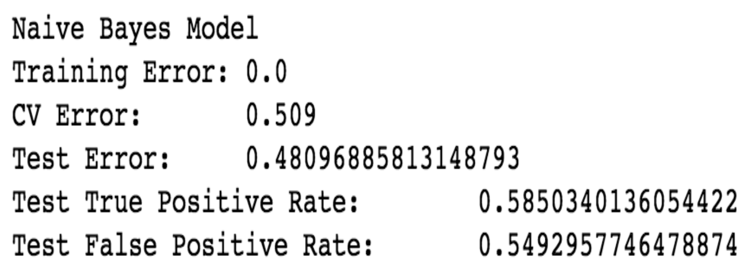
From Figure 21 and 22, again, the results are not exciting, in both datasets, the Decision Tree model give test error is around 0.48, more or less the same as the benchmark models.



```
Naive Bayes Model
Training Error: 0.0
CV Error:       0.49
Test Error:     0.47750865051903113
Test True Positive Rate:       0.5306122448979592
Test False Positive Rate:      0.4859154929577465
```

Figure 21: Decision Tree under frequency dataset

[10]From http://scikit-learn.org/stable/modules/naive$_b$*ayes.html*
[11]From http://scikit-learn.org/stable/modules/tree.html



```
Naive Bayes Model
Training Error: 0.0
CV Error:       0.509
Test Error:     0.48096885813148793
Test True Positive Rate:        0.5850340136054422
Test False Positive Rate:       0.5492957746478874
```

Figure 22: Decision Tree under sentiment dataset

5.7 Random Forests

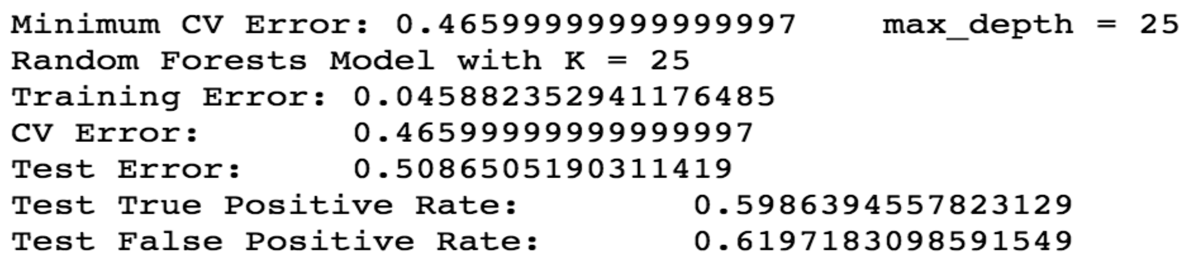
5.7.1 Model

Random forest is based on the single decision tree. Instead of using all the p predictors, it use only some of them, a popular choice is $\sqrt{p}$. Then generate B samples of such smaller version of trees and average the results. The reason is that we can reduce the variance by reducing the uncorrelated quantities and using smaller number of the predictor in each tree will essentially reduce the correlations.

In our project, we use the function [12] **RandomForestClassifier** in **sklearn.ensemble** package. We control the number of predictors picked by the parameter **max_depth**, which indicates the distance from the root to the bottom leaves. We use cross-validation to choose the best **max_depth** ranging from 1 to 50.

5.7.2 Results

From figure 23 and 24, in both datasets, we can see the **max_depth** = 25 gives us the best CV-error, which is around 0.465. In general, Random Forest model is better than logistic, naive Bayesian, Rocchio and single tree method in terms of estimated test error.



```
Minimum CV Error: 0.46599999999999997    max_depth = 25
Random Forests Model with K = 25
Training Error: 0.045882352941176485
CV Error:       0.46599999999999997
Test Error:     0.5086505190311419
Test True Positive Rate:        0.5986394557823129
Test False Positive Rate:       0.6197183098591549
```

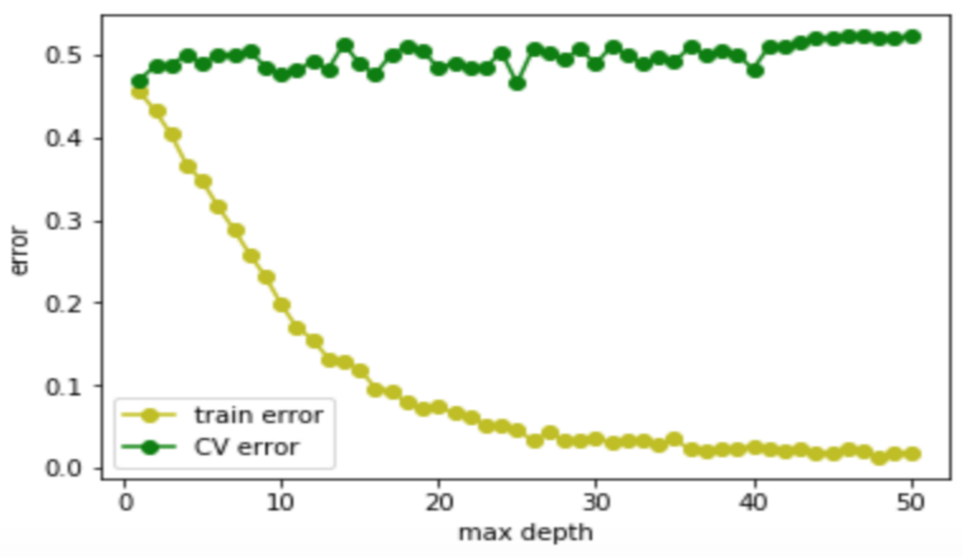


Figure 23: Random Forest under frequency dataset

[12]From http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

```
Minimum CV Error: 0.4689999999999999     max_depth = 24
Random Forests Model with K = 24
Training Error: 0.020588235294117685
CV Error:       0.4689999999999999
Test Error:     0.46712802768166095
Test True Positive Rate:        0.5510204081632653
Test False Positive Rate:       0.4859154929577465
```

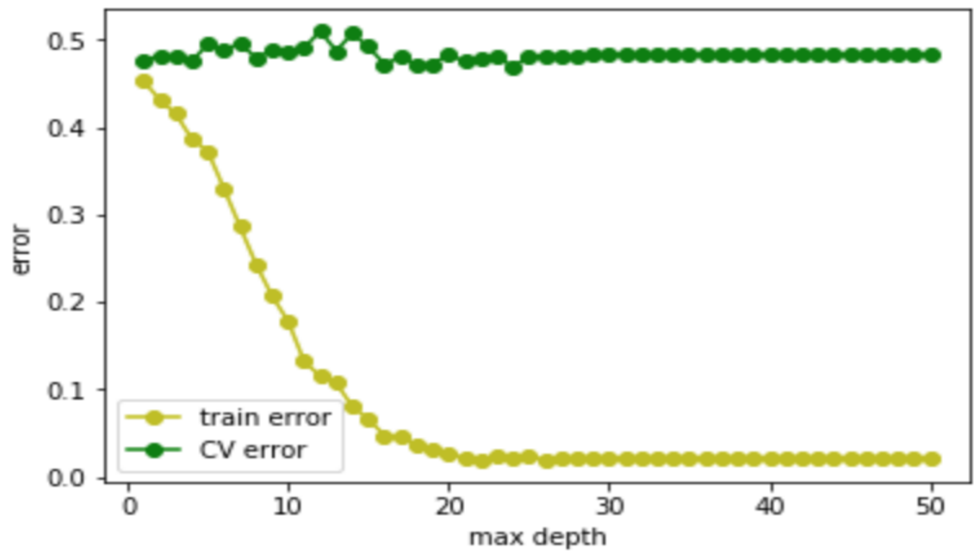


Figure 24: Random Forest under sentiment dataset

5.8 Support Vector Classifier

5.8.1 Model

In a p-dimensional space, a nonlinear hyperplane is a flat subspace of p-1 dimensions, and specified by the polynomial: $0 = f(X_1, X_2, \cdots, X_p) = \beta_0 + \sum_{j=1}^{p} \beta_{j1} X_j + \sum_{j=1}^{p} \beta_{j2} X_j^2$. Then we can classify each observation $\bar{\mathbf{X}}_{\mathbf{i}} = (x_{i1}, x_{i2}, \cdots, x_{ip})$ by:

$$\begin{aligned} y_i &= +1 \quad \text{if} \quad f(\bar{X}) > 0 \\ y_i &= -1 \quad \text{if} \quad f(\bar{X}) < 0 \end{aligned}$$

In constructing margin classifier, we want to maximize the margin M, (the minimum distance from the hyperplane to any data) and constraint on the hyperplane that can separate two groups with some fixed amount of violation C. C can be chosen from cross validation. Fitting:

$$\begin{aligned} \underset{\beta_0,\cdots,\beta_{p2}}{\text{maximize}} \quad & M \\ \text{subject to} \quad & \sum_{j=1}^{p} \beta_{j1}^2 + \sum_{j=1}^{p} \beta_{j2}^2 = 1 \\ & y_i(\beta_0 + \sum_{j=1}^{p} \beta_{j1} X_{ij} + \sum_{j=1}^{p} \beta_{j2} X_{ij}^2) \geq M(1-\epsilon_i),\ \forall i \\ & \epsilon_i \geq 0 \\ & \sum_{i=1}^{n} \leq C \end{aligned}$$

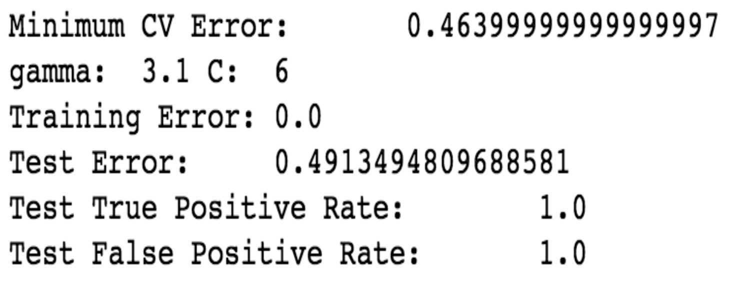
This optimization problem has a dual problem:

$$\begin{aligned}
&\underset{\alpha_1,\cdots,\alpha_p}{\text{maximize}} && \sum_{i=1}^{n}\frac{\alpha_i}{y_i}-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j K(\bar{x_i},\bar{x_j})\\
&\text{subject to} && \sum_{i=1}^{n}\alpha_i=0\\
& && 0\le\frac{\alpha_i}{y_i}\le\lambda,\ \forall i\\
&\text{with decision boundary:} && f(\bar{x})=h(\bar{\alpha})+\sum_{j=1}^{n}\alpha_i K(\bar{x_i},\bar{x_j})
\end{aligned}$$

$K(\bar{x},\bar{x'})$ is a kernal and it quantifies the similarity between $\bar{x}$ and $\bar{x'}$. The choice of K determines the shape of the decision boundary and in this project, we choose a **Radial** kernal: $K(\bar{x},\bar{x'}) = \exp(-\gamma\|\bar{x}-\bar{x'}\|^2)$. We also use cross-validation to choose γ. We use function **SVC** in **sklearn.svm** [13] to generate the SVC classifier and choose C with range from 1 to 15 and γ with range from 0.1 to 4 in the cross-validation.

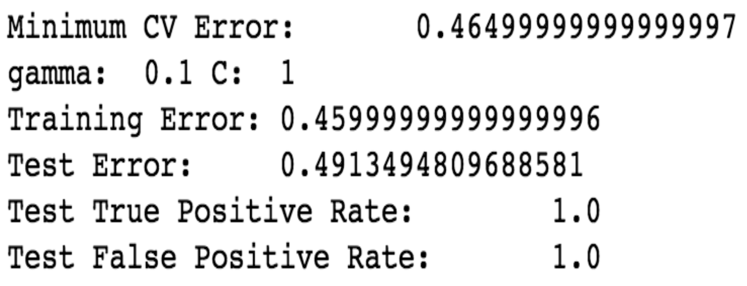
5.8.2 Results

From figure 25 and 26, we can see the estimated test errors are quite similar for 2 dataset. For the frequency dataset, the 0 training error indicates the overfitting. Also noticed that in both cases, the TPR and FPR are equal to 1, indicates that SVM is biased towards Y=1.



```
Minimum CV Error:       0.46399999999999997
gamma:  3.1 C:  6
Training Error: 0.0
Test Error:     0.4913494809688581
Test True Positive Rate:        1.0
Test False Positive Rate:       1.0
```

Figure 25: SVM under frequency dataset



```
Minimum CV Error:       0.46499999999999997
gamma:  0.1 C:  1
Training Error: 0.45999999999999996
Test Error:     0.4913494809688581
Test True Positive Rate:        1.0
Test False Positive Rate:       1.0
```

Figure 26: SVM under sentiment dataset

[13] From http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

6 Model Comparison and Conclusion

In Section 5, we have built several model and Table 1 lists the results under the frequency dataset; Table 2 is the lists the results under the sentiment dataset.

We choose the good model by comparing the estimated error (cross-validation error). Under the frequency dataset, Under the frequency dataset, KNN, Random forest and SVM give smaller estimated error and perform better than the benchmark model. However, to compare the real test error, no one beat the Reverse model, indicating that under the frequency dataset, our model failed to give a better results than an arbitrage guessing.

Under the sentiment dataset, KNN, Ridge, LASSO, Naive Bayes, Random forest and SVM give small estimated error relative to benchmark model. When compare the test error, although again no model beat the Reverse Model, the Random Forest model give a fairly good test error.

Table 1: Model Comparison under Frequency Dataset

Model	Estimated Test Error	Test Error	TPR	FPR
All Bullish	0.465	0.491	1	1
Reverse	0.475	0.467	0.524	0.458
KNN	0.458	0.498	0.537	0.535
Ridge	0.487	0.539	0.633	0.718
LASSO	0.487	0.540	0.633	0.718
Rocchio	0.493	0.498	0.510	0.507
Naive Bayes	0.495	0.526	0.531	0.585
Decision Tree	0.490	0.478	0.531	0.486
Random Forest	0.466	0.509	0.599	0.620
SVM	0.464	0.491	1	1

Table 2: Model Comparison under Sentiment Dataset

Model	Estimated Test Error	Test Error	TPR	FPR
All Bullish	0.465	0.491	1	1
Reverse	0.475	0.467	0.524	0.458
KNN	0.466	0.474	0.612	0.563
Ridge	0.470	0.498	0.980	0.993
LASSO	0.470	0.498	0.980	0.993
Rocchio	0.498	0.547	0.639	0.739
Naive Bayes	0.465	0.491	1	1
Decision Tree	0.509	0.481	0.585	0.549
Random Forest	0.469	0.467	0.55	0.486
SVM	0.465	0.491	1	1

So far, we used two different ways to generate our model matrix, the frequency and the sentiment, from the daily news and build 2 benchmark and 7 classification models to predict the stock up and down based on the daily news. It turns out that although the test error is still relative high, around 0.46, the benchmark model, Reverse model is more stable than other models. In future work, to further improve our classification, we have some proposed modification:

1. **Fetch more data** In general, more data we can get, the more accurate the model we can build. We only have 1700 training data overall and 700 cv-training data in the cv-set, which is fairly not enough for our frequency dataset, since there are thousands of predictors. Hence, in future, we will try to get more data; probably top 40 headlines from Year 2000.

2. **Consider the lag** As we have mentioned in Section 3.2.1, there must be the lagging effect in the stock prediction based on the news. For example, stock market may have internalized the effect of finance news even before they are officially published; on the other hand, stock market may react upon on the technology news and political news only after they are officially publish. In future, when we build the model, we will consider the lag from 0 to 7 and pick the best model in the specific lag, that is, predict the stock movement in day $i+1, i+2, \cdots, i+7$ based on the news on day i.

3. **Specify the days** We can also build model specified the stock movement of particular days. For example, it seems the stock market always plunge on Monday, we can build models on the daily news or Friday and weekend news only to predict Monday stock movement, and so on and so forth.