Project Report

Default of Credit Card Clients

Yi Wang

Banner ID: B01629467

Brown ID: 140304323

Data Science Initiative at Brown University

Github repository: https://github.com/yahowang/Data1030Project/tree/master

Introduction

Because of the advance of financial crediting system, customers are able to make an expense with their credibility and pay off in the future. Yet, it is challenging yet vital for banks to determine whether their customers would have the ability to pay off their credits to minimize the risk of capital loss. Therefore, one of the ways of doing so is to track historical transactions and payments to target potential customers who will not meet payoff requirements. These customers will be classified and will have a greater chance of default in the shortcoming future.

In this project, the dataset was obtained from banks in Taiwan and was meant to target the case of customers default payments. This is a binary classification problem and the target valuable is *whether a customer will default or not* based on different features and historical payments. This project is interesting and meaningful because it will help banks to target potential default payments such that they may do some actions to prevent capital loss.

Data Descriptions

Number of features: 24

Number of instances: 30000

No Missing data

Feature Name	Domain
X1: Amount of the given credit	Real numbers
X2: Sex	1 = male; 2 = female
X3: Education	1 = graduate school; 2 = university; 3 = high school; 0,4,5,6 = others
X4: Marriage	1 = married; 2 = single; 3 = divorce; 0 = others
X5: Age	Integer
X6: September payment delay status	-2 = No consumption -1 = paid in full 0 = paid minimum amount 1 = delay for one month 2 = delay for two months 3 = delay for three months 4 = delay for four months 5 = delay for five months 6 = delay for six months 7 = delay for seven months 8 = delay for eight months 9 = delay for nine months and above
X7: August payment delay status	
X8: July payment delay status	
X9: June payment delay status	
X10: May payment delay status	
X11: April payment delay status	
X12: September bill statement amount	Real number
X13: August bill statement amount	
X14: July bill statement amount	
X15: June bill statement amount	
X16: May bill statement amount	
X17: April bill statement amount	
X18: September payment amount	
X19: August payment amount	
X20: July payment amount	
X21: June payment amount	
X22: May payment amount	
X23: April payment amount	
X24: Default or not (Response Var.)	1 = Yes; 0 = No

Figure 1: Feature Descriptions

EDA

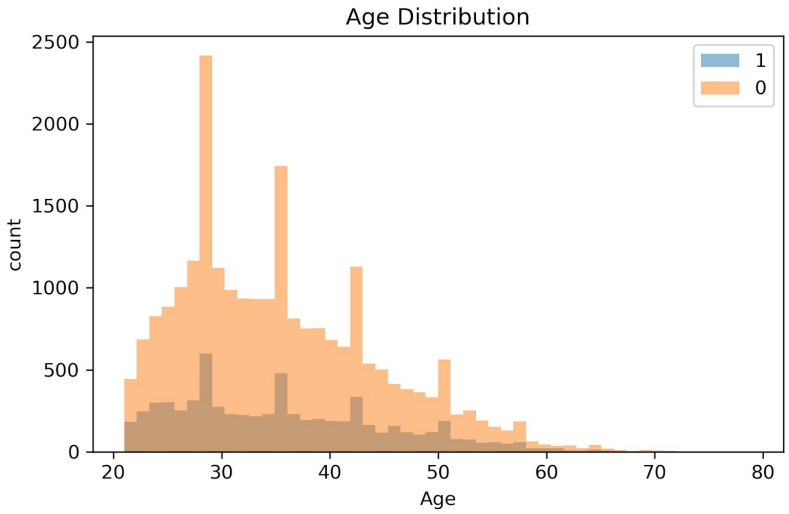

Figure 2: Age distribution between the default and not default payment groups

There is no significant difference between default payment status and age. However, age is right-skewed under each group, meaning that most people are young to middle-aged in the dataset.

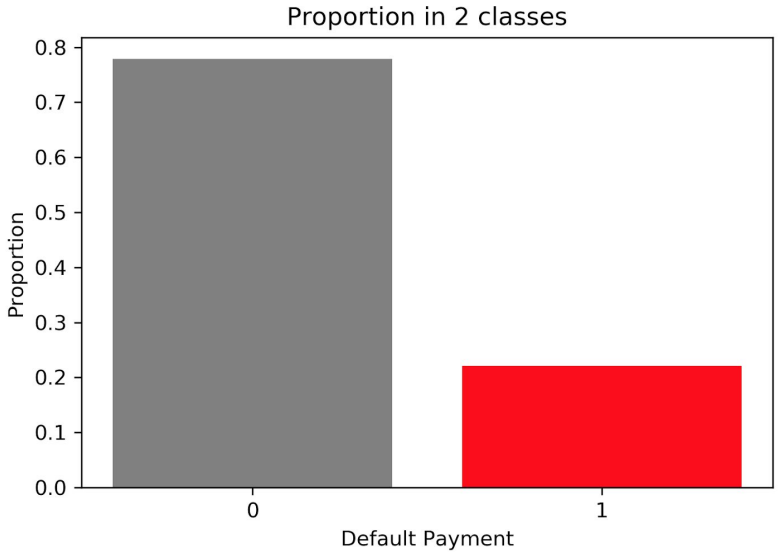

Figure 3: Class proportions

The data set is imbalanced since the non-default class is very huge yet only 22% is default. This suggests that we should carefully choose our metric when tuning models.

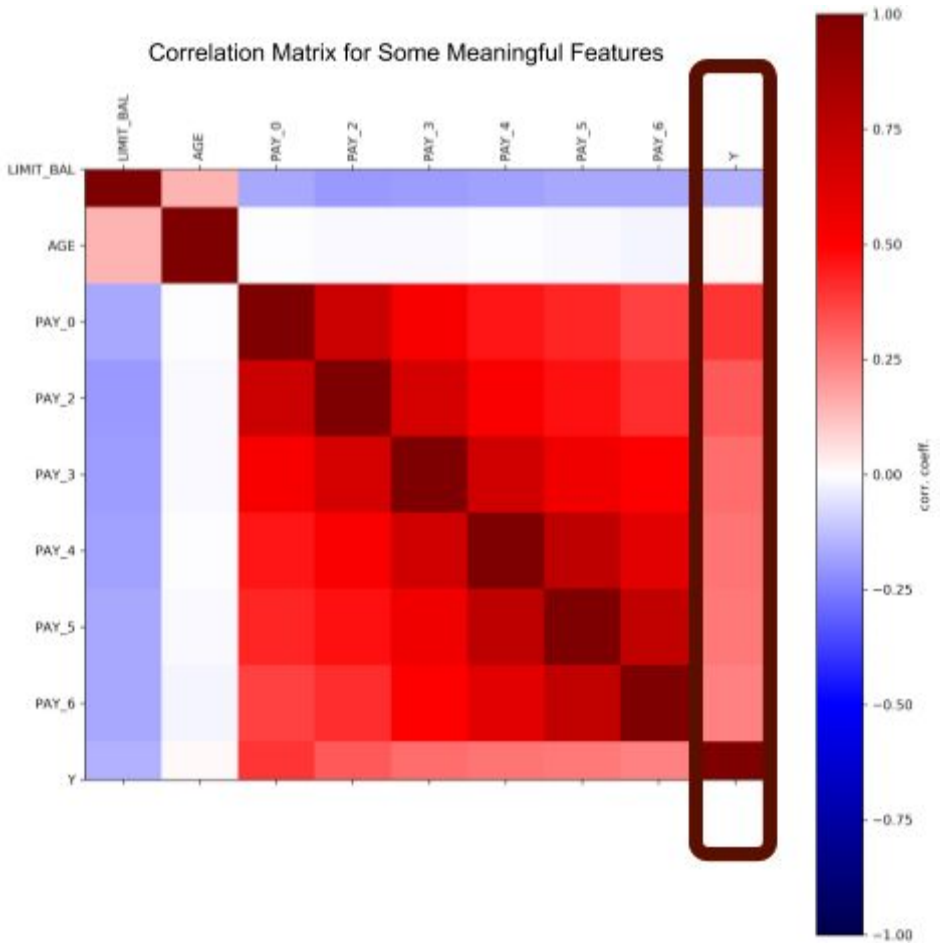

Figure 4: Correlation matrix for Feature X5 - X11 and F24

The correlation matrix suggests that more given credit (limited balance) leads to less chance of default. It also suggests that the payment delay status in that most recent month matters to determine whether a person will default.

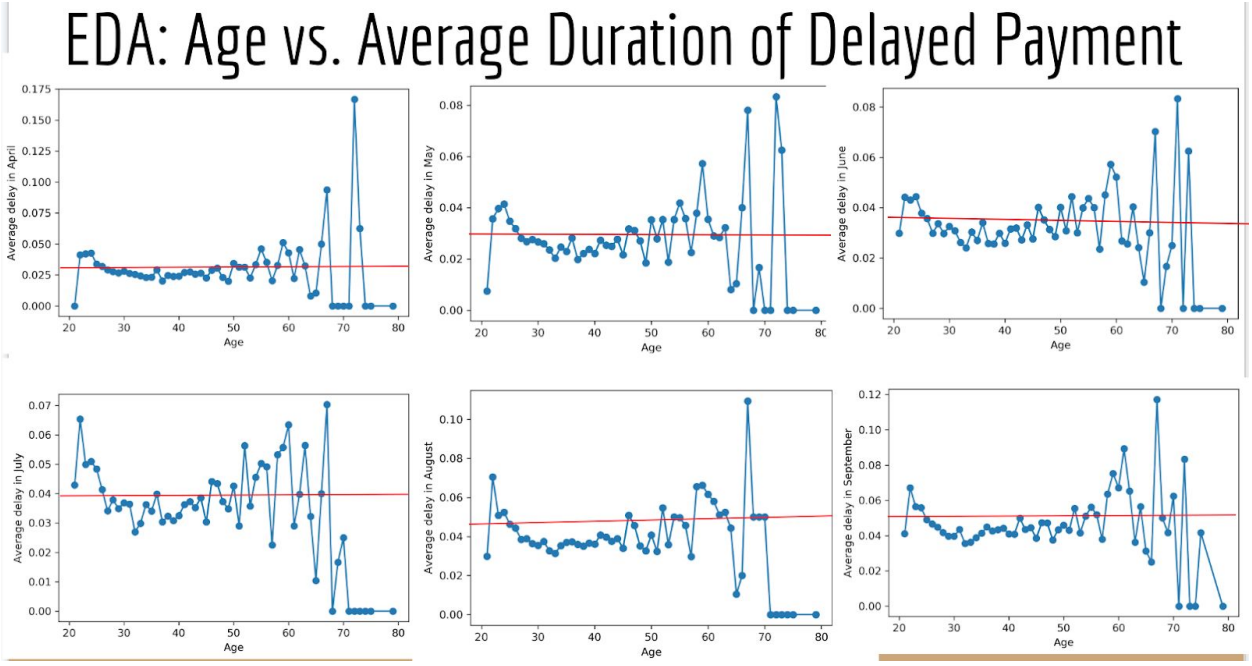

Figure 5: Age and average duration of delayed payment in each month

In each month, younger people tend to have smaller fluctuations on duration of delayed payment yet older people tend to be more volatile. This is probably because the age is right skewed such that the variance is small. It can also be the case that older people died such that they would not be able to pay off their balance.

Methods

Preprocessing:

- For X1 and X5, I used MinMaxScaler.
- For X2 - X4, I used OneHot Encoder.
- Special*: For X6 - X11, Each feature contains both categorical and ordinal values, thus, I first extracted the categorical values by OneHot and then set them to a default value in the original columns. Then I treated the original columns as MinMaxScaler to preserve the linear-ordinal information.
- For X12 - X23, I used Standard Scaler.
- For X24, I changed the name of the feature to Y.

- There are 86 columns after preprocessing.

Each model uses stratified k-folds to produce the best model, where k = 5. To take account of the imbalancement, the idea is to use F1 score as a metric for evaluating models. Thus, F1 score is used to evaluate each cross validation model within each fold. I also applied accuracy score to evaluate the best model among the chosen best cross-validated models. With a mixture of F1 score and accuracy score, each modeling will be able to produce a reasonable best model that dealing with the imbalancement after tuning. Also, the uncertainties of random splitting data is measured by doing different random_state and calculate the mean and standard deviation.
Five models were being considered initially. Yet the Support Vector Classifier was not workable for this dataset as it was too large to construct a SVC model. Thus, I had four models at the end.

1) *Naive Bayes (NB)*
No tuning is available for this model. Yet, I can still use stratified k-folds to take amount for the uncertainty when making splits and using 9 different random_state to generate the best model under each random_state.

Stratified test size: 20%
Stratified train size in each fold: 64%
Stratified cross validation size in each fold: 16%
Random_state: 1, 4, 9, 16, 25, 36, 49, 64, 81
Number of folds: 5

2) *Logistic Regression (LR)*
Logistic regression is a workable model for a classification problem as we assume the features have some linearity with the response variable and we can apply sigmoid function to calculate the probability of being in class 1 (default). It can then be converted to classification prediction if we provide a threshold. I still use 9 different random_state and 5 folds to train the model. Alpha is the weight for the coefficient of each feature and is ranging from 11 numbers form 10^-5 to

10^5. The L1 penalty term is added and solver is set to “saga” to avoid overfitting. The max iteration is set to 10000 to prevent endless improvement steps of the model.

Stratified test size: 20%
Stratified train size in each fold: 64%
Stratified cross validation size in each fold: 16%
Max iteration for each model training: 10000
Random_state: 1, 4, 9, 16, 25, 36, 49, 64, 81
Alpha: logspace(-5, 5, 11)
Number of folds: 5

3) *Random Forest (RF)*
Another suitable model for this problem is random forest classifier. In this case, the parameters for tuning are max_depth and min_samples_split. I still use 9 different random_state and 5 folds to train the model. The number of inner estimators is set to 50.

Stratified test size: 20%
Stratified train size in each fold: 64%
Stratified cross validation size in each fold: 16%
Random_state: 1, 4, 9, 16, 25, 36, 49, 64, 81
Max_depth: 1 - 10
Min_samples_split: 2 - 10
Number of folds: 5

4) *K-Nearest Neighbors (KNN)*
KNN classifier is a useful classifier for this problem. The weights for each data point is set to distance such that the prediction of a point is based on weighted majority votes of its neighbors. I still use 9 different random_state and 5 folds to train the model.

Stratified test size: 20%

Stratified train size in each fold: 64%

Stratified cross validation size in each fold: 16%

Random_state: 1, 4, 9, 16, 25, 36, 49, 64, 81

N_neighbors: 100, 200, 300, 400, 500, 600, 700, 800, 900

Number of folds: 5

Results

The baseline accuracy is 0.77883 with highly imbalanced data.

Since Naive Bayes does not have anything to tune, the best scores are displayed for each random_state.

random_state	best_score
1	0.42287361845266697
4	0.40935672514619886
9	0.4138581192552439
16	0.41105025005953794
25	0.4174618320610687
36	0.4093896713615024
49	0.4132192106514503
64	0.4203303684879289
81	0.42059463379260337

The rest of the three models, we come up with the best parameters:

random_state	LR	RF	KNN
1	Alpha: 0.1	Max_depth:8 Min_samples_split: 3	N_neighbors: 200
4	Alpha: 0.001	Max_depth:8 Min_samples_split: 8	N_neighbors: 200
9	Alpha: 1	Max_depth:9 Min_samples_split: 8	N_neighbors: 200
16	Alpha: 0.001	Max_depth:9 Min_samples_split: 8	N_neighbors: 200
25	Alpha: 0.001	Max_depth:8 Min_samples_split: 8	N_neighbors: 200
36	Alpha: 0.001	Max_depth:8 Min_samples_split: 4	N_neighbors: 200
49	Alpha: 0.01	Max_depth:7 Min_samples_split: 5	N_neighbors: 200
64	Alpha: 1	Max_depth:9 Min_samples_split: 5	N_neighbors: 200
81	Alpha: 0.001	Max_depth:7 Min_samples_split: 2	N_neighbors: 200

The mean, standard deviation, and best score, best parameter(s) of each model:

	NB	LR	RF	KNN
Mean	0.41534826991868906	0.8193703703703703	0.8214259259259259	0.8025740740740741
Standard Deviation	0.004831305070268036	0.002490448144730449	0.003100289822285545 6	0.0024483694638575 654
Best Score	0.42287361845266697	0.8241666666666667	0.8271666666666667	0.807
Best Parameter(s)		Alpha: 0.001	Max_depth:8 Min_samples_split: 4	N_neighbors: 200

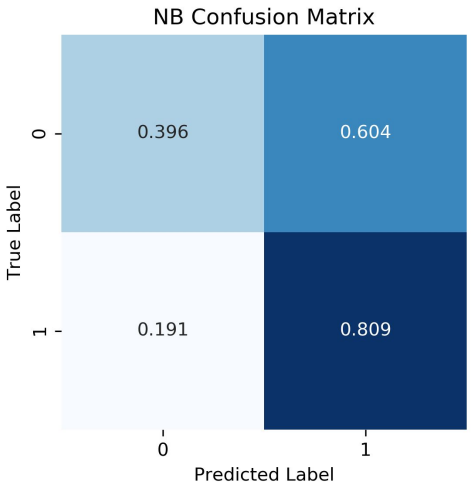

Figure 6: Confusion matrix for Naive Bayes

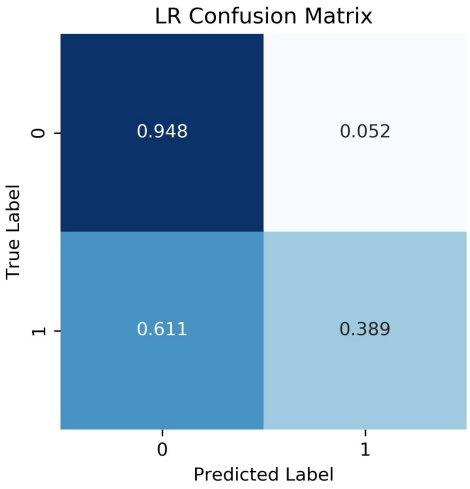

Figure 7: Confusion matrix for Logistic Regression

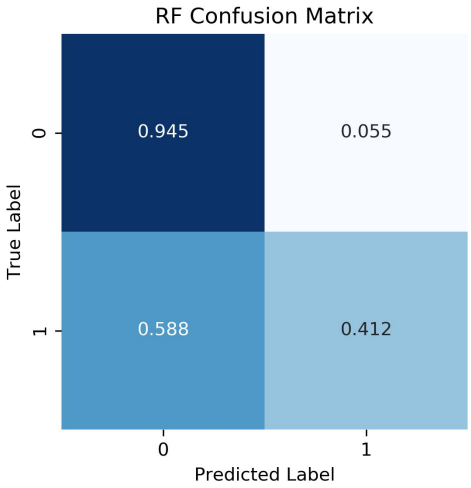

Figure 8: Confusion matrix for Random Forest

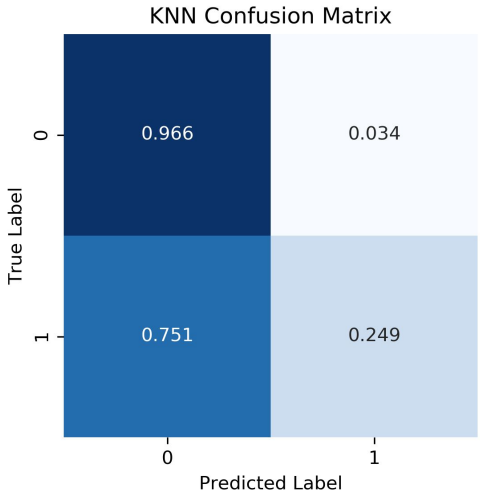

Figure 9: Confusion matrix for KNN

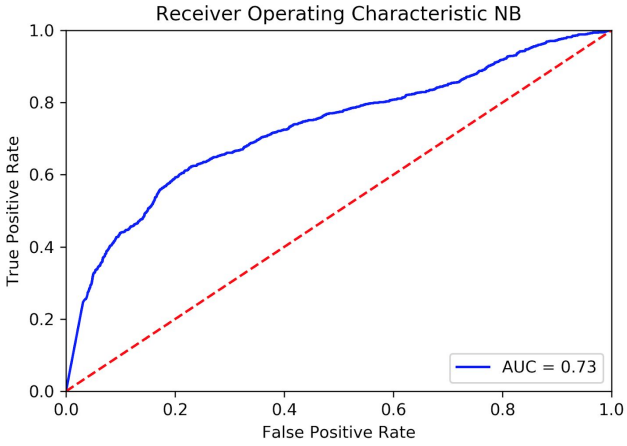

Figure 10: Receiver Operating Characteristic for NB

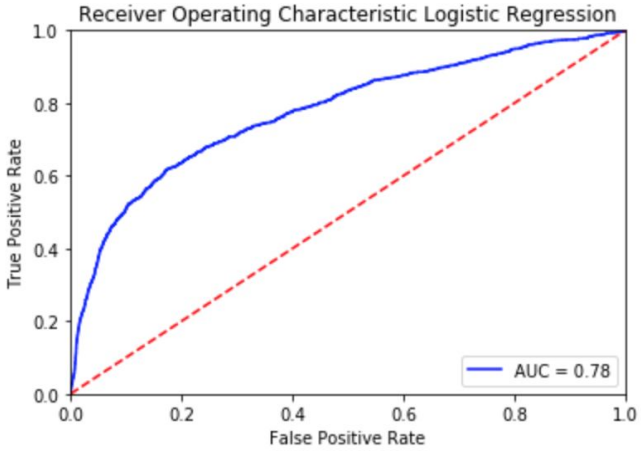

Figure 11: Receiver Operating Characteristic for LR

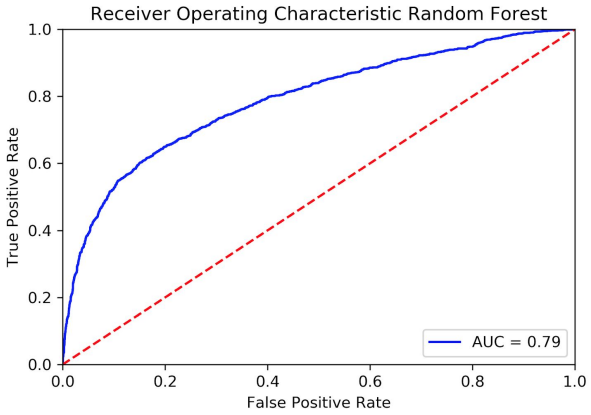

Figure 12: Receiver Operating Characteristic for RF

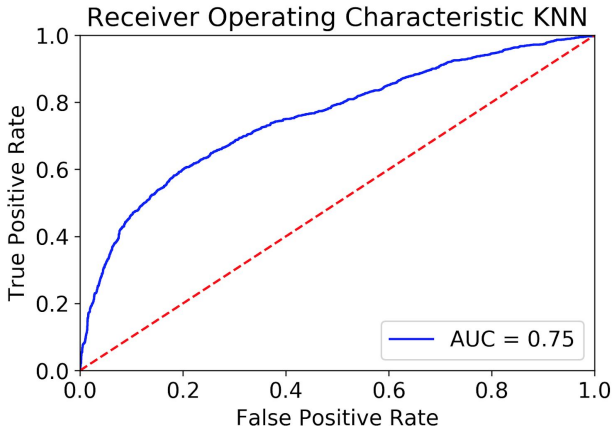

Figure 13: Receiver Operating Characteristic for KNN

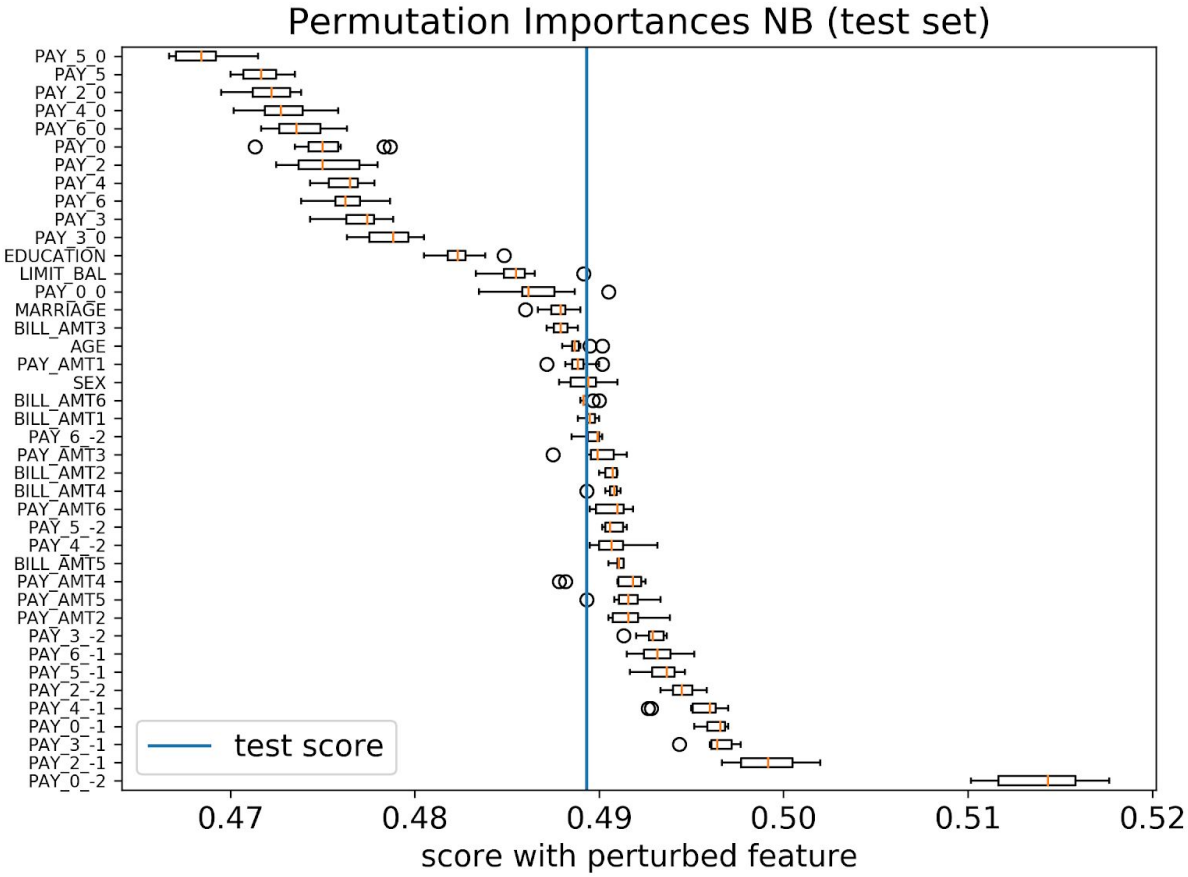

Figure 14: Permutation Importances for NB

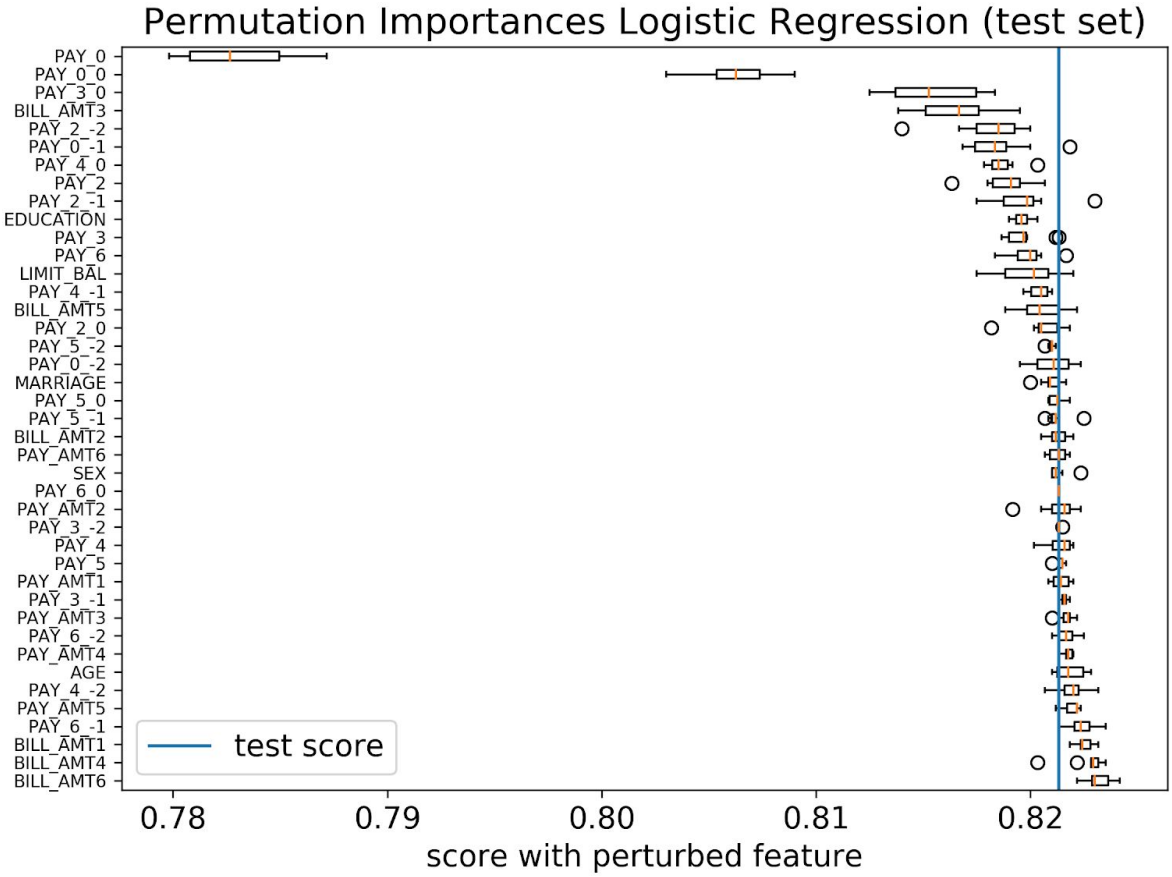

Figure 15: Permutation Importances for LR

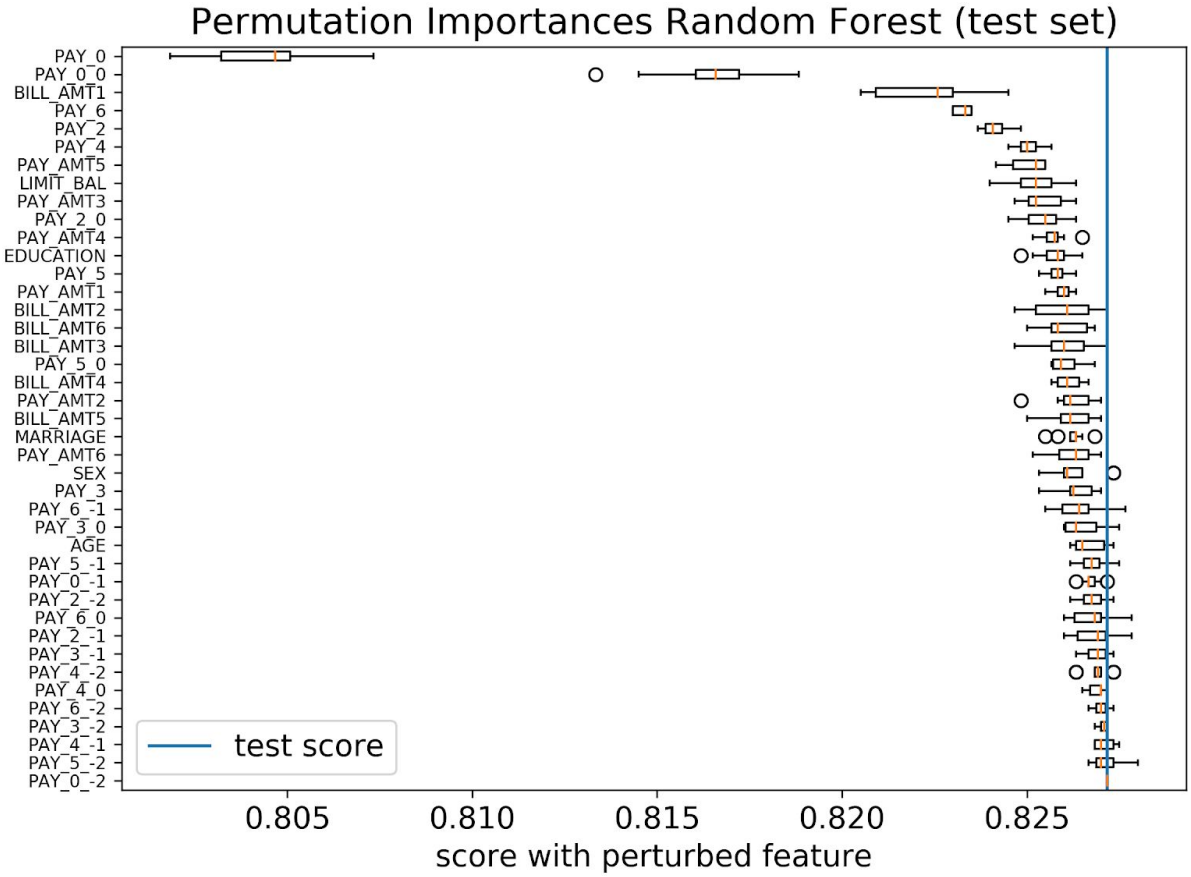

Figure 16: Permutation Importances for RF

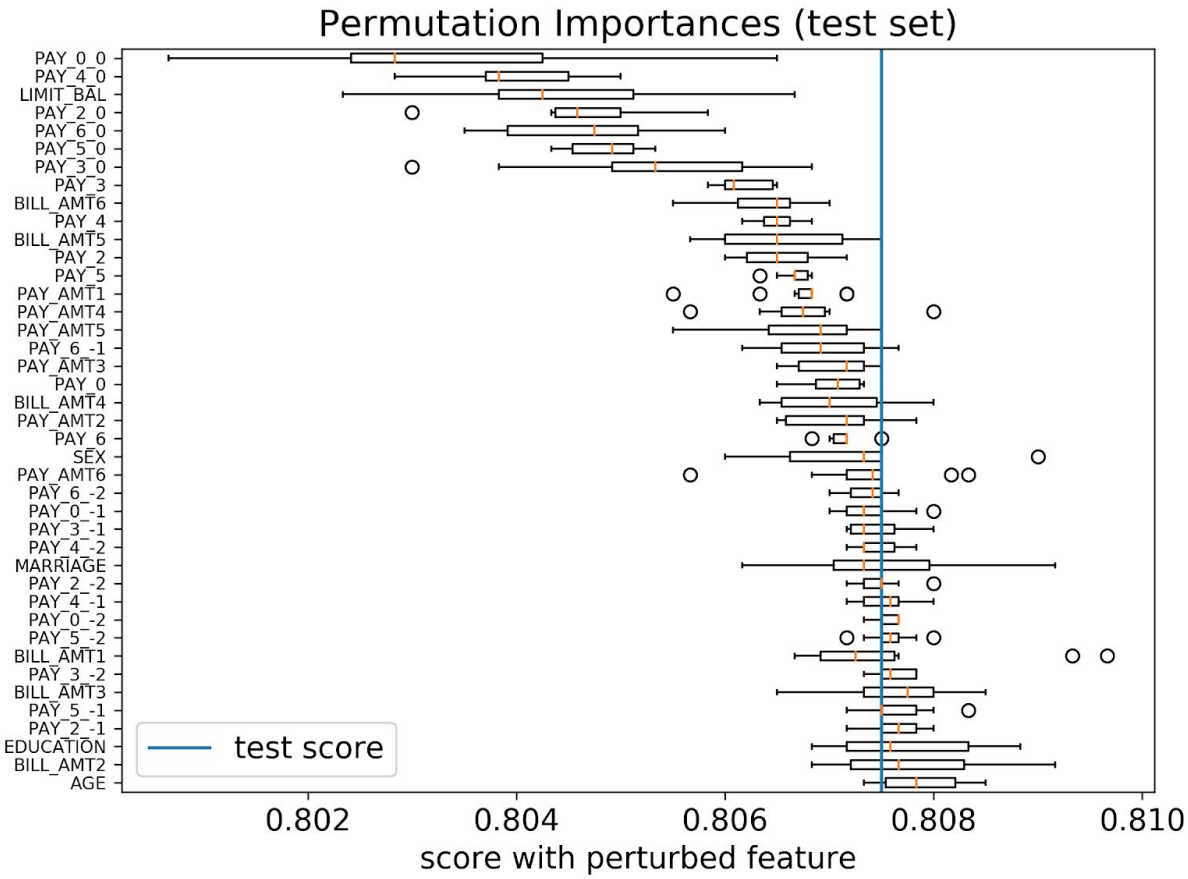

Figure 17: Permutation Importances for KNN

Analysis:

Naive Bayes

The best result for Naive Bayes is much worse than the baseline accuracy, even worse than tossing a coin. Even if we predict the other way, the accuracy is only around 58.5% and is not good enough. However, the area under the ROC is reasonably high. Moreover, the detection rate is very high for Naive Bayes because it classifies most of the points as (+1) yet the false positive rate is high as well. Many features can affect the model as we can see the huge "S" shape in Figure 14.

This is definitely not a feasible solution for the problem.

Logistic Regression

This is better than the baseline accuracy. The best test score is relatively high among different models. The precision and recall is balanced we have the false positive rate being low yet the detection rate reaches as high as it can get, although it is not very high enough because of the nature of the data set. The area under the ROC is high. The top 5 most important features are Pay_0 (X6), Pay_0_0 (X6_0), Pay_3_0 (X8_0), Bill_AMT3 (X14), and Pay_2_-2 (X7_-2).

Random Forest

RF is the best among the other models with a high accuracy rate and relatively high detection rate and a small false positive rate. The area under the ROC is also the largest. The top 5 most important features are Pay_0 (X6), Pay_0_0 (X6_0), Bill_AMT1 (X12), Pay_6 (X11), and Pay_2 (X7).

K-Nearest-Neighbors

The test accuracy is slightly better than the baseline accuracy. The area under the ROC is reasonably high as well. However, the detection rate is not high enough. The top 5 features are Pay_0_0 (X6_0), Pay_4_0 (X9_0), LIMIT_BAL (X1), Pay_2_0 (X7_0), Pay_6_0 (X11_0).

In short, the best model is Random Forest with the best accuracy and relatively high detection rate. The important features are Pay_0 (X6), Pay_2 (X7), Pay_6 (X11), and Bill_AMT1 (X12). All other sub-features can be derived with the Special* part in preprocessing. This suggests that the most recent payment delay status, the second most recent payment delay status, the least recent payment delay status, and the most recent billing statement amount matters the most when detecting bank credit card default.

Outlook

The next thing for this project is using bootstrapping in each model to deal with the imbalancement and try to boost the detection rate as the amount of data points is enough. The weak spot for NB is that it is assuming feature independence. It is not tunable and should be avoided. For LR, there could be some non-linear relationships and it cannot detect such relationships. We could do some feature engineering by adding some power of features in the data. For RF, the number of inner estimators may not be enough. We could use more estimators to test it. For KNN, it does not depend on any probability measure and the number of neighbors may not be able to detect points at the class boundary well. We should apply some probability mass to the points at the boundary to improve the detection rate.

Additional technique could be a neural network with backpropagation. Additional data could be the credit scores from third party agencies to boost the detection rate.

References

Yeh, I. C., & Lien, C. H. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications, 36(2), 2473-2480.

Islam, Sheikh Rabiul, William Eberle, and Sheikh Khaled Ghafoor. "Credit default mining using combined machine learning and heuristic approach." arXiv preprint arXiv:1807.01176 (2018).

Data Source: http://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients