Prediction Analysis of the Cardholder's Default Payments in Taiwan.

Lin Yi 117020168 Yang Yipeng 117020338 Fan Gang 116010045

May 22, 2020

Abstract

In the 2000s, the banks in Taiwan over-issued cash and credits cards that they aimed to perform risk prediction. The major purpose of risk prediction is to use related financial information to predict individual customer's credit risk. In this study, our goal was going to predict the customer's default payment using their payment data. We set the judging criteria of models as score, and tried different methods including logistic regression, LDA, KNN, SVM and tree models. As a result, we found that the random forest model gives the best predicting performance over the other models tested.

Introduction

In the 2000s, the banks in Taiwan over-issued cash and credits cards. This brought a huge risk to banks if the cardholders choose to default payment. Thus, the banks in Taiwan aimed to perform risk prediction, whose purpose is to predict individual customers' credit risk and to reduce the damage and uncertainty. Many statistical methods have been used to develop models for risk prediction. In order to explore the best statistical method for risk prediction, the payment data in 2005 was collected from a bank in Taiwan in this study. We aimed to predict whether a customer will default payment using their payment data, and we employed methods including logistic regression, linear discriminant analysis, KNN, support vector machine, decision tree and random forest. After comparing the performance of these different methods, we decided to select random forest as our final approach for the purpose of risk prediction.

Materials and Methods

Data

Data Description

The data used in this study contains 30000 observations of 24 variables. Among the included variables, X2-X5 are the demographic information of the cardholders, while X1, X6-X23 are the financial conditions and behaviors for 6 months. X24, default payment, is a binary variable. In this study, we used X24 as response, and other 23 variables as predictors:

X1: (Quantitative) the amount of credit given to his/her family (in NT dollars)

X2: (Categorical) gender (male = 1, female = 0)

X3: (Categorical) education (graduate school = 1, university = 2, high school = 3, others = 4)

X4: (Categorical) marital status (married = 1, single = 2, others = 3)

X5: (Quantitative) age (year)

X6 – X11: (Categorical) past payment records from April to September 2005 (-1 = pay duly; 1 = payment delay for one month;.; 9 = payment delay for nine months and above.)

X12 – X17: (Quantitative) the amount of bill statements from April to September 2005 (in NT dollars)

X18 – X23: (Quantitative) the amount of previous payments from April to September 2005 (in NT dollars)

X24: (Categorical) default payment (Yes = 1, No = 0)

Data Preprocessing

In our study, we standardized the dataset to avoid the problem that may be generated by different range of each variable. And we (equally and randomly) seperated these 30000 observations into 3 categories: training set, validation set and test set. We used the data in the training set to train our models, and then we tested the model with the validation set. Finally when we found the model that can best predict data, we furthur trained this model with all the data in the training set and validation set. Then we tested the fitted model using the test set.

We standardized and seperated the data set using the code below:

```
# Load the data
Credit = read.csv("C:/Users/45037/Desktop/default of credit card clients.csv",skip = 1)
Credit = na.omit(Credit)

# Standardize the dataset
x =  scale(Credit[, 2:24])
default.payment.next.month = Credit[, "default.payment.next.month"]
Credit = data.frame(x, default.payment.next.month)

# Seperate out 10000 observations into the test set
Credit_idx = createDataPartition(Credit$default.payment.next.month,p=2/3,list=FALSE)
Credit_trn = Credit[Credit_idx, ]
Credit_tst = Credit[-Credit_idx, ]

# Seperate out 10000 observations into the validation set
Credit_idx = createDataPartition(Credit_trn$default.payment.next.month,p=1/2,list=FALSE)
Credit_valid = Credit_trn[-Credit_idx, ]
Credit_trn = Credit_trn[Credit_idx, ]
```

By observing our data, 77% of the observations have the response variable as 0 (which implies that 77% of these customers will not default next month). Thus when we fit the model using this dataset, the predicted value might be biased to the class 0. In order to solve this problem, we will use the SMOTE approach to balance the training set:

```
# SMOTE
Credit_trn$default.payment.next.month = as.factor(Credit_trn$default.payment.next.month)
Credit_trn = SMOTE(default.payment.next.month~., Credit_trn, perc.over=150)
Credit_trn$default.payment.next.month = as.numeric(Credit_trn$default.payment.next.month)-1
```

Apart from that, the correlation among different variables could lead bias to their significant level, so we the plot out the correlation graph of variables and obtained that several features are indeed highly correlated with each other. Thus, there might exist the multicollinearity problem.

```
# Correlation
bc = cor(x)
corrplot.mixed(bc, tl.pos="lt")
```

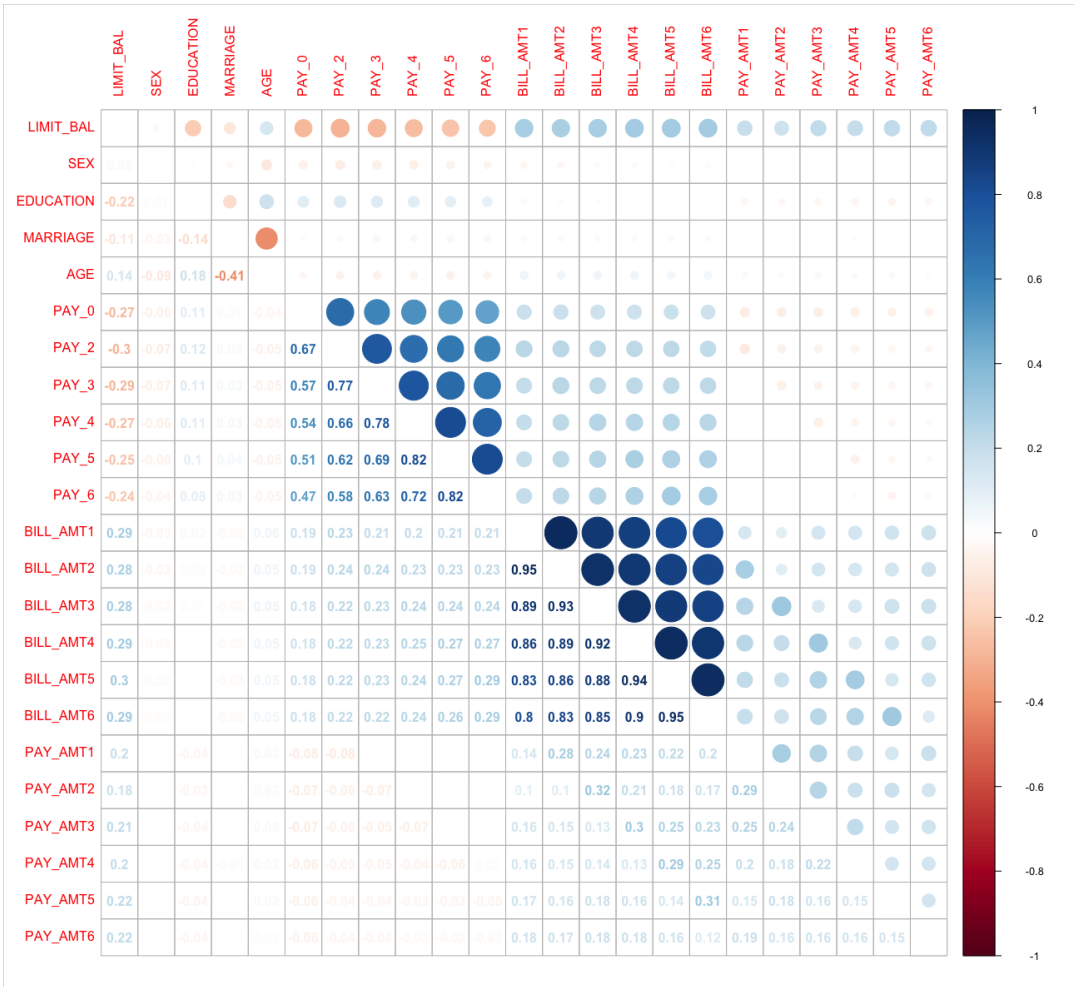


Figure 1: Correlation

VIF is the benchmark to evaluate the degree of multicollinearity. Usually, we consider a VIF larger than 10 as the existence of the multicollinearity problem. Hence, we obtained that several predictors are above 10, so we might truly face the problem of multicollinearity.

```
glm.fit=glm(default.payment.next.month~.,data=Credit_trn,family="binomial")
summary(glm.fit)
```

```
vif(glm.fit)
```

```
 LIMIT_BAL       SEX EDUCATION  MARRIAGE       AGE     PAY_0     PAY_2     PAY_3     PAY_4     PAY_5     PAY_6
  1.597347  1.022891  1.159670  1.231957  1.278795  1.626343  2.790059  3.457761  4.062822  4.688180  3.279933
BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5
27.905181 52.579707 38.611271 29.096883 29.536185 16.806263  1.514341  1.916846  1.561566  1.327852  1.473072
 PAY_AMT6
  1.143400
```

Figure 2: VIF

Since several approaches, for example, PCR and LDA, required the pre-requisite condition that all the variables should follow the Guassian distribution, we printed out each variable seperately by plots which set that x-axis to be the variable's values and the y-axis as the times of occurence. From the graph, we noticed that although some variables were not perfectly follow the Guassian distribution, however, their distribution shapes seemingly to be part of the Gussian distribution. Thus, we assume that the dataset indeed satisfied this pre-requisite condition.

```
par(mfrow=c(3,2))
hist(Credit$LIMIT_BAL,main="",col="lightsteelblue3")
hist(Credit$AGE,main="",col="lightsteelblue3")
hist(Credit$BILL_AMT1,main="",col="lightsteelblue3")
hist(Credit$BILL_AMT2,main="",col="lightsteelblue3")
hist(Credit$BILL_AMT3,main="",col="lightsteelblue3")
hist(Credit$BILL_AMT4,main="",col="lightsteelblue3")
hist(Credit$BILL_AMT5,main="",col="lightsteelblue3")
hist(Credit$BILL_AMT6,main="",col="lightsteelblue3")
hist(Credit$PAY_AMT1,main="",col="lightsteelblue3")
hist(Credit$PAY_AMT2,main="",col="lightsteelblue3")
hist(Credit$PAY_AMT3,main="",col="lightsteelblue3")
hist(Credit$PAY_AMT4,main="",col="lightsteelblue3")
hist(Credit$PAY_AMT5,main="",col="lightsteelblue3")
hist(Credit$PAY_AMT6,main="",col="lightsteelblue3")

par(mfrow=c(1,1))
```

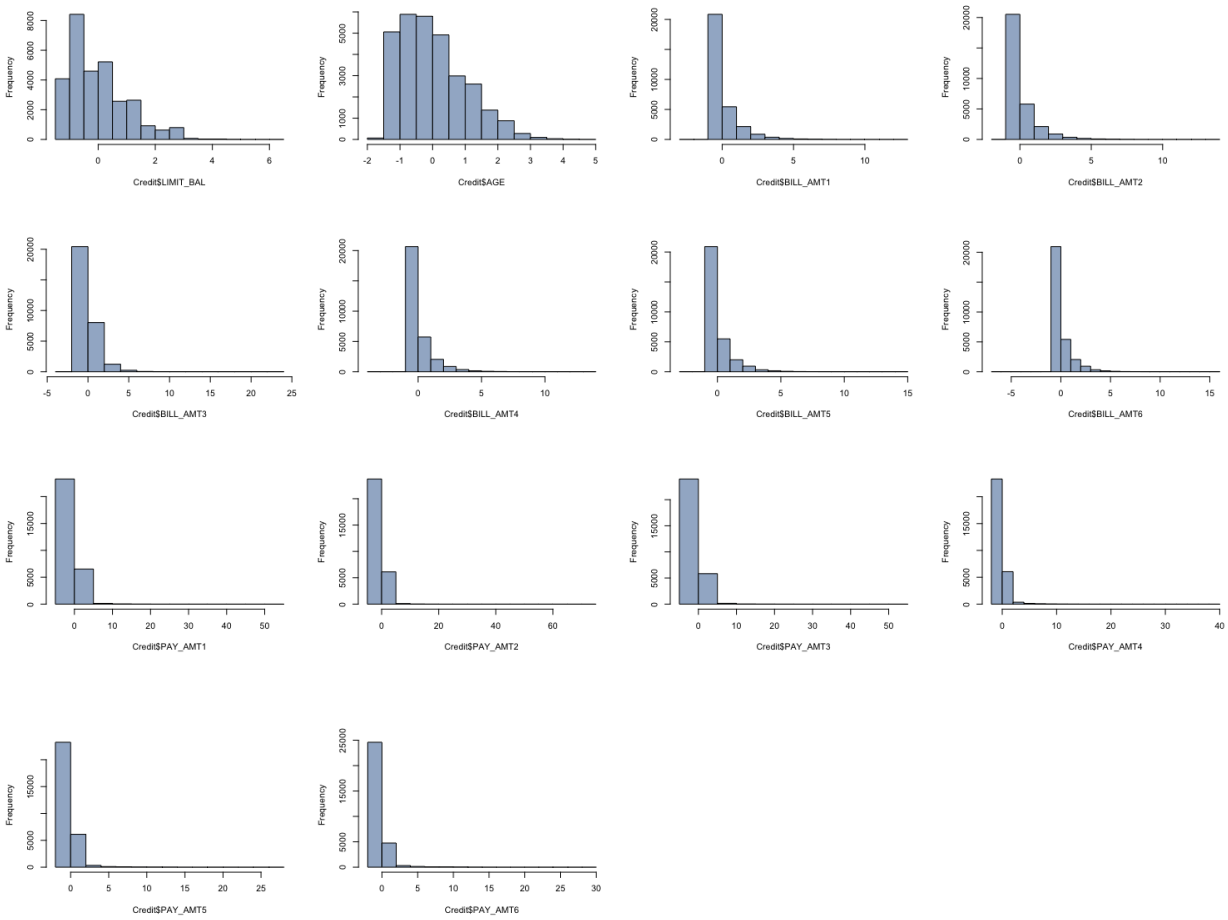


Figure 3: Normal Distribution

Method

Model Evaluation Method

To evaluate prediction performance of different models, we set our judging criteria as score, and accuracy for reference. When a customer choose to default next month, it will bring a huge risk to the bank. Thus we need to give a penalty when a model predicts a customer who is going to default next month to not default. In other words, when an observation has the response variable as value 1 but the model predicted it to be 0, we will punish the model in terms of its score. From the historical data and papers, we calculated and learned that the cost of false predicting 1 as 0 is 5 times higher than the cost of false predicting 0 as 1. Considering the total credit given to the customers that will default next month, the formula of the score is given by:

$$Score = 1 \times (default\ labeled\ default) + 1 \times (not\ default\ labeled\ not\ default)$$

$$-1 \times (not\ default\ labeled\ default) - 5 \times (default\ labeled\ not\ default)$$

Logistic Regression Model

We launched several approaches to tackle the multicollinearity problem: using the stepwise approach to do the best subset selection, dimension reduction such as PCR and LDA, or Ridge and Lasso regression to reduce weight of unimportant predictors. First of all, we started with the logistic regression as the base-line model. We fitted the training dataset into the model and obtained its performance as followed:

```
# Fit the basic model
glm.fit=glm(default.payment.next.month~.,data=Credit_trn,family="binomial")
summary(glm.fit)

# Find the threshold that can maximize the score
thre = seq(1,99)/100
payment_logi = rep(NA,10)
thershold = rep(NA,10)
result = rep(NA,99)
score = rep(NA,99)
glm.prob=predict(glm.fit,Credit_valid,type="response")
for (j in 1:99){
  glm.pred=rep(0,length(Credit_valid[, 24]))
  glm.pred[glm.prob>thre[j]]=1
  table.glm = table(Credit_valid[, 24], glm.pred)
  result[j] = mean(Credit_valid[, 24] == glm.pred)
  score[j] = table.glm[1,1]+table.glm[2,2]-table.glm[1,2]-5*table.glm[2,1]
}
thershold_logi = which.max(score)/100
thershold_logi
score_logi = score[which.max(score)]
score_logi
accuracy_logi = result[which.max(score)]
accuracy_logi
# Accuracy: 0.7981
# Score: 1274
```

Now we observed that almost half of the predictors are insignificant to the regression. This phenomenon might be due to the multicollinearity problem. In order to determine which of the variables are significant to the regression, we tried to do feature selection by using backward-stepwise selection:

```
Call:
glm(formula = default.payment.next.month ~ ., family = "binomial",
    data = Credit_trn)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2467  -1.0644   0.0389   1.0550   2.9396

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.223875   0.024914  -8.986  < 2e-16 ***
LIMIT_BAL   -0.162864   0.031820  -5.118 3.08e-07 ***
SEX         -0.054297   0.023538  -2.307 0.021070 *
EDUCATION   -0.119379   0.026148  -4.565 4.98e-06 ***
MARRIAGE    -0.104115   0.025619  -4.064 4.82e-05 ***
AGE          0.021804   0.026194   0.832 0.405181
PAY_0        0.556836   0.031096  17.907  < 2e-16 ***
PAY_2        0.153445   0.039029   3.932 8.44e-05 ***
PAY_3        0.034959   0.043515   0.803 0.421757
PAY_4        0.041546   0.046956   0.885 0.376276
PAY_5       -0.060991   0.049702  -1.227 0.219769
PAY_6        0.038055   0.041738   0.912 0.361893
BILL_AMT1   -0.520668   0.131011  -3.974 7.06e-05 ***
BILL_AMT2   -0.039539   0.178117  -0.222 0.824326
BILL_AMT3    0.500846   0.152302   3.289 0.001007 **
BILL_AMT4   -0.188830   0.131580  -1.435 0.151259
BILL_AMT5    0.452251   0.132427   3.415 0.000638 ***
BILL_AMT6   -0.247734   0.101382  -2.444 0.014543 *
PAY_AMT1    -0.316473   0.061451  -5.150 2.60e-07 ***
PAY_AMT2    -0.324400   0.070745  -4.585 4.53e-06 ***
PAY_AMT3    -0.002409   0.042796  -0.056 0.955112
PAY_AMT4    -0.272708   0.055673  -4.898 9.66e-07 ***
PAY_AMT5     0.056437   0.038939   1.449 0.147238
PAY_AMT6    -0.048334   0.034373  -1.406 0.159675
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 12183  on 8787  degrees of freedom
Residual deviance: 10634  on 8764  degrees of freedom
AIC: 10682

Number of Fisher Scoring iterations: 4
```

Figure 4: Logistic Summary

```
# Try stepwise subset selection
glm.new = step(glm.fit, direction = "backward")
summary(glm.new)

# Now all the selected variables are significant to the regression
thre = seq(1,99)/100
payment_logi = rep(NA,10)
thershold = rep(NA,10)
result = rep(NA,99)
score = rep(NA,99)
glm.prob=predict(glm.new,Credit_valid,type="response")
for (j in 1:99){
  glm.pred=rep(0,length(Credit_valid[, 24]))
  glm.pred[glm.prob>thre[j]]=1
  table.glm = table(Credit_valid[, 24], glm.pred)
  result[j] = mean(Credit_valid[, 24] == glm.pred)
  score[j] = table.glm[1,1]+table.glm[2,2]-table.glm[1,2]-5*table.glm[2,1]
}
thershold_logi = which.max(score)/100
thershold_logi
score_logi = score[which.max(score)]
score_logi
accuracy_logi = result[which.max(score)]
accuracy_logi
# Accuracy: 0.7718
# Score: 1328
```

After the feature selection, we found the score and accuracy were improved. Then we also launched the ridge using the whole set of predictors to reduce the contribution of the unimportant variables, and the lasso to do the feature selection. We found that both approaches' performance indeed improved comparing to the logistic regression.

```
# Ridge
x = model.matrix(default.payment.next.month~., Credit)[,-1]
y = Credit[,"default.payment.next.month"]
x_trn_matrix = model.matrix(default.payment.next.month~., Credit_trn)[,-1]
x_tst_matrix = model.matrix(default.payment.next.month~., Credit_valid)[,-1]
cv.ridge = cv.glmnet(x_trn_matrix, Credit_trn[, 24],
                     alpha=0, family="binomial", nfolds = 10)
bestlam=cv.ridge$lambda.min

out = glmnet(x,y,alpha=0,lambda=bestlam)
ridge.coef = predict(out,type="coefficients",s=bestlam)
ridge.coef

thre = seq(1,100)/100
result = rep(NA,100)
score = rep(NA,100)

ridge.mod = glmnet(x_trn_matrix,Credit_trn[, 24],alpha=0,lambda=bestlam,family="binomial")
ridge.pred = predict(ridge.mod,s=bestlam,newx=x_tst_matrix)
for (j in 1:100){
  glm.pred=rep(0,length(Credit_valid[, 24]))
  glm.pred[ridge.pred>thre[j]]=1
```

```
  glm.table = table(Credit_valid[, 24], glm.pred)
  result[j] = (glm.table[1,1]+glm.table[2,2])/sum(glm.table)
  score[j] = glm.table[1,1]+glm.table[2,2]-glm.table[1,2]-5*glm.table[2,1]
}
thershold_ridge = which.max(score)/100
thershold_ridge
score_ridge = score[which.max(score)]
score_ridge
accuracy_ridge = result[which.max(score)]
accuracy_ridge
# Accuracy: 0.7893
# Score: 1378
```

```
# Lasso
out=glmnet(x,y,alpha=1,lambda=bestlam)
lasso.coef=predict(out,type="coefficients",s=bestlam)
lasso.coef
thre = seq(1,100)/100
payment_lasso = rep(NA,10)
thershold = rep(NA,10)
result = rep(NA,100)
score = rep(NA,100)

lasso.mod = glmnet(x_trn_matrix,Credit_trn[, 24],alpha=1,lambda=bestlam,family="binomial")
lasso.pred = predict(lasso.mod,s=bestlam,newx=x_tst_matrix)
for (j in 1:100){
  glm.pred = rep(0,length(Credit_valid[, 24]))
  glm.pred[lasso.pred>thre[j]]=1
  table.lasso = table(Credit_valid[, 24], glm.pred)
  result[j] = (table.lasso[1,1]+table.lasso[2,2])/sum(table.lasso)
  score[j] = table.lasso[1,1]+table.lasso[2,2]-table.lasso[1,2]-5*table.lasso[2,1]
}

thershold_lasso = which.max(score)/100
thershold_lasso
score_lasso = score[which.max(score)]
score_lasso
accuracy_lasso = result[which.max(score)]
accuracy_lasso
# Accuracy: 0.7844
# Score: 1380
```

Principal Components Regression

Then we tried the Principal Components Regression (PCR) to reduce the dataset's dimension. It turned out that 6 components were enough to obtain a model that has a low enough prediction error, so we applied these components as our new factors to fit the PCR. The result turned out that it could only slightly improve the performance.

```
# PCR
thre = seq(1,100)/100
payment_pca = rep(NA,10)
thershold = rep(NA,10)
```

```
result = rep(NA,100)
score = rep(NA,100)
pcr = pcr(default.payment.next.month~., data=Credit_trn,
          scale =FALSE, validation = "CV", family=binomial)
validationplot(pcr, val.type = "MSEP")
pca.pred = predict(pcr, Credit_valid, ncomp=6)
for (j in 1:100){
  pca.prob=rep(0,length(Credit_valid[, 24]))
  pca.prob[pca.pred>thre[j]]=1
  pca.table = table(Credit_valid[, 24],pca.prob)
  result[j] = (pca.table[1,1]+pca.table[2,2])/sum(pca.table)
  score[j] = pca.table[1,1]+pca.table[2,2]-pca.table[1,2]-5*pca.table[2,1]
}
thershold = mean(which.max(score))/100
thershold
accuracy_pca = result[which.max(score)]
accuracy_pca
score_pca = score[which.max(score)]
score_pca
# Accuracy: 0.7901
# Score: 1338
```

Linear Discriminant Analysis

Next, we considered LDA. A good performance of LDA depends on some assumptions:

1. Normality assumption: observations are drawn from the normal distribution with common variance in each class.
2. Non-multicollinearity assumption: performance of an LDA model might become worse if the correlation between variables increased.

Although we had already verify the normality condition, however, some of the variables were not perfectly followed the normal distribution and there was a multicollinearity problem is our data set. Hence, we did not highly expect LDA would work well in this study, but we still tried it and considered LDA method as a candidate.

```
# LDA
model_lda = lda(default.payment.next.month ~ ., data = Credit_trn)
temp = predict(model_lda, newdata = Credit_valid)
prediction_lda = ifelse(temp$x > 0.5, 1, 0)
error_rate_lda = sum((prediction_lda - Credit_valid[, 24]) ^ 2) / length(prediction_lda)
score = 0
for (i in 1:length(prediction_lda)) {
  if (prediction_lda[i] == 0 && Credit_valid[, 24][i] == 0) score = score + 1
  if (prediction_lda[i] == 1 && Credit_valid[, 24][i] == 1) score = score + 1
  if (prediction_lda[i] == 1 && Credit_valid[, 24][i] == 0) score = score - 1
  if (prediction_lda[i] == 0 && Credit_valid[, 24][i] == 1) score = score - 5
}
1 - error_rate_lda
score
# Accuracy: 0.8017
# Score: 1186
```

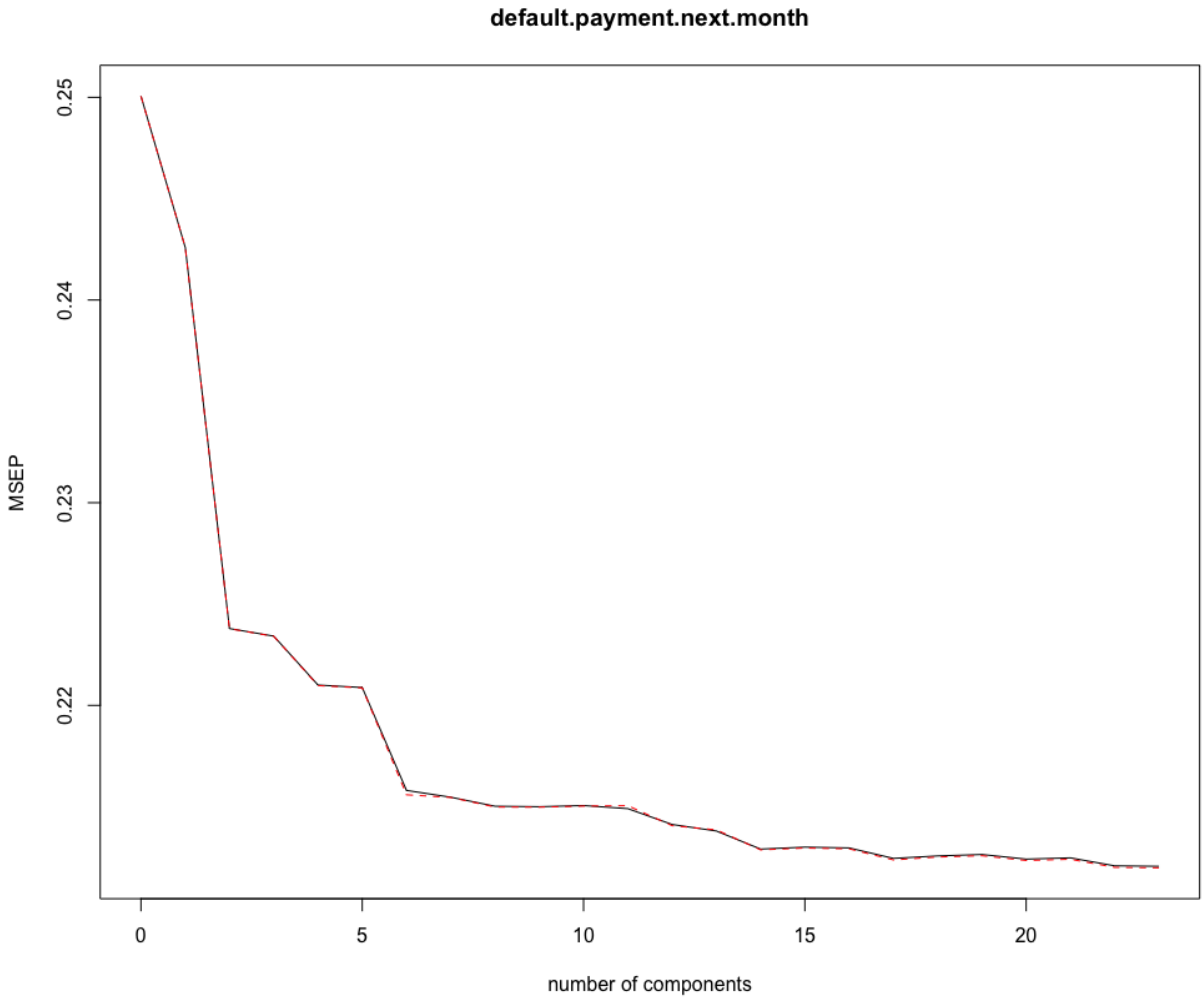

default.payment.next.month
MSEP
0.25
0.24
0.23
0.22
0
5
10
15
20
number of components

Figure 5: PCR

KNN

For KNN, it might have better performance than linear models when the decision boundary is not linear. Thus, we applied KNN method to our data. We selected k using validation set approach. In rule of thumb, k is usually around square root of size of training set. We tried k from 20 to 300, each with an interval of 20. The plot and result show that when k equals to 160, the score reached the highest. Thus, we chose k=160 in our KNN model.

```
# KNN
set.seed(101)
train.x = as.matrix(Credit_trn[,-24])
train.y = Credit_trn$default.payment.next.month
test.x = as.matrix(Credit_valid[,-24])
acc = rep(0,15)
score_knn = rep(0,15)
for (i in 1:15){
  knn_pred = knn(train.x,test.x,train.y,k=20*i,prob = FALSE)
  t_knn = table(knn_pred, Credit_valid$default.payment.next.month)
  acc[i] = mean(knn_pred == Credit_valid$default.payment.next.month)
  score_knn[i] = t_knn[1,1]+t_knn[2,2]-t_knn[2,1]-5*t_knn[1,2]
}
score_knn
which.max(score_knn)
acc
knn_k = seq(20,300,30)
plot(knn_k,score_knn,type = "b")
set.seed(101)
knn.pred = knn(train.x,test.x,train.y,k=160,prob = FALSE)
t.knn = table(knn.pred, Credit_valid$default.payment.next.month)
mean(knn.pred == Credit_valid$default.payment.next.month)
t.knn[1,1]+t.knn[2,2]-t.knn[2,1]-5*t.knn[1,2]
# Accuracy: 0.7792
# Score: 1496
```

Support Vector Machine

For the SVM approach, we performed feature selection: fit a basic SVM model to the data set and then select the 8 variables that have the largest absolute value of coefficients. Then we train the new SVM model based on these 8 variables that should be most "useful".

We also discovered that it gives the best description when the kernel is set to be "linear" instead of "radial" or "sigmoid". This might be caused by the fact that several variables in our data set are categorical variables. Because these categorical variables are discrete and they can only take a limited number of values, when they are mapped to a higher dimensional space (during the process of the SVM approach, the data will be mapped to a higher dimensional space), the decision boundary may tend to be linear.

```
# SVM
# Feature selection
fit_svm = svm(default.payment.next.month ~., data = Credit_trn, kernel = "linear", cost = 0.001, scale =
Credit_trn = Credit_trn[, c(order(abs(coef(fit_svm)[2:24]))[16:23], 24)]
Credit_valid = Credit_valid[, c(order(abs(coef(fit_svm)[2:24]))[16:23], 24)]

# Training the model
```

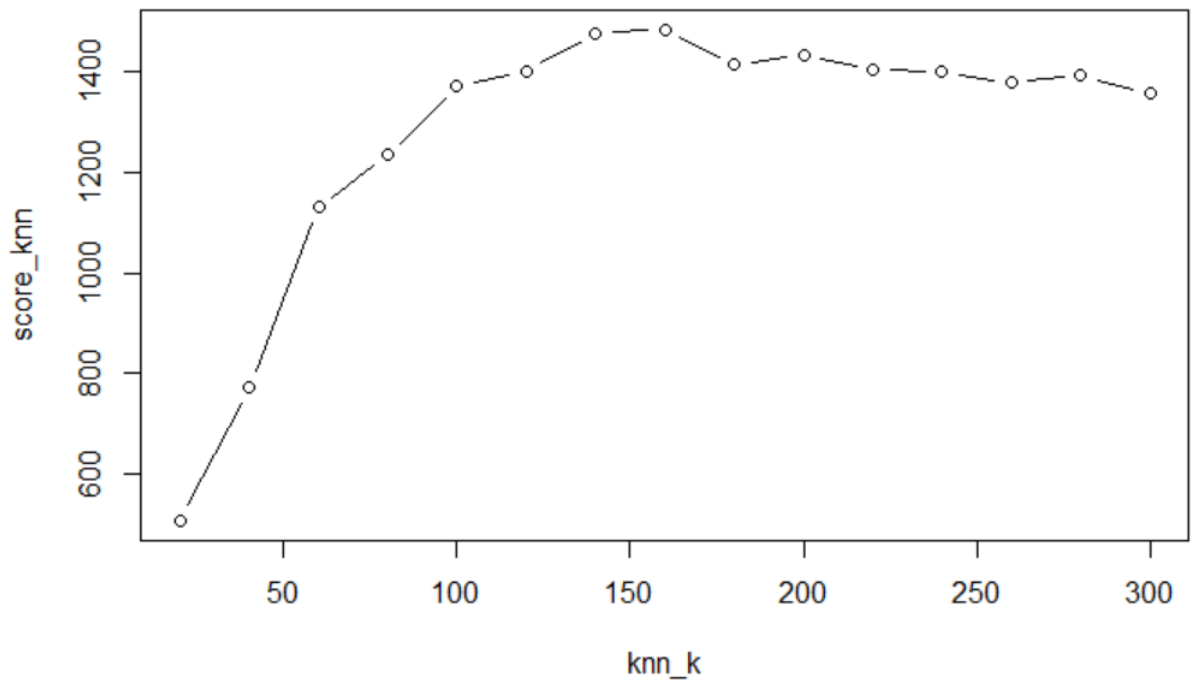


Figure 6: KNN

```
fit_svm = svm(default.payment.next.month ~., data = Credit_trn, kernel = "linear", cost = 1, scale = FA
prediction_svm = predict(fit_svm, newdata = Credit_valid)
prediction_svm = as.vector(ifelse(as.numeric(prediction_svm) > 1.5, 1, 0))
error_rate_svm = sum((prediction_svm - Credit_valid[, 9]) ^ 2) / length(prediction_svm)
score = 0
for (i in 1:length(prediction_svm)) {
  if (prediction_svm[i] == 0 && Credit_valid[, 9][i] == 0) score = score + 1
  if (prediction_svm[i] == 1 && Credit_valid[, 9][i] == 1) score = score + 1
  if (prediction_svm[i] == 1 && Credit_valid[, 9][i] == 0) score = score - 1
  if (prediction_svm[i] == 0 && Credit_valid[, 9][i] == 1) score = score - 5
}
1 - error_rate_svm
score
# Accuracy: 0.7876
# Score: 1392
```

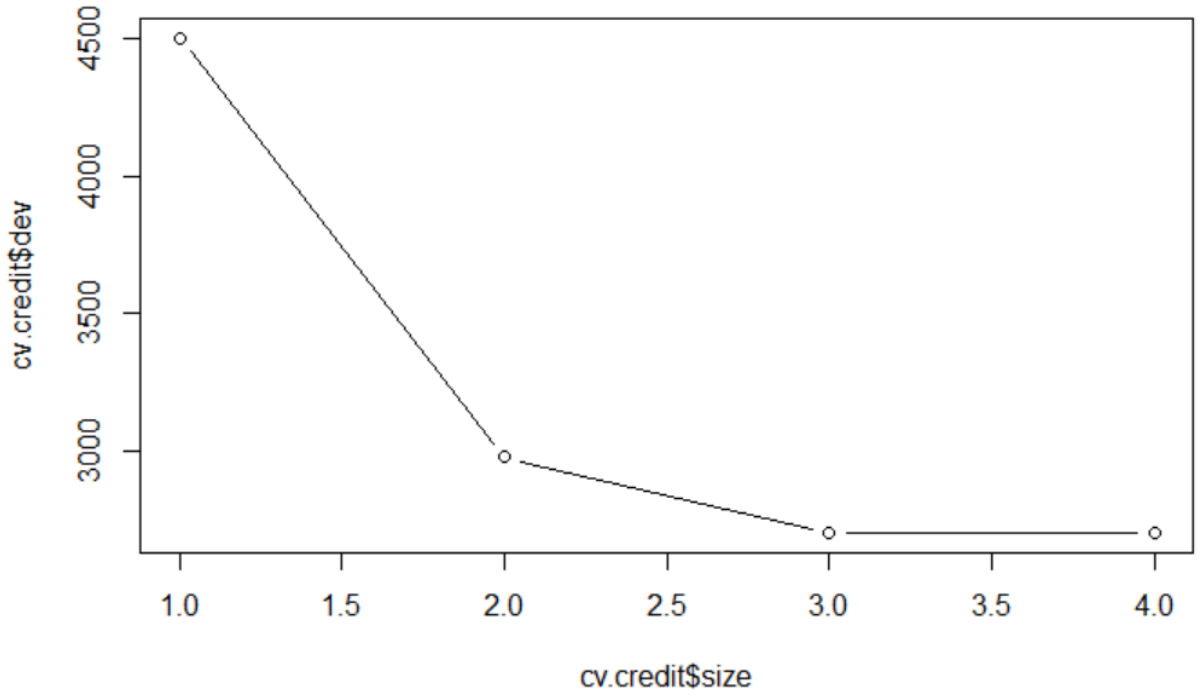
Decision Tree

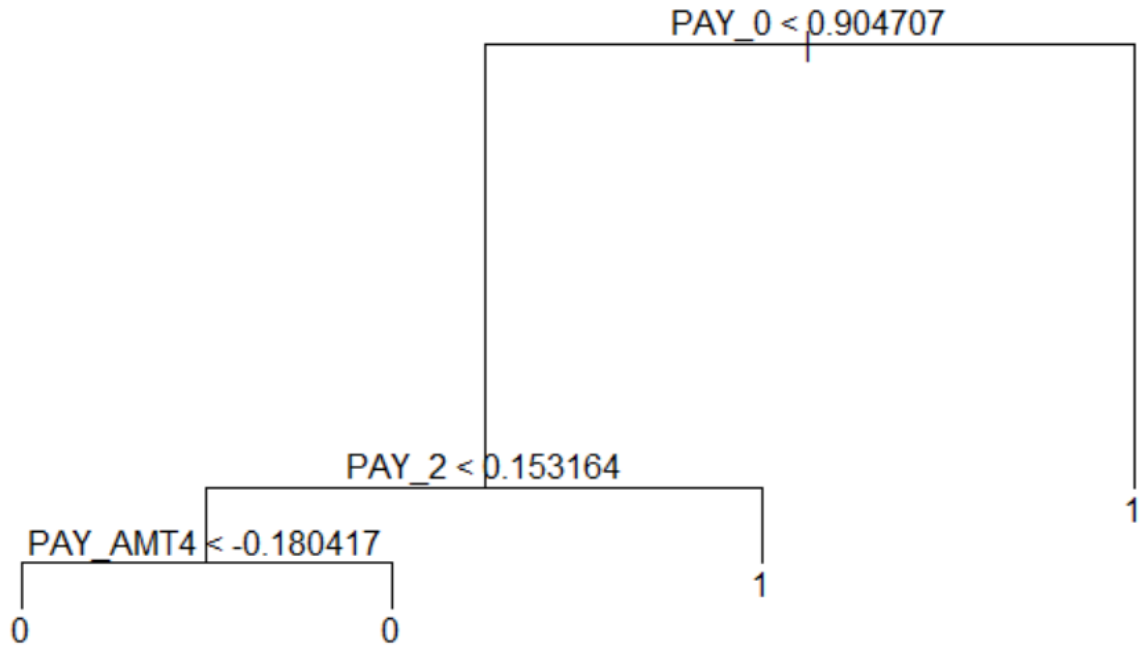
For decision tree, we first built up a tree, and used cross validation to see whether we need to prune. The unpruned tree has 4 terminal nodes and then we used cross validation to prune the tree. The plot shows that increasing size from 1 to 4, the test error decreased, so we kept our tree unpruned. From the structure plot of the tree, we learned that the variables actually used are PAY_0, PAY_2, and PAY_AMT2. PAY_0, which is the past payment records in April, is the most important feature since it is in the first layer.

```
# Decision Tree
payfactor.trn = as.factor(Credit_trn$default.payment.next.month)
```

```r
payfactor.valid = as.factor(Credit_valid$default.payment.next.month)
trn = data.frame(Credit_trn[,-24], payfactor.trn)
valid = data.frame(Credit_valid[,-24], payfactor.valid)
tree.fit = tree(payfactor.trn~.,trn)
summary(tree.fit)
set.seed(100)
cv.credit = cv.tree(tree.fit,FUN=prune.misclass)
cv.credit
plot(cv.credit$size,cv.credit$dev,type = 'b')
plot(tree.fit)
text(tree.fit,pretty = 0)
tree.pred = predict(tree.fit,valid,type="class")
t.tree=table(tree.pred,valid$payfactor.valid)
(t.tree[1,1]+t.tree[2,2])/length(valid$payfactor.valid)
t.tree[1,1]+t.tree[2,2]-t.tree[2,1]-5*t.tree[1,2]
# Accuracy: 0.8049
# Score: 1370
```

```
Classification tree:
tree(formula = payfactor.trn ~ ., data = trn)
Variables actually used in tree construction:
[1] "PAY_0"    "PAY_2"    "PAY_AMT4"
Number of terminal nodes:  4
Residual mean deviance:  1.154 = 10140 / 8784
Misclassification error rate: 0.3075 = 2702 / 8788
$size
[1] 4 3 2 1

$dev
[1] 2703 2703 2976 4502

$k
[1] -Inf    0  273 1419

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
```





Random Forest

Further, we tried random forest approach. It can reduce the variance of the trees, which provides an improvement over decision trees. In addition, we can directly use all of our variables without dimension reduction and feature selection. Each time, a random selection of m predictors will be chosen from the full set of predictors. We tried different number of m, and found that when m=1, score achieved its highest.

```r
# Random Forest
set.seed(13)
vscore = rep(0,6)
for (i in 1:6){
  vrf = randomForest(payfactor.trn~.,mtry=i,data=trn)
  vpred = predict(vrf,valid)
  tablev = table(vpred,valid$payfactor.valid)
  vscore[i] = tablev[1,1]+tablev[2,2]-tablev[2,1]-5*tablev[1,2]
}
vscore
which.max(vscore)
set.seed(13)
rf.fit = randomForest(payfactor.trn~.,mtry=1,data = trn,importance=TRUE)
rf.pred = predict(rf.fit,valid)
mean(rf.pred==valid$payfactor.valid)
t.rf = table(rf.pred,valid$payfactor.valid)
t.rf[1,1]+t.rf[2,2]-t.rf[2,1]-5*t.rf[1,2]
# Accuracy: 0.7714
# Score: 1816
```

Results and Discussion

Results

After trying all six datamining methods, we can summarize our result (using the highest score obtained in each method):

method	accuracy	score
Logistic	0.7981	1274
Logistic + Best subset	0.7718	1328
Ridge	0.7893	1378
Lasso	0.7844	1380
PCR	0.7901	1338
LDA	0.8017	1186
KNN	0.7792	1496
SVM	0.7876	1392
Decision Tree	0.8049	1370
Random Forest	0.7714	1816

Thus, we found that the random forest model earns the highest score among all data mining methods, which is 1816. We choose the random forest model to be the final model and train it with all the data except those in the test set, and finally get the score of 1768 on test dataset.

```r
# Final Approach - Random Forest
# Data Preprocess
Credit = read.csv("C:/Users/45037/Desktop/default of credit card clients.csv",skip = 1)
Credit = na.omit(Credit)
x =  scale(Credit[, 2:24])
default.payment.next.month = Credit[, "default.payment.next.month"]
Credit = data.frame(x, default.payment.next.month)
Credit_idx = createDataPartition(Credit$default.payment.next.month,p=2/3,list=FALSE)
```

```
Credit_trn = Credit[Credit_idx, ]
Credit_tst = Credit[-Credit_idx, ]
Credit_trn$default.payment.next.month = as.factor(Credit_trn$default.payment.next.month)
Credit_trn = SMOTE(default.payment.next.month~., Credit_trn, perc.over=150)

# Fit the Model
set.seed(13)
rf.fit = randomForest(default.payment.next.month~.,mtry=1,
                      data = Credit_trn,importance=TRUE)
importance(rf.fit,type=1)
varImpPlot(rf.fit)
rf.pred = predict(rf.fit,Credit_tst)
mean(rf.pred==Credit_tst$default.payment.next.month)
t.rf = table(rf.pred,Credit_tst$default.payment.next.month)
t.rf[1,1]+t.rf[2,2]-t.rf[2,1]-5*t.rf[1,2]
# Accuracy: 0.7638
# Score: 1768
```

Discussion

As the base-line model, we viewed logistic regression's accuracy and score as the basic benchmark for future comparison. Feature selection could lower the influence of insignificant predictors and so as the method of adding a penalty term, so stepwise subset seletion together with ridge and lasso could perform better under the explosure of multicollinearity problem. Dimension deduction was the linear combination of the existing predictors, so there existed the condition that the less important predictors were put more weight while constructing the new variables. Hence, the performance improved not much under PCR. Due to the situation that some variables were not compeletely followed the normal distribution and the existence of the multicollinearity performance, LDA indeed could hardly perform better. SVM model had the best performance when the kernel is set to be "linear" instead of "radial" or "sigmoid". It may be mainly because many variables are categorical and only take limited values, so that data can be divided by linear boundary. However, result shows that SVM model in our study gives us the lowest score.

Among these models, nonlinear models, for example, KNN, decision tree, and random forest outperform others. It suggests that the data probably has nonlinear decision boundary. In addition, flaws in our dataset such as multicollinearity problem have little effect on the prediction capability of these models. Hence, these models have

Comparing to all other models, random forest gives the most satisfactory prediction result. It can not only overcome the flaws of our dataset, but also decrease the variance of decision trees because of its advanced algorithm. Hence, we chose random forest as our final approach to reach our prediction goal. We trained model using all the training and validation data. The final approach gives us score of 1768 on test dataset, which is close to the estimated score using validation set approach (since the validation set has the same sample size with the test set, we can evaluate the score with the previous estimated one). The test accuracy is 0.7691, which is also not bad. Overall, the model shows good prediction performance.

Conclusion

In this project, we mainly used score as our criteria to choose the model which can best describe our data. We put more penalty on false negative error, since this type will generate more cost in the reality. In the data pre-processing step, we scaled data to avoid the issue that may be caused by different units of features and used SMOTE to deal with imbalanced data.

Considering the potential multicollinearity problem, we tried logistic regression with feature selection, lasso and ridge regression, LDA, KNN, SVM and Trees. From the accuracy aspect, the abilities of models are not much different, however, if we look at scores, they are quite different. And the random forest model gave the highest score among all these methods. So finally, we chose random forest as our final approach. When we test this approach using the data in the test set, we earned a score as 1768 which is one of the highest scores we have found.

References

1. Swallow, L. (2020). Random Forest. Retrieved from: https://blog.csdn.net/xingxing1839381/article/details/81273793

2. NIeson, T. (2020). Solutions to multicollinearity problems. Retrieved from: https://blog.csdn.net/nieson2012/article/details/48980491

3. Microstrong. (2020). Explanation of the principal component analysis. Retrieved from: https://blog.csdn.net/program_developer/article/details/80632779

4. 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.

5. The dataset can be retrieved from the UCI Machine Learning Repository website: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients