**Predict DJIA Movements Using Daily News**

Rui Shen, Tianjiao Wang, Yang Hu, Yan Yu

1. **Introduction**
   1. Project Overview

Social sentiments are proved to have great influences on the financial markets. In this project, we use 8-year daily news headlines from Reddit WorldNews Channel to predict the directions of DJIA movements. It is a binary classification task using unstructured data. Hence, we have two subtasks in our project: one is to transform the text data into structured data; another is to use the data to predict the movements.

First, we collected the data set from Kaggle. The historical news data was obtained using web scraping techniques and DJIA data was downloaded directly from Yahoo Finance. We then transform the data into term-document sparse matrix for the following analysis. To begin the training process, we use basic models such as logistic regression and apply it on our testing set and generate the crosstab results. Next, we conduct dimension reduction and variable selection techniques to improve the results, including LASSO/Ridge and PCA. We also implement advanced models such as Random Forest, LDA and Social Sentiment Analysis. Finally, we compare the results from those models and reasonable explanations.

* 1. Problem Statement

Our goal is to apply what we learned from the Statistical Machine Learning II classes into this real world financial issue. The main tasks and challenges are:

1. Obtain and preprocess the data using web scraping and NLP techniques
2. Train a reliable classifier
3. Compare different models and analyze their strengths and weaknesses
4. Make an applicable tool to serve as a good trading assistance

1. **Data Set and Text Processing**
   1. Data Set Summary

The news dataset includes top 25 daily news headlines from Aug-08-2008 to Jul-01-2016, in total 1989×25 headlines. The volatility dataset includes the rise (category 1) or fall (category 0) flag of each day’s DJIA index compared with the past 5-day moving average, which are also from the same time period.

In order to train our model, we split the total dataset from Jan-01-2015. The first 7 year’s data is the training set and last 2 year’s data is the testing set. The base rate of category 1 is 54.19% for training set and 50.79% for testing set, so the data is well-balanced.

* 1. Text Preprocessing

1. Stemming

Since we mainly use the bag-of-words representations for our text, the first step of text processing is to stem the words and avoid the impact of word transformation on text vectorizing. We use the SnowballStemmer to remove morphological affixes of each word in all the headlines, but still keep the sentence structure unchanged.

1. Word parsing

Since we have multiple daily top news, we first get a single string by pasting 25 pieces of news into one. Then we use CountVectorizer to parse strings into words or phrases.

CountVectorizer creates a sparse matrix of which the columns are word features and the rows are dates. For the parameters of CountVectorizer, we only select words with length over two. Also, we add ‘english’ stop word list, which can filter out the most common words in English.

Instead use the bag-of-words representation, we use n-grams representation. N-grams refers to combinations of words. We observe that single words often fail to carry the true meaning of the original text. Phrases can better carry the sentiments of news. For example, a sentence with “north korea” may indicate very different messages from the one with both “north” and “korea”.

We use the train data to fit the processor then transform both train data and test data into term-document matrices. This ensures the train and test set share the same word features. That is to say, the two matrices have same columns names. The train\_mat is a 1608-by-1719100 matrix and the test\_mat is a 378-by-1719100 matrix. We will use these two matrices for further analysis.

1. Normalization

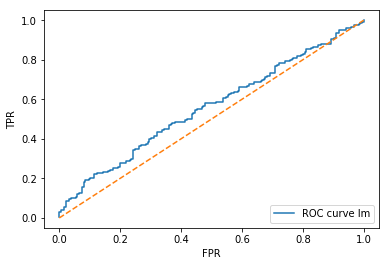
News on different dates have different lengths. We consider using word frequency instead of word count. TfidVectorizer can normalize the count vectors by “Term Frequency – Inverse Document” Frequency (TF-IDF). Here we use TfidfTransformer to transform the word count matrix into the word frequency matrix. Matrices returned by TfidfTransformer have same dimensions with the input matrices.

1. **Models and Performances**
   1. Logistic Regression

The target is to find a relationship between the words frequency and the moving direction of the stock market index. Logistic Regression uses a sigmoid transformation to predict the probability of an upward drift. We set the threshold to 0.5. If the probability is larger than 0.5, we see it as a prediction of rise, otherwise, we see it as a prediction of drop. Logistic is a very simple model, so it has low variance. It is a good benchmark of other methods.

We fit the model using the train data and predict the trend of each date. The misclassification rate on the test set is 45.2%. ROC curve is significantly above the random classifier and has an AUC of 0.546. The confusion matrix result shows:

|  |  |  |
| --- | --- | --- |
| Actual \ Predicted | Down | Up |
| Down | 87 | 99 |
| Up | 72 | 120 |



* 1. Logistic Lasso Regression

In NLP, there are lots of words that do not express any positive or negative feeling about the events. Their explanation power may be trivial, but Lasso regression can solve this problem. We use LogisticRegressionCV with penalty = ‘l1’, and the result shows the remaining words are:

*['bin laden', 'court rule', 'new zealand', 'north korea', 'nuclear weapon', 'sexual abus', 'world cup', 'world largest']*

From the result we can see the keywords are very explainable for macroeconomics and geopolitical events, but unfortunately, the result is discouraging. The test misclassification rate is 49.47%, while AUC is 0.497, which is worse than the logistic regression without penalty.

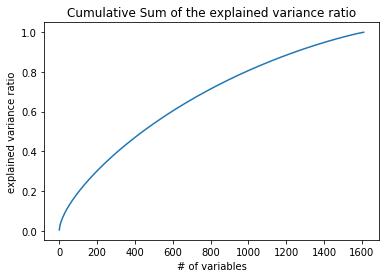
* 1. Logistic Ridge Regression

Sometimes ridge regression is better than lasso regression because we don’t know what the specific effects of the X variables, but we suppose it may be some overfitting. So we need some penalty to make it shrink.

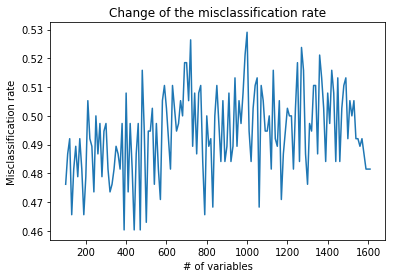
This time we use the LogisticRegressionCV again and set the penalty as ‘l2’, but the result is still unfavorable. The misclassification rate is 49.2%. The AUC is 0.5. And the confusion matrix shows the classifier totally predict everything to be positive.

* 1. Principal Component Analysis (PCA)

There are more than 17000 word features in the term-document matrix. We consider use dimension reduction techniques to filter down the word list and only keep reasonable words. Previously we used document frequency as a cutoff to do the simple filter: we dropped the words that only show up very few times and the ones that show up in almost all the documents. Here we use PCA to further shorten the length of the word list.



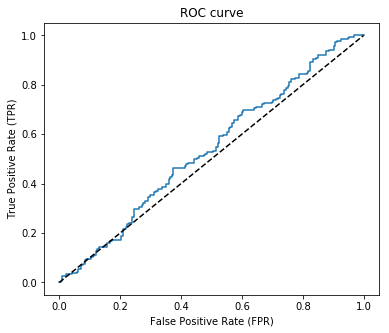
From the above plot, we observed that around 1000 components could already explain 80% of the variance. However, as the number of variable increases, the cumulative sum of the explained variance ratio grows at a relatively slow speed, almost linear.



After going through the number of components to find the optimal point, we find that there is no obvious pattern for the change of the misclassification rates. The best result appeared when using 390, 440 or 470 component. We then investigate more into the specific model of 390 components.

The misclassification rate on the test set is 45.50%. The confusion matrix result shows:

|  |  |  |
| --- | --- | --- |
| Actual \ Predicted | Down | Up |
| 72 | 113 | 106 |
| Up | 66 | 134 |



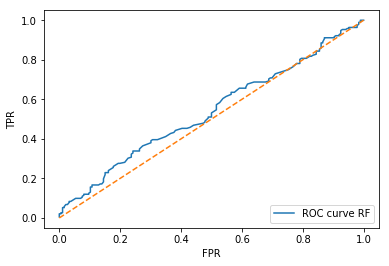
The ROC curve is better than the random classifier, but obviously it is not good enough. This lead us to further try other approaches.

* 1. Random Forests

Random Forests provides a stable result with low bias and variance, and we tried to use Random Forests to deal with this data. We set the number of trees = 500 and oob\_score = true.

We fit the model using the train data and predict the trend of each date. The out-of-bag misclassification rate is 47.8 while the misclassification rate on the test set is 48.9%. ROC curve is above the random classifier and has an AUC of 0.538. The confusion matrix result shows:

|  |  |  |
| --- | --- | --- |
| Actual \ Predicted | Down | Up |
| Down | 56 | 130 |
| Up | 57 | 135 |



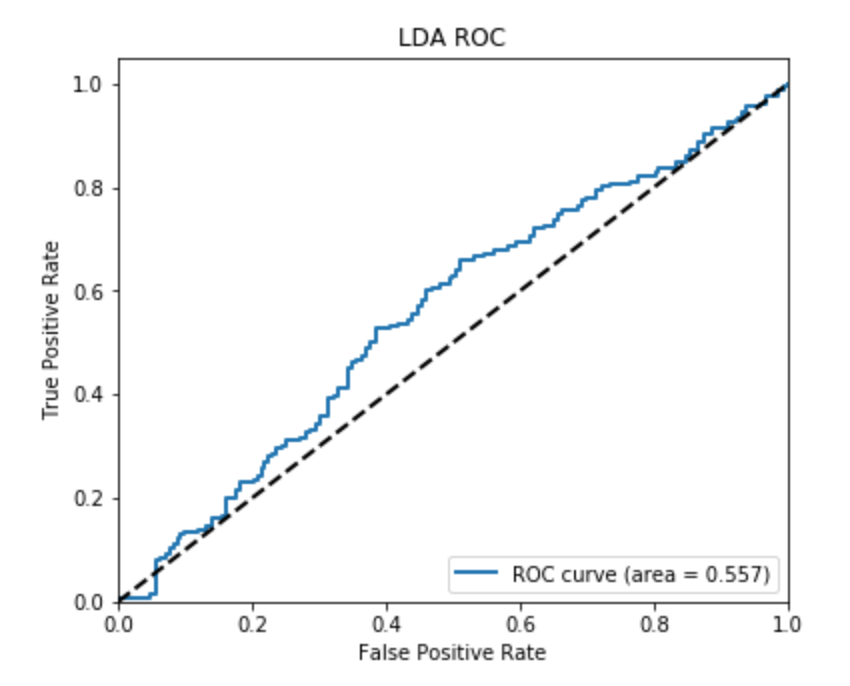
* 1. Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation identifies topics in a set of documents. LDA assumes each document has a mixture of topics and words are distributed according to topics. We assume there are two topics: up and down. Words are generated according to the corresponding Dirichlet distribution. Here we set the number of components = 2, learning method='batch' and max\_iter=100.

We fit the model using the train data and predict the topic of each date. We obtain the label probability from the fitted model and predict the label to be 1 if the probability is larger than 0.5 and 0 otherwise. The misclassification rate on the test set is 43.9%. The confusion matrix result is shown below:

|  |  |  |
| --- | --- | --- |
| Actual \ Predicted | Down | Up |
| Down | 113 | 79 |
| Up | 87 | 99 |

ROC curve is significantly above the random classifier and has an AUC of 0.557.



Also, we list the top 20 words given the label. We obtain the distribution over words for each topic by calculating and ranking the normalized LDA components:

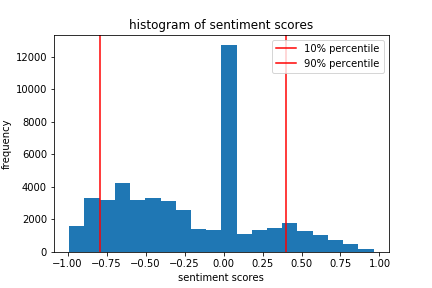
*Top 20 words for label 0: ['north korea' 'year old' 'human right' 'prime minist' 'unit state'  
 'bbc news' 'saudi arabia' 'west bank' 'julian assang' 'new zealand'  
 'war crime' 'south korea' 'north korean' 'al jazeera' 'edward snowden'  
 'middl east' 'al qaeda' 'kim jong' 'islam state' 'vladimir putin']  
Top 20 words for label 1: ['north korea' 'year old' 'prime minist' 'human right' 'bbc news'  
 'north korean' 'unit state' 'new zealand' 'south korea' 'saudi arabia'  
 'west bank' 'war crime' 'kim jong' 'pirat bay' 'middl east' 'al qaeda'  
 'hong kong' 'south africa' 'court rule' 'polic offic']*

The lists of words for two labels have a large intersection. From the list of words, we do observe some patterns for volatility changes. Top words of label 1 are more closely related to geopolitics while words of label 0 are more diverse.

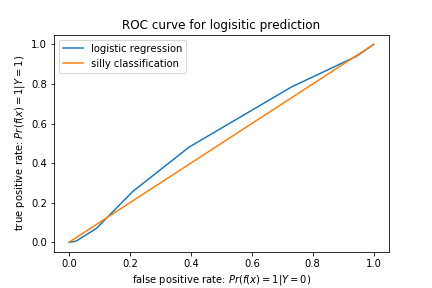
* 1. Sentiment Analysis

Until now, all our results are based on bag-of-words models. We distinguish positive and negative words by simply learning from the training dataset. Another approach of sentiment analysis is to use lexicons of sentiment-related words to tell how positive or negative are the headlines.

In practice, we use the VADER package to rate each headline, and get the compound scores ranging from -1 to 1. The distribution of scores (in the training set) are as follow:



As we can see, there is a large proportion of headlines which is neutral (score close to 0). In reality, only very positive or very negative news will have an impact on market and cause volatility rise. Therefore, we leave out most of the neutral headlines and only count on the number of extremely positive or negative news (score above 90% percentile or below 10% percentile) to predict the changing direction of DJIA.



By applying a logistic regression model, we get a 45.76% misclassification rate on the test set. The ROC curve is above the random classifier, but still could not be stated as satisfying.

1. **Conclusion**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Logistic | Logistic Lasso | Logistic Ridge | PCA | Random Forest | LDA | Sentiment Analysis |
| Misclassification rate | 45.2% | 49.47% | 49.2% | 45.5% | 48.9% | 43.9% | 45.76% |
| AUC | 54.6% |  | 50.0% |  | 53.8% | 55.7% | 54.2% |

In this paper, we investigate whether social sentiments are predictive to stock market movements. We use NLP techniques and applied machine learning model to predict DJIA movements. The results show that news does have some prediction power.

The simplest model--logistic regression model do very well. However, considering we have 17360 words, but only 1611 samples, the logistic regression may also have the problem with overfitting. In order to fix the problem of overfitting, we add some penalty on the model, which leads to logistic lasso regression and logistic ridge regression. The lasso and ridge show poor results and we think the reason might be that each word does not have great prediction power to predict the stock movement, so lasso just shrink it to zero.

PCA seems to be a good dimension reduction tool to shorten our word list and make better predictions. We noticed that within the 17000 words, 1000 components could already explain 80% of the variance. The result seems inspiring. However, as the number of components increases, the cumulative sum of the explained variance ratio grows at a relatively slow speed, almost linear. Also, after going through iterations to find the optimal number of components, we find that there is no obvious pattern for the change of the misclassification rates. Although the misclassification rate of the optimal classifier on the test set is 45.50%, which is not bad, the ROC curve seems not good enough compared to the random classifier. This lead us to further try other approaches.

The Random Forests still cannot beat the simplest logistic regression. This maybe is because the random forests method has a bad performance on very high dimension and sparse datasets.

LDA shows the best result among all models. LDA is a generative topic model and goes well with bag-of -word assumptions. It is particularly useful in distinguish different text topics.

The Sentiment Analysis model also does not give a satisfying result. The model takes advantage of the sentiment-related words lexicon, which simplifies the sentiment analysis. However, this practice also leads to information losses, as the model is unable to take full use of the training set and will fail to identify some special but ‘neutral’ words such as Trump or Iran, which are quite important for market movements.

Our research proves the prediction power of news does exist but it is very weak. Predicting market movements is a difficult task, especially given very limited information. We are interested in introducing sequence models such as LSTM model in the future to capture the time series feature of our data.