Credit Default Prediction via Machine Learning

*Biancheng Wang and Tianyi Xia**

Abstract

To figure out the best method which can be used for risk management in developing countries in terms of the credit industry, this research aims at the case of default payments in Taiwan and applies nine machine learning methods for classification on this data with highly correlated features. Since the dataset is a little imbalanced, F_1 score and AUC are used instead of Accuracy. After feature engineering, feature selection, and parameter tuning, models are evaluated based on the performance metrics above. Gradient boosting and random forest, two tree-based ensemble methods, are the top two methods, and the latest payment status is the most important variable here.

Keywords: Credit Default; Gradient Boosting; Random Forest; ROC

1. Introduction

Increased competition and growing pressure for revenue generation have led credit-granting and other financial institutions to search for more effective ways to attract new creditworthy customers, and at the same time, control losses. For a credit card firm, the main purpose of risk prediction is to use financial information to predict individual customers' credit risk to reduce the damage. With the help of those predictive models, the financial company can make different business decisions in terms of different customers.

Many statistical methods, including discriminant analysis, logistic regression, Bayes Classifier, nearest neighbor decision trees and neural networks have been used to forecast credit risk (I-Cheng Yeh & Che-hui Lien, 2009). Credit risk here means the probability of a delay in the repayment of the credit granted (Paolo, 2001).

However, unlike the United States with three main credit bureaus, in some developing countries, for example, in China, credit reporting is just an emerging industry. Sometimes, credit card companies may be faced with limited and even highly correlated features since most of those features might be history of payment. Also, the proportion of observations with default payment is always small which may lead to some class imbalance problem.

Therefore, it is important to figure out how good popular methods are on such large imbalanced datasets with limited and highly correlated features, which may also be meaningful for industry when taking the reality of the developing industry into consideration. This is also the main goal of this research.

In this paper, we will not only use the method I-Cheng Yeh and Che-hui Lien (2009) used but also introduce two more machine learning methods (random forest and gradient boosting). Besides, we will conduct feature engineering before modeling and select features using methods including regularization to tackle with the problem caused by high correlated features. In practice, misclassification costs of two kinds of error are different. Therefore, also because of the imbalanced data we have, some other performance metrics instead of accuracy will be used, including F_1 score and AUC.

In the next section, we review the modeling methodologies and performance metrics. Then, we describe the real credit card data in Taiwan and conduct data cleaning. Section 4 is dedicated to the application of these statistical methods, including some modeling details. In section 5, we compare the model performance among models and also with the result from I-Cheng Yeh and Che-hui Lien (2009). Finally, we conclude in Section 6.

*Department of Statistics, UCLA

2. Modeling Methodologies and Performance Metrics

2.1 Modeling Methodologies

1) **Logistic Regression**

Logistic regression is one of the generalized linear model(GLM). For a binary outcome Y_i, we assume that $Y_i|X_i \sim Bernoulli(\pi_i)$, thus our aim is to predict $\pi_i = \mathbb{E}[Y_i|X_i]$. We use the logit function to link π_i with the linear predictor $X_i^T\beta$. Then we find the $\hat{\beta}$ that maximizes the likelihood given our training data. For a new data point X_k from testing set, we can calculate $\pi_k = exp(X_k^T\beta)/[1 + exp(X_k^T\beta)]$. For a natural threshold 0.5, we predict Y_k as 0 if $\pi_k < 0.5$, otherwise as 1.

To avoid overfitting, we could combine logistic regression with ridge regression or lasso regression. Since there is linear predictor $X_i^T\beta$ in the logistic regression model, we need to check that whether those categorical variables are transferred to numerics with reasonable orders. Besides, linearity itself is a strong assumption. GLM models may perform poorly when the response has complex relationship with variables.

We use the function "glm()" to perform the logistic regression. We use the function "cv.glmnet()" and "glmnet()" in R package "glmnet" to choose the parameter and perform the ridge/lasso logit models.

2) **K-Nearest Neighbor Classifiers (KNN)**

The idea of KNN is that: For each row of the test set, the k nearest training set vectors (according to some kind of distance) are found. Then we calculate the mean of the k responses. If the mean is less than 0.5, we predict the response of the row of test set as 0, otherwise as 1.

Here we meet the problem of choosing k. Our first thought is to choose k by AUC, since it is now our criterion of judging the performance of methods. Then we notice that AUC gets higher as our k increases. A possible reason is that by using k neighbors, the mean of the k responses could be $\{1/k, 2/k, ..., k/k\}$, providing k turning points in the ROC curve. As we increase k, the ROC curve would be smoother, leading to an increasing AUC value. Therefore, AUC is not a good standard for us to decide which k to choose.

So we still use error rate to pick an appropriate k: Separate the training set into two parts, data_train1 and data_train2. For each k, perform k-nearest neighbor classification of data_train2 using data_train1. Then we calculate the error rate. By varying k in a range from 1 to 20, we choose the k with the smallest error rate.

We use the function "kknn()" in R package "kknn" to perform the KNN method.

3) **Linear Discriminant Analysis (LDA)**

LDA is commonly used in dimension reduction before classification, while it can also be used as a linear classifier. To approach the two classes problem, LDA assumes the class densities $p(X_i|Y_i = 0)$ and $p(X_i|Y_i = 1)$ are both normally distributed and have the same covariance matrix.

Our purpose is to calculate and compare $p(Y_i = 0|X_i)$ and $p(Y_i = 1|X_i)$. From the Bayesian point of view, we know that $p(Y_i = g|X_i) \propto p(X_i|Y_i = g)p(Y_i = g)$. The model learns parameter of the normal distributions of class densities, then for a new data point X_i from the test set, we can calculate $p(Y_i = g|X_i), g \in \{0, 1\}$. If we choose natural threshold 0.5, we predict Y_i as 0 if $p(Y_i = 0|X_i) > p(Y_i = 1|X_i)$, otherwise as 1.

Notice that LDA could go wrong if the assumptions mentioned above don't hold. Also, when two variables are highly correlated, the inverse of covariance matrix is unstable. In the expression of $log(p(Y_i = g|X_i))$, we see the linear term of X_i, which leads to a linear decision boundary.

We use the function "lda()" in R package "MASS" to perform the LDA method.

4) **Quadratic Discriminant Analysis (QDA)**

QDA is closely related to LDA, they both have the assumption that the class densities $p(X_i|Y_i = 0)$ and $p(X_i|Y_i = 1)$ are normally distributed. The difference is that QDA doesn't assume that the covariance matrix of each class is identical.

Like LDA, we need to check whether those assumptions in QDA fit for the data. Besides, since QDA has much more parameters than LDA, it is more likely to have overfitting problem when data size is small. As its name implies, QDA leads to a quadratic decision boundary.

We use the function "qda()" in R package "MASS" to perform the QDA method.

5) **Naive Bayes Classifier (NBC)**

Although NBC has a quite simplified assumption, it works quite well in many real applications. Again we need to estimate the class densities $p(X_i|Y_i = g), g \in \{0, 1\}$. Now we assume that all the features are independent. Thus we have $p(X_i|Y_i = g) = \prod_{j=1}^{p} p(X_i^{(j)}|Y_i = g), g \in \{0, 1\}$. Under this assumption, we reduce p-dimensional density to product of p univariate densities. Also we always assume that $p(X_i^{(j)}|Y_i = g)$ is a normal density, leading to Guassian Naive Bayes.

Notice that if the features of our data are highly correlated, NBC could perform poorly.

We use the funtion "naiveBayes()" in R package "e1071" to perform the NBC method.

6) **Classification And Regression Tree (CART)**

To create a CART model, we need to select input variables and split points on these variables until a suitable tree is constructed. In the classification problem, we always use Gini index function to decide the purity of the tree nodes. For our binary classification problem, $G = 2p(1 - p)$. For each variable, we can find the best split point that gives the lowest Gini index G_i. Then among all the possible variables, we find the smallest G_i and use the corresponding variable (with the best split point) to split the node to two child nodes.

By repeating the steps to child nodes, we can construct the tree. We always set a stopping criterion to decide when to stop splitting, and then prune the tree to improve its performance. For a new data point X_i in the test set, it would fall into one of the leaf node. If this leaf node has more observations (in training set) with response 0, then we predict X_i's response Y_i as 0, otherwise as 1.

The classification rule of a tree model makes it easy to understand and interpret the result. But if the structure of our data is complex, classification tree may be too simple and not very effective.

We use the function "tree()" in R package "tree" to perform the Classification Tree method.

7) **Random Forest**

Random forest is an ensemble learning method for classification. Instead of just constructing one tree as in the classification tree model, we grow a multitude of decision trees using the training data. Suppose that there are N observations in training set, for each tree, the sample of these N cases is randomly taken and with replacement. At each node, we randomly select m out of all M variables into consideration of splitting the node. We usually grow each tree to the largest extent without pruning. We predict the response of new data from testing set by choosing the majority votes given by all trees built in the training process.

Random forest could correct the overfitting problem in classification tree method. On the other side, because of the randomness, we have little control of what the model does.

We use the function "randomForest()" in R package "randomForest" to perform the Random Forest method.

8) **Support Vector Machine (SVM)**

Support Vector Machine is a supervised learning method used for classification. The idea is that we want to find a hyperplane to separate our data into two classes. Although there are many hyperplanes that might classify the data, we prefer the one with largest margin between the two classes. In other

words, we choose the hyperplane that has the largest distance to the nearest training data point of any class. The intuition here is that the widest margin would take biggest change in X to switch the label.

If the data is not linearly separable in the original space, we could try the kernel trick to map the data into a much higher-dimensional space, and expect that the separation would be easier in that space.

We use the function "svm()" in R package "e1071" to perform the SVM method.

9) **Gradient Boosting**

Gradient boosting is a well-known prediction model as an ensemble of weak classifiers, typically decision trees. The idea of boosting is that: When we iteratively add weak classifiers, the data will be reweighted in a way such that those misclassified examples would gain weights so that the future classifiers will focus more on fixing the mistakes. Gradient boosting could be regarded as an optimization problem. Our purpose is to minimize the loss we defined. We use a gradient descent procedure when adding weak classifiers. It is an additive model, which means that a new classifier is added at a time and the existing classifiers would not be changed.

To understand it from algorithm point of view, let y be our response variable and x be a vector of input variables. The training set is $\{(x_1, y_1), ..., (x_n, y_n)\}$. The goal is to find $\hat{F}(x)$, an approximation of a function $F(x)$ that minimizes the expectation value of given loss function $L(y, F(x))$:

$$\hat{F} = \underset{F}{argmin}\, \mathbb{E}_{x,y}[L(y, F(x))]$$

To show the algorithm in a compact form:

1) Initialize the model with a constant value:

$$F_0(x) = \underset{\gamma}{argmin} \sum_{i=1}^{n} L(y_i, \gamma)$$

2) For $m = 1$ to M:

   a) Compute so-called pseudo-residuals:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

   for $i = 1, ..., n$.

   b) Fit a base learner $h_m(x)$ to pseudo-residuals. Use the training set $\{(x_i, r_{im})\}_{i=1}^{n}$ to train the model.

   c) Compute multiplier γ_m by solving the linear search:

$$\gamma_m = \underset{\gamma}{argmin} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

   d) Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

3) Output $F_M(x)$.

For a tree-based gradient boosting, we can compare it to random forest. The similarity is that, they both use an ensemble of multiple trees to create more powerful prediction models for classification. While unlike the random forest method that builds and combines a forest of randomly different trees in parallel, the key point of gradient boosted decision trees is that they build a series of trees. Here each tree is trained to correct the mistakes of previous trees in the series.

To control the model complexity, one important parameter is the number of small decision trees in the ensemble. Another one is the learning rate, which controls how the gradient boosts the tree algorithm. When the learning rate is high, each tree would put much emphasis on correcting the mistakes of previous model. This results in a more complex individual tree and increases the overall complexity of our model. These two parameters are usually tuned together. Since when we lower the learning rates, more trees are needed to maintain the model complexity. Besides, the max depth parameter for individual tree also has an effect of model complexity. Small depth makes model simpler. Since we assume each tree to be a weak classifier, the max depth is usually set as three to five.

One advantage of tree-based gradient boosting is that we don't need to do feature scaling in advance. Also, the feature can be a mix of binary, categorical and continuous types. A downside is it is hard to interpret, compared to an individual tree. Sometimes, the training process may require heavy computation.

We use the function "train()" in R package "caret" to choose the parameters in gradient boosting and use the function "gbm()" in R package "gbm" to perform the gradient boosting method.

2.2 Performance Metrics

1) **Accuracy**

Accuracy is a well-used metric to judge the model performance. It is the ratio of correctly classified samples within all the samples in the testing set. One downside of this metric is that when we have imbalanced data, we could get high accuracy by frequently predicting the majority group. Since our data is little imbalanced, we don't take accuracy as our core metric.

2) F_1 **Score**

F_1 score takes both recall and precision into consideration. The traditional F_1 score is

$$F_1 = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = 2\frac{precision \times recall}{precision + recall}$$

Higher precision and higher recall lead to a higher F_1 score. Notice that this metric does not take the true negatives into account and emphasizes on the how well we can get the true positives. Since in the credit default case, we do care more about picking out those clients who would default(true positive), F_1 may be a reasonable measure.

3) **Area Under the Curve (AUC)**

The ROC AUC is most often used for model comparison in machine learning. A receiver operating characteristic curve(ROC curve), is a graphical plot that illustrates the diagnostic ability of a binary classifier. For various threshold, the sensitivity against the specificity is plotted. The ROC curve visualizes the tradeoff between sensitivity and specificity. For our imbalanced data, we prefer ROC AUC to accuracy and will take it as our measure of model performance.

3. Data Description

The dataset [1] contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. Among total 30000 observations, 6636 observations (22.12%) are the cardholders with default payment. Therefore, we just have a modest imbalanced data problem in this case. So we decide not to use oversampling or undersampling method in this case. We omit 54 observations with missing data and transform some categorical variables to dummy variables, for example, Sex and Marriage. Since there are some unknown categories in Education, we

[1] Lichman, M. (2013). UCI Machine Learning Repository, http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.

will handle this in next section. Our response variable is default payment next month (1=Yes, 0=No). There are 23 variables as predictors as follows:

Name of Variable	Meaning and Level/Value
Limit Balance	Amount of given credit (NT dollar)
Sex	Gender (0=female, 1=male)
Education	(0=unknown, 1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
Marriage	Marital Status (1=married, 2=single, 3=others)
Age	Age in years
PAY_1-PAY_6	Repayment status (1-6) months ago (-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.)
BILL_AMT_1-BILL_AMT_6	Amount of bill statement X (1-6) months ago (NT dollar)
PAY_AMT_1-PAY_AMT_6	Amount of previous payment X (1-6) months ago (NT dollar)

For a better understanding, here we give more explanations about the variable. PAY_1, BILL_AMT_1 and PAY_AMT_1 track the repayment status, amount of bill statement and amount of previous payment in September, 2005 respectively. PAY_6, BILL_AMT_6 and PAY_AMT_6 track the repayment status, amount of bill statement and amount of previous payment in April, 2005 respectively.

The correlation plot is showed in the Figure 1. To take a look at collinearity, here we plot the absolute value of the correlation. The features which stand for history of repayment status are highly correlated as expected. So does the history of bill statement.

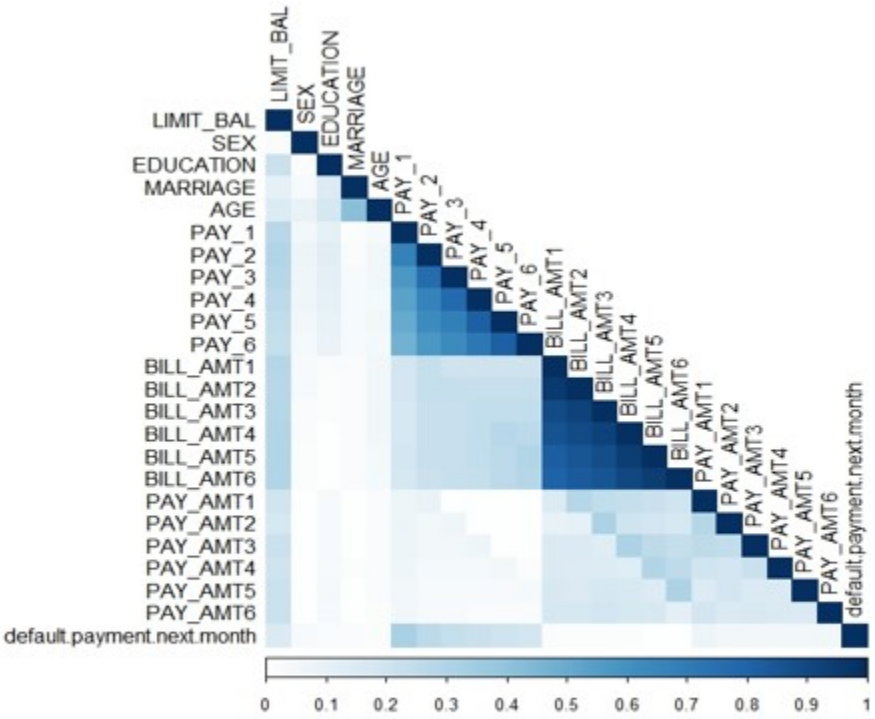


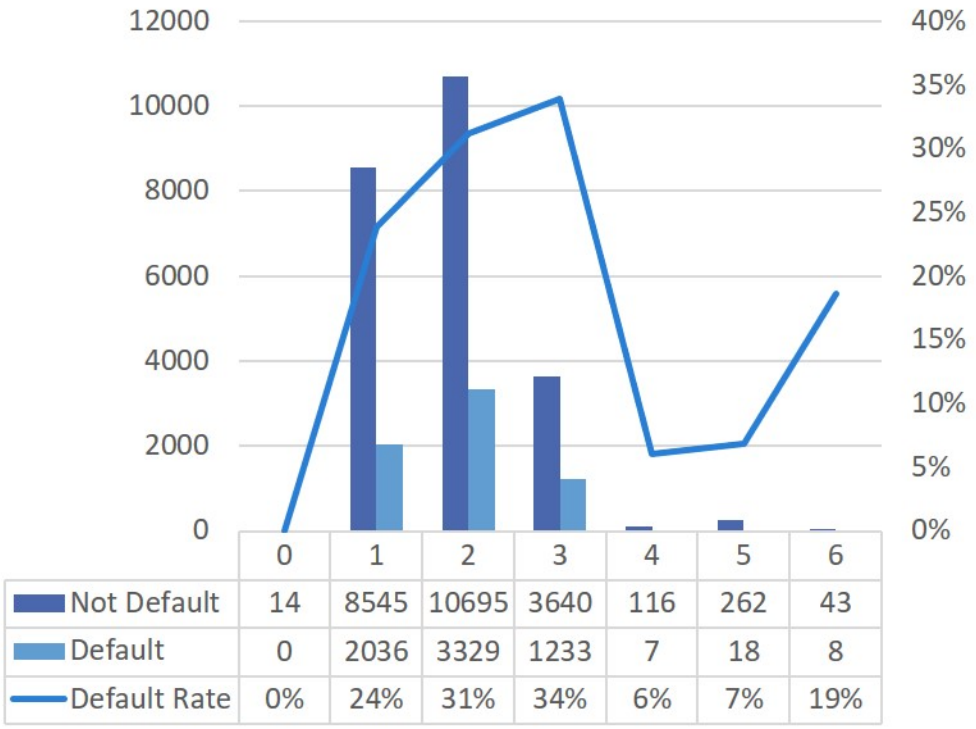
Figure 1: Correlation Plot

The data is randomly divided into two groups, one for model training and the other for testing the model.

4. Modeling Details

4.1 Feature Engineering

In the categorical variable, there are 345 observations with unknown education level (Education=0, 5, 6) and 123 observations with education level "Other" (Education=4). We try to find the pattern of data by finding the relationship between education and default rate.



	0	1	2	3	4	5	6
Not Default	14	8545	10695	3640	116	262	43
Default	0	2036	3329	1233	7	18	8
Default Rate	0%	24%	31%	34%	6%	7%	19%

Figure 2: Histogram of Education

From Figure 2, the proportion of default increases as Education increases from 1 to 3. Also, the default rates in group 0, 4, 5 and 6 are all lower than the counterpart in group 1. Therefore, in order to represent this trend, we set Education=0 for observations with Education equals 4, 5 or 6.

Next, from the correlation plot, it seems that Bill Amount does not contribute a lot to the response variable on its own. We would like to do some bivariate analysis to figure out whether there are some interesting interactions with other features. Then we can create some new features based on what we found.

Figure 3 shows that there exists some sort of vertical wall at the point where Limit Balance equals 500000. These guys with higher limit balance may have different customer behavior. Thus we create a variable special_customer which would be the status indicating whether a customer is granted a balance limit over 500000.

Besides, it seems that the default samples generally lie near diagonal lines, which means that defaulters' monthly bills often reach their balance limits. We divide Bill Amount each month by Limit Balance and denote them as a series of variables close_1 to close_6.

As we can see in Figure 4, there is a clear trend between BILL_AMT_X+1 and PAY_AMT_X. This plot displays the bill against the payment over the same month. For example, BILL_AMT_2 is the bill statement of August which should be paid in September and PAY_AMT_1 is the payment in September. Defaulters always center on the left side of the plots, which means that they pay a very low portion of their bills

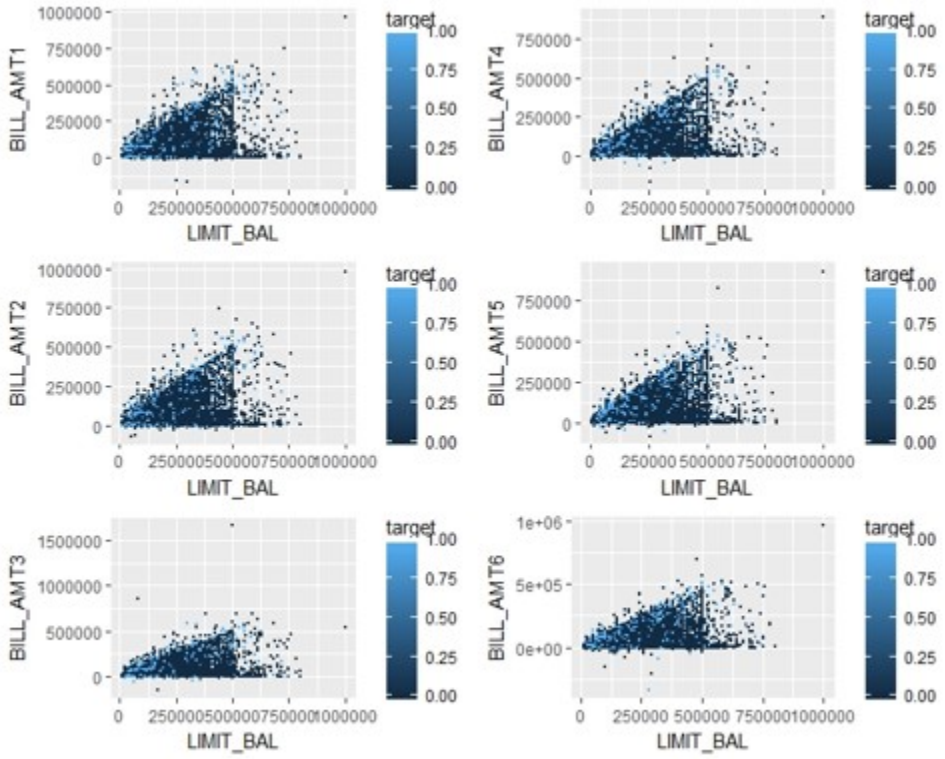


Figure 3: Limit Balance vs Bill Amount

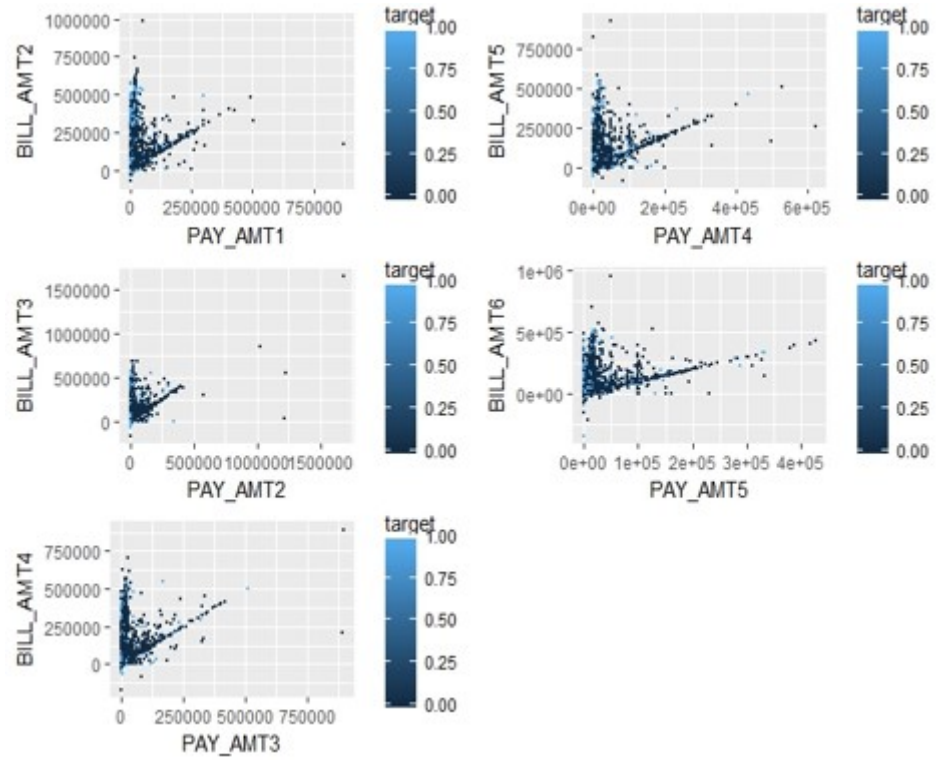


Figure 4: Pay Amount vs Bill Amount

consistently. We divide PAY_AMT_X by BILL_AMT_X+1 and denote it as pbr_X which captures the pay back ratio in that month.

Then, after feature engineering, we use all features we create as well as original features to train our model.

4.2 Feature Selection

Since our data size is large enough, for some methods such as Gradient Boosting, the model itself will learn the importance of those features. We could put all the variables into the model in this case.

While for generative models such as LDA, QDA and Naive Bayes, we need to perform feature selection in order to meet the assumptions of those models. Notice we have high correlations between some of the predictors, it may lead to unstable inverse of covariance matrices of LDA and QDA. Also, the assumption of independence in Naive Bayes would fail. We would manually choose those features with low correlation into our generative models.

For logistic regression, we should also perform model selection. We use step-wise selection to pick out those significant features, then run the logistic regression with these variables. Also, our lasso logit model has an effect of feature selection.

4.3 Parameter Tuning

Parameter tuning is an important part before we decide the exact model to use. We use a common way to deal with this part: Cross validation with grid search.

For choosing the regularization parameter λ, we use the "cv.glmnet" function in R package "glmnet" to do the cross validation.

For choosing k in KNN and "ntree", "mtry" in random forest, we use a simpler way. Since our data size is relatively large and cross validation is quite time-consuming, we just separate our training data into two parts. For each grid, we use one part to train the model and calculate error rate on the other part. Then we choose the parameter grid with the smallest error rate.

To control the complexity of gradient boosting model, we need to tune the parameters "n.trees"","interaction.depth" and "shrinkage". In practice, we use the function "train" in R package "caret" to do the cross validation.

5. Results

5.1 Model Evaluation

We list accuracy, F_1 score and AUC in the following table. Also, we list the accuracy in the paper for comparison.

	Accuracy	F_1 Score	AUC	Accuracy (I.-C. Yeh, C.-h. Lien, 2009)
Logit	0.8100	0.3585	0.7272	0.82
Lasso	0.8082	0.3411	0.7288	-
Ridge	0.8065	0.3195	0.7283	-
KNN	0.8069	0.3810	0.7399	0.84
LDA	0.8094	0.3612	0.7222	0.74
QDA	0.7699	0.5223	0.7359	-
NBC	0.8021	0.5155	0.7309	0.79
CT	0.8224	0.4538	0.6987	0.83
RF	0.8182	0.4724	0.7664	-
SVM	0.8206	0.4460	0.7286	-
GBM	0.8227	0.4786	0.7908	-

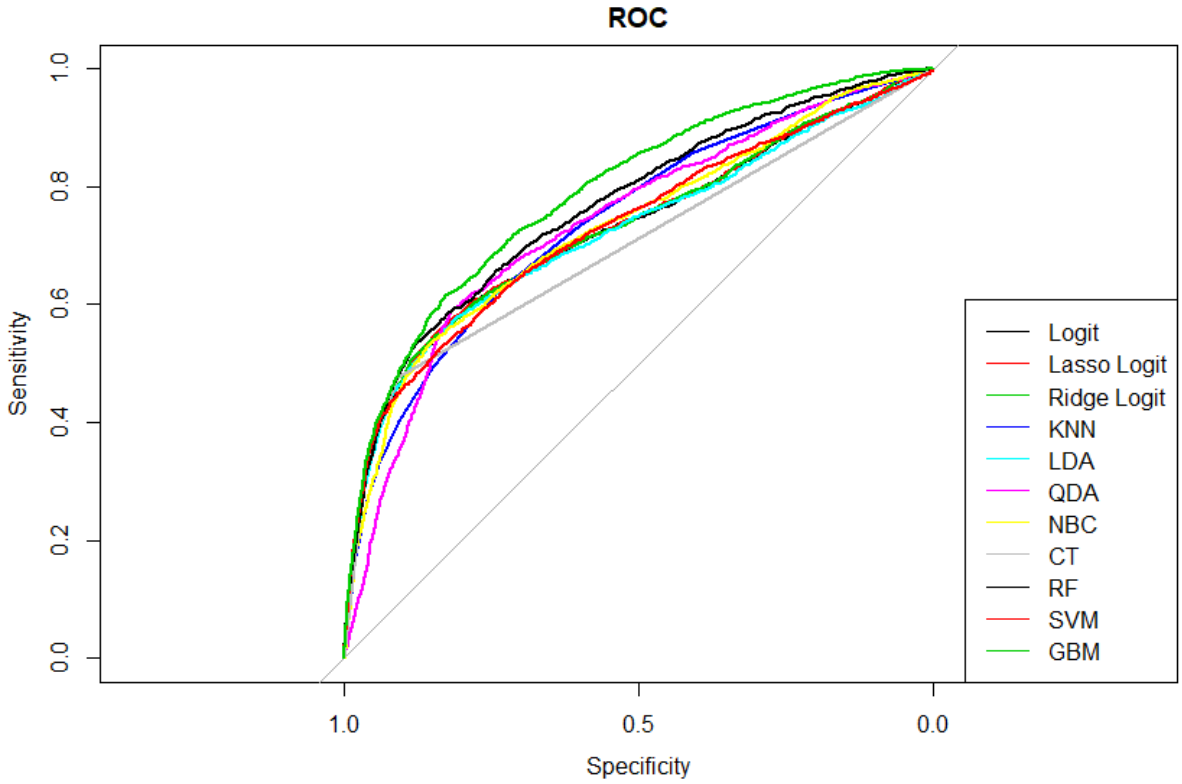


Figure 5: ROC curve

I-Cheng Yeh and Che-hui Lien (2009) use error rate and area ratio as the criterion to judge the models. They also use "Sorting Smoothing Method" to estimate the real probability of default. They do the simple regression, using the real probability of default as the response and the predictive probability of default as the variable, and calculate the coefficient R^2. Under their metric, artificial neural network has largest area ratio and R^2, thus is regarded as the best method to predict probability of default.

Under the error rate measure, we get similar results as stated in the paper. There isn't much difference among various methods, and the error rates are around 0.2. Unlike the paper, we focus on the classification of clients, rather than estimating the probability of default. The reason is that in the testing set we only have real information of binary classification, while it is impossible to know the real probability of default for each client. When I-Cheng Yeh and Che-hui Lien (2009) tried to estimate the real probability of default, much noise may be included, leading to a poor estimate. Then the R^2 measure would be meaningless, or at

least, not convincing.

Under our AUC metric, gradient boosting gives the highest AUC value of 0.7908. Random forest comes the second with an AUC of 0.7664. All the other methods fall below the bar of 0.74. These two methods have the best performance on our data probably because they are tree-based ensemble methods. Since our data size is large and the features are somehow correlated, these two relatively complex models can dig into the structure and figure out those influential variables during the process of bagging or boosting.

5.2 Interpretation

Since we don't focus too much on the interpretation part, we just check the relative influence given by the best model gradient boosting. The top ten are listed in the following table.

Variable	Relative Influence
PAY_1	38.31
PAY_2	3.97
PAY_3	3.45
BILL_AMT1	3.19
PAY_AMT1	3.04
close_2	2.94
close_1	2.91
close_3	2.28
PAY_AMT3	2.25
LIMIT_BAL	2.19

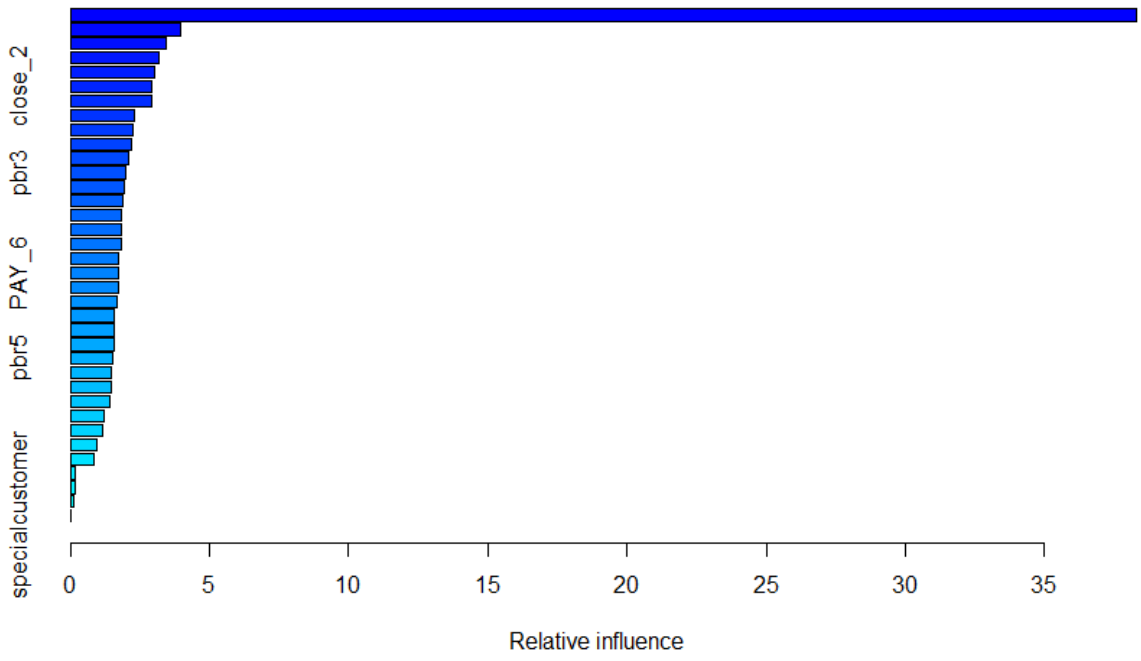


Figure 6: Relative influence in GBM

We can conclude that PAY_1 is a dominating variable in our model, with a relative influence of 38.31. This

actually makes sense since whether a client would default next month is closely related to his payment status this month. One who defaulted before is likely to default again. There are several other features such as latest bill amount and payment amount that may play a role here. If bill amount is high and payment amount is low, the client could be more likely to default next month.

6. Further Discussion

In the paper, we focus on the performance of various machine learning models on our credit default data. Since our data is a little imbalanced and has strong correlation within some of the features, some models may fit better for this kind of data. We conclude that under the metric of AUC, gradient boosting has the best performance in our case.

Our study can be extended or improved in the following aspects. First, we haven't included neural network in the study, since we use R as the coding language and neural network is often performed in TensorFlow, and is extremely time consuming in R when we add multiple layers. In the original paper, the authors argued that artificial neural network is the best model under their measure. We could check how it works under our metric. Furthermore, we could try some techniques of dimension reduction. PCA can be applied to our feature matrix, then we can try the PC logistic regression to see if we can make any improvement on the original logistic regression. Also, we can try to fix the imbalance in our data. Though we think the data imbalance is not very severe in our case, we can still try the Synthetic Minority Over-sampling Technique(SMOTE) and check if the performance of these models would be different.

7. Reference

[1] Yeh, I-Cheng, and Che-hui Lien. "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients." Expert Systems with Applications 36, no. 2 (2009): 2473-2480.

[2] Giudici, Paolo. "Bayesian data mining, with application to benchmarking and credit scoring." Applied Stochastic Models in Business and Industry 17, no. 1 (2001): 69-81.

[3] https://www.kaggle.com/cloudly/credit-default?scriptVersionId=1754738

[4] https://en.wikipedia.org/wiki/Gradient_boosting

[5] https://www.coursera.org/learn/python-machine-learning/lecture/emwn3/gradient-boosted-decision-trees