Predicting the credit card default using machine learning algorithms

Md Rayhanul Islam [*1] **Eranda Sooriyarachchi** [*1 2]

Abstract

Loans are the backbone of the banking industry. Hence it is vital for loans to be profitable to banks. To avoid losses from loans, banks must avoid defaulters as much as possible. By identifying who is likely to default, the banks can provide the possible defaulters with incentives to prioritize on making the payment and in turn minimize the risk of default. To bring about the greatest profit margins to banks it is important to minimize losses from defaulters, banks need to recover the money they have given out in loans. Providing incentives to repay loans by a stochastic process is inefficient and costly. Given parameters such as income, age, credit history, and many others, we have developed a machine learning approach to accurately identify the clients who will be defaulters in the next month so that banks can proactively provide incentives to prevent loan defaulting while maximizing their profits. We solved this banking sector problem using 11 machine learning models and certain data preparation techniques. The data normalization was performed by the MinMaxScaler library in sklearn. We achieved the maximum accuracy of predicting the defaulters in the next month to be 94.08% by using the Random Forest model on a normalized dataset. The three best classifiers identified for this dataset to have over 90% accuracy were the Decision tree, Random Forest, and XGBoost models.

1. Introduction

This project would enable banks to identify customers who are likely to default in the following month, so that banks can proactively try to prevent its occurrence. The banks can provide loan repayment incentives to customers likely to default so that they can recover the loans they have given to their customers. Without having this information, banks would have to randomly provide incentives to their customers which could be costly, yet unable to solve the issue at hand. This is why we decided to apply machine learning models on the default of credit card clients dataset to predict potential defaulters with high accuracy.

The dataset (Yeh, 2017) titled "default of credit card clients Data Set" from UCI Machine Learning Repository contains a list of 30,000 bank customers' data from Taiwan. Each client has a unique ID and there are 25 attributes, one binary label for each customer whether they would default or not, and 23 explanatory variables. The variables are as follows: 1: Amount of the given credit (NT dollar),2: Gender, 3: Education level, 4: Marital status ,5: Age (year), 6 - 11: History of past payment(repayment status from April to September, 2005), 12-17: Amount of bill statement from April to September, 2005 (NT dollar), 18-23: Amount of previous payment from April to September, 2005 (NT dollar). As this dataset has mostly categorical features with numeric values, to provide a solution for this classification problem, we decided to use the following models: Logistic Regression, Linear Perceptron, Support Vector Machine, Naive Bayes, K-Nearest Neighbors, Decision Tree, Random Forest, AdaBoot, XgBoost, and Multi-layer Perceptron(MLP). [1]

2. Related Work

Predicting the credit loan defaulters for banks is important because they need to generate profit and sustain themselves in a competitive environment. The authors in (wor, b) applied a few machine learning algorithms to predict the defaulters in the next month. The algorithms used are Decision tree, Logistic regression, KN Neighbors, Clustering K-Means, Naive Bayes, Adaboost, Random forest classifier, and XgBoost to achieve a maximum accuracy of 79.18% in the Random forest approach.
The authors in (wor, a) have applied three classifier models, Logistic Regression, Random Forest, and XGBoost, out of which XgBoost was the top classifier to most accurately predict the defaulters of the next month with an 81.96% accuracy.
It is important to note that both of these sources used the same dataset that was used by us.

3. Methodology

This section describes the intuition behind machine learning models, dataset collection, and data preprocessing which in-

[1] The source code available in (sou)

cludes data normalization and balancing, generation of new features, and feature selection. It also describes the choice of hyper-parameters for different models in our dataset.

3.1. Dataset Collection

The dataset titled "default of credit card clients Data Set" from UCI dataset (Yeh, 2017) contains a list of 30,000 bank customers' data. There are 23 explanatory variables and a binary label as a response variable for each customer, whether or not they become the defaulter in October. The specific features are LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE, Repayment status from April to September (PAY_6 to PAY_1), Amount of bill statement from April to September (BILL_AMT6, to BILL_AMT1), Amount of previous payment from April to September (PAY_AMT1 to PAY_AMT6).

3.2. Data pre-processing

To make our results more accurate and unbiased, we planned to use 80% of the data for training (4000 samples) and the remaining 20% (1000 samples) for testing purposes. Among the 80 % of the data, we used k-fold cross-validation to train our model. To identify the optimum value of k, we followed this paper (Marcot & Hanea, 2021) and considered 10 as the optimum value for k. In the 10-fold cross-validation, 9 folds are used to train the model, and the remaining 1 fold is used as the validation set; this process is repeated 10 times before calculating the average training performance of the model.

3.2.1. Data Normalization

The dataset contains different features with different probability distributions and value ranges with different measuring scales. However, for better fitting the machine learning algorithms, we need to normalize our data without distorting the ranges of values or losing information. For example, Figure 1 shows that the mean value of Limit_bal and BILL_AMT1 is higher than that of Pay_1, Pay_2, etc. Therefore, the normalization of the entire dataset will impact prediction accuracy. We normalized the dataset using sklearn built-in library named MinMaxScaler. The MinMaxScaler scaler normalization works using the Equation 1 given below. The normalized description of the whole dataset is provided in Figure 2.

$$x_{normalized} = \frac{x - min(x)}{max(x) - min(x)} \tag{1}$$

3.2.2. Generations of new features:

The problem with this dataset is that it has many categorical features which hamper the classification models to perform well. Furthermore, we know that better encoding of categorical data means better model performance. In this dataset, Gender, Education, Payment history (Pay_1 to Pay_6), etc. are the categorical data. Therefore, it is important to convert these categorical data into binary-valued features. In this example, the value of features Payment history (Pay_1 to Pay_6) lies between -1 to 9, where -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above. Figure 3 shows how the distributions of credit card defaulters between PAY_1 and Limit_bal. Here in this dataset, we used the one-hot encoding approach to convert these categorical features (Pay_1 to Pay_6) to the binary-valued features. Similarly, we have also created a list of auxiliary features to effectively describe the information of the dataset. Table 1 shows all the newly created datasets along with their descriptions.

3.2.3. Feature selection:

To identify the best features impacting credit defaulters, we used the Feature selection method using the Pearson correlation method. It calculates the Pearson correlation (provided in Equation 2) between each feature with credit card defaulter and then drops all the features with negative correlations with the defaulters. The Pearson correlation values of the existing features are provided in Figure 4.

$$r = \frac{\sum(x_i - x')(y_i - y')}{\sqrt{\sum(x_i - x')^2(y_i - y')^2}} \tag{2}$$

3.2.4. Data Balancing:

The individual count on default.payment.next.month columns of the dataset is provided in Table 2. This table shows that 23364 samples are 0 and the remaining 6636 are 1. The number of not being defaulters is higher than being defaulters, which makes the dataset imbalanced and may hamper the prediction accuracy. Therefore, we balanced our dataset using a library named SMOTE with a hybrid approach, which increases the number of samples of the minority class and decreases the samples of the majority class. After the data augmentation, the new dataset in our experiment is now more balanced, provided in Table 3.

3.3. Machine learning models and the intuition behind selection

The selected problem of predicting bank loan defaulters is a classification problem; therefore, we applied multiple classification models, for example, Linear Perceptron algo-

Attribute	Description
Payment_score	It is the summation of all Payment 1 to Payment 6.
Bill_amt_score	It is calculated by multiplying payment score with average bill amount
BUSTED_LIMIT	It represents whether someone busts his/her limit.
PAYMENT_TO_CHARGES_RATIO	It is the ratio between the sum of all payments and bills.
BILL_TO_BALANCE	It is the ratio between the average bill and the balance limit.
LAST_BILL_TO_BALANCE	Ration between last bill and balance limit.
Pay_1 to 6	It is converted to binary value based on payment duly or delay
TOTAL_PAID_ON_TIME	Number of timely payments
AGE_less_30	It represents the age group less than 30.
AGE_less_40	It represents the age group less than 40 and older than 29.
AGE_less_50	It represents the age group less than 50 and older than 39.
AGE_less_60	It represents the age group less than 60 and older than 49.
Age_greater_60	It represents the age group older than 59.
Relationship_Married	It is true if married and false otherwise.
Relationship_Single	It is true if married and false otherwise.
Relationship_Others	It is true if married and false otherwise.
EDUCATION_grad_school	It represents who completes grad school.
EDUCATION_university	It represents who completes University.
EDUCATION_high_school	It represents who completes high school.
EDUCATION_others	It represents who has other certificates.
EDUCATION_ukn	It represents the education for unknown education.

Table 1. The list of newly generated features

	default.payment.next.month
0	23364
1	6636
Total	30000

Table 2. The value counts of default.payment.next.month column in the unbalanced dataset

	default.payment.next.month
0	11010
1	11981
Total	22991

Table 3. The value counts of default.payment.next.month column in the balanced dataset

rithms, Logistic Regression, Support Vector Machine, and K-nearest neighbors, on the dataset. The following section briefly describes the used machine learning models and the reason for selecting these models.

- **Logistic Regression:** This classification problem contains a binary class, which is defaulters or not. As we know, Logistic Regression works well in classifying binary data, so we choose it. We further know logistic regression works well for cases where the dataset is linearly separable. Although our data is not linearly separable, we used it to know how this model performs.
- **Linear Perceptron:** Although the decision boundary of this dataset is non-linear, we use Linear perceptron to compare results with Logistic Regression.
- **Support Vector Machine:** SVM allows us to classify linearly separable data. Even though the dataset is not linearly separable, we used the rbf kernel to make SVM workable in our data.
- **Naive Bayes:** As it performs well in cases of categorical input variables compared to numerical variables. As the dataset is categorical and small, we use Naive Bayes.
- **K-Nearest Neighbors**: As the data is numerical and plottable in N-dimensional space, that is the reason we choose knn models.
- **Decision Tree**: When new features are introduced in the existing data, all features in the dataset become binary-valued. As the decision tree works well with binary features and can easily build the decision tree, that's the reason we considered the decision tree to apply to our dataset.
- **Random Forest**: Although the Decision Tree performs well compared to other models, the intuition behind choosing Random Forest is to apply the Bagging technique and see how ensemble learning improves performance.

- **AdaBoot**: AdaBoot used to see how priorities to the misclassified sampled improved performances.

- **XgBoost**: This model is selected to see how extreme boosting can give more priority to the misclassified samples and correct it in the next iteration.

- **Multi-layer Perceptron(MLP)**: As the problem is a classification problem, we can use MLP, a feedforward artificial neural network with at least three layers, to classify our dataset.

3.4. Experimental Environment:

The experiment runs on the MacBook Pro with Processor 2 GHz Quad-Core Intel Core i5 and 16 GB RAM. Python 3.7 is used, and Sklearn and Matplotlib libraries are used to experiment. Apart from this, we also run this script in Google Kolab.

3.5. Choices of hyper-parameter:

The performance of machine learning models depends on how well hyper-parameters are selected. The choice of hyper-parameters for logistic regression, linear perception, Naive Bayes and QDA is very straightforward because we do not need to worry about it. However, when it comes to Support Vector Machine (SVM), k-nearest neighbors (knn), Decision Tree, and Random Forest, we need to identify a set of hyper-parameters like depth of the tree, kernel function, alpha value, number of estimators, depth of the tree, etc. depending on the models. The following sections will discuss how we selected hyper-parameters for individual machine-learning models.

3.5.1. Finding hyper-parameter for SVM:

Figure 5 shows that linear separation between loan defaulters and not being defaulters is impossible if we plot the data with respect to limit balance and bill amount. Therefore, we need a kernel function to make SVM workable in this dataset. Here, we used a brute force parameter search to find the optimum value for the kernel function and C (error value). Our experimental result found that training accuracy is better when C (error value) is 2 or 3, and the kernel function is poly for unbalanced data; however, when the balanced dataset is used, the choice of C becomes 1-5, and the Kernel becomes rbf (provided in Figure 7). The result shows that machine learning models outperform in the balanced data compared to imbalanced data; therefore, we will use only the balanced dataset in the remaining experiment.

3.5.2. Finding optimum k for KNN:

The Elbow method (Mabayoje et al., 2019) is used to find the optimum value for k-nearest neighbors (knn). Figure 8 plots the error rate concerning different k values in the x-axis. Here, we can see that the error value is minimum for k= 24, and 31. Therefore, we can consider anyone from these two values. For the sake of simplicity, we picked k=31 and run the knn on the test data considering that value of k.

3.5.3. Finding hyper-parameter for Decision Tree

For the decision tree, we created a decision tree classifier for different numbers of nodes and depths for each alpha value. Once we found a decision tree with a minimum error rate, we selected that tree's hyper-parameter for running our model. Figure 9 and Figure 10 show the best alpha for our experiment. We get the best decision tree for **alpha: 8.110879909225043e-05**.

3.5.4. Finding hyper-parameter for Random Forest

The hyper-parameters for Random Forest are the number of estimators, the depth of the tree, and many more; however, in this experiment, we only considered the number of estimators and depth. We ran the random forest for the set of combinations of estimators and depth and found the minimum error when the number of estimators is 26, 38, 53, and so on, and the depth is 14. However, considering the biased variance trade-off, we selected n_estimators 26 and depth 14 for running the random forest.

3.5.5. Finding hyper-parameter for XGBoost

Our experiment on the credit dataset shows the best hyper-parameters for XGBoost are binary: logistic as the objective function, max depth is 30, the number of estimators is 100, learning rate 0.1, and scale weight 5.

3.5.6. AdaBoost: n_estimators

Although many hyperparameters exist for AdaBoost, here, we have considered the maximum number of estimators as the parameters. Then we used the Elbow technique to find the best number of estimators for our dataset. In our case, n_estimators for this data is 90.

3.5.7. MultiLayer Perceptron (MLP)

The maximum iterations are 90, and the learning rate is 0.01.

3.5.8. NO HYPER-PARAMETERS

No Hyper-parameters for Logistic regression, Linear Perceptron, Naïve Bayes and QDA.

4. Experimental Results

Once the hyperparameters selection is completed, we selected those selected parameters to run the selected machine learning models. After that, we used accuracy, True Positive Rate (TPR), and False Positive Rate (FPR) to compare our results.

4.1. Performance Evaluation Metric

The accuracy and ROC_AUV value of different machine learning models are calculated using the confusion matrix. For better understanding, the confusion matrix is provided in Table 4.1, and later, the process of calculating accuracy, TPR, and FPR is also addressed.

	Predicted True	Predicted False
Actual True	TP	FN
Actual False	FP	TN

Table 4. Confusion Matrix

Now, the accuracy can be calculated from the confusion matrix using the following equation-

$$Accuracy = \frac{TP + TN}{P + N}$$

Furthermore, The plotting of ROC value is accomplished using TPR and FPR value, where

$$TPR = \frac{TP}{TP + FN}$$
$$FPR = \frac{FP}{TP + FP}$$

4.2. Result Analysis

This section describes the experimental results analysis of the different machine learning models on the UCI credit dataset.

Table 5 compares the training and testing accuracy of the machine learning models in the normalized data set. The highest accuracy score of 0.9188 was recorded for the Decision tree model, and the lowest accuracy score of 0.7647 was recorded for Linear Perceptron model. Overall, only 3 models show an accuracy score above 90%, which are Decision Tree, Random Forest and XgBoost models with accuracies of 0.9188, 0.9392 and 0.9228 respectively. Furthermore, according to the result, the least performing models are Linear Perceptron, QDA, and Naive Bayes in both the test and training datasets.

Model	Training accuracy	Testing accuracy
Logistic Regression	0.8498	0.8490
Linear Perceptron	0.7647	0.8103
SVM	0.8616	0.8573
Decision Tree	0.9188	0.9106
Random Forest	0.9392	0.9408
Knn	0.8621	0.8632
AdaBoost	0.8641	0.8638
MLP	0.8617	0.8617
XgBoost	0.9228	0.9228
Naive Bayes	0.8019	0.8021
QDA	0.8019	0.8106

Table 5. The training and testing accuracy of different machine learning models

Figure 12 shows the Receiver operating characteristic (ROC) curve depicts the diagnostic ability of the models from the normalized dataset. The ROC curves compare the true positive rate against the false positive rate of models, where the true positive rate is the probability of detection, whereas the false positive is the possibility of a false alarm. The AUC represents the degree of separability, which implies the model's ability to distinguish between different classes. From Figure 12, it is clear that the Decision tree, Random forest and XGBoost have values above 0.90. Random Forest has the highest value 0.94. Conversely, Linear Perceptron has the lowest value, 0.81. Therefore, it can be concluded that Decision tree, Random forest, and XgBoost are the top classifiers in this dataset.

Model	Work A (wor, a)	Work B (wor, b)	Our work
Logistic Regression	0.8054	0.7807	0.8490
Linear Perceptron	-	-	0.8103
SVM	-	-	0.8573
Decision Tree	-	0.7688	0.9106
Random Forest	0.8173	0.7918	0.9408
Knn	-	0.7752	0.8632
AdaBoost	-	0.7660	0.8638
MLP	-	-	0.8617
XgBoost	0.8196	0.7087	0.9228
Naive Bayes	-	0.7874	0.8021
QDA	-	-	0.8106
Clustering K-means	-	0.3142	-

Table 6. Comparison of accuracies of with two existing works

Table 6 compares the accuracy of the models with two existing project discussed in the related works section of the paper. Project B has the lowest accuracy of 0.3142 with the Clustering K means model, which we haven't used and the highest accuracy of 0.7918 using the Random Forest model, which is consistent with our best-performing model.

In the case of Project A, out of the three models used, the lowest accuracy is Logistic Regression at 0.7807, which is consistent with our results if we just picked the same three models. The highest accuracy in Related work A is using XgBoost with 0.8196 accuracies, which is our second highest performing model. Compared to both the related works, the accuracy of our models is superior. The highest accuracy of our models is over 94%, which others have not been able to achieve. Additionally, our lowest accuracy of 81.06% is similar to the highest accuracy of both related works.

5. Conclusions

This project provides a solution to the problem banks face when trying to mitigate losses due to loan defaults by accurately identifying the defaulters of the next month. Therefore, banks can be proactive with their strategies to avoid it. For this purpose, we used 11 machine learning models on a balanced and normalized dataset. To measure the performance of the models used, we used the accuracy of the model and ROC curve on each model. These results showed that the Decision Tree, Random Forest, and XGBoost had testing accuracies of 0.9106, 0.9408, and 0.9228, respectively. Hence, the Random Forest model was the top classifier with ROC value of 0.94 and an accuracy of 94.08%. Future work could be done on this model to tune the hyper-parameters for even higher accuracy.

References

Source code. https://drive.google.com/file/d/1T3v0aDVQ-BYLR7UzKcaoEo3-XkdjzFmA/view?usp=sharing. Accessed: Dec 09, 2022.

Medium. https://ameyband.medium.com/default-of-credit-card-clients-dataset-classification-evaluation-2c46c858d981, a. Accessed: Dec 09, 2022.

Kaggle. https://www.kaggle.com/code/samuelpaquettepar/default-payments-of-credit-card-clients-analysis#Model-Experimentation, b. Accessed: Dec 09, 2022.

Mabayoje, M. A., Balogun, A. O., Jibril, H. A., Atoyebi, J. O., Mojeed, H. A., and Adeyemo, V. E. Parameter tuning in knn for software defect prediction: an empirical analysis. 2019.

Marcot, B. G. and Hanea, A. M. What is an optimal value of k in k-fold cross-validation in discrete bayesian network analysis? *Computational Statistics*, 36(3):2009–2031, 2021.

Yeh, I.-C. UCI machine learning repository, 2017. URL https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients.

	count	mean	std	min	25%	50%	75%	max
ID	30000.0	15000.500000	8660.398374	1.0	7500.75	15000.5	22500.25	30000.0
LIMIT_BAL	30000.0	167484.322667	129747.661567	10000.0	50000.00	140000.0	240000.00	1000000.0
SEX	30000.0	1.603733	0.489129	1.0	1.00	2.0	2.00	2.0
EDUCATION	30000.0	1.853133	0.790349	0.0	1.00	2.0	2.00	6.0
MARRIAGE	30000.0	1.551867	0.521970	0.0	1.00	2.0	2.00	3.0
AGE	30000.0	35.485500	9.217904	21.0	28.00	34.0	41.00	79.0
PAY_1	30000.0	-0.016700	1.123802	-2.0	-1.00	0.0	0.00	8.0
PAY_2	30000.0	-0.133767	1.197186	-2.0	-1.00	0.0	0.00	8.0
PAY_3	30000.0	-0.166200	1.196868	-2.0	-1.00	0.0	0.00	8.0
PAY_4	30000.0	-0.220667	1.169139	-2.0	-1.00	0.0	0.00	8.0
PAY_5	30000.0	-0.266200	1.133187	-2.0	-1.00	0.0	0.00	8.0
PAY_6	30000.0	-0.291100	1.149988	-2.0	-1.00	0.0	0.00	8.0
BILL_AMT1	30000.0	51223.330900	73635.860576	-165580.0	3558.75	22381.5	67091.00	964511.0
BILL_AMT2	30000.0	49179.075167	71173.768783	-69777.0	2984.75	21200.0	64006.25	983931.0
BILL_AMT3	30000.0	47013.154800	69349.387427	-157264.0	2666.25	20088.5	60164.75	1664089.0
BILL_AMT4	30000.0	43262.948967	64332.856134	-170000.0	2326.75	19052.0	54506.00	891586.0
BILL_AMT5	30000.0	40311.400967	60797.155770	-81334.0	1763.00	18104.5	50190.50	927171.0
BILL_AMT6	30000.0	38871.760400	59554.107537	-339603.0	1256.00	17071.0	49198.25	961664.0
PAY_AMT1	30000.0	5663.580500	16563.280354	0.0	1000.00	2100.0	5006.00	873552.0
PAY_AMT2	30000.0	5921.163500	23040.870402	0.0	833.00	2009.0	5000.00	1684259.0
PAY_AMT3	30000.0	5225.681500	17606.961470	0.0	390.00	1800.0	4505.00	896040.0
PAY_AMT4	30000.0	4826.076867	15666.159744	0.0	296.00	1500.0	4013.25	621000.0
PAY_AMT5	30000.0	4799.387633	15278.305679	0.0	252.50	1500.0	4031.50	426529.0
PAY_AMT6	30000.0	5215.502567	17777.465775	0.0	117.75	1500.0	4000.00	528666.0
default.payment.next.month	30000.0	0.221200	0.415062	0.0	0.00	0.0	0.00	1.0

Figure 1. The dataset description before normalization

	count	mean	std	min	25%	50%	75%	max
ID	30000.0	15000.500000	8660.398374	1.000000	7500.750000	15000.500000	22500.250000	3.000000e+04
LIMIT_BAL	30000.0	167484.322667	129747.661567	10000.000000	50000.000000	140000.000000	240000.000000	1.000000e+06
SEX	30000.0	1.603733	0.489129	1.000000	1.000000	2.000000	2.000000	2.000000e+00
EDUCATION	30000.0	1.842267	0.744494	1.000000	1.000000	2.000000	2.000000	4.000000e+00
MARRIAGE	30000.0	1.557267	0.521405	1.000000	1.000000	2.000000	2.000000	3.000000e+00
AGE	30000.0	35.485500	9.217904	21.000000	28.000000	34.000000	41.000000	7.900000e+01
PAY_1	30000.0	0.772733	0.419073	0.000000	1.000000	1.000000	1.000000	1.000000e+00
PAY_2	30000.0	0.852067	0.355040	0.000000	1.000000	1.000000	1.000000	1.000000e+00
PAY_3	30000.0	0.859567	0.347442	0.000000	1.000000	1.000000	1.000000	1.000000e+00
PAY_4	30000.0	0.883000	0.321426	0.000000	1.000000	1.000000	1.000000	1.000000e+00
PAY_5	30000.0	0.901067	0.298577	0.000000	1.000000	1.000000	1.000000	1.000000e+00
PAY_6	30000.0	0.897367	0.303484	0.000000	1.000000	1.000000	1.000000	1.000000e+00
BILL_AMT1	30000.0	51223.330900	73635.860576	-165580.000000	3558.750000	22381.500000	67091.000000	9.645110e+05
BILL_AMT2	30000.0	49179.075167	71173.768783	-69777.000000	2984.750000	21200.000000	64006.250000	9.839310e+05
BILL_AMT3	30000.0	47013.154800	69349.387427	-157264.000000	2666.250000	20088.500000	60164.750000	1.664089e+06
BILL_AMT4	30000.0	43262.948967	64332.856134	-170000.000000	2326.750000	19052.000000	54506.000000	8.915860e+05
BILL_AMT5	30000.0	40311.400967	60797.155770	-81334.000000	1763.000000	18104.500000	50190.500000	9.271710e+05
BILL_AMT6	30000.0	38871.760400	59554.107537	-339603.000000	1256.000000	17071.000000	49198.250000	9.616640e+05
PAY_AMT1	30000.0	5663.580500	16563.280354	0.000000	1000.000000	2100.000000	5006.000000	8.735520e+05
PAY_AMT2	30000.0	5921.163500	23040.870402	0.000000	833.000000	2009.000000	5000.000000	1.684259e+06
PAY_AMT3	30000.0	5225.681500	17606.961470	0.000000	390.000000	1800.000000	4505.000000	8.960400e+05
PAY_AMT4	30000.0	4826.076867	15666.159744	0.000000	296.000000	1500.000000	4013.250000	6.210000e+05
PAY_AMT5	30000.0	4799.387633	15278.305679	0.000000	252.500000	1500.000000	4031.500000	4.265290e+05
PAY_AMT6	30000.0	5215.502567	17777.465775	0.000000	117.750000	1500.000000	4000.000000	5.286660e+05
default.payment.next.month	30000.0	0.221200	0.415062	0.000000	0.000000	0.000000	0.000000	1.000000e+00
EDUCATION_grad_school	30000.0	0.352833	0.477859	0.000000	0.000000	0.000000	1.000000	1.000000e+00

Figure 2. The normalized dataset description

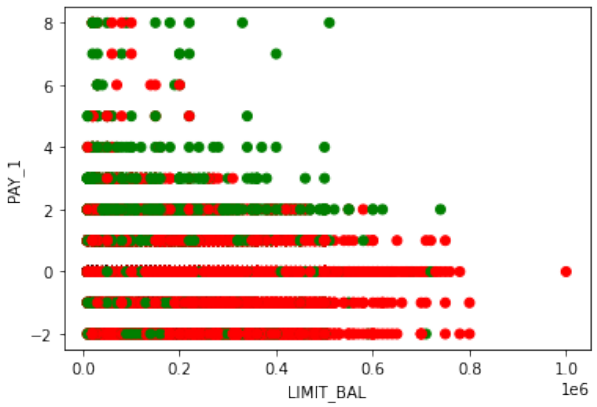

Figure 3. The distribution of credit card defaulters to PAY_1

```
ID : -0.01395195483898623
LIMIT_BAL : -0.1535198763935075
SEX : -0.0399605777054416
EDUCATION : 0.02800607765624998
MARRIAGE : -0.024339215683403748
AGE : 0.013889834301963243
PAY_1 : 0.3247937284786222
PAY_2 : 0.26355120167216306
PAY_3 : 0.23525251372491646
PAY_4 : 0.2166136368424244
PAY_5 : 0.2041489138761648
PAY_6 : 0.1868663616535438
BILL_AMT1 : -0.019644197143221534
BILL_AMT2 : -0.014193218088215737
BILL_AMT3 : -0.014075518043214781
BILL_AMT4 : -0.010156495880289721
BILL_AMT5 : -0.006760463841014728
BILL_AMT6 : -0.005372314914815501 6
PAY_AMT1 : -0.07292948777785141
PAY_AMT2 : -0.058578706582900986
PAY_AMT3 : -0.056250350990331204
PAY_AMT4 : -0.05682740089288779
PAY_AMT5 : -0.05512351562108832
PAY_AMT6 : -0.05318334032612812
default.payment.next.month : 1.0
```

Figure 4. The Pearson correlation values of all existing features

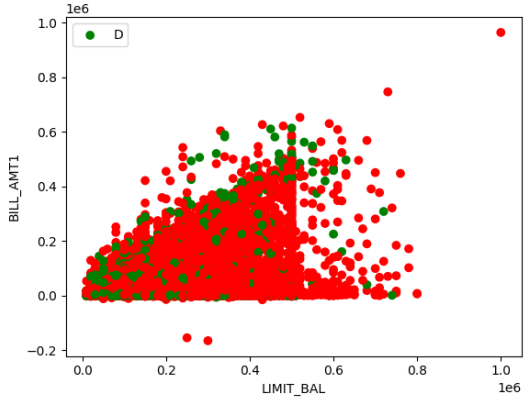

Figure 5. The distribution of loan defaulter with Limit_bal and Bill_amt1

```
kernel : linear, C: 0.01, a: 0.8180555555555555
kernel : linear, C: 0.1, a: 0.8216666666666667
kernel : linear, C: 1, a: 0.8216666666666667
kernel : linear, C: 2, a: 0.8219444444444445
kernel : linear, C: 3, a: 0.8219444444444445
kernel : linear, C: 5, a: 0.8219444444444445
kernel : poly, C: 0.01, a: 0.8152777777777778
kernel : poly, C: 0.1, a: 0.8222222222222222
kernel : poly, C: 1, a: 0.8280555555555555
kernel : poly, C: 2, a: 0.83
kernel : poly, C: 3, a: 0.83
kernel : poly, C: 5, a: 0.8297222222222222
kernel : rbf, C: 0.01, a: 0.815
kernel : rbf, C: 0.1, a: 0.8183333333333334
kernel : rbf, C: 1, a: 0.8283333333333334
kernel : rbf, C: 2, a: 0.8286111111111111
kernel : rbf, C: 3, a: 0.8283333333333334
kernel : rbf, C: 5, a: 0.8283333333333334
```

Figure 6. The hyper-parameter for SVM in the unbalanced with all features dataset

```
kernel : linear, C: 0.01, a: 0.7883291047480971
kernel : linear, C: 0.1, a: 0.849583182312432
kernel : linear, C: 1, a: 0.8510329829648423
kernel : linear, C: 2, a: 0.8510329829648423
kernel : linear, C: 3, a: 0.8510329829648423
kernel : linear, C: 5, a: 0.8510329829648423
kernel : poly, C: 0.01, a: 0.7459224356650961
kernel : poly, C: 0.1, a: 0.8133381660021747
kernel : poly, C: 1, a: 0.8354476259514316
kernel : poly, C: 2, a: 0.8379847770931497
kernel : poly, C: 3, a: 0.83943457774556
kernel : poly, C: 5, a: 0.8412468285610728
kernel : rbf, C: 0.01, a: 0.7857919536063791
kernel : rbf, C: 0.1, a: 0.8321855744835085
kernel : rbf, C: 1, a: 0.8535701341065604
kernel : rbf, C: 2, a: 0.8535701341065604
kernel : rbf, C: 3, a: 0.8535701341065604
kernel : rbf, C: 5, a: 0.8535701341065604
```

Figure 7. The hyper-parameter for SVM in the balanced

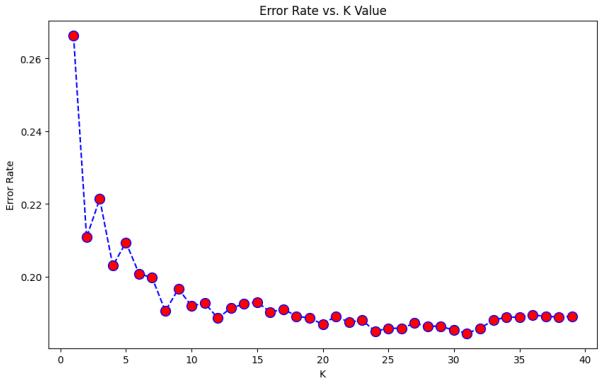

Figure 8. The k value of knn using Elbow method

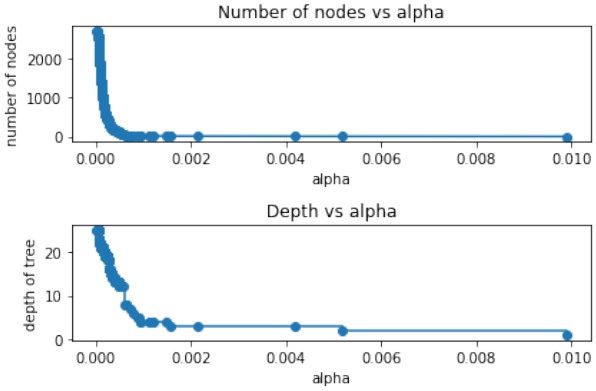

Figure 9. Number of nodes vs alpha

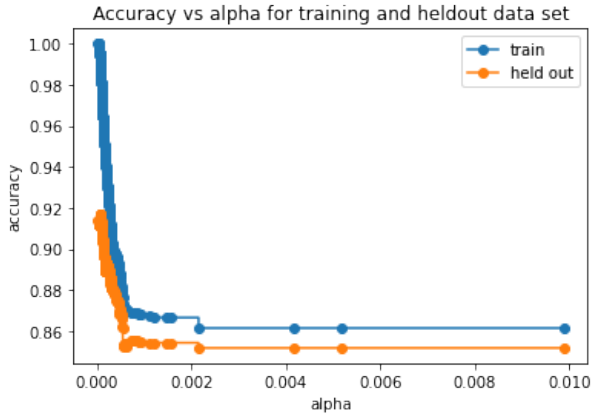

Figure 10. The accuracy value of decision tree with different value of alpha

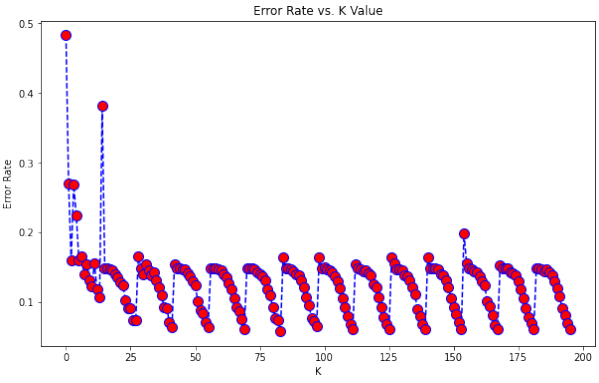

Figure 11. The hyper-parameters selection for Random Forest

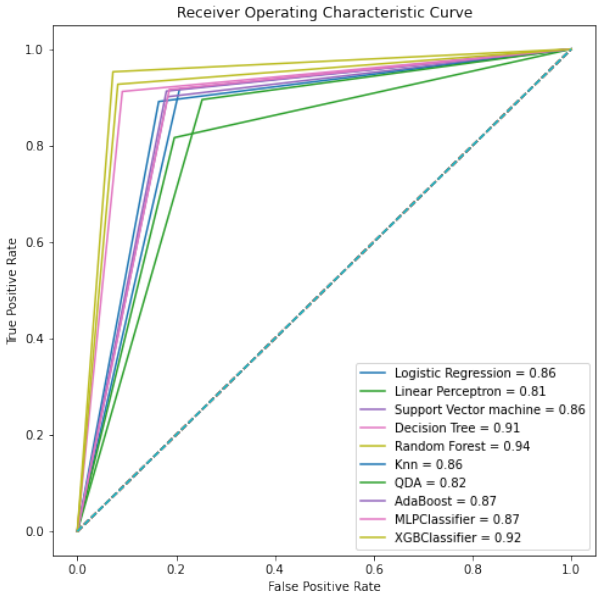

Figure 12. The AUC-ROC Curve