Text Classification for Stock Market Prediction

Abstract Stock market is influenced by many factors and is difficult to predict. Financial news can help to solve this problem. In this report, text classification models including Naïve Bayes model and three other models in the *nltk* module have been implemented and tested. These models enable investors to make better investment decisions.

1 Introduction

Stock market prices fluctuate greatly and depend on many factors. It is almost impossible to quantify all these factors. Therefore, stock market prediction can be a tough thing at most times. Is there any way to approach stock prices as close as possible? Financial news can be used to solve this problem. That is because when good news is discussed, a company's stock will soar and vice versa. Here, a large amount of financial news is provided to predict whether the price of a stock will go up or down.

2 Models

2.1 The Text Classification Task

The stock market prediction can be converted into a text classification task. Our goal is to implement models that maximize the jointly likelihood of word features and target labels. The task description is as follows:

Input:

- documents (refers to financial news about a target stock that publish in the same day)
- a fixed set of classes $\mathrm{C} = \{C_1, C_2\}$ (refers to the positive and negative prediction for stock market here)

Output:

- a predicted class $\mathrm{c} \in \mathrm{C}$ (whether the stock will go up or down)

2.2 Naïve Bayes Model

The Naïve Bayes classifier is based on the bayes formula and max likelihood estimation, through which we can get the most likely class.

$$\begin{aligned} c_{MAP} &= argmax_{c\in C} p(c|d) \\ &= argmax_{c\in C} \frac{p(d|c)p(c)}{p(d)} \\ &= argmax_{c\in C} p(d|c)p(c) \end{aligned} \tag{1}$$

$$= argmax_{c \in C} p(c_j) \prod_{w \in V} p(w|c)^{count(v,c)}$$

where *w* refers to the types in document *d*.

The probability of class c_j can be calculated as follows:

$$\mathrm{p}(c_j) = \frac{the\ number\ of\ documents\ of\ class\ c_j}{total\ number\ of\ documents} \tag{2}$$

When it comes to the probability $p(w|c)$, it is noticeable that two special processing techniques are needed. Firstly, Laplace smoothing is applied to better deal with the words that never appear in certain class.

$$p(w_i|c) = \frac{count(w_i,c)+1}{\sum_{w \in V} count(w,c)+|V|} \tag{3}$$

Also, to avoid underflow, sum of logs of probabilities is used instead of multiplying probabilities as below:

$$C_{NB} = argmax_{c_j \in C} \log p(c_j) + \sum count(w_i, c_j) \log p\ (w_i|c_j) \tag{4}$$

2.3 Other models in *nltk* module

Before constructing models, word features of the financial news need to be extracted. Often, whether a word appears in a class is more important than the frequency of its appearance. As there are massive types in the financial news corpus, it is inefficient to pay attention to all words. Therefore, stopwords have been removed here by using the stopwords dictionary, and only some words of high frequency are focused and boolean features--whether the words appear in a document are selected as features for classification. Here, 1000 high frequency words were tested first.

Here, three classifiers in the *nltk* module are tested.

A. The Decision Tree Classifier

The Decision Tree Classifier can summarize decision rules from a series of data with features and labels, and present these rules with tree graph structure to solve classification problems. The core of decision tree lies in the feature selection for branch. Usually, information gain or information gain rate is used as the selection criterion. Here, the model needs to select different features (appearance of different words) to best construct a tree.

B. The SVM Classifier

SVM is a classifier that finds a hyperplane in space to separate samples. It only focuses on support vectors near the boundary line. It usually uses a criterion called "classification interval" to evaluate the classification. When we encounter the problem of linear inseparability, a kernel function is used to transform the space from the original linear space to a higher dimensional space.

C. The Logistic Regression Classifier

Logistic Regression aims to find a function that maps the value predicted actual by the linear function into interval of [0,1]. Often, sigmoid function is chosen to generate the prediction function:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \tag{5}$$

Where θ represents the parameter vector, and $h_\theta(x)$ indicates the probability to be the positive class for a input x.

3 Experiment

A. Results

Tab.1 the experiment results

Classifier	Recall	Precision	Accuracy	F1-score
Na ïve Bayes	63.8%	96.7%	63.0%	76.9%
Decision Tree	64.8%	97.6%	64.7%	77.9%
SVM	71.3%	80.7%	67.1%	75.7%
Logistic Regression	72.4%	78.4%	67.2%	75.3%

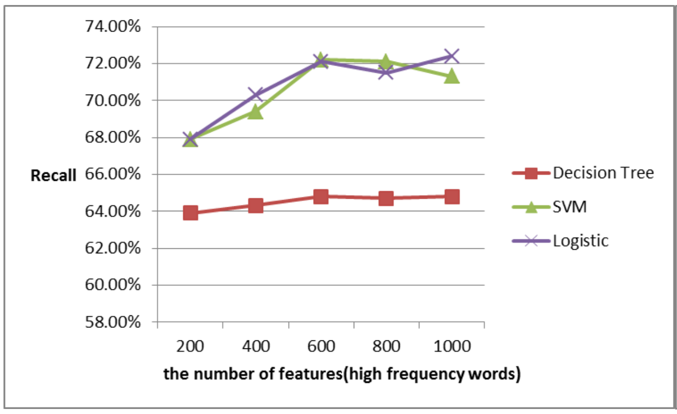
From the results above, precision is quite high while the recall is much lower, for the Na ïve Bayes Classifier. This may be the result of dominance of probabilities $\mathrm{p}(c_1)$ and $\mathrm{p}(c_2)$, which are much larger than the conditional probability $p(d|c)$. As the probability of positive class $\mathrm{p}(c_1)$ is larger than $\mathrm{p}(c_2)$, the classifier tends to predict the stock price to go up . Therefore, to avoid this bias, smoothing for class probability is preferred. In expressions:

$$\mathrm{p}(c_i) = \frac{p(c_i)+\alpha}{1+\alpha * M} \tag{6}$$

where M is the number of classes and α is manually assigned.
After this processing, the F1-score of Na ïve Bayes classifier improved. But the ability to predict the negative class is still unsatisfying.

B. Classifiers comparison

The number of features may influence the performance of classifiers. So different count of high frequency words were tested for the 3 classifiers in *nltk* and their performance are compared. After removing stopwords, there are only 1553 types in our corpus, so the top 200,400,600,800 and 1000 high frequency were tested and the results are below.



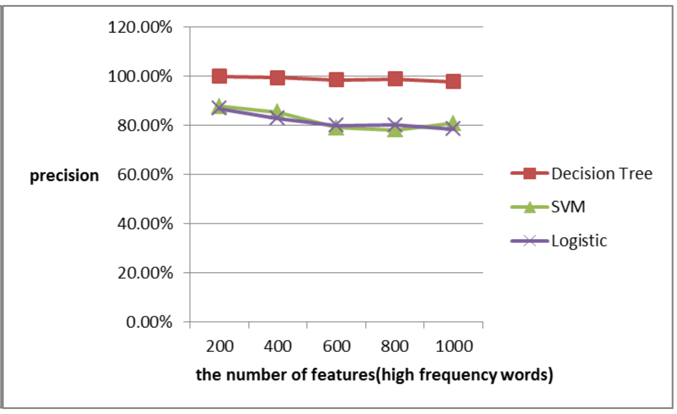
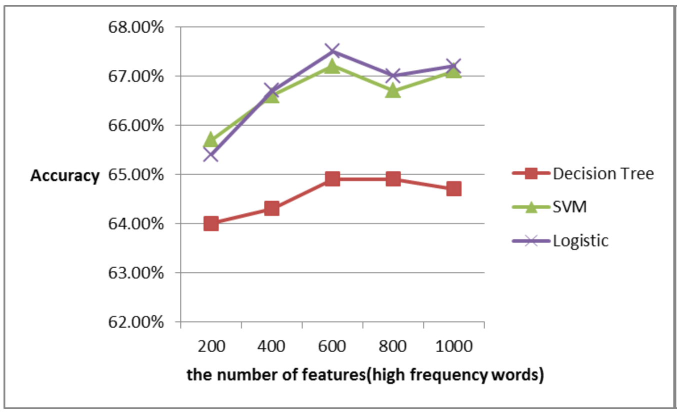


Fig.1 Recall of Classifiers

Fig.2 Precision of Classifiers



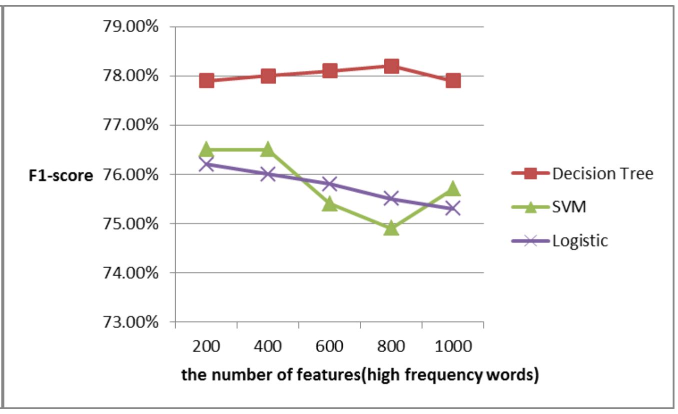


Fig.3 Accuracy of Classifiers

Fig.4 F1-scores of Classifiers

In terms of classifiers, the results show that the Decision Tree tends to predict the stock price to go up, of which the precision is quite high but the recall is much lower. SVM and Logistic classifiers have similar performance, whose ability to deal with the unbalanced data is better.

When it comes to the number of features, 600 features seem to be a good choice for decision tree, while SVM and Logistic classifiers seem to perform better on fewer features, if we focus on F1-score.

4 Conclusion

In this report, Na ïve Bayes classifier was constructed and other three classifiers in *nltk* module were tested. The Na ïve Bayes classifier tends to predict the stock price to go up, so the prediction for the negative class is not that ideal. For the three classifiers in *nltk* module, different numbers of high frequency words were tested and compared, and the results show that the decision tree performed differently in discriminating ability between positive and negative classes. In comparison, the other two classifiers are more comprehensive.

There still remains much work to be done. Naive Bayesian algorithm can not guarantee the same results, because there are different smoothing algorithms. For classification of imbalanced samples, it’s necessary to use some skills to make sure that the classifier won’t bias towards a particular category. Parameters of smoothing and feature selection are critical to the performance of the classifiers.