Code ▾

Project: Credit card default prediction

Feb.13th 2024 Chenbo Huang, Jiayi Wu, Selina Wang, Yiting Hu, Ziao Xiong

1. Introduction

1.1 Problem description Predicting whether a client will default on their credit card payment has been a significant part of risk assessment for credit card companies. In our project, we used 8 models to predict the case of default based on a collection of demographic, repayment status and amount records of clients in Taiwan from April to September 2005.

1.2 Data description The dataset is acquired from Kaggle: https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset?resource=download (https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset?resource=download)

This dataset contains information on default payments, demographic factors, history of repayment status, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. It contains records of 30000 customers with 25 features. Below is a description of them:

- ID: ID of each client
- LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit)
- SEX: Gender (1 = male, 2 = female)
- EDUCATION: (0 = others, 1 = graduate school, 2 = university, 3 = high school, 4 = others, 5 = unknown, 6 = unknown)
- MARRIAGE: Marital status (0 = others, 1 = married, 2 = single, 3 = others)
- AGE: Age in years Scale of PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, and PAY_6: -2 = no consumption, -1 = paid in full, 0 = use of revolving credit (paid minimum only), 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
- PAY_0: Repayment status in September, 2005
- PAY_2: Repayment status in August, 2005
- PAY_3: Repayment status in July, 2005
- PAY_4: Repayment status in June, 2005
- PAY_5: Repayment status in May, 2005
- PAY_6: Repayment status in April, 2005
  All amount of bill statement are shown in New Taiwan dollar (NT dollar)
- BILL_AMT1: Amount of bill statement in September, 2005
- BILL_AMT2: Amount of bill statement in August, 2005
- BILL_AMT3: Amount of bill statement in July, 2005
- BILL_AMT4: Amount of bill statement in June, 2005
- BILL_AMT5: Amount of bill statement in May, 2005
- BILL_AMT6: Amount of bill statement in April, 2005
- PAY_AMT1: Amount of previous payment in September, 2005
- PAY_AMT2: Amount of previous payment in August, 2005
- PAY_AMT3: Amount of previous payment in July, 2005
- PAY_AMT4: Amount of previous payment in June, 2005

- PAY_AMT5: Amount of previous payment in May, 2005
- PAY_AMT6: Amount of previous payment in April, 2005
- default.payment.next.month: Default payment (0 = no, 1 = yes)

2. Exploratory Data Analysis

Hide

```
ccard = read.csv("/Users/selina.wang/Desktop/Junior Spring/DAT500S final project/Credit_
card_default_prediction/UCI_Credit_Card.csv")
```

```
Warning message:
R graphics engine version 15 is not supported by this version of RStudio. The Plots tab
will be disabled until a newer version of RStudio is installed.
```

Hide

```
attach(ccard)
str(ccard)
```

```
'data.frame':	30000 obs. of  25 variables:
 $ ID                        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ LIMIT_BAL                 : num  20000 120000 90000 50000 50000 50000 500000 100000 1
40000 20000 ...
 $ SEX                       : int  2 2 2 2 1 1 1 2 2 1 ...
 $ EDUCATION                 : int  2 2 2 2 2 1 1 2 3 3 ...
 $ MARRIAGE                  : int  1 2 2 1 1 2 2 2 1 2 ...
 $ AGE                       : int  24 26 34 37 57 37 29 23 28 35 ...
 $ PAY_0                     : int  2 -1 0 0 -1 0 0 0 0 -2 ...
 $ PAY_2                     : int  2 2 0 0 0 0 0 -1 0 -2 ...
 $ PAY_3                     : int  -1 0 0 0 -1 0 0 -1 2 -2 ...
 $ PAY_4                     : int  -1 0 0 0 0 0 0 0 0 -2 ...
 $ PAY_5                     : int  -2 0 0 0 0 0 0 0 0 -1 ...
 $ PAY_6                     : int  -2 2 0 0 0 0 0 -1 0 -1 ...
 $ BILL_AMT1                 : num  3913 2682 29239 46990 8617 ...
 $ BILL_AMT2                 : num  3102 1725 14027 48233 5670 ...
 $ BILL_AMT3                 : num  689 2682 13559 49291 35835 ...
 $ BILL_AMT4                 : num  0 3272 14331 28314 20940 ...
 $ BILL_AMT5                 : num  0 3455 14948 28959 19146 ...
 $ BILL_AMT6                 : num  0 3261 15549 29547 19131 ...
 $ PAY_AMT1                  : num  0 0 1518 2000 2000 ...
 $ PAY_AMT2                  : num  689 1000 1500 2019 36681 ...
 $ PAY_AMT3                  : num  0 1000 1000 1200 10000 657 38000 0 432 0 ...
 $ PAY_AMT4                  : num  0 1000 1000 1100 9000 ...
 $ PAY_AMT5                  : num  0 0 1000 1069 689 ...
 $ PAY_AMT6                  : num  0 2000 5000 1000 679 ...
 $ default.payment.next.month: int  1 1 0 0 0 0 0 0 0 0 ...
```

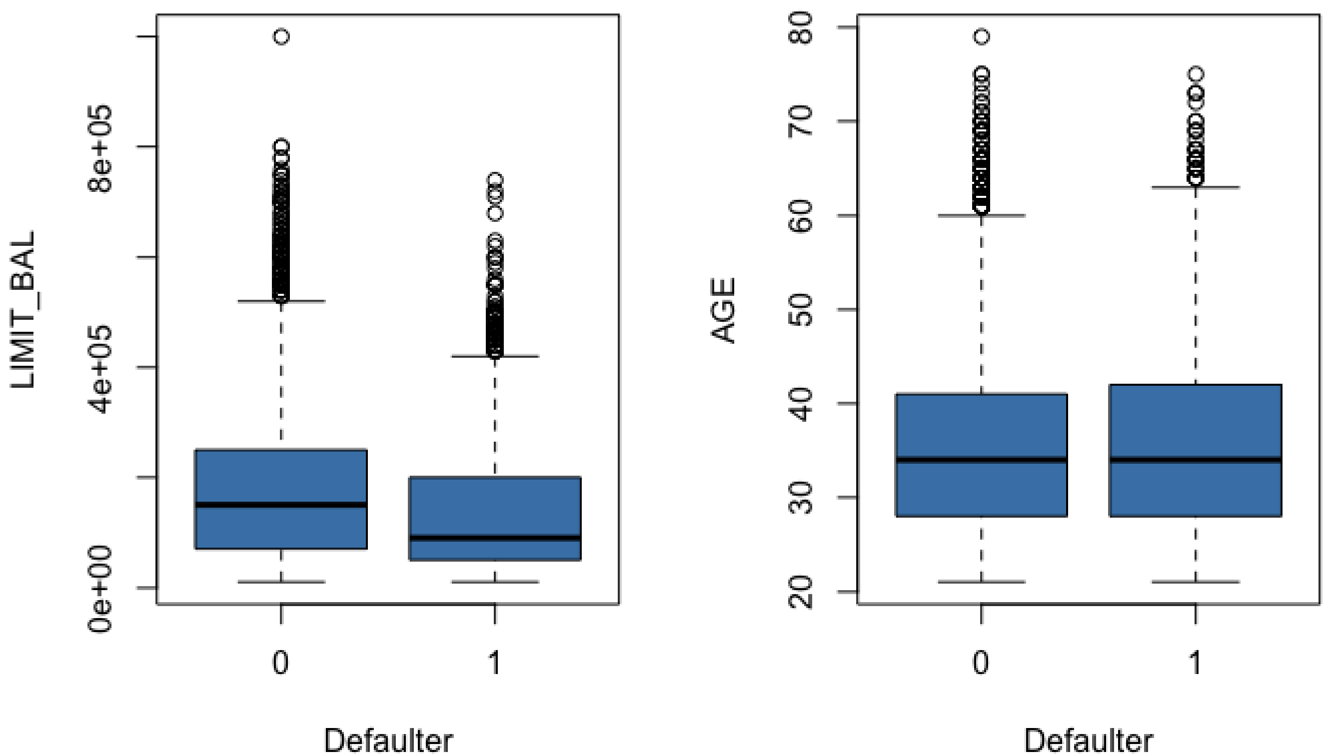
Explore the numerical variables: LIMIT_BAL and AGE

Hide

```
par(mfrow=c(1,2))

boxplot(LIMIT_BAL~default.payment.next.month, xlab = "Defaulter", col = "steelblue")
boxplot(AGE~default.payment.next.month, xlab = "Defaulter", col = "steelblue")
```

Hide

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



Explore the categorical variables: SEX, EDUCATION, MARRIAGE

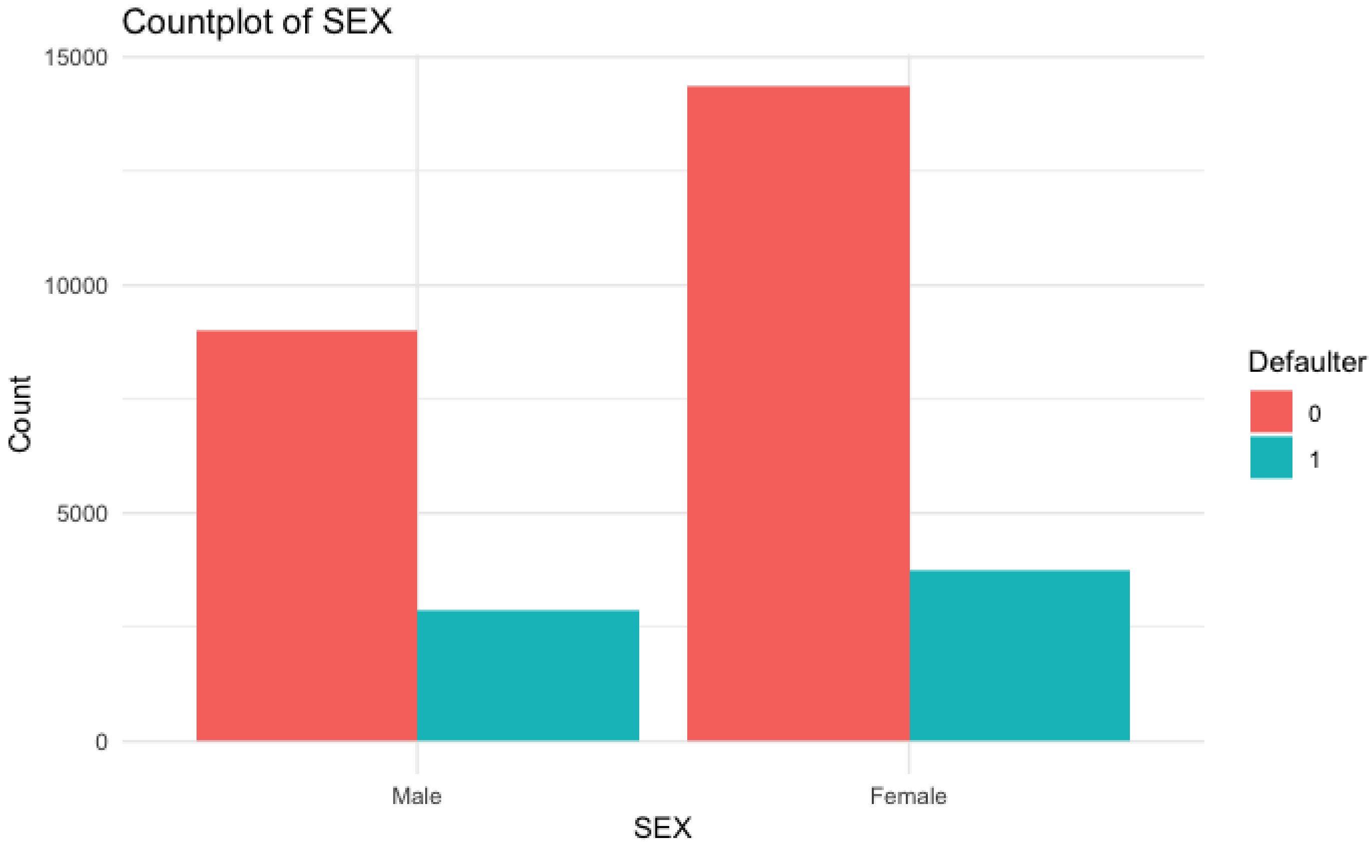
Hide

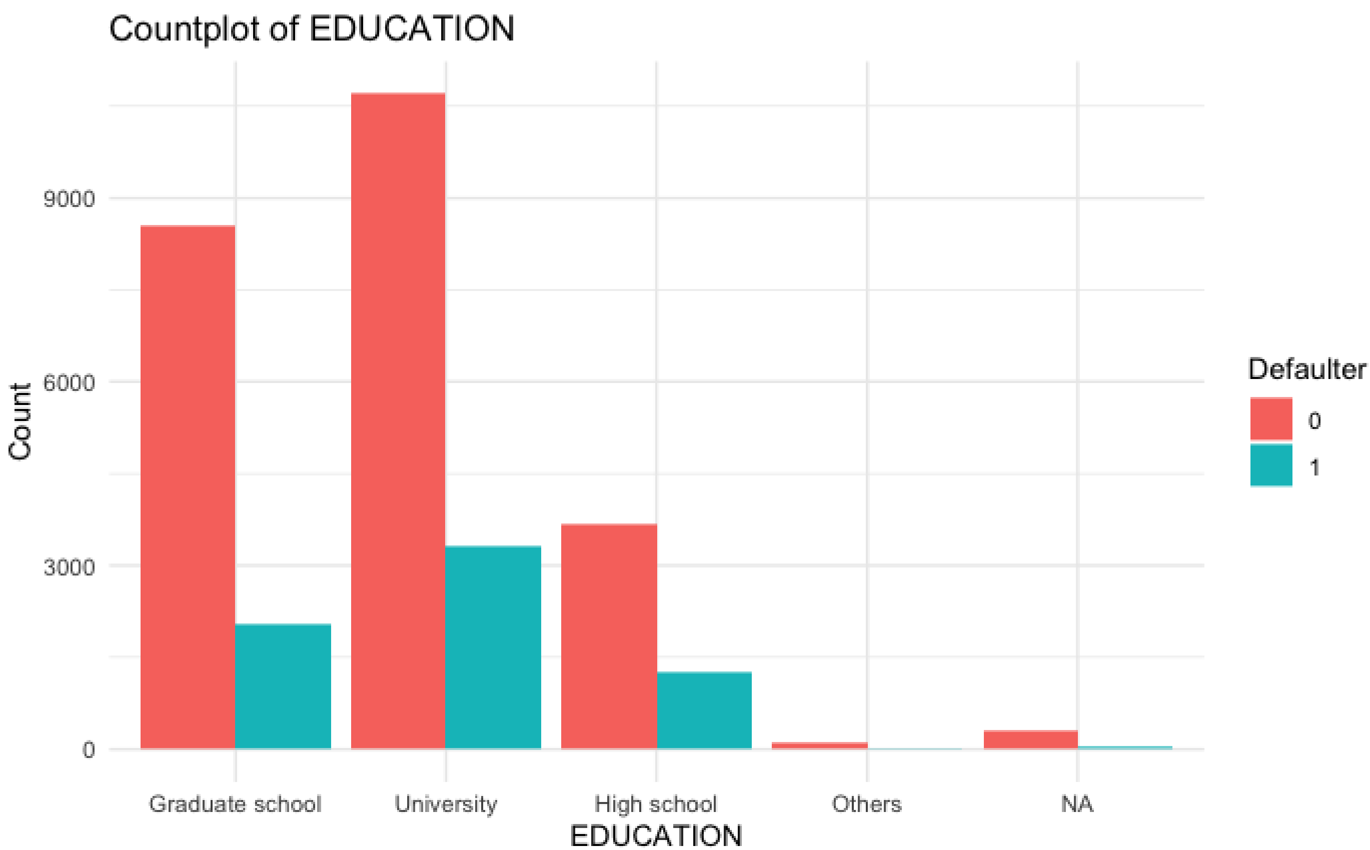
```
library(ggplot2)
library(dplyr)
```

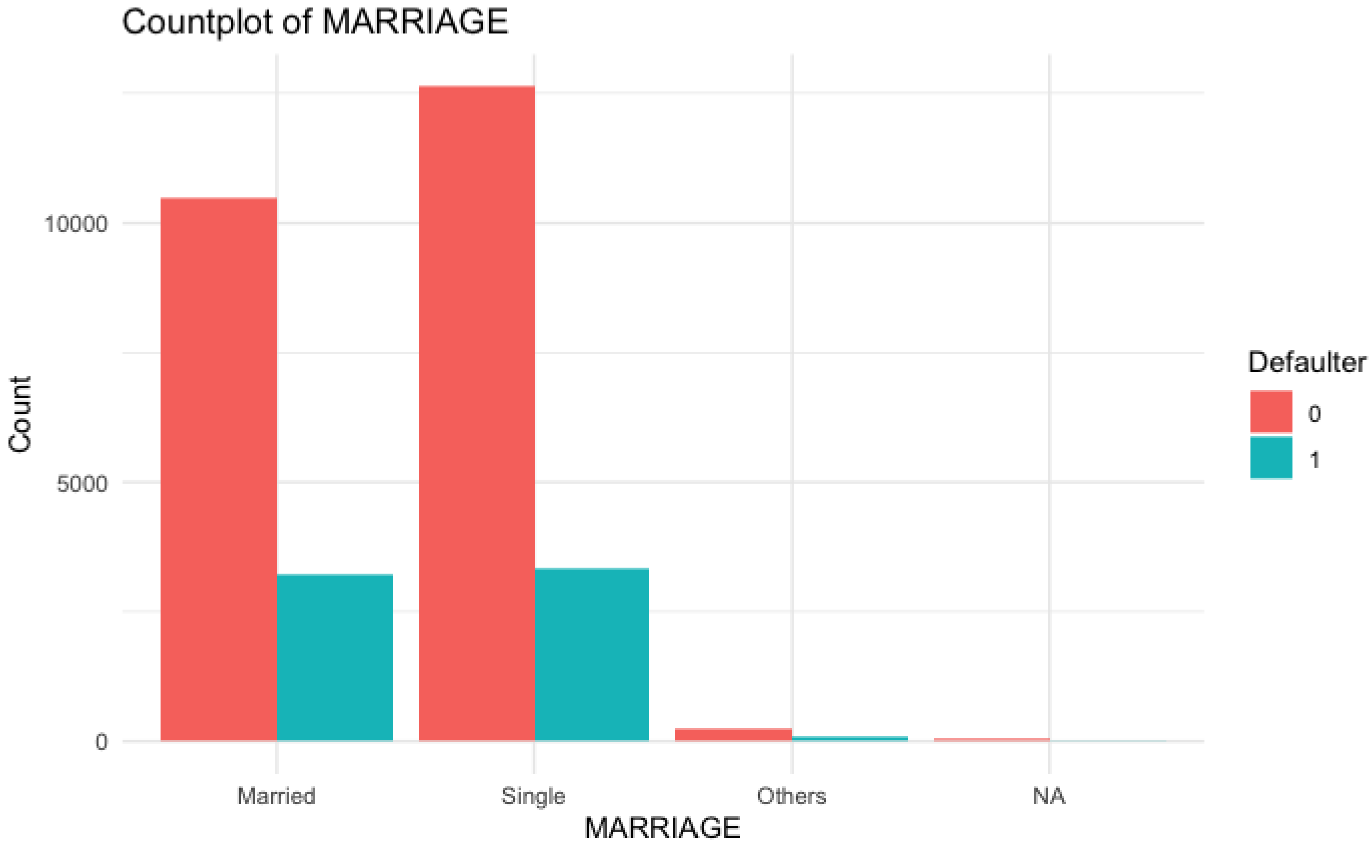
```
Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union
```

Hide

```
categorical_features <- c('SEX', 'EDUCATION', 'MARRIAGE')
ccard_cat <- ccard[categorical_features]
ccard_cat$Defaulter <- ccard$default.payment.next.month

# Replace values with labels
ccard_cat$SEX <- factor(ccard_cat$SEX, labels = c("Male", "Female"))
ccard_cat$EDUCATION <- factor(ccard_cat$EDUCATION, levels = c(1, 2, 3, 4), labels = c("G
raduate school", "University", "High school", "Others"))
ccard_cat$MARRIAGE <- factor(ccard_cat$MARRIAGE, levels = c(1, 2, 3), labels = c("Marrie
d", "Single", "Others"))

# Plot countplots
for (col in categorical_features) {
  count_plot <- ggplot(ccard_cat, aes(x = !!sym(col), fill = factor(Defaulter))) +
    geom_bar(position = "dodge") +
    labs(title = paste("Countplot of", col),
         x = col,
         y = "Count",
         fill = "Defaulter") +
    theme_minimal()

  plot(count_plot)
}
```


Countplot of SEX
15000
10000
5000
0
Count
Male
Female
SEX
Defaulter
0
1

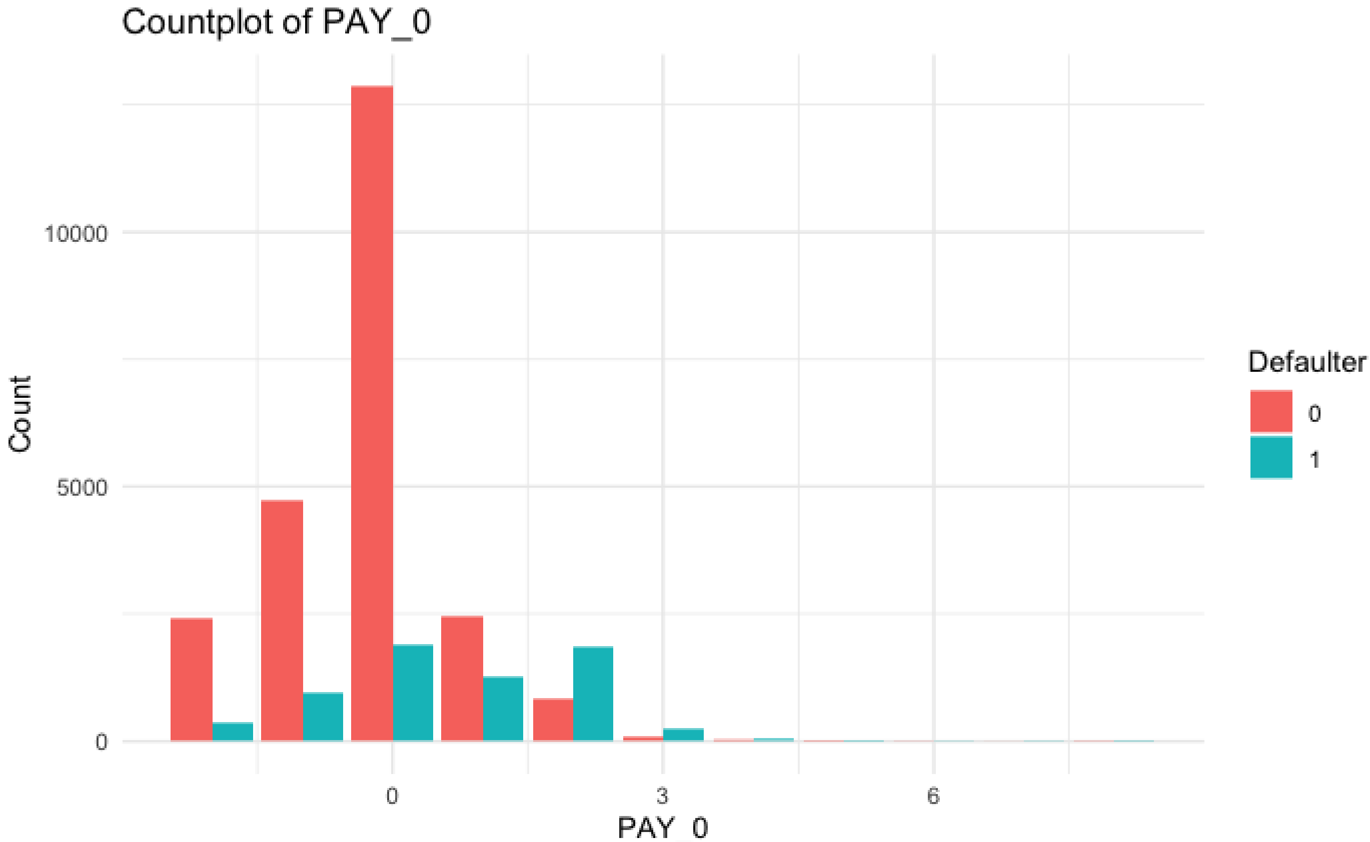

Countplot of EDUCATION
9000
6000
3000
0
Count
Graduate school
University
High school
Others
NA
EDUCATION
Defaulter
0
1

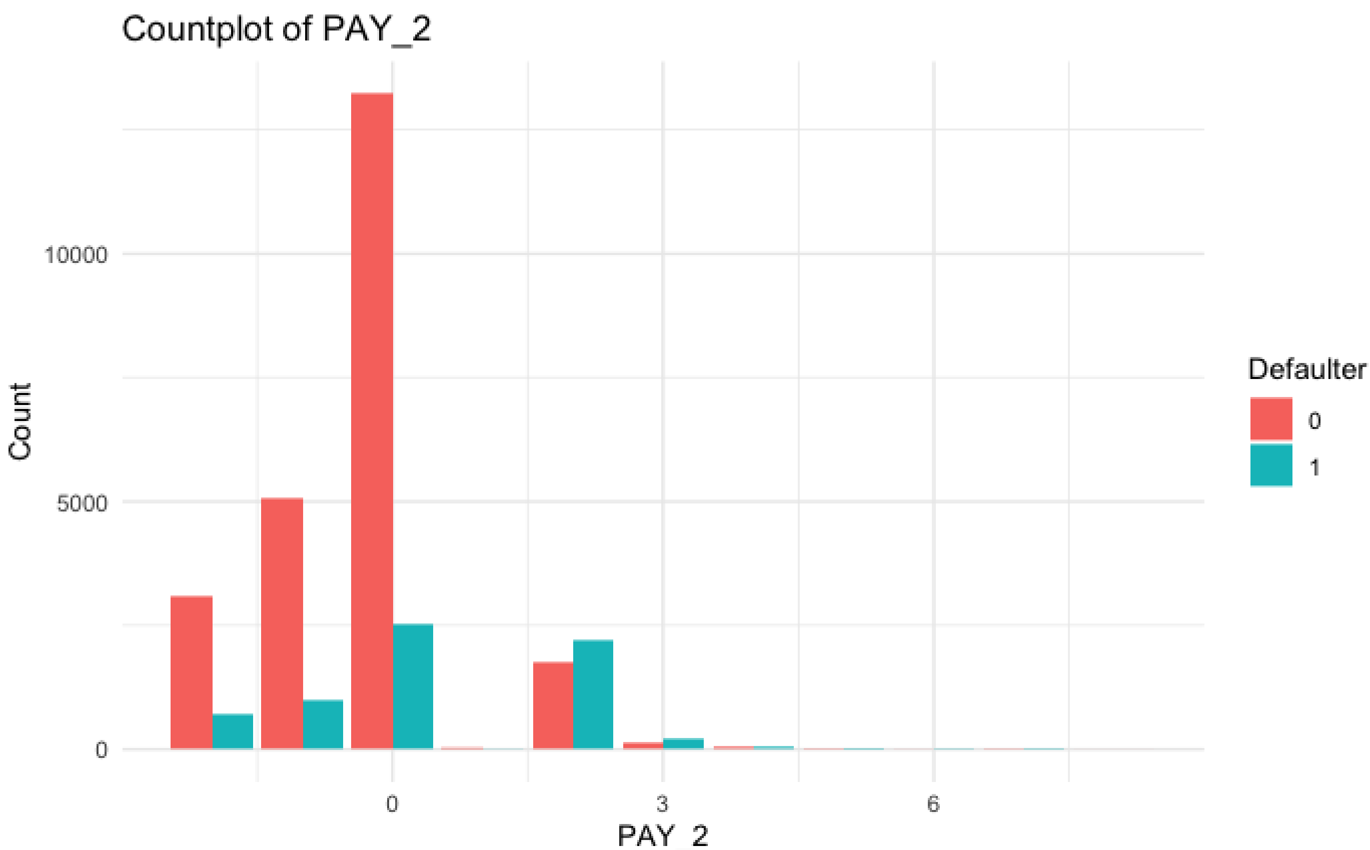


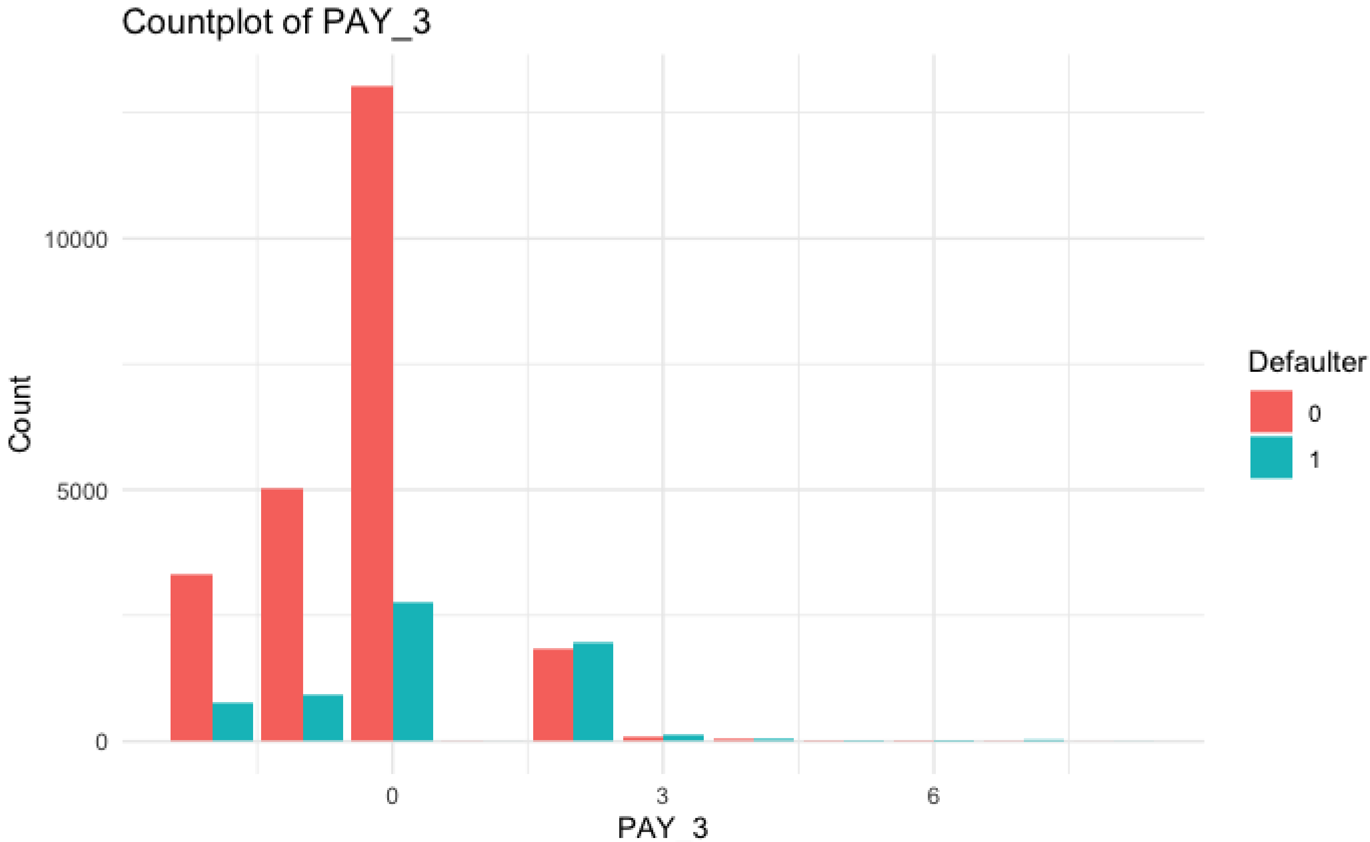
Explore the history of repayment status: PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, and PAY_6

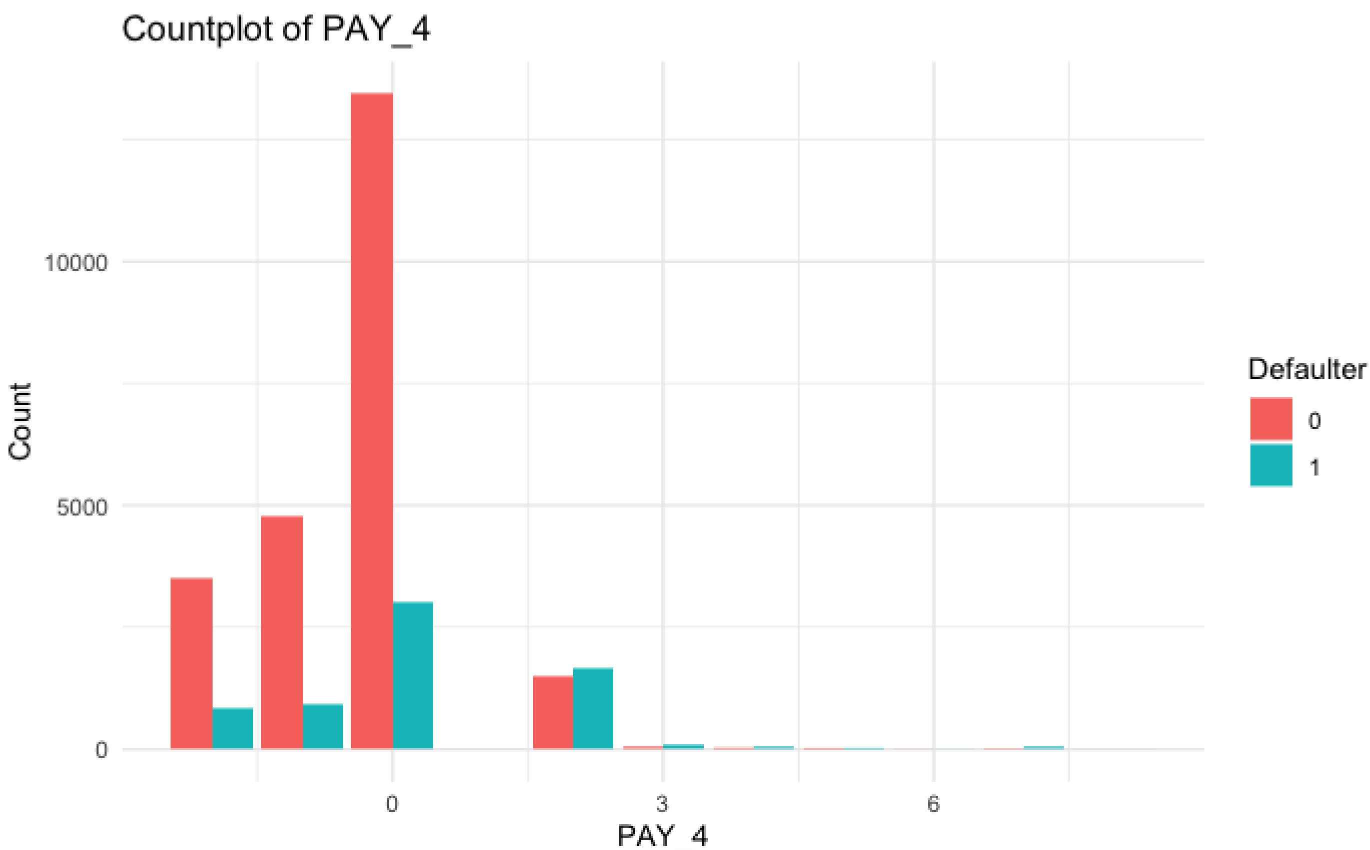
Hide

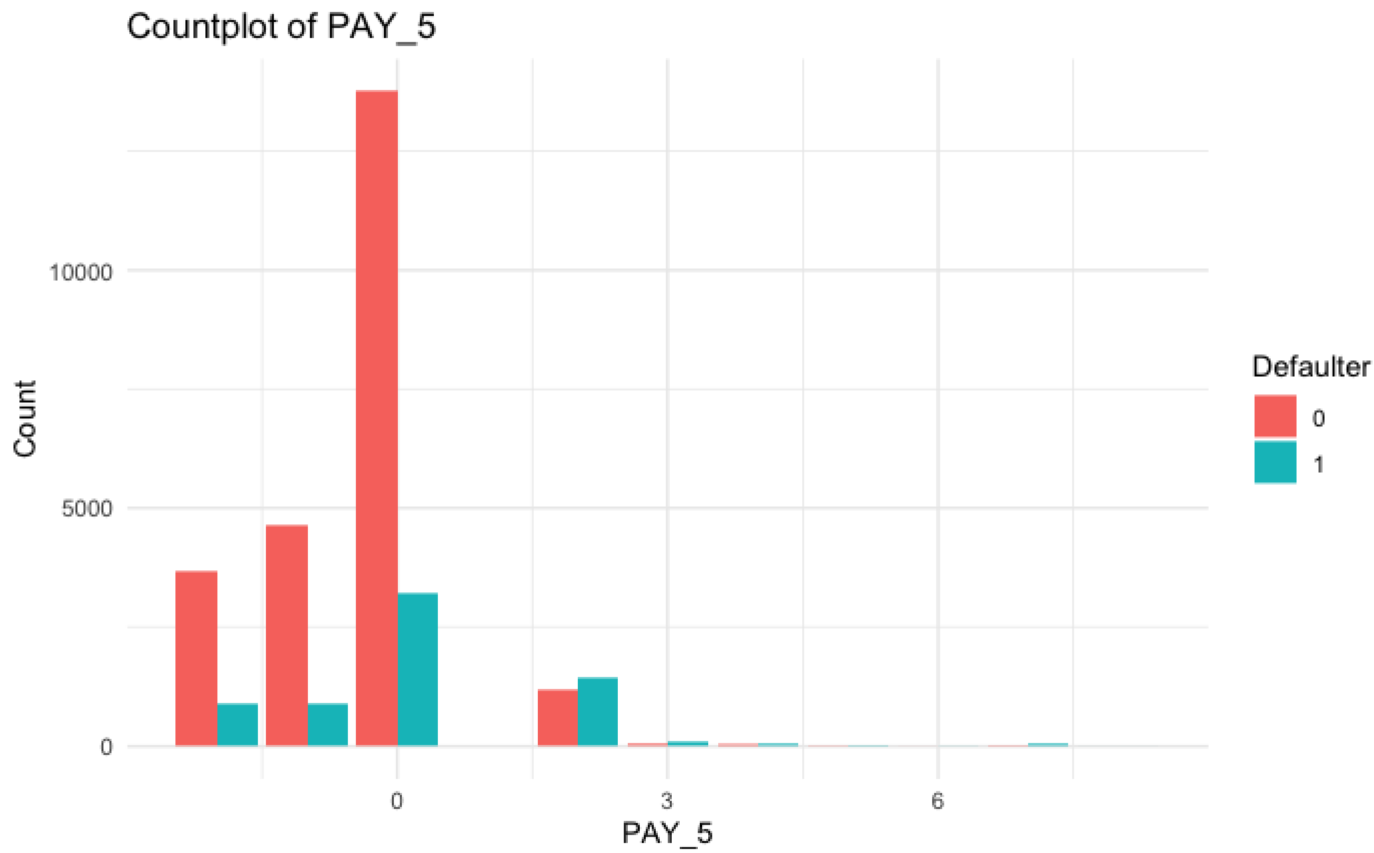
```
PAY = c("PAY_0", "PAY_2", "PAY_3", "PAY_4", "PAY_5", "PAY_6")

for (i in PAY) {
  count_plot <- ggplot(ccard, aes(x = !!sym(i), fill = factor(default.payment.next.mont
h))) +
    geom_bar(position = "dodge") +
    labs(title = paste("Countplot of", i),
         x = i,
         y = "Count",
         fill = "Defaulter") +
    theme_minimal()

  plot(count_plot)
}
```

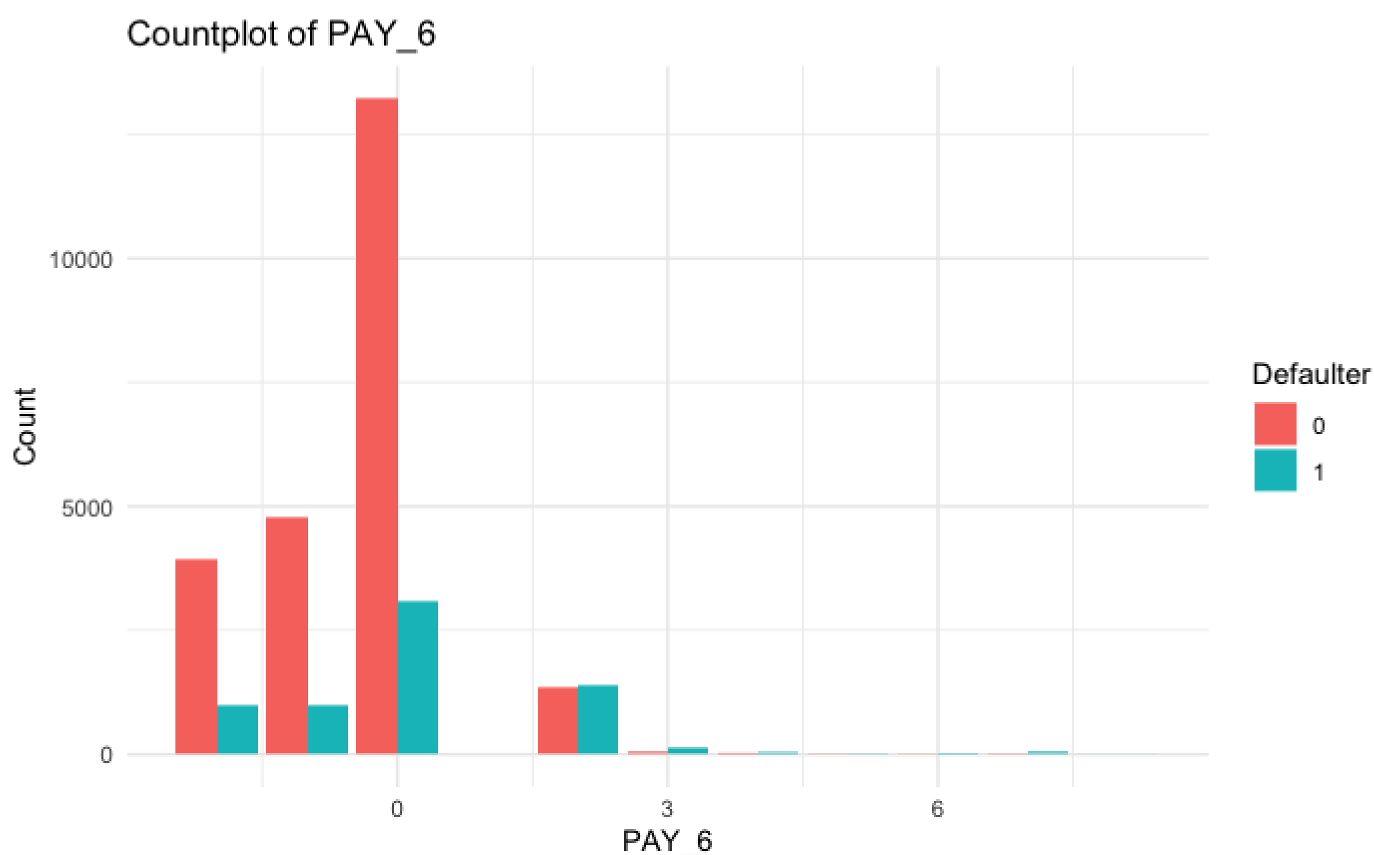

Countplot of PAY_0
Count
10000
5000
0
0
3
6
PAY_0
Defaulter
0
1


Countplot of PAY_2
Count
10000
5000
0
0
3
6
PAY_2
Defaulter
0
1


Countplot of PAY_3
Count
10000
5000
0
0
3
6
PAY_3
Defaulter
0
1


Countplot of PAY_4
Count
10000
5000
0
0
3
6
PAY_4
Defaulter
0
1

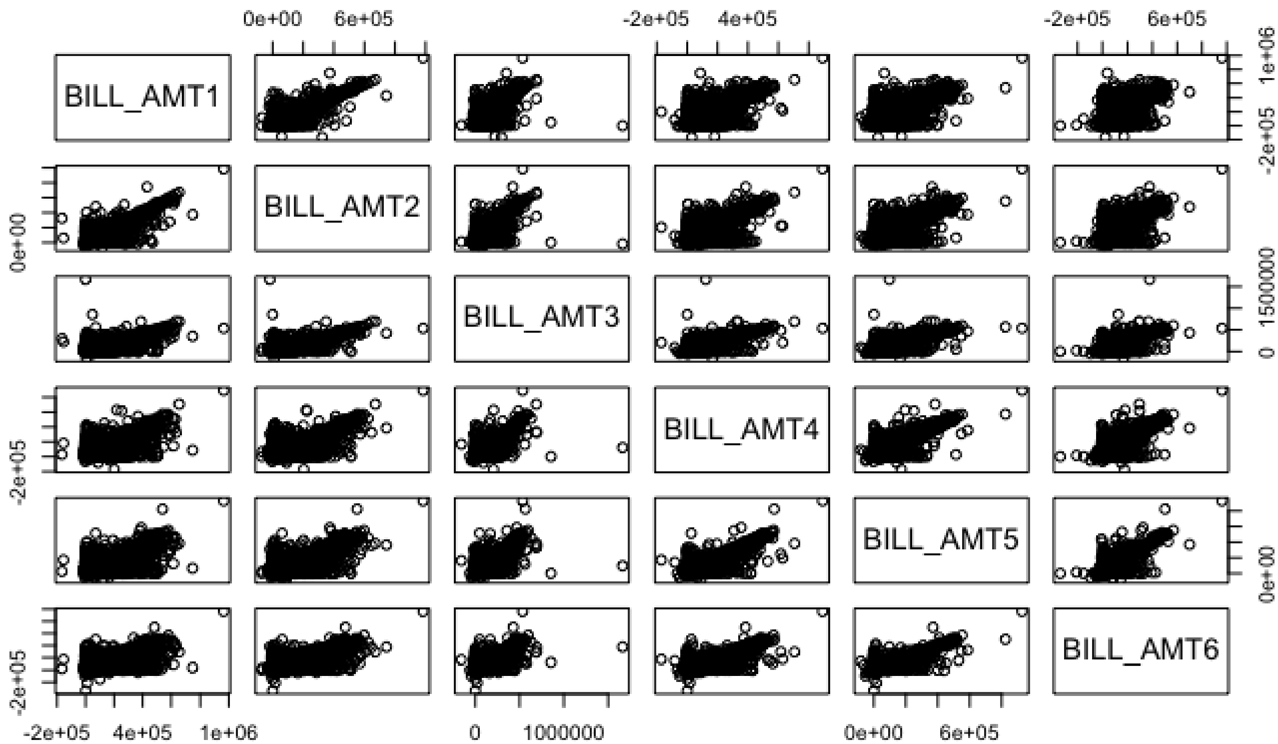




Explore the amount of bill statement and previous payment for the past 6 months:

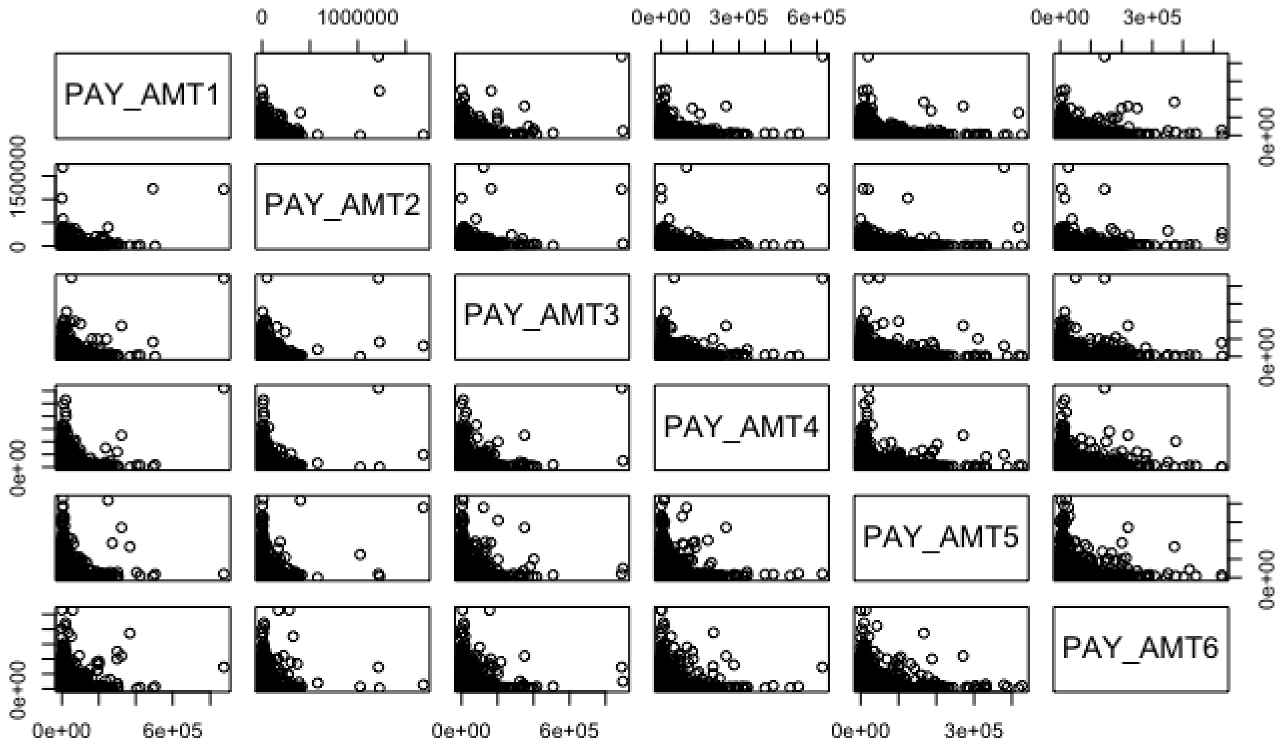
Hide

```
BILL_AMT = c("BILL_AMT1", "BILL_AMT2", "BILL_AMT3", "BILL_AMT4", "BILL_AMT5", "BILL_AMT
6")
pairs(ccard[BILL_AMT])
```



Hide

```
PAY_AMT = c("PAY_AMT1", "PAY_AMT2", "PAY_AMT3", "PAY_AMT4", "PAY_AMT5", "PAY_AMT6")
pairs(ccard[PAY_AMT])
```



Check the correlation between variables

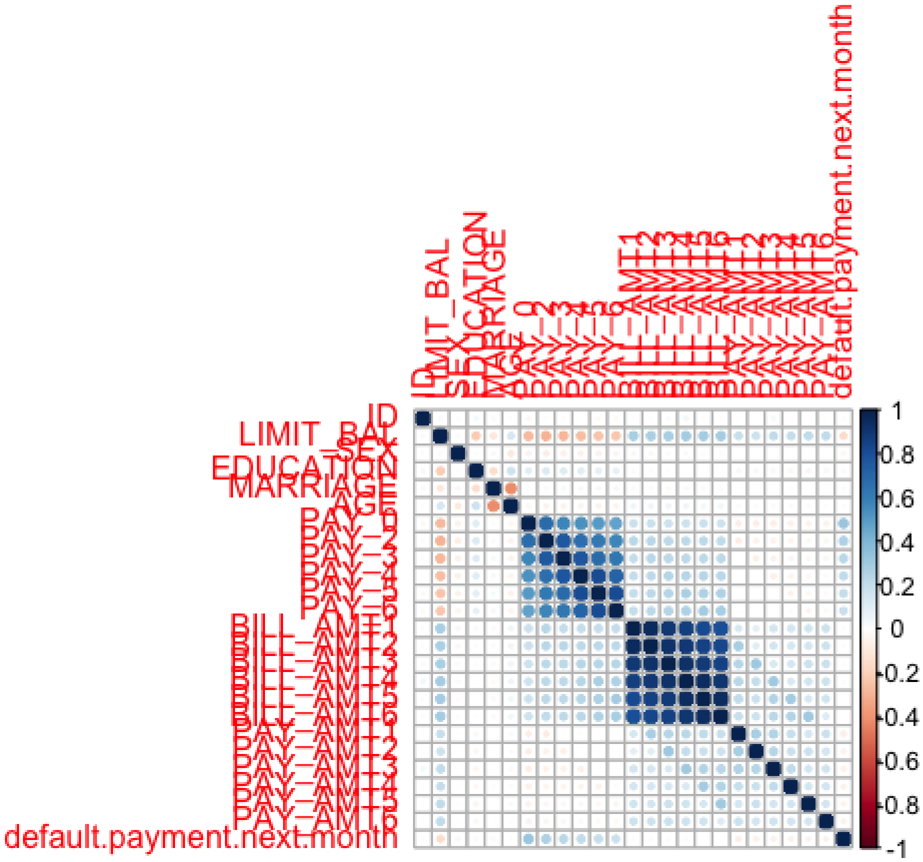
Hide

```
library(corrplot)
```

```
corrplot 0.92 loaded
```

Hide

```
data <- ccard
correlation_matrix <- cor(data)
corrplot(correlation_matrix)
```

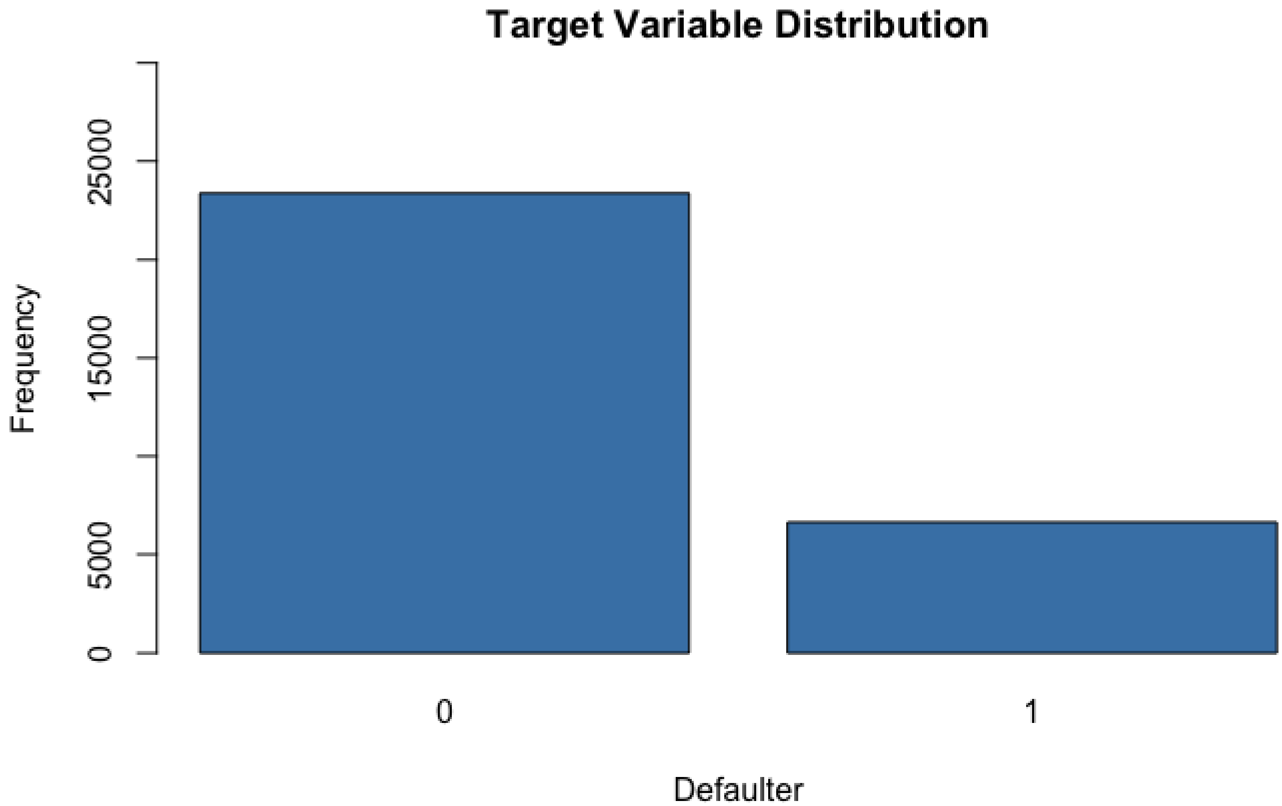


As shown by the plot, the BILL_AMT variables are highly correlated. And, the closer the two months, the more correlated the payment status variables (i.e. PAY_2 and PAY_3). Therefore, in predictive analysis, we are going to use a new predictor to represent the BILL_AMT variables instead of including all of them in the model.

Explore the target variable default.payment.next.month

Hide

```
freq <- table(default.payment.next.month)
barplot(freq, col = "steelblue", main = "Target Variable Distribution", ylim=c(0,30000),
xlab = "Defaulter", ylab = "Frequency")
```



3. Feature Engineering

Rename the variables

Hide

```
column_names <- list(PAY_0 = "PAY_SEPT", PAY_2 = "PAY_AUG", PAY_3 = "PAY_JUL", PAY_4 =
"PAY_JUN", PAY_5 = "PAY_MAY", PAY_6 = "PAY_APR", BILL_AMT1 = "BILL_AMT_SEPT", BILL_AMT2
= "BILL_AMT_AUG", BILL_AMT3 = "BILL_AMT_JUL", BILL_AMT4 = "BILL_AMT_JUN", BILL_AMT5 = "B
ILL_AMT_MAY", BILL_AMT6 = "BILL_AMT_APR", PAY_AMT1 = "PAY_AMT_SEPT", PAY_AMT2 = "PAY_AMT
_AUG", PAY_AMT3 = "PAY_AMT_JUL", PAY_AMT4 = "PAY_AMT_JUN", PAY_AMT5 = "PAY_AMT_MAY", PAY
_AMT6 = "PAY_AMT_APR", default.payment.next.month = "IsDefaulter"
)

for (old_name in names(column_names)) {
  new_name <- column_names[[old_name]]
  names(ccard)[names(ccard) == old_name] <- new_name
}

names(ccard)
```

```
 [1] "ID"            "LIMIT_BAL"     "SEX"           "EDUCATION"     "MARRIAGE"      "AG
E"           "PAY_SEPT"
 [8] "PAY_AUG"       "PAY_JUL"       "PAY_JUN"       "PAY_MAY"       "PAY_APR"       "BI
LL_AMT_SEPT" "BILL_AMT_AUG"
[15] "BILL_AMT_JUL"  "BILL_AMT_JUN"  "BILL_AMT_MAY"  "BILL_AMT_APR"  "PAY_AMT_SEPT"  "PA
Y_AMT_AUG"   "PAY_AMT_JUL"
[22] "PAY_AMT_JUN"   "PAY_AMT_MAY"   "PAY_AMT_APR"   "IsDefaulter"
```

Combine and Delete features

Hide

```
ccard1 = ccard
attach(ccard1)
```

```
The following objects are masked from ccard:

    AGE, EDUCATION, ID, LIMIT_BAL, MARRIAGE, SEX
```

Hide

```
# Combine EDUCATION = 0, 4 or 5 to 4
ccard1$EDUCATION = ifelse(ccard1$EDUCATION==0 | ccard1$EDUCATION==4 | ccard1$EDUCATION==
5 |ccard1$EDUCATION==6, 4, ccard1$EDUCATION)

# Combine MARRIAGE = 0, 3 to 3
ccard1$MARRIAGE = ifelse(ccard1$MARRIAGE==0 | ccard1$MARRIAGE==3, 3, ccard1$MARRIAGE)

# Combine PAY_month = 4, 5, 6, 7, 8 to 4 for every month
ccard1$PAY_SEPT = ifelse(ccard1$PAY_SEPT==4 | ccard1$PAY_SEPT==5 |ccard1$PAY_SEPT==6 | c
card1$PAY_SEPT==7 | ccard1$PAY_SEPT==8, 4, ccard1$PAY_SEPT)
ccard1$PAY_AUG = ifelse(ccard1$PAY_AUG==4 | ccard1$PAY_AUG==5 |ccard1$PAY_AUG==6 | ccard
1$PAY_AUG==7 | ccard1$PAY_AUG==8, 4, ccard1$PAY_AUG)
ccard1$PAY_JUL = ifelse(ccard1$PAY_JUL==4 | ccard1$PAY_JUL==5 |ccard1$PAY_JUL==6 | ccard
1$PAY_JUL==7 | ccard1$PAY_JUL==8, 4, ccard1$PAY_JUL)
ccard1$PAY_JUN = ifelse(ccard1$PAY_JUN==4 | ccard1$PAY_JUN==5 |ccard1$PAY_JUN==6 | ccard
1$PAY_JUN==7 | ccard1$PAY_JUN==8, 4, ccard1$PAY_JUN)
ccard1$PAY_MAY = ifelse(ccard1$PAY_MAY==4 | ccard1$PAY_MAY==5 |ccard1$PAY_MAY==6 | ccard
1$PAY_MAY==7 | ccard1$PAY_MAY==8, 4, ccard1$PAY_MAY)
ccard1$PAY_APR = ifelse(ccard1$PAY_APR==4 | ccard1$PAY_APR==5 |ccard1$PAY_APR==6 | ccard
1$PAY_APR==7 | ccard1$PAY_APR==8, 4, ccard1$PAY_APR)

# Delete PAY_AUG, PAY_JUL, PAY_JUN = 1 data #not sure???????????
ccard1 = subset(ccard1, !(PAY_AUG == 1 | PAY_JUL == 1 | PAY_JUN == 1))

# Delete irrelevant variable ID
ccard1 = subset(ccard1, select = -c(ID))
```

Convert categorical variables from int to factor

Hide

```
categorical_vars <- c("SEX","EDUCATION","MARRIAGE","PAY_SEPT","PAY_AUG", "PAY_JUL", "PAY
_JUN", "PAY_MAY", "PAY_APR", "IsDefaulter")
ccard1[categorical_vars] = lapply(ccard1[categorical_vars], as.factor)
str(ccard1)
```

```
'data.frame':	29972 obs. of  24 variables:
 $ LIMIT_BAL    : num  20000 120000 90000 50000 50000 50000 500000 100000 140000 20000
...
 $ SEX          : Factor w/ 2 levels "1","2": 2 2 2 2 1 1 1 2 2 1 ...
 $ EDUCATION    : Factor w/ 4 levels "1","2","3","4": 2 2 2 2 2 1 1 2 3 3 ...
 $ MARRIAGE     : Factor w/ 3 levels "1","2","3": 1 2 2 1 1 2 2 2 1 2 ...
 $ AGE          : int  24 26 34 37 57 37 29 23 28 35 ...
 $ PAY_SEPT     : Factor w/ 7 levels "-2","-1","0",..: 5 2 3 3 2 3 3 3 3 1 ...
 $ PAY_AUG      : Factor w/ 6 levels "-2","-1","0",..: 4 4 3 3 3 3 3 2 3 1 ...
 $ PAY_JUL      : Factor w/ 6 levels "-2","-1","0",..: 2 3 3 3 2 3 3 2 4 1 ...
 $ PAY_JUN      : Factor w/ 6 levels "-2","-1","0",..: 2 3 3 3 3 3 3 3 3 1 ...
 $ PAY_MAY      : Factor w/ 6 levels "-2","-1","0",..: 1 3 3 3 3 3 3 3 3 2 ...
 $ PAY_APR      : Factor w/ 6 levels "-2","-1","0",..: 1 4 3 3 3 3 3 2 3 2 ...
 $ BILL_AMT_SEPT: num  3913 2682 29239 46990 8617 ...
 $ BILL_AMT_AUG : num  3102 1725 14027 48233 5670 ...
 $ BILL_AMT_JUL : num  689 2682 13559 49291 35835 ...
 $ BILL_AMT_JUN : num  0 3272 14331 28314 20940 ...
 $ BILL_AMT_MAY : num  0 3455 14948 28959 19146 ...
 $ BILL_AMT_APR : num  0 3261 15549 29547 19131 ...
 $ PAY_AMT_SEPT : num  0 0 1518 2000 2000 ...
 $ PAY_AMT_AUG  : num  689 1000 1500 2019 36681 ...
 $ PAY_AMT_JUL  : num  0 1000 1000 1200 10000 657 38000 0 432 0 ...
 $ PAY_AMT_JUN  : num  0 1000 1000 1100 9000 ...
 $ PAY_AMT_MAY  : num  0 0 1000 1069 689 ...
 $ PAY_AMT_APR  : num  0 2000 5000 1000 679 ...
 $ IsDefaulter  : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
```

Create a new variable AVG_BILL_AMT to replace highly correlated BILL_AMT features

Hide

```
attach(ccard1)
```

```
The following objects are masked from ccard1 (pos = 3):

    AGE, BILL_AMT_APR, BILL_AMT_AUG, BILL_AMT_JUL, BILL_AMT_JUN, BILL_AMT_MAY, BILL_AMT_
SEPT, EDUCATION,
    IsDefaulter, LIMIT_BAL, MARRIAGE, PAY_AMT_APR, PAY_AMT_AUG, PAY_AMT_JUL, PAY_AMT_JU
N, PAY_AMT_MAY,
    PAY_AMT_SEPT, PAY_APR, PAY_AUG, PAY_JUL, PAY_JUN, PAY_MAY, PAY_SEPT, SEX

The following objects are masked from ccard:

    AGE, EDUCATION, LIMIT_BAL, MARRIAGE, SEX
```

Hide

```
ccard1$AVG_BILL_AMT = 1/6 * (BILL_AMT_SEPT + BILL_AMT_AUG + BILL_AMT_JUL + BILL_AMT_JUN
+ BILL_AMT_MAY + BILL_AMT_APR)
ccard1 = subset(ccard1, select = -c(BILL_AMT_SEPT, BILL_AMT_AUG, BILL_AMT_JUL, BILL_AMT_
JUN, BILL_AMT_MAY, BILL_AMT_APR))
names(ccard1)
```

```
 [1] "LIMIT_BAL"    "SEX"          "EDUCATION"    "MARRIAGE"     "AGE"          "PAY_SEP
T"     "PAY_AUG"      "PAY_JUL"
 [9] "PAY_JUN"      "PAY_MAY"      "PAY_APR"      "PAY_AMT_SEPT" "PAY_AMT_AUG"  "PAY_AMT
_JUL"  "PAY_AMT_JUN"  "PAY_AMT_MAY"
[17] "PAY_AMT_APR"  "IsDefaulter"  "AVG_BILL_AMT"
```

Handle missing data

Hide

```
dim(ccard1)
```

```
[1] 29972    19
```

Hide

```
ccard1 = na.omit(ccard1)
dim(ccard1)
```

```
[1] 29972    19
```

Handle outliers

Hide

```
# 3 sigma method

numeric_cols = sapply(ccard1, is.numeric)

valid_rows <- rep(TRUE, nrow(ccard1))

for (i in which(numeric_cols)) {
  mean_col <- mean(ccard1[, i])
  sd_col <- sd(ccard1[, i])
  lower_bound <- mean_col - 3 * sd_col
  upper_bound <- mean_col + 3 * sd_col
  # update valid_rows based on 3 sigma bound
  valid_rows <- valid_rows & (ccard1[, i] >= lower_bound & ccard1[, i] <= upper_bound)
}

ccard2 = ccard1[valid_rows,]

dim(ccard2)
```

```
[1] 27314    19
```

4. Predictive Analysis

Perform train-test split

Hide

```
set.seed(1)

train = sample(nrow(ccard2), nrow(ccard2)*0.7, replace = FALSE)
ccard.train = ccard2[train,]
ccard.test = ccard2[-train,]
```

4.1 Logistic Regression

Hide

```
glm.fit = glm(IsDefaulter~., data = ccard.train, family = binomial)

glm.probs = predict(glm.fit, ccard.test, type = "response")

glm.pred = rep(0, nrow(ccard.test))
threshold = 0.5
glm.pred[glm.probs > threshold] = 1

table(glm.pred, ccard.test$IsDefaulter)
```

```
glm.pred    0    1
       0 6042 1163
       1  323  667
```

Hide

```
logiER = mean(glm.pred != ccard.test$IsDefaulter)
logiER
```

```
[1] 0.1813301
```

4.2. LDA

Hide

```
library(MASS)
```

```
Attaching package: ‘MASS’

The following object is masked from ‘package:dplyr’:

    select
```

Hide

```
lda.fit = lda(IsDefaulter~., data = ccard.train)

lda.pred = predict(lda.fit, ccard.test)
lda.class = lda.pred$class

table(lda.class, ccard.test$IsDefaulter)
```

```
lda.class    0    1
        0 6007 1129
        1  358  701
```

Hide

```
ldaER = mean(lda.class != ccard.test$IsDefaulter)
ldaER
```

```
[1] 0.1814521
```

4.3. QDA

Hide

```
qda.fit = qda(IsDefaulter~., data = ccard.train)

qda.pred = predict(qda.fit, ccard.test)
qda.class = qda.pred$class

table(qda.class, ccard.test$IsDefaulter)
```

```
qda.class    0    1
        0 5923 1076
        1  442  754
```

Hide

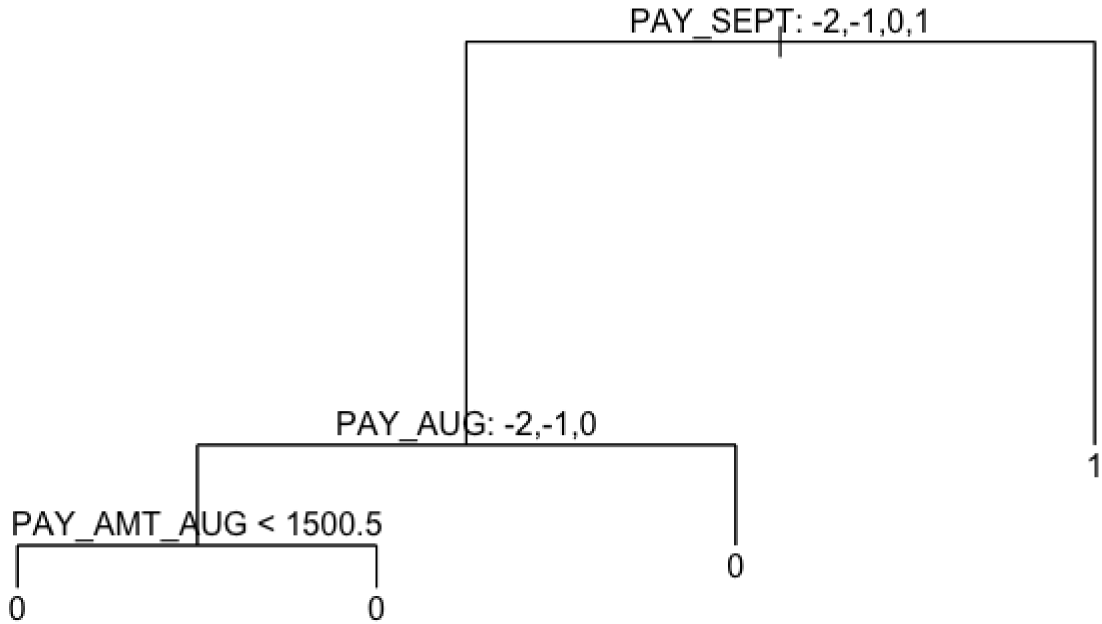
```
qdaER = mean(qda.class != ccard.test$IsDefaulter)
qdaER
```

```
[1] 0.1852349
```

4.4. Classification Tree

Hide

```
library(tree)

tree.ccard = tree(IsDefaulter~., ccard.train)
plot(tree.ccard)
text(tree.ccard, pretty=0)
```



Hide

```
tree.pred = predict(tree.ccard, ccard.test, type="class")

table(tree.pred, ccard.test$IsDefaulter)
```

```
tree.pred    0    1
        0 6092 1219
        1  273  611
```

Hide

```
treeER = mean(tree.pred != ccard.test$IsDefaulter)
treeER
```

```
[1] 0.1820622
```

4.5. Bagging

Hide

```
# Tuning the hyperparameter ntree using Grid search

library(randomForest)
```

```
randomForest 4.7-1.1
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The following object is masked from 'package:dplyr':

    combine

The following object is masked from 'package:ggplot2':

    margin
```

Hide

```
set.seed(1)

testER = c()
ntreeList = seq(300,500,50)
for (ntree in ntreeList) {
  bag.ccard = randomForest(IsDefaulter~., data = ccard.train, mtry = ncol(ccard.train)-
1, ntree = ntree, importance=FALSE)
  bag.pred = predict(bag.ccard, ccard.test)
  #print(mean(bag.pred != ccard.test$IsDefaulter))
  testER = append(testER, mean(bag.pred != ccard.test$IsDefaulter))
}
print(min(testER))
```

```
[1] 0.1885296
```

Hide

```
#plot(ntreeList, testER)
```

4.6. Random Forest

Tune the hyperparameter mtry using Grid search

Hide

```
# usually mtry = sqrt(numPredictors) gives the best performance

set.seed(1)
testER = c()
mtryList = seq(2,6,1)
for (mtry in mtryList) {
  rf.ccard = randomForest(IsDefaulter~., data = ccard.train, mtry = mtry, ntree = bestnt
ree, importance=FALSE)
  rf.pred = predict(rf.ccard, ccard.test)
  #print(mean(rf.pred != ccard.test$IsDefaulter))
  testER = append(testER, mean(rf.pred != ccard.test$IsDefaulter))
}
print(min(testER))
```

```
[1] 0.1846248
```

Hide

```
#plot(mtryList, testER)
```

Hide

```
bestmtry = 4
rfER = min(testER)
```

Tune the hyperparameter ntree suing Grid Search

Hide

```
set.seed(1)
testER = c()
ntreeList = seq(250,500,50)
for (ntree in ntreeList) {
  rf.ccard = randomForest(IsDefaulter~., data = ccard.train, mtry = bestmtry, ntree = nt
ree, importance=FALSE)
  rf.pred = predict(rf.ccard, ccard.test)
  #print(mean(rf.pred != ccard.test$IsDefaulter))
  testER = append(testER, mean(rf.pred != ccard.test$IsDefaulter))
}
print(min(testER))
```

```
[1] 0.1849908
```

Hide

```
#plot(ntreeList, testER)
```

Hide

```
bestntree = 400
rfER = testER[4]
```

Build the best model, and show the importance of predictors

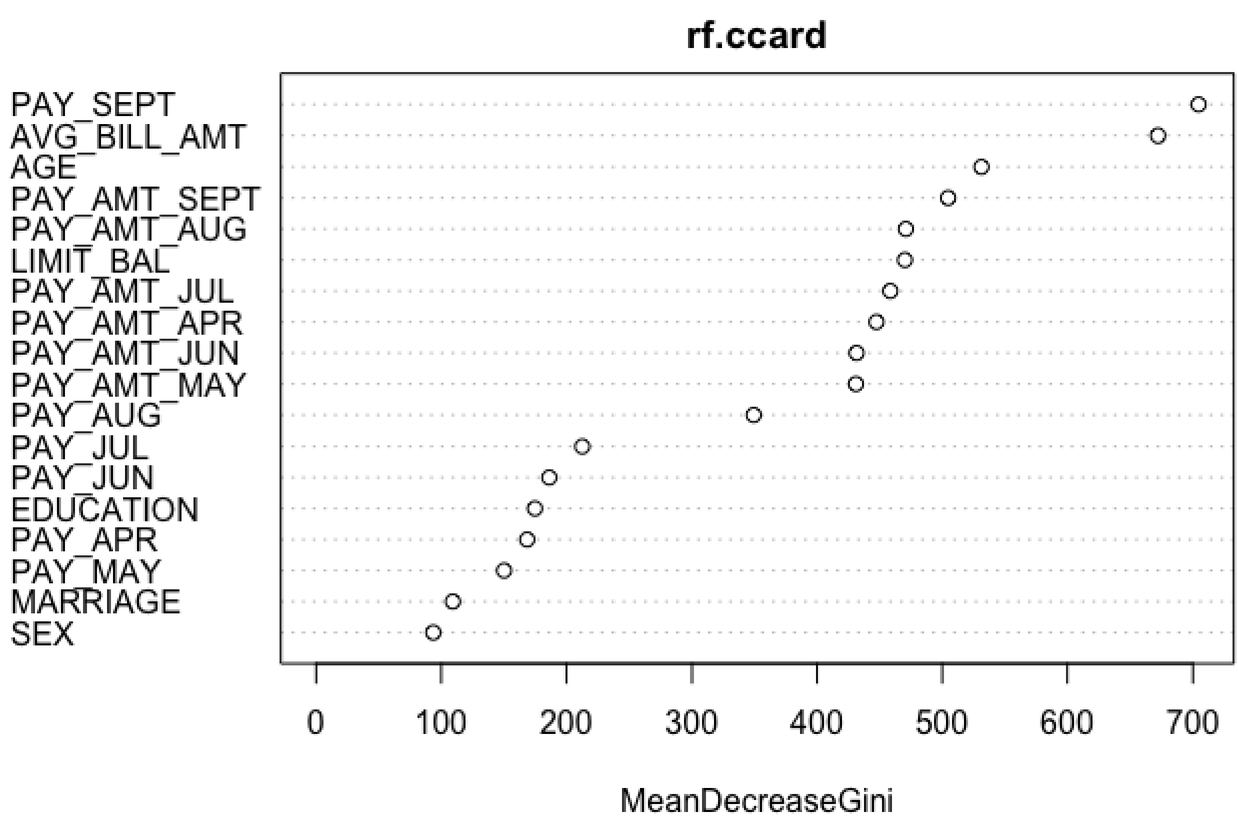
Hide

```
set.seed(1)
rf.ccard = randomForest(IsDefaulter~., data = ccard.train, mtry = bestmtry, ntree = best
ntree, importance=FALSE)
rf.pred = predict(rf.ccard, ccard.test)
rfER = mean(rf.pred != ccard.test$IsDefaulter)
rfER
```

```
[1] 0.185723
```

Hide

```
varImpPlot(rf.ccard)
```



Hide

```
sorted_indices = order(importance(rf.ccard)[,1], decreasing = TRUE)
importance(rf.ccard)[sorted_indices, ]
```

```
    PAY_SEPT AVG_BILL_AMT          AGE PAY_AMT_SEPT  PAY_AMT_AUG    LIMIT_BAL  PAY_AMT_J
UL  PAY_AMT_APR  PAY_AMT_JUN
   704.50942    672.06273    531.38104    504.73734    470.96414    470.18627    458.509
43    447.48306    431.55589
 PAY_AMT_MAY      PAY_AUG      PAY_JUL      PAY_JUN    EDUCATION      PAY_APR      PAY_M
AY     MARRIAGE          SEX
   431.04533    349.36921    212.76499    186.38232    174.86209    168.69740    150.266
57    109.28519     93.60734
```

4.7. Boosted Trees

Preprocessing the data

Hide

```
library(xgboost)
```

```

Attaching package: ‘xgboost’

The following object is masked from ‘package:dplyr’:

    slice
```

Hide

```
set.seed(1)

# Convert categorical variables to numerical values -- Ordinal encoding
ccard3 = lapply(ccard2, function(x) { if(is.factor(x)) as.numeric(as.character(x)) else
x})
ccard3 = data.frame(ccard3)

# One-hot encoding
# library(fastDummies)
# categorical_preds = c("SEX","EDUCATION","MARRIAGE","PAY_SEPT","PAY_AUG", "PAY_JUL", "P
AY_JUN", "PAY_MAY", "PAY_APR")
# ccard3 = dummy_cols(ccard2, select_columns = categorical_preds, remove_first_dummy = T
RUE) # remove the first level dummy to avoid multicollinearity
# ccard3 = ccard3[, !colnames(ccard3) %in% categorical_preds]
# colnames(ccard3) = make.names(colnames(ccard3)) # make valid names

# Recreate the train-test split
ccard.train = ccard3[train,]
ccard.test = ccard3[-train,]

x_train = ccard.train[, -which(names(ccard.train) == "IsDefaulter"), drop = FALSE]
x_train = as.matrix(sapply(x_train, as.numeric))
y_train = ccard.train$IsDefaulter
dtrain = xgb.DMatrix(data = x_train, label = y_train)

x_test = ccard.test[, -which(names(ccard.test) == "IsDefaulter"), drop = FALSE]
x_test = as.matrix(sapply(x_test, as.numeric))
y_test = ccard.test$IsDefaulter
dtest = xgb.DMatrix(data = x_test)
```

Tuning shrinkage

Hide

```
# grid search with k-fold cross validation
shrinkage = seq(from = 0.1, to = 0.5, by = 0.1)

best_ER <- 1
best_eta <- NULL
for (eta in shrinkage) {
  set.seed(1)
  cv <- xgb.cv(
    data = dtrain,
    eta = eta, #
    objective = "binary:logistic",
    nfold = 5,
    metrics = "error",
    nrounds = 10,
    early_stopping_rounds = 10,
    verbose = FALSE
  )

  mean_ER = cv$evaluation_log$test_error_mean[length(cv$evaluation_log$test_error_mean)]
  if (mean_ER < best_ER) {
      best_ER = mean_ER
      best_eta = eta
  }
}
best_ER
```

```
[1] 0.1836393
```

Hide

```
best_eta
```

```
[1] 0.2
```

Tuning nrounds*

Hide

```
ntreeList = seq(from = 4, to = 12, by = 2)

best_ER <- 1
best_ntree <- NULL
for (ntree in ntreeList) {
  set.seed(1)
  cv <- xgb.cv(
    data = dtrain,
    eta = best_eta,
    nrounds = ntree,
    objective = "binary:logistic",
    nfold = 5,
    metrics = "error",
    early_stopping_rounds = 10,
    verbose = FALSE
  )

  mean_ER = cv$evaluation_log$test_error_mean[length(cv$evaluation_log$test_error_mean)]
  if (mean_ER < best_ER) {
      best_ER = mean_ER
      best_ntree = ntree
  }
}
best_ER
```

```
[1] 0.1836393
```

Hide

```
best_ntree
```

```
[1] 10
```

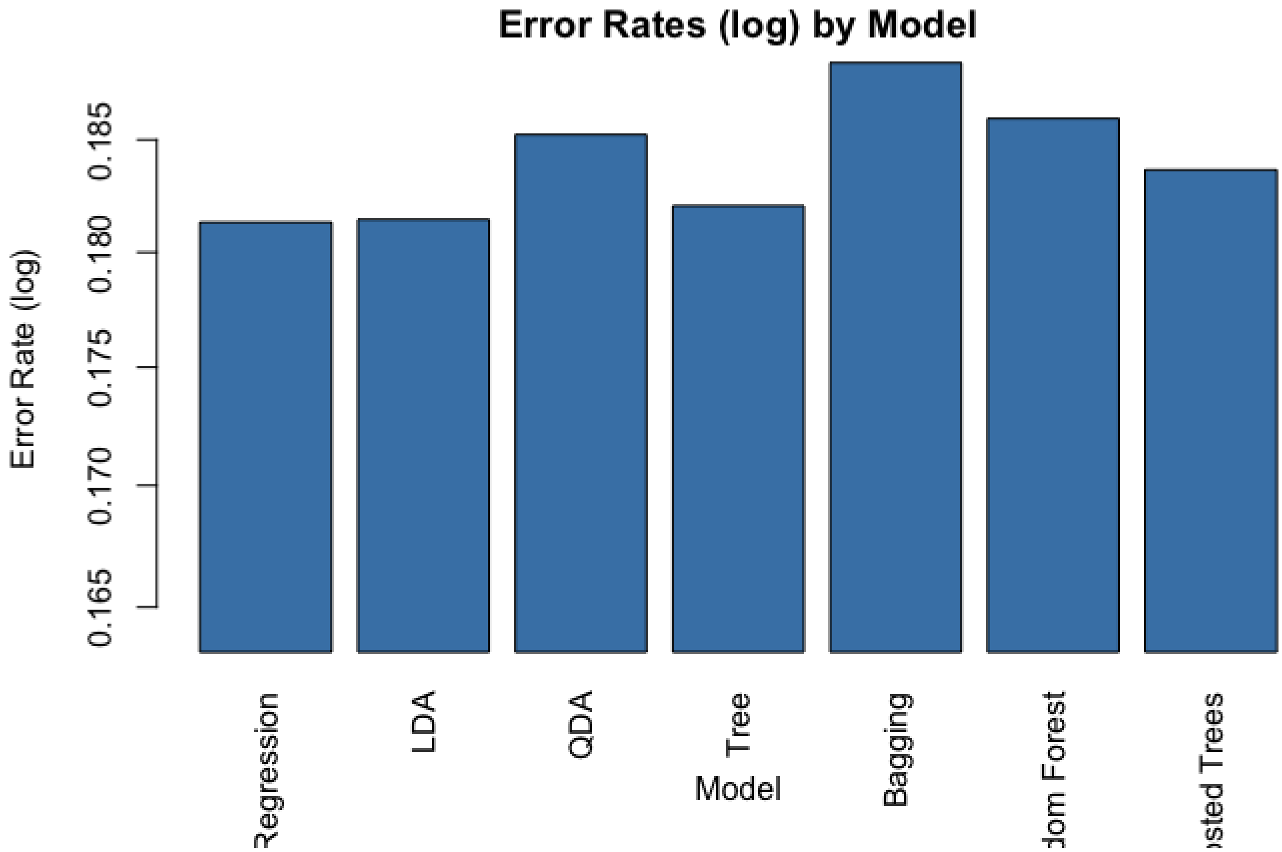
Hide

```
boostER = best_ER
```

5. Model Evaluations

Hide

```
# need Boosted tree and Neural network
models = c("Logistic Regression", "LDA", "QDA", "Tree", "Bagging", "Random Forest", "Boo
sted Trees")
errorRates = c(logiER, ldaER, qdaER, treeER, baggingER, rfER, boostER)
barplot(errorRates, names.arg = models, col = "steelblue", main = "Error Rates (log) by
Model",
        xlab = "Model", ylab = "Error Rate (log)", log = "y", las=3)
```



Hide

```
errorRates
```

```
[1] 0.1813301 0.1814521 0.1852349 0.1820622 0.1885296 0.1859671 0.1836393
```

6. Conclusion

While most models demonstrate similar precision in predicting default cases, Logistic Regression, LDA, and decision trees outperform the others. The relationship between our listed predictors and the target is straightforward and easily interpretable, with the decision boundary between defaulters and non-defaulters being close to linear.

The most significant predictors include the repayment amount and status from September, the average amount of the bill statement, age, and credit limit. Recent repayment information proves to be more predictive, with data from consecutive months showing high correlation. Moreover, clients who didn't pay on time usually made payments two months late, while only a small number paid just one month late.