**Credit Rating**

Yuwen Jin 10455173

1. **Motivation**

Credit risk is a risk that occurs in bank loans or investment bonds. It’s the risk of default by borrowers, and may cause economic losses, increasing company financing costs and so on. In this case, providing appropriate credit ratings and detecting credit changes become important.

1. **Data** 
   1. **Data collection**

In this project, I use the file professor provided. Basically it’s a credit rating data set about some companies under different time points. In order to detect credit rating change, I create a new column determines when the credit rating is going to change and whether the credit rating has just changed.

* 1. **Data process**

Missing value: I write a function to detect and show characteristics of missing values. If the distribution of missing value is similar to the target variable(difference in the binary distribution is less than 5%), I think it’s missing not at random. If data is missing completely at random or at least at random, I fill the blank with mean value for quantitative columns and with ‘missing’ for qualitative columns. If it’s missing not at random, I create new labels/number for the blank because there must be some hidden information. In this project, there’s no missing value detected.

Data type: I write a function to classify qualitative and quantitative features. In this data set, all useful predictors(columns except date and information about company names/keys) are quantitative.

Feature selection: In order to simplify modeling process and avoid overfitting, I try PCA(Principal Component Analysis) and RFE(Recursive feature elimination) with logistic regression to reduce features. Also Random forest comes with its own future selection functionality. PCA projects each data point onto only the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible. And PFE uses a machine learning model for multiple rounds of training. After each round the features corresponding to multiple weight coefficients are eliminated, and the next round of training is performed based on the new feature set.

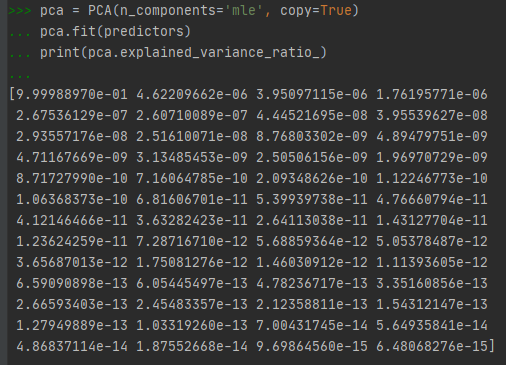
1. **Model**
   1. **Model selection**

The baseline model I use in this project is Multi-class learning with Logistic Regression. This strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes.

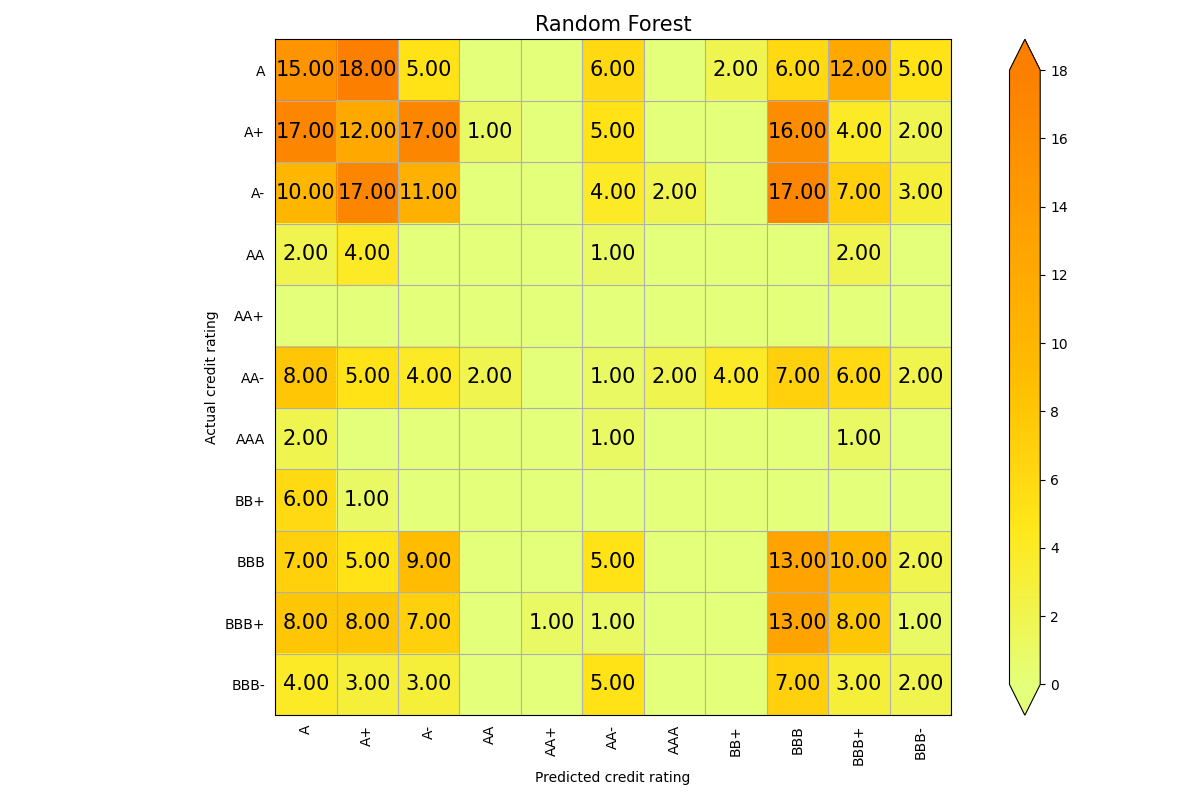
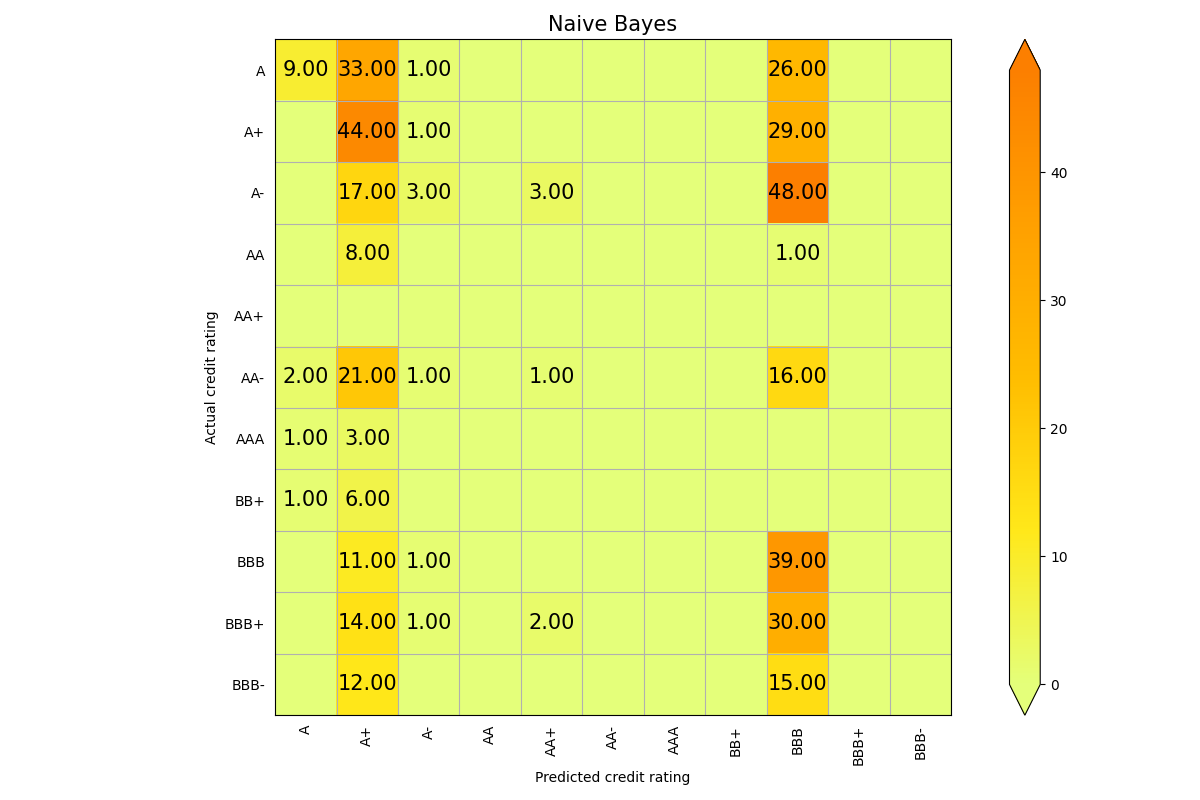
Other models I use in this project are Random Forest and Naive Bayes. Random Forest can process data of high dimension (with many features) without making feature selection. Naive Bayes has stable classification efficiency, and can handle multiple classification tasks as well.

1. **Result and Analysis**
   1. **Feature selection**

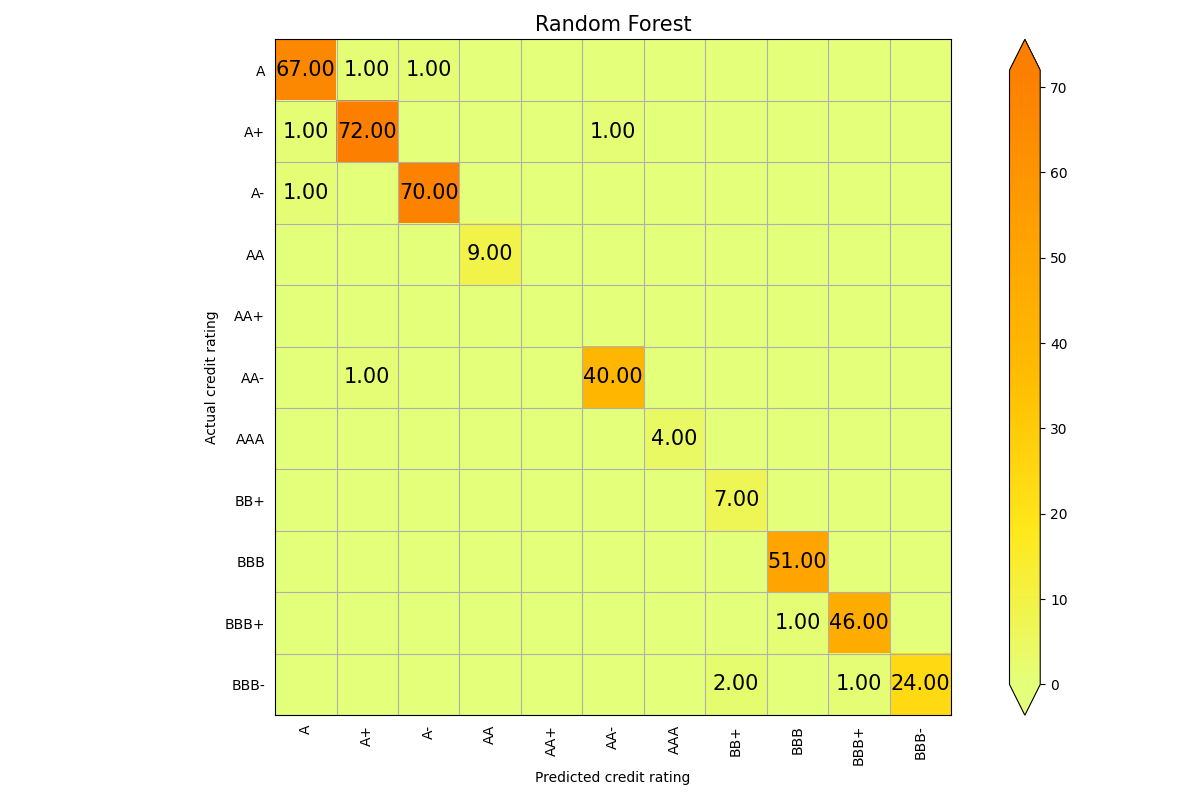
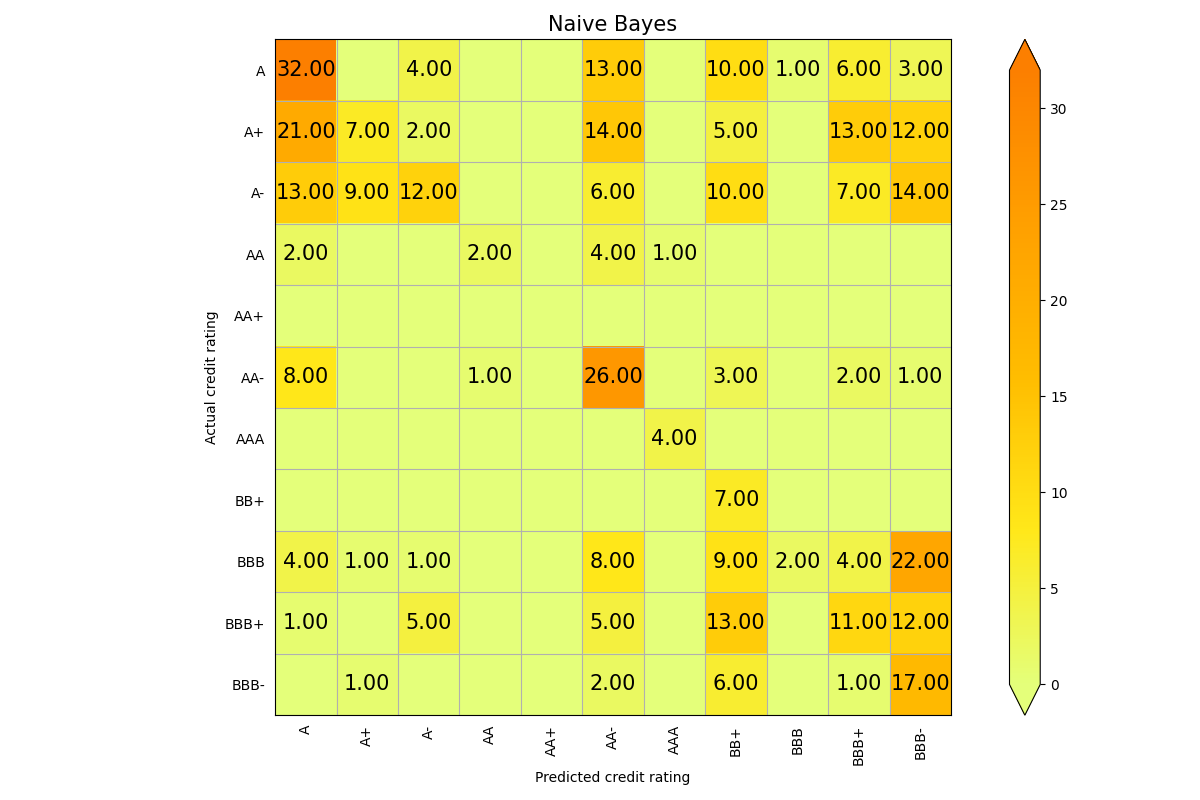
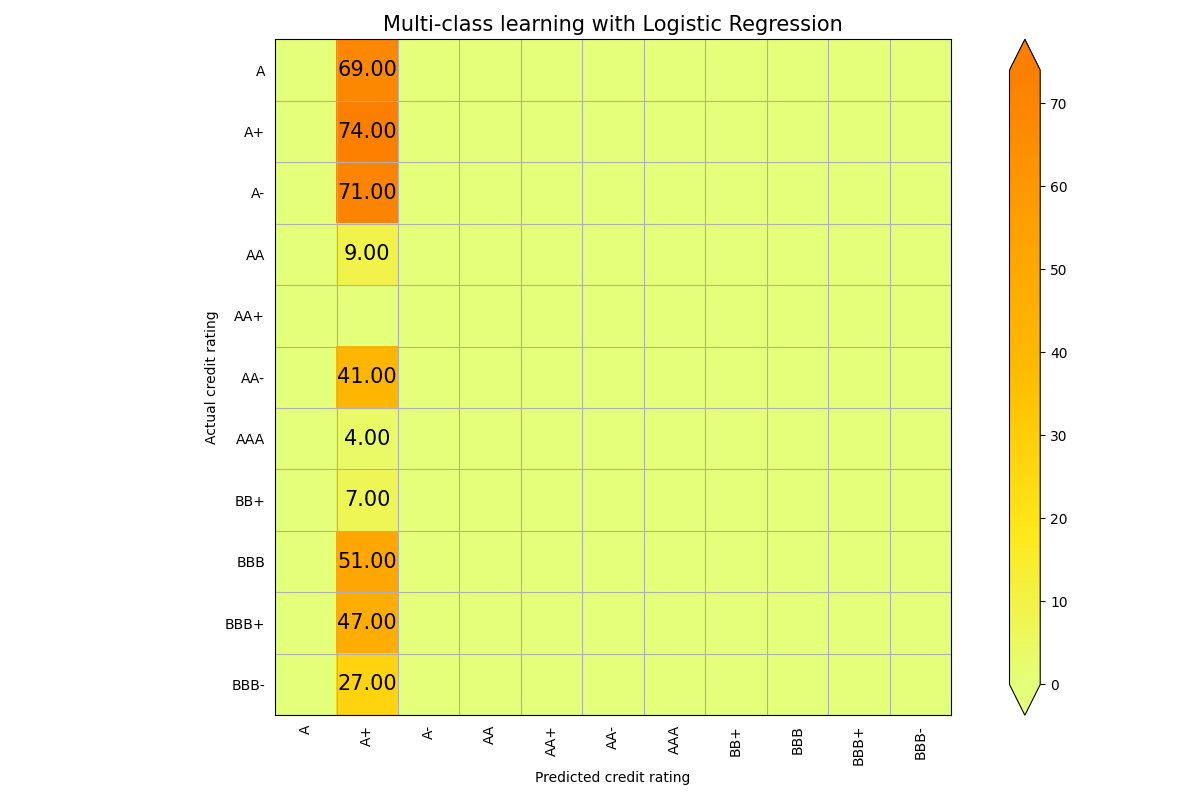
Applying PCA on this data set, the results shows that only one component can include 99.9% information. It saves a lot of time fitting models with transformed data.



However, when I do so, accuracy on both Naive Bayes and Random Forest are low around 20%, and the confusion matrix looks not good.



Then I tried PRE. It keeps 26 features out of 53, and below is confusion matrix of 3 machine learning methods, with each model’s name as title.



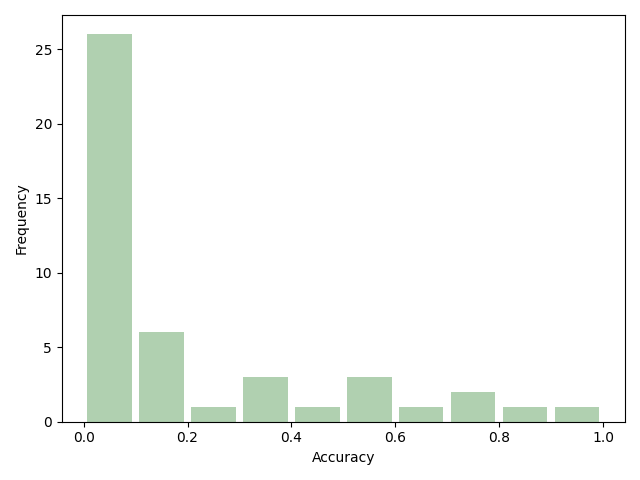
Also to ensure that future selection doesn’t influence the prediction too much, I fit model for another time using all predictors. All the 3confusion matrix when I fit models with the complete data data are exactly the same as the 3 plots above(with PRE).

Therefore, appropriate feature selection method, which is PRE for example in this project, does help to save time and reduce computational complexity.

* 1. **Credit rating**

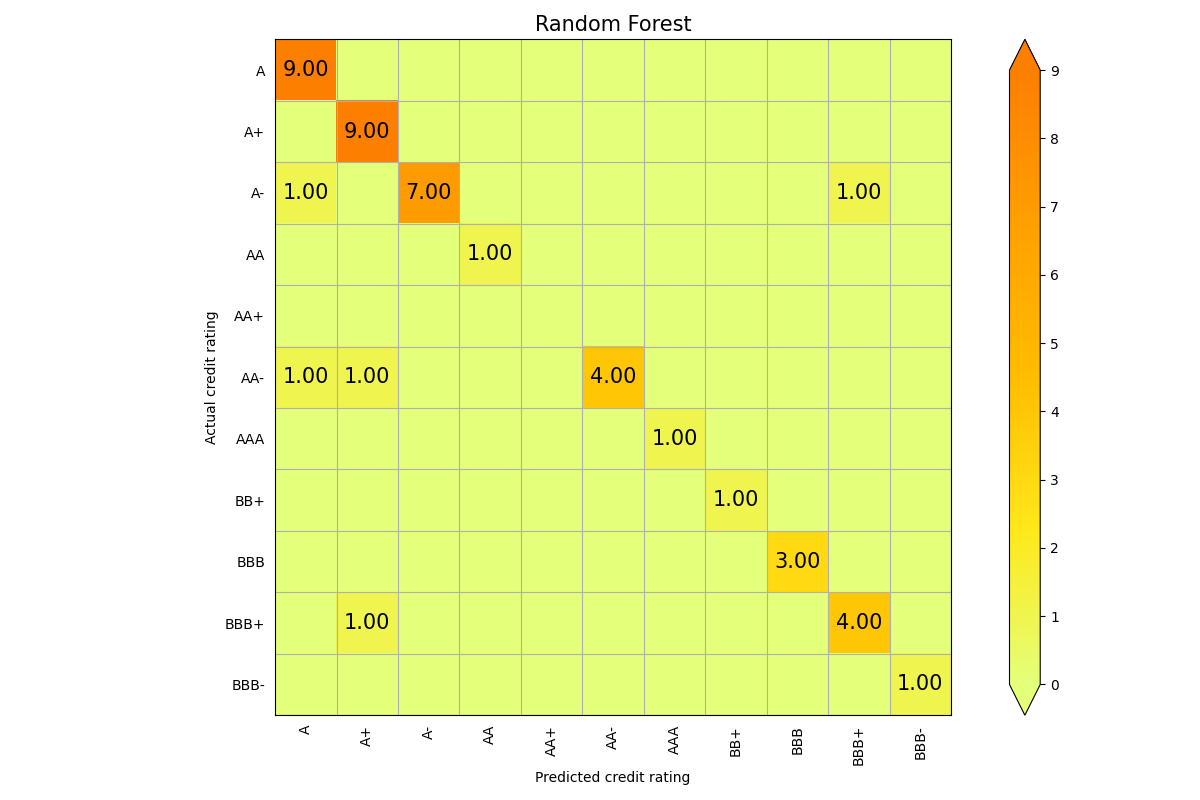
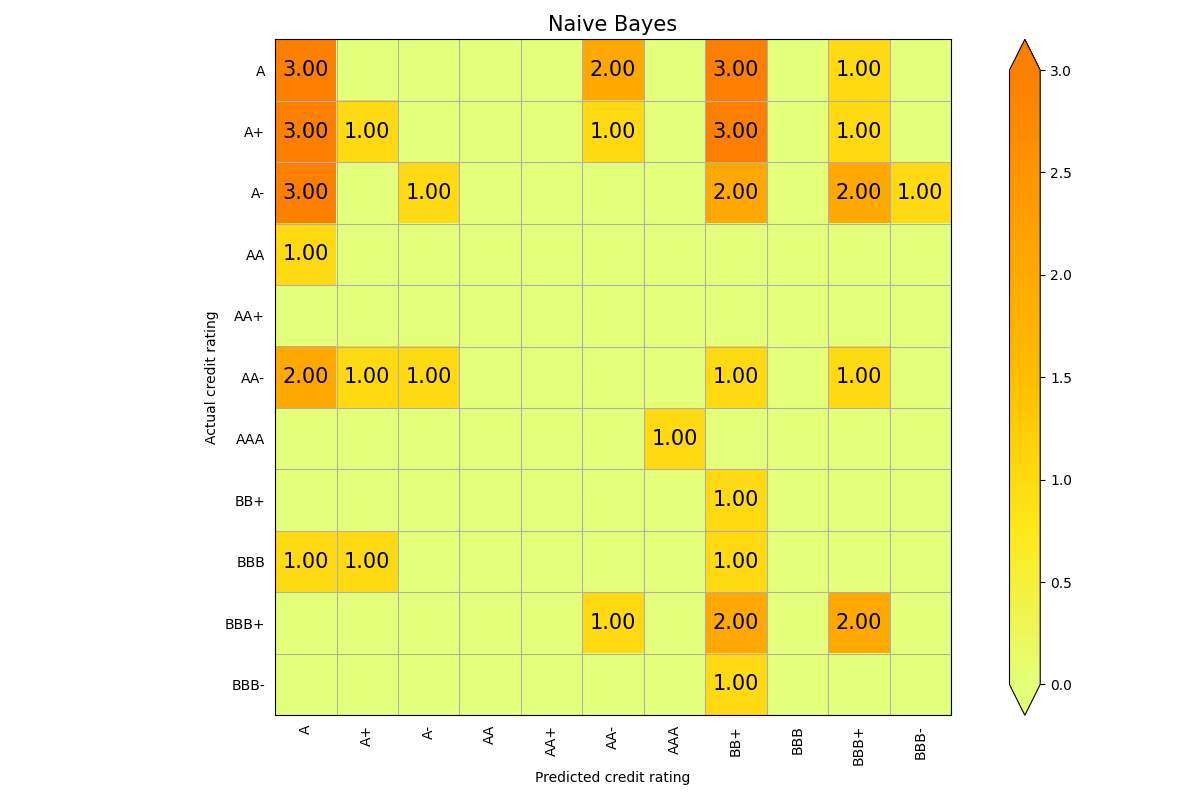
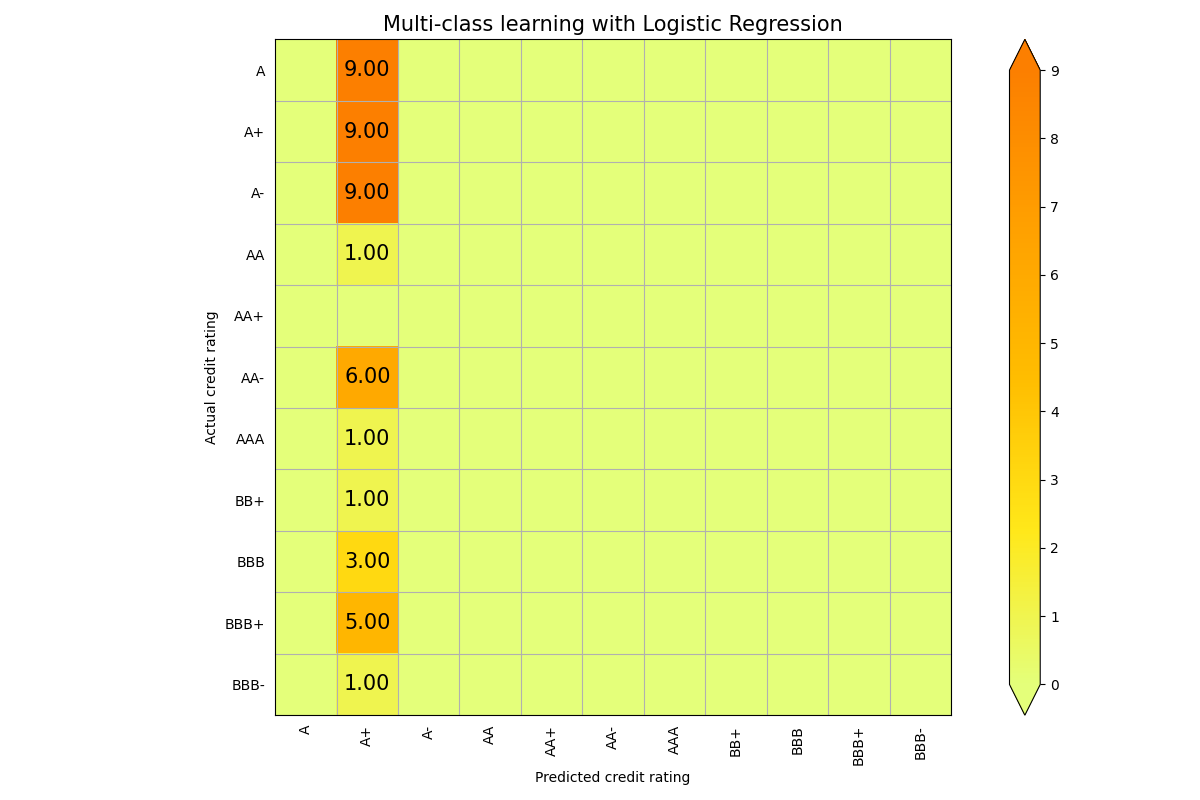
At the beginning, I randomly select some data from the whole data set as training and testing. When I use my baseline method, the predictions are totally useless. Naive Bayes doesn’t work well either(with 29.5% accuracy), where as Random Forest give me an incredibly high accuracy. (Confusion matrix are shown in 4.1)

Then I realize that there may be some correlation between a company's data at different recording times, and credit ratings tend not to change very frequently or dramatically. It may be improper to split data sets into training and testing in this way and say a model must be good or not on this kind of data set, since we are very likely to extrapolate the current credit rating combining the records of previous and subsequent records of the same company. So I did another test. Because there are only 45 companies in this data set, in case sample space is not large enough, I imitate the cross validation method, run a loop for 45 times and every time rate one company based on other companies’ information using Random Forest, which preforms the best at previous test. This time the accuracy becomes really terrible. Mean accuracy is only around 20% with the same machine learning method. This is a histogram showing accuracy distribution in all attempts.



Then I have reason to say that a company’s historical credit rating records helps a lot in credit rating. at least for this data set, this group of features, it is so. And in this data set, the correlation between the credit ratings of different companies may not be as strong as imagined. If we have more companies’ records, maybe things will change.

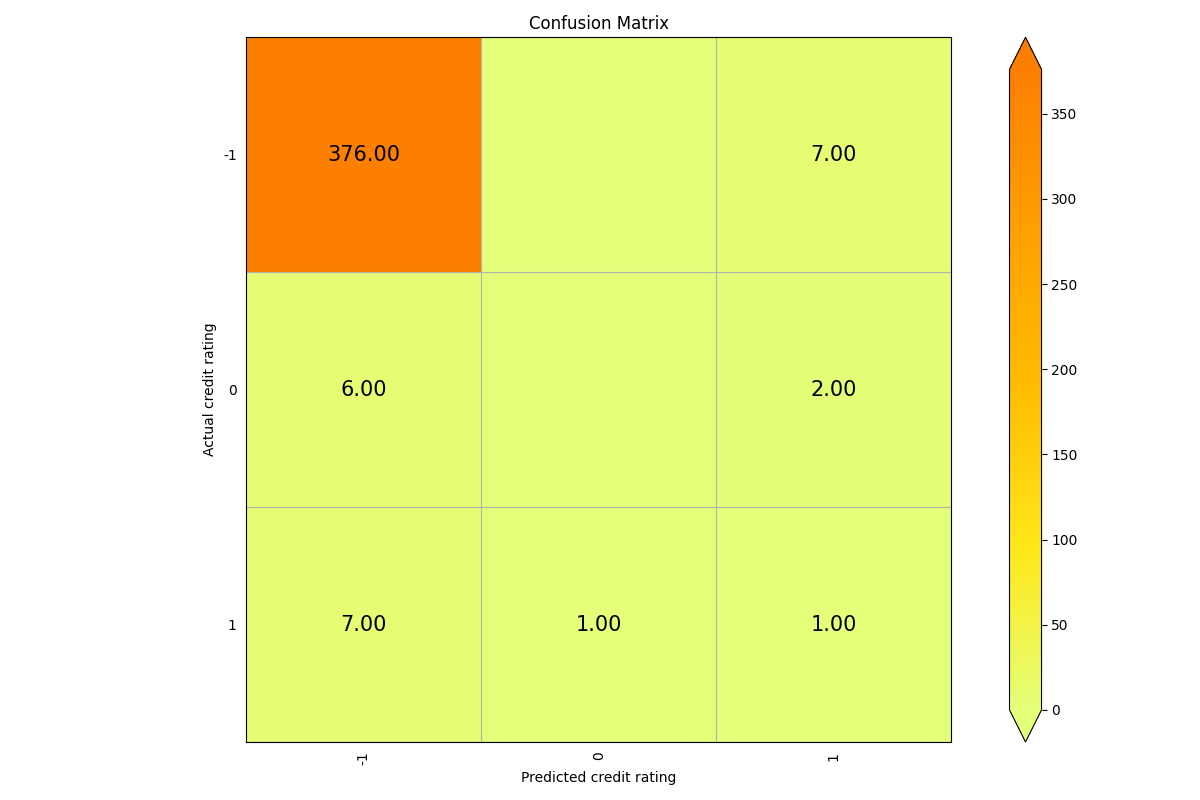
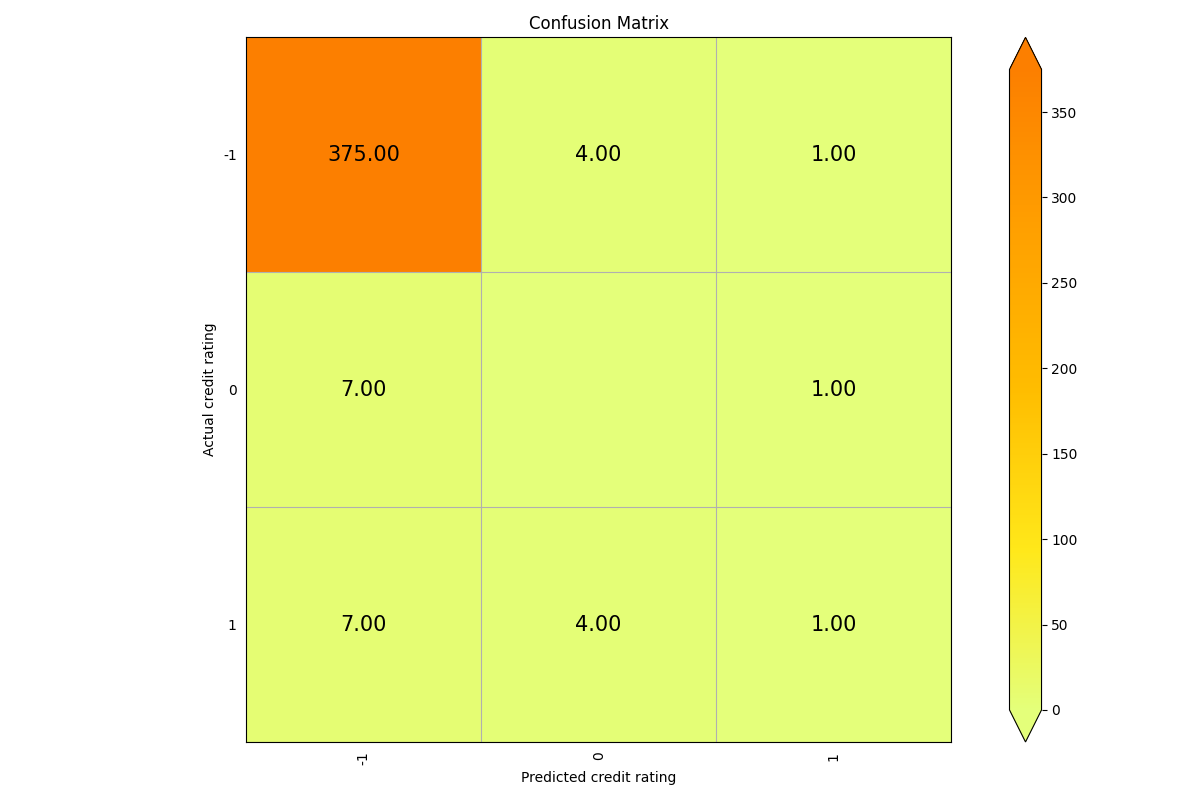
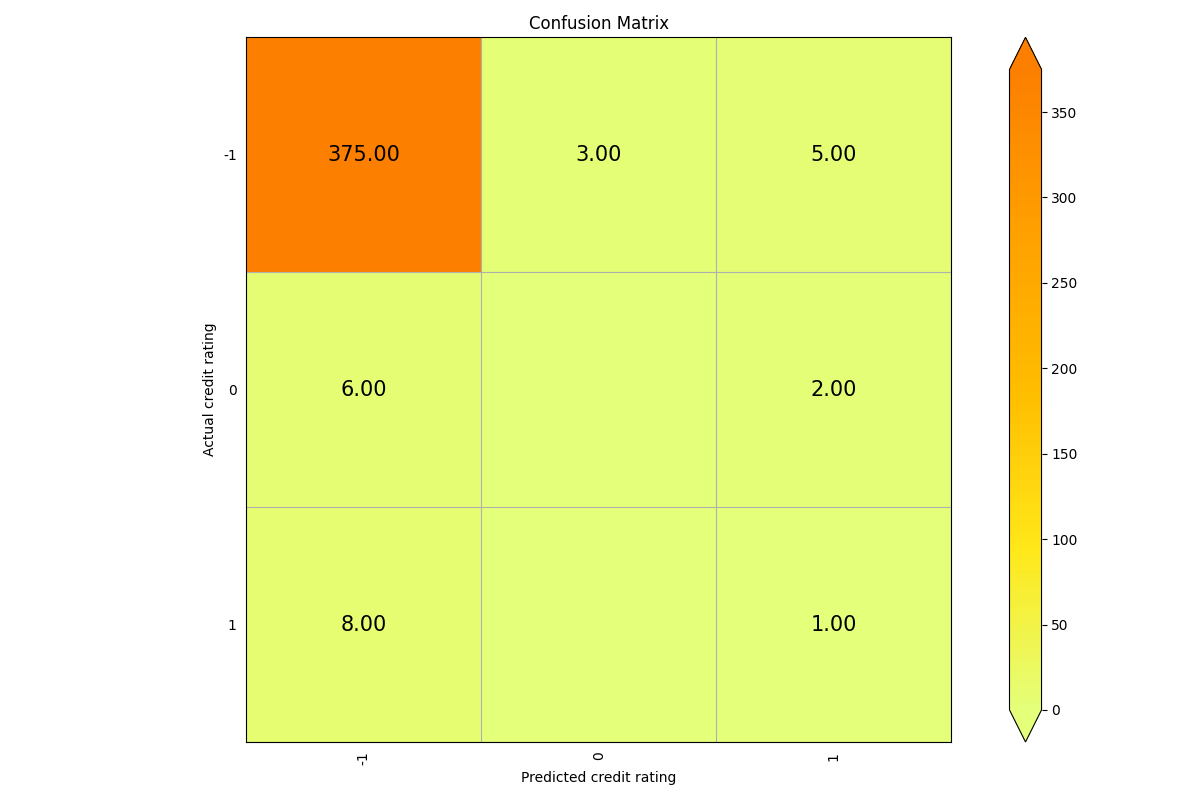
Then, because we care more about ‘now’ than the past, I use all companies’ most recent records as testing data to fit the model and predict again. This time Random Forest gives accuracy of 88.89%. Not as high as the first time, when we randomly split the data, but is not low either. And it makes more sense than before because we don’t use future information to predict ‘now’. My baseline and Naive Bayes both gives unreliable predictions.



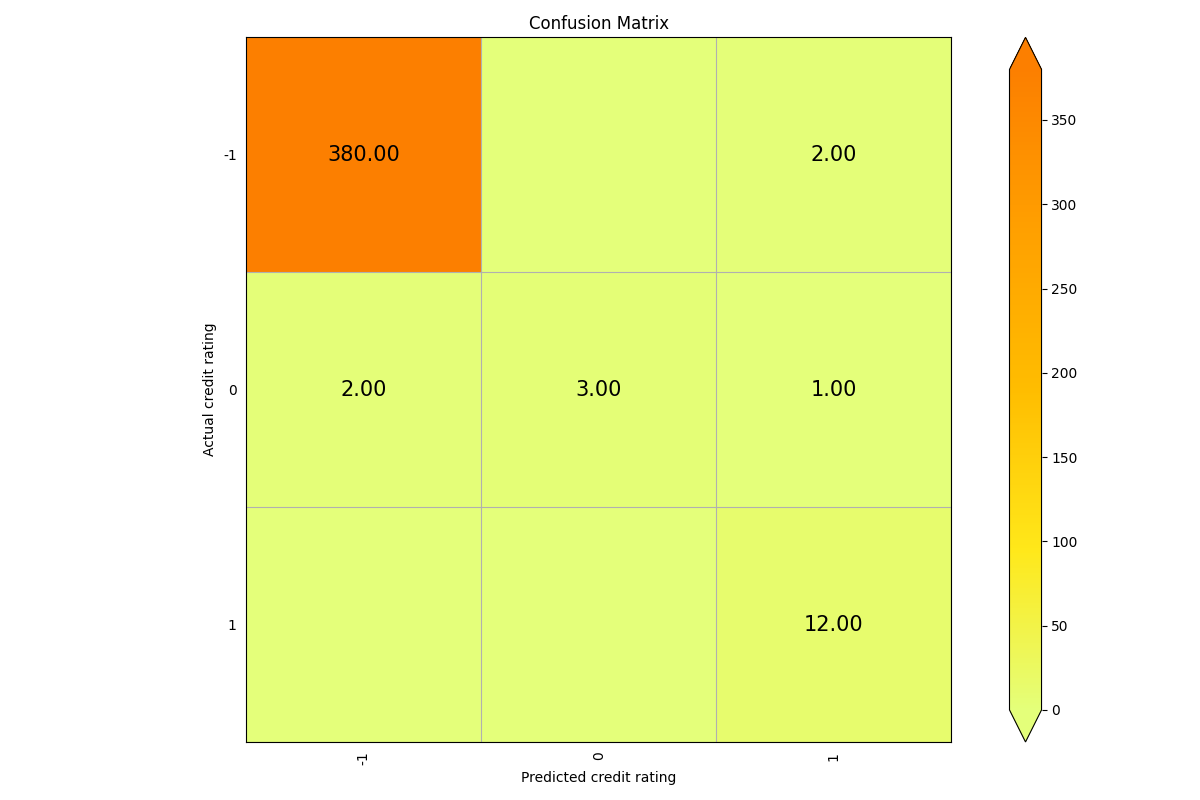
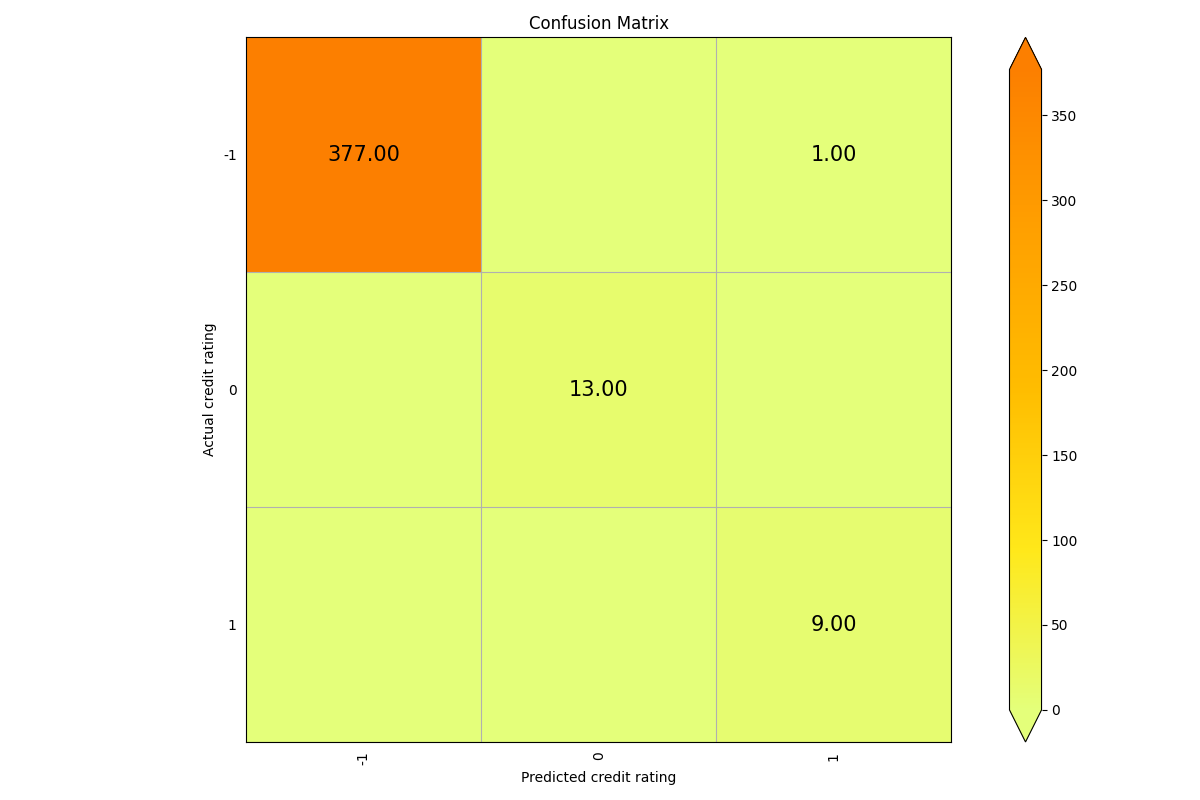
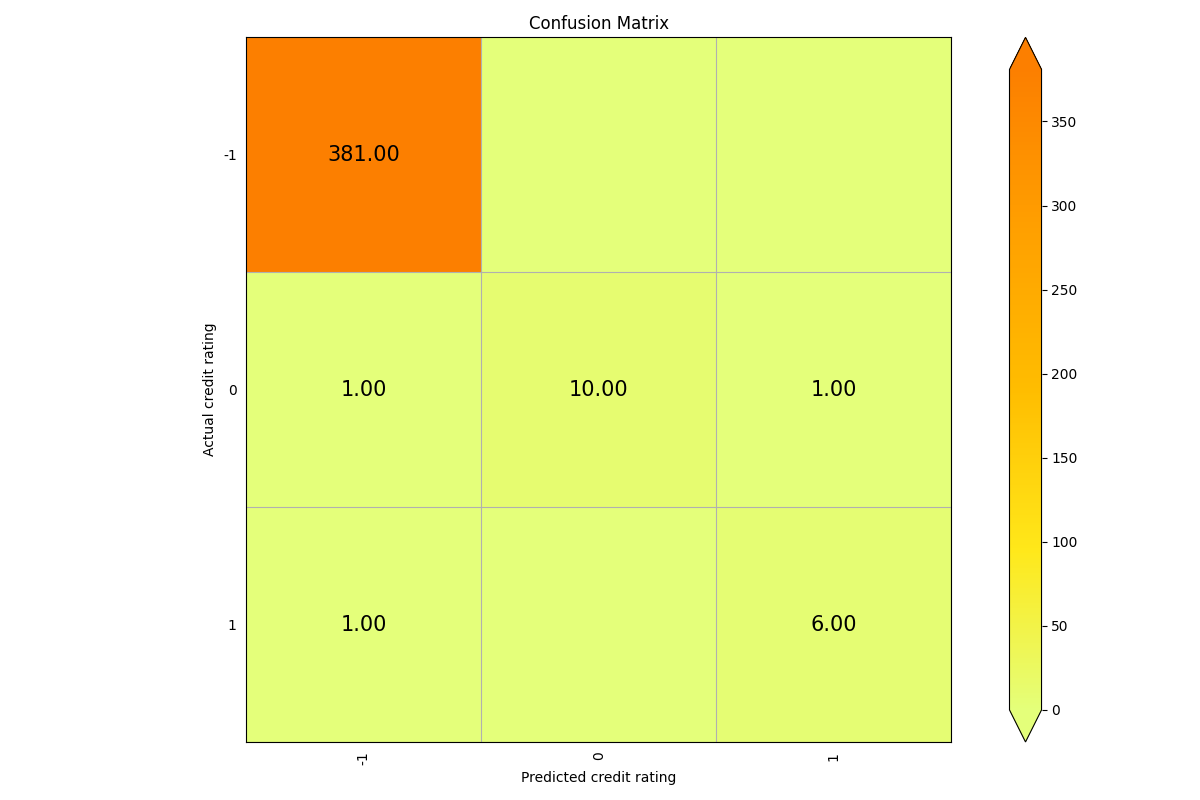
* 1. **Credit change detect**

Except for credit rating itself, we also care about it change. If we know that a company’s credit is going to change, we can make corresponding measures in advance. Therefore, I add a new column in original data set. 0 stands for ‘everything is fine’, 1 means the rating is going to change, and 2 means it just changed in that time point of recording.

First I use Random Forest and randomly split and train and test for 100 times. Though the mean accuracy is higher than 97%, the confusion matrix looks not that good.



The high accuracy is just because most records appear to be 0, and the classifier set most predictions to 0. So I try Naive Bayes instead. This time the mean accuracy of 100 attempts are even higher, it’s 99.5%. And the confusion matrix is much more reasonable.



Different prediction targets require different models. Then for credit rating, Random Forest works the best among the 3, while in credit change detecting, Naive Bayes becomes more reliable. Multi-class learning with Logistic Regression did not perform well in this project.

1. **Future work**
   1. Get familiar with Google Colab and try more future selection and machine learning methods such as Gaussian Process, which is taking forever running on my own CPU
   2. Try the same methods on similar data(recent information of the same companies, the same features, or find more companies’ records) to expand the project.
   3. Fine tuning on models

**Appendix**

I uploaded my source code for this project to github:

<https://github.com/StarryYJ/690-credit-rating>