Analysis of Credit Card Default Against Cardholders' Background

Yu-Chen (Amber) Lu

Date: 2018-12-21

Abstract

To find out the best fit algorithm for amount of given credit in NT dollars against other factors, which are important variables using Bayesian information criterion.

0. Introduction

High credit card default rate can make a business in trouble even bankrupt. The propose of this project is to predict whether a cilent defaults on her/his credit card, the business can underwrite credit cards more carefully to the potential clients who cannot pay bills with high probability.

In order to get more insight of this dataset, I did exploratory data analysis using `ggplot2`. Another perceptive of this project is to predict the limit balance of a credit card. Thus, I used "Default of Credit Card Clients Dataset", which was downloaded from https://archive.ics.uci.edu/ml/machine-learning-databases/00350/.

1) About the data

This dataset 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.

2) About the variables

(1) ID: ID of each client
(2) LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit
(3) SEX: Gender (1=male, 2=female)
(4) EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
(5) MARRIAGE: Marital status (1=married, 2=single, 3=others)
(6) AGE: Age in years
(7) PAY_0: Repayment status in September, 2005 (-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)
(8) PAY_2: Repayment status in August, 2005 (scale same as above)
(9) PAY_3: Repayment status in July, 2005 (scale same as above)
(10) PAY_4: Repayment status in June, 2005 (scale same as above)
(11) PAY_5: Repayment status in May, 2005 (scale same as above)
(12) PAY_6: Repayment status in April, 2005 (scale same as above)
(13) BILL_AMT1: Amount of bill statement in September, 2005 (NT dollar)
(14) BILL_AMT2: Amount of bill statement in August, 2005 (NT dollar)
(15) BILL_AMT3: Amount of bill statement in July, 2005 (NT dollar)
(16) BILL_AMT4: Amount of bill statement in June, 2005 (NT dollar)

(17) BILL_AMT5: Amount of bill statement in May, 2005 (NT dollar)
(18) BILL_AMT6: Amount of bill statement in April, 2005 (NT dollar)
(19) PAY_AMT1: Amount of previous payment in September, 2005 (NT dollar)
(20) PAY_AMT2: Amount of previous payment in August, 2005 (NT dollar)
(21) PAY_AMT3: Amount of previous payment in July, 2005 (NT dollar)
(22) PAY_AMT4: Amount of previous payment in June, 2005 (NT dollar)
(23) PAY_AMT5: Amount of previous payment in May, 2005 (NT dollar)
(24) PAY_AMT6: Amount of previous payment in April, 2005 (NT dollar)
(25) default.payment.next.month: Default payment (1=yes, 0=no)

3) Citation

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.

4) Library sources

```r
library(plyr)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.1

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(tidyr)
library(leaps)
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:tidyr':
##
##     expand

## Loading required package: foreach

## Loaded glmnet 2.0-16
```

```r
library(tree)
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.4
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(class)
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```r
library(gam)
```

```
## Loading required package: splines

## Loaded gam 1.16
```

```r
set.seed(1)
```

1. Exploratory Data Analysis

```r
CreditCard = read.csv(file="UCI_Credit_Card.csv", header=TRUE)
# names(CreditCard) = sapply(names(CreditCard), tolower)
head(CreditCard)
```

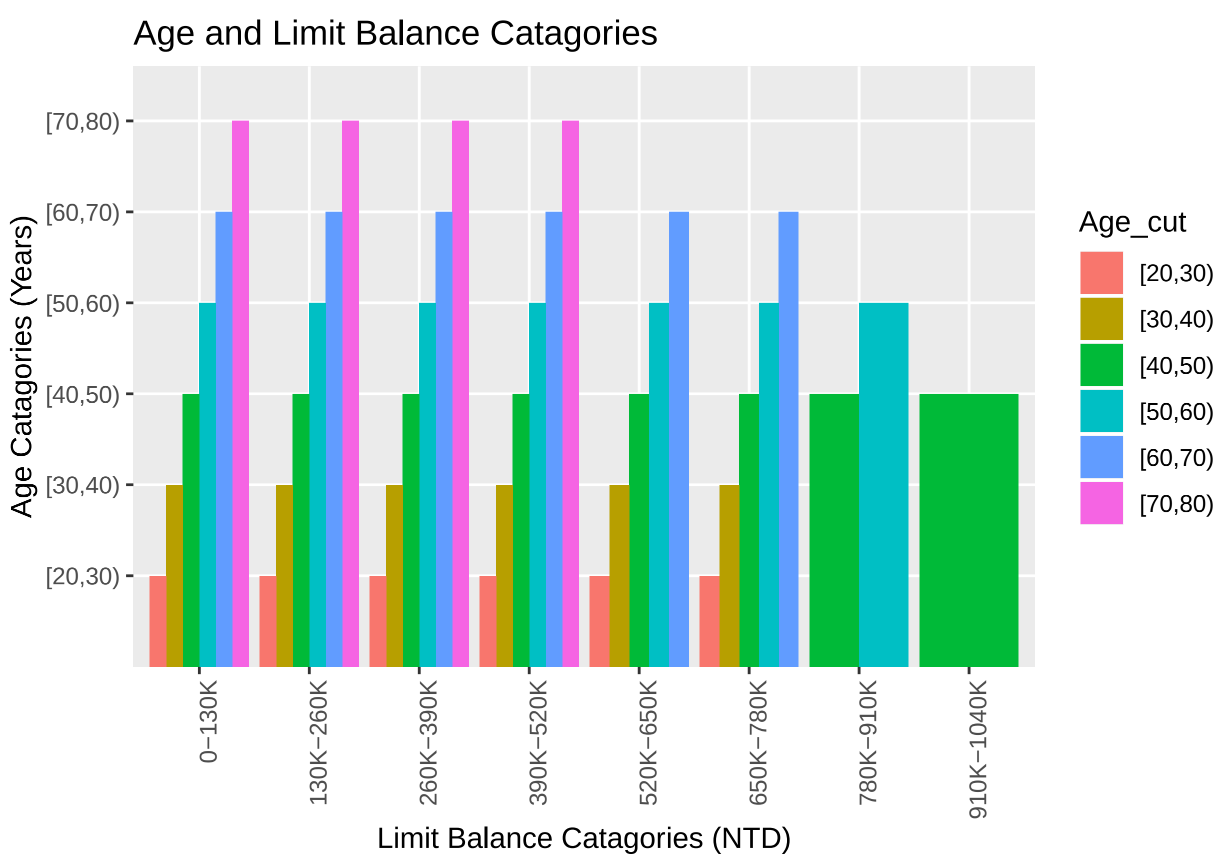
```
##   ID LIMIT_BAL SEX EDUCATION MARRIAGE AGE PAY_0 PAY_2 PAY_3 PAY_4 PAY_5
## 1  1     20000   2         2        1  24     2     2    -1    -1    -2
## 2  2    120000   2         2        2  26    -1     2     0     0     0
## 3  3     90000   2         2        2  34     0     0     0     0     0
```

```
## 4  4     50000   2         2        1  37     0     0     0     0     0
## 5  5     50000   1         2        1  57    -1     0    -1     0     0
## 6  6     50000   1         1        2  37     0     0     0     0     0
##   PAY_6 BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6
## 1    -2      3913      3102       689         0         0         0
## 2     2      2682      1725      2682      3272      3455      3261
## 3     0     29239     14027     13559     14331     14948     15549
## 4     0     46990     48233     49291     28314     28959     29547
## 5     0      8617      5670     35835     20940     19146     19131
## 6     0     64400     57069     57608     19394     19619     20024
##   PAY_AMT1 PAY_AMT2 PAY_AMT3 PAY_AMT4 PAY_AMT5 PAY_AMT6
## 1        0      689        0        0        0        0
## 2        0     1000     1000     1000        0     2000
## 3     1518     1500     1000     1000     1000     5000
## 4     2000     2019     1200     1100     1069     1000
## 5     2000    36681    10000     9000      689      679
## 6     2500     1815      657     1000     1000      800
##   default.payment.next.month
## 1                          1
## 2                          1
## 3                          0
## 4                          0
## 5                          0
## 6                          0
```

```
glimpse(CreditCard)
```

```
## Observations: 30,000
## Variables: 25
## $ ID                         <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ...
## $ LIMIT_BAL                  <dbl> 20000, 120000, 90000, 50000, 50000,...
## $ SEX                        <int> 2, 2, 2, 2, 1, 1, 1, 2, 2, 1, 2, 2,...
## $ EDUCATION                  <int> 2, 2, 2, 2, 2, 1, 1, 2, 3, 3, 3, 1,...
## $ MARRIAGE                   <int> 1, 2, 2, 1, 1, 2, 2, 2, 1, 2, 2, 2,...
## $ AGE                        <int> 24, 26, 34, 37, 57, 37, 29, 23, 28,...
## $ PAY_0                      <int> 2, -1, 0, 0, -1, 0, 0, 0, 0, -2, 0,...
## $ PAY_2                      <int> 2, 2, 0, 0, 0, 0, 0, -1, 0, -2, 0, ...
## $ PAY_3                      <int> -1, 0, 0, 0, -1, 0, 0, -1, 2, -2, 2...
## $ PAY_4                      <int> -1, 0, 0, 0, 0, 0, 0, 0, 0, -2, 0, ...
## $ PAY_5                      <int> -2, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, ...
## $ PAY_6                      <int> -2, 2, 0, 0, 0, 0, 0, -1, 0, -1, -1...
## $ BILL_AMT1                  <dbl> 3913, 2682, 29239, 46990, 8617, 644...
## $ BILL_AMT2                  <dbl> 3102, 1725, 14027, 48233, 5670, 570...
## $ BILL_AMT3                  <dbl> 689, 2682, 13559, 49291, 35835, 576...
## $ BILL_AMT4                  <dbl> 0, 3272, 14331, 28314, 20940, 19394...
## $ BILL_AMT5                  <dbl> 0, 3455, 14948, 28959, 19146, 19619...
## $ BILL_AMT6                  <dbl> 0, 3261, 15549, 29547, 19131, 20024...
## $ PAY_AMT1                   <dbl> 0, 0, 1518, 2000, 2000, 2500, 55000...
## $ PAY_AMT2                   <dbl> 689, 1000, 1500, 2019, 36681, 1815,...
## $ PAY_AMT3                   <dbl> 0, 1000, 1000, 1200, 10000, 657, 38...
## $ PAY_AMT4                   <dbl> 0, 1000, 1000, 1100, 9000, 1000, 20...
## $ PAY_AMT5                   <dbl> 0, 0, 1000, 1069, 689, 1000, 13750,...
## $ PAY_AMT6                   <dbl> 0, 2000, 5000, 1000, 679, 800, 1377...
## $ default.payment.next.month <int> 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

```r
# Get some plots of the data set
CreditCard_plot = CreditCard
CreditCard_plot$LIM_cut = cut(as.numeric(as.character(CreditCard_plot$LIMIT_BAL)),
                              c((0:8)*130000), right = FALSE,
                              labels = c("0-130K", "130K-260K", "260K-390K",
                                         "390K-520K", "520K-650K", "650K-780K",
                                         "780K-910K", "910K-1040K")) # Categorize LIMIT_BAL
CreditCard_plot$Age_cut = cut(as.numeric(as.character(CreditCard_plot$AGE)),
                              c(seq(20,80,10)), right = FALSE) # Categorize Defualt Rate

# Convert format
CreditCard_plot$default.payment.next.month =
  as.character(CreditCard_plot$default.payment.next.month)
CreditCard_plot$EDUCATION = as.character(CreditCard_plot$EDUCATION)

# Plot 1 ---------------------------------------------------------------------------
ggplot(data=CreditCard_plot, aes(LIM_cut, Age_cut)) +
  geom_bar(stat = "identity", aes(fill = Age_cut), position = "dodge") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs( title = "Age and Limit Balance Catagories", x = "Limit Balance Catagories (NTD)",
        y = "Age Catagories (Years)")
```

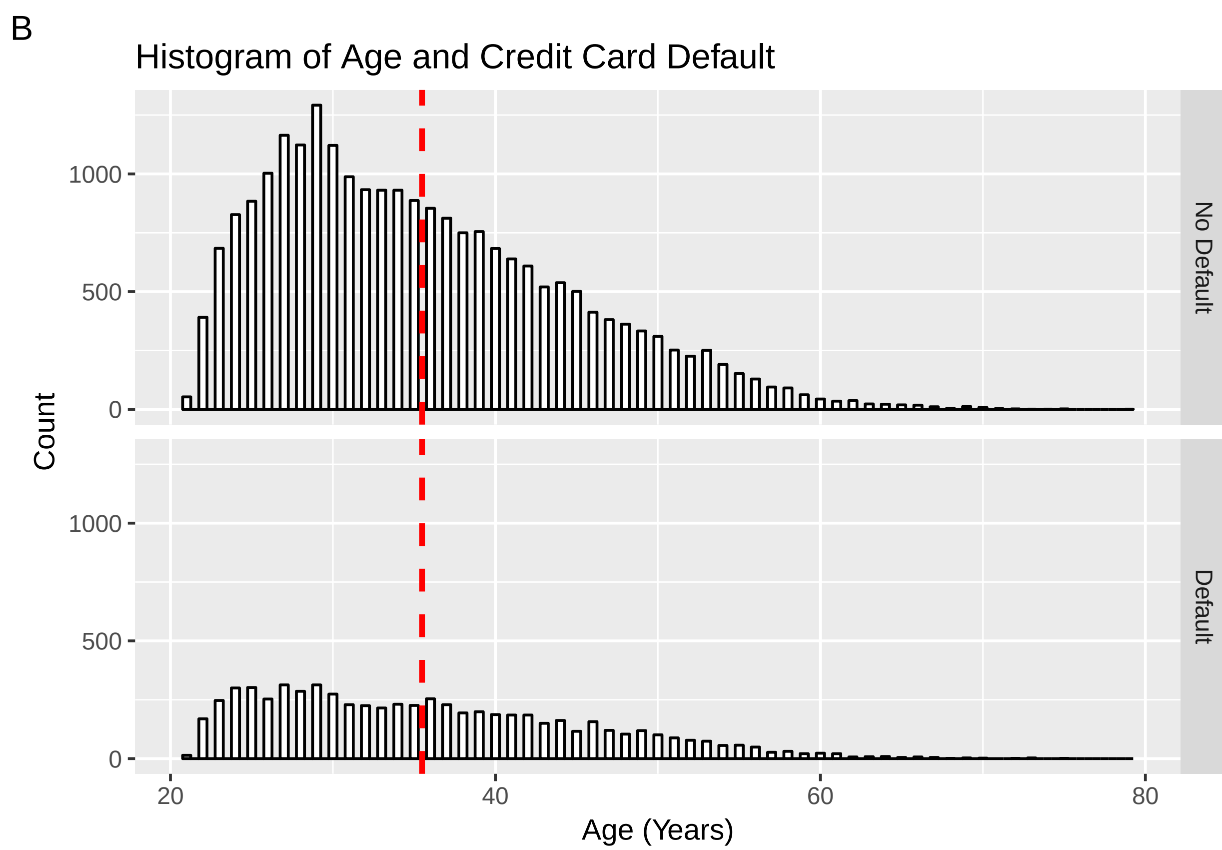



```
## Comment: Middle ages have higher amount of given credit in NT dollars
```

```r
# Plot 2 ---------------------------------------------------------------------------
```

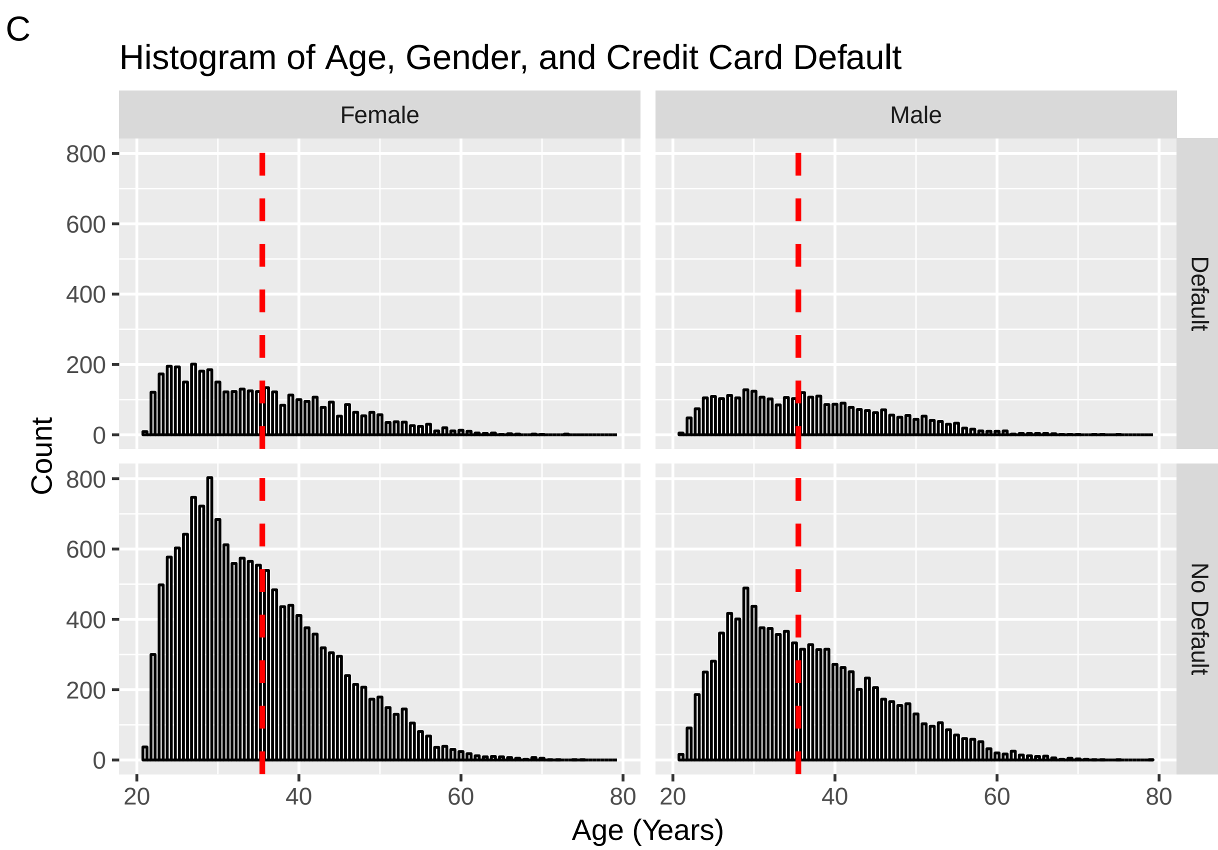
```r
# The original dataset in default.payment.next.month column shows 0 and 1,
# which mean credit card does not default and do default respectively.
# It is not so clear for someone who does not know this dataset,
# so I changed the labels of default.payment.next.month on the plot

# First, given a list for converting the labels
default_names = list('0' ="No Default", '1'= "Default")

# Then, define a labeller to convert labels of default.payment.next.month
default_labeller = function(variable,value){
  return(default_names[value])
}

ggplot(data=CreditCard_plot, aes(x=AGE)) +
  geom_histogram(binwidth=.5, colour="black", fill="white") +
  facet_grid(default.payment.next.month ~., labeller=default_labeller) +
  geom_vline(data=CreditCard_plot, aes(xintercept=mean(AGE, na.rm=T)),
             linetype="dashed", size=1, colour="red") +
  labs(title = "Histogram of Age and Credit Card Default", x = "Age (Years)",
       y = "Count", tag = "B")
```

```
## Warning: The labeller API has been updated. Labellers taking `variable`and
## `value` arguments are now deprecated. See labellers documentation.
```

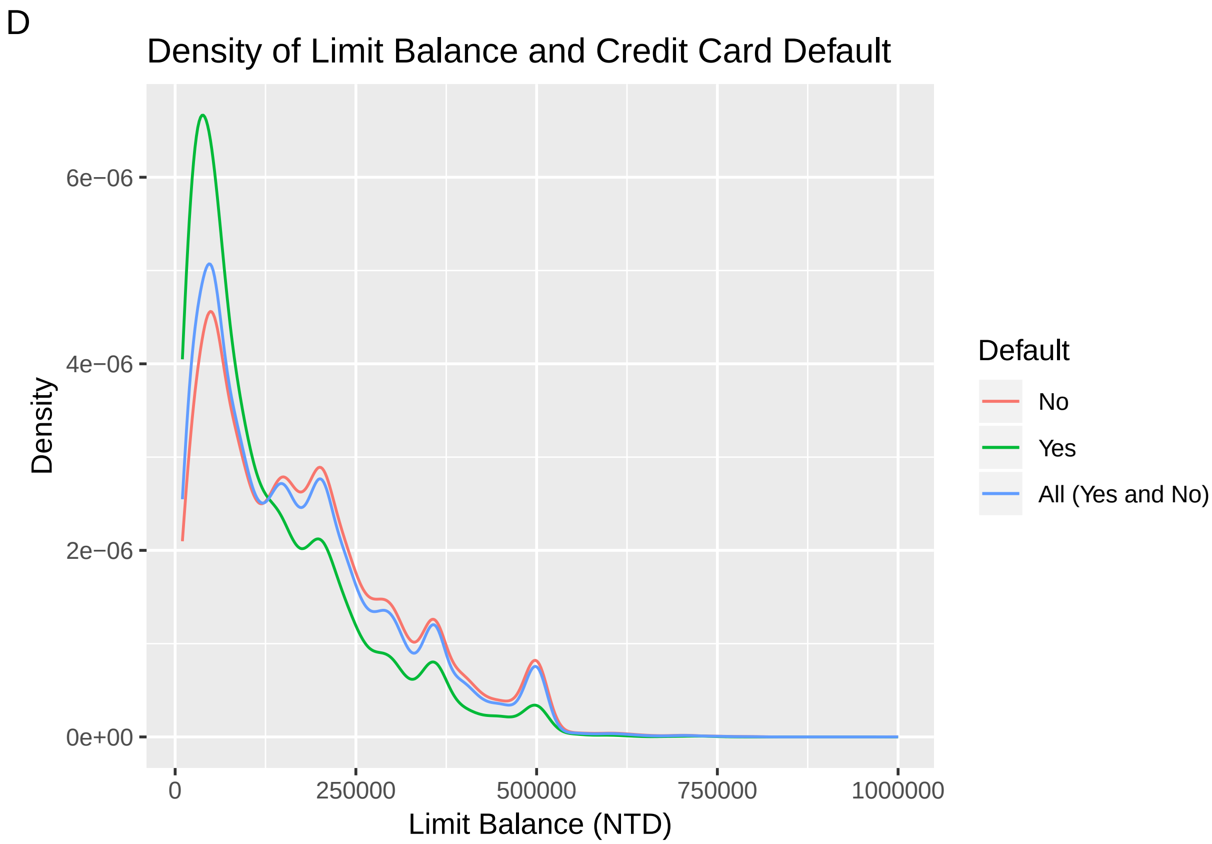



```
## Comment: The bar charts shows it is lower percentage of credit card default
##          for people between age 25 and age 40. Also, most of the clients are
```

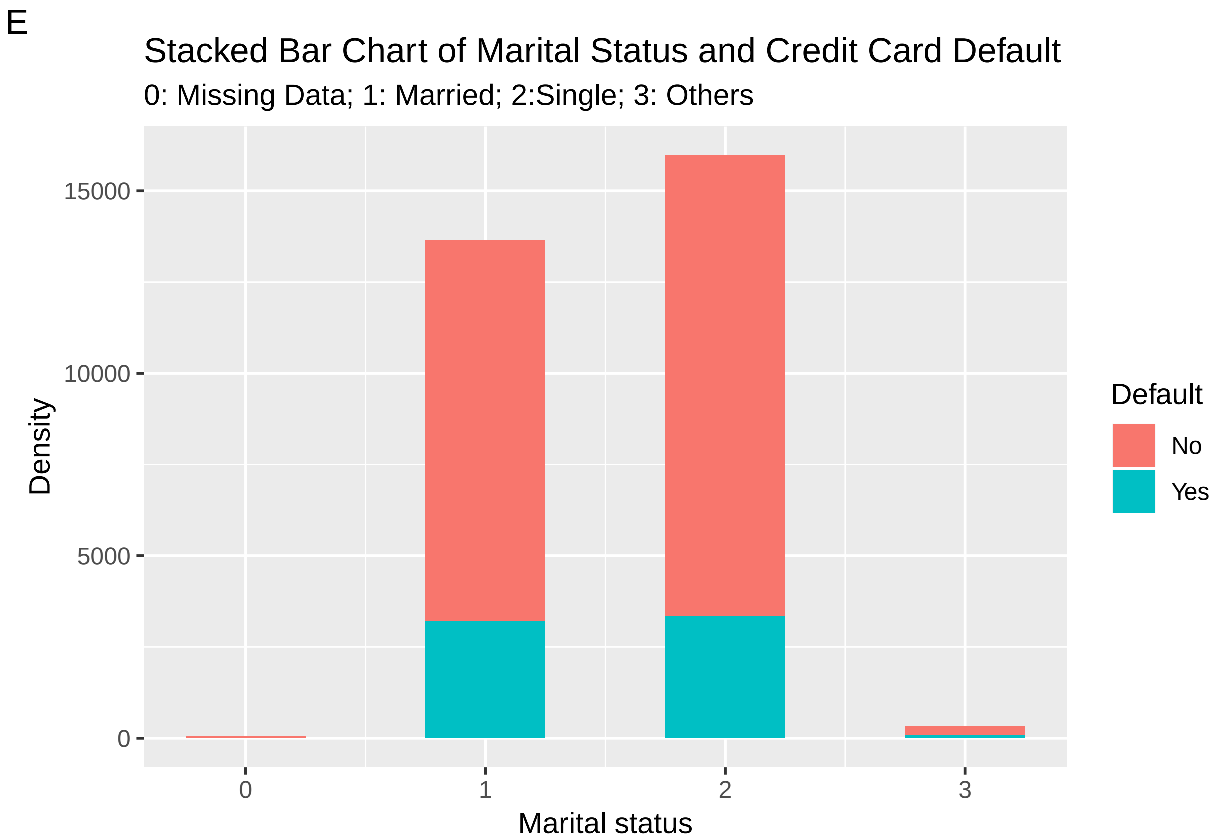
```
##          between age 25 and age 35.

# Plot 3 ---------------------------------------------------------------------------
CreditCard_plot2 = CreditCard_plot
CC_Default = CreditCard_plot$default.payment.next.month
CreditCard_plot2$default.payment.next.month[CC_Default=="1"] = "Default"
CreditCard_plot2$default.payment.next.month[CC_Default=="0"] = "No Default"
CreditCard_plot2$SEX[CreditCard_plot$SEX=="1"] = "Male"
CreditCard_plot2$SEX[CreditCard_plot$SEX=="2"] = "Female"

ggplot(data=CreditCard_plot2, aes(x=AGE)) +
  geom_histogram(binwidth=.5, colour="black", fill="white") +
  facet_grid(default.payment.next.month ~SEX) +
  geom_vline(data=CreditCard_plot2, aes(xintercept=mean(AGE, na.rm=T)),
             linetype="dashed", size=1, colour="red") +
  labs(title = "Histogram of Age, Gender, and Credit Card Default", x = "Age (Years)",
       y = "Count", tag = "C")
```




```
# Plot 4 ---------------------------------------------------------------------------
ggplot(data=CreditCard_plot, aes(x=LIMIT_BAL, colour=default.payment.next.month)) +
  stat_density(geom="line",position="identity") +
  stat_density(geom="line", aes(color = "default.payment.next.month")) +
  labs(title = "Density of Limit Balance and Credit Card Default",
       x = "Limit Balance (NTD)", y = "Density", tag = "D") +
  scale_colour_discrete(name="Default", breaks=c("0", "1", "default.payment.next.month"),
                        labels=c("No", "Yes", "All (Yes and No)"))
```

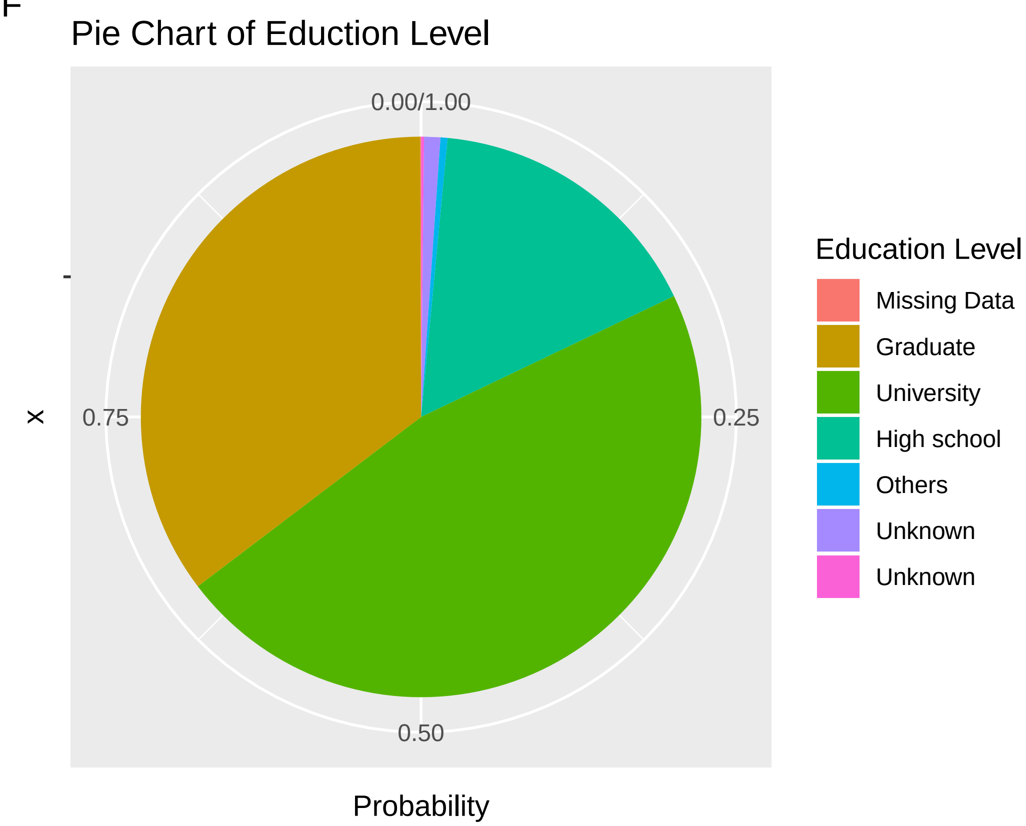



```
## Comment: Light blue line, which represents the density of credit card default,
##          has a high peak at limit balance about 10000 NTD. It might tell us
##          that credit card might be too easy to approve without careful
##          considerations of applicants' credit score.

# Plot 5 ---------------------------------------------------------------------
ggplot(data=CreditCard_plot, aes(MARRIAGE, fill = default.payment.next.month)) +
  labs(title = "Stacked Bar Chart of Marital Status and Credit Card Default",
       subtitle = ("0: Missing Data; 1: Married; 2:Single; 3: Others"),
       x = "Marital status", y = "Density", tag = "E") + geom_histogram(bins = 7) +
  scale_fill_discrete(name="Default", breaks=c("0", "1"), labels=c("No", "Yes"))
```




```
## Comment: Most of cardholders were married or single.
##          Single had less probabilities of credit card default by percentage.

# Plot 6 ------------------------------------------------------------------------
edu_count_table = as.data.frame(table(CreditCard_plot$EDUCATION))
edu_count_table$Prob = edu_count_table$Freq / sum(edu_count_table$Freq)
colnames(edu_count_table) = c("Edu", "Freq", "Prob" )

ggplot(data=edu_count_table, aes(x="", y=Prob, fill=Edu)) +
  geom_bar(width = 1, stat = "identity") + coord_polar("y", start=0) +
  labs(title = "Pie Chart of Eduction Level", y = "Probability", tag = "F") +
  scale_fill_discrete(name="Education Level", breaks=seq(0,6),
                      labels=c("Missing Data", "Graduate", "University",
                               "High school", "Others", "Unknown", "Unknown"))
```

F




```
## Comment: More than two-third cardholders have Bachelor's degree.
```

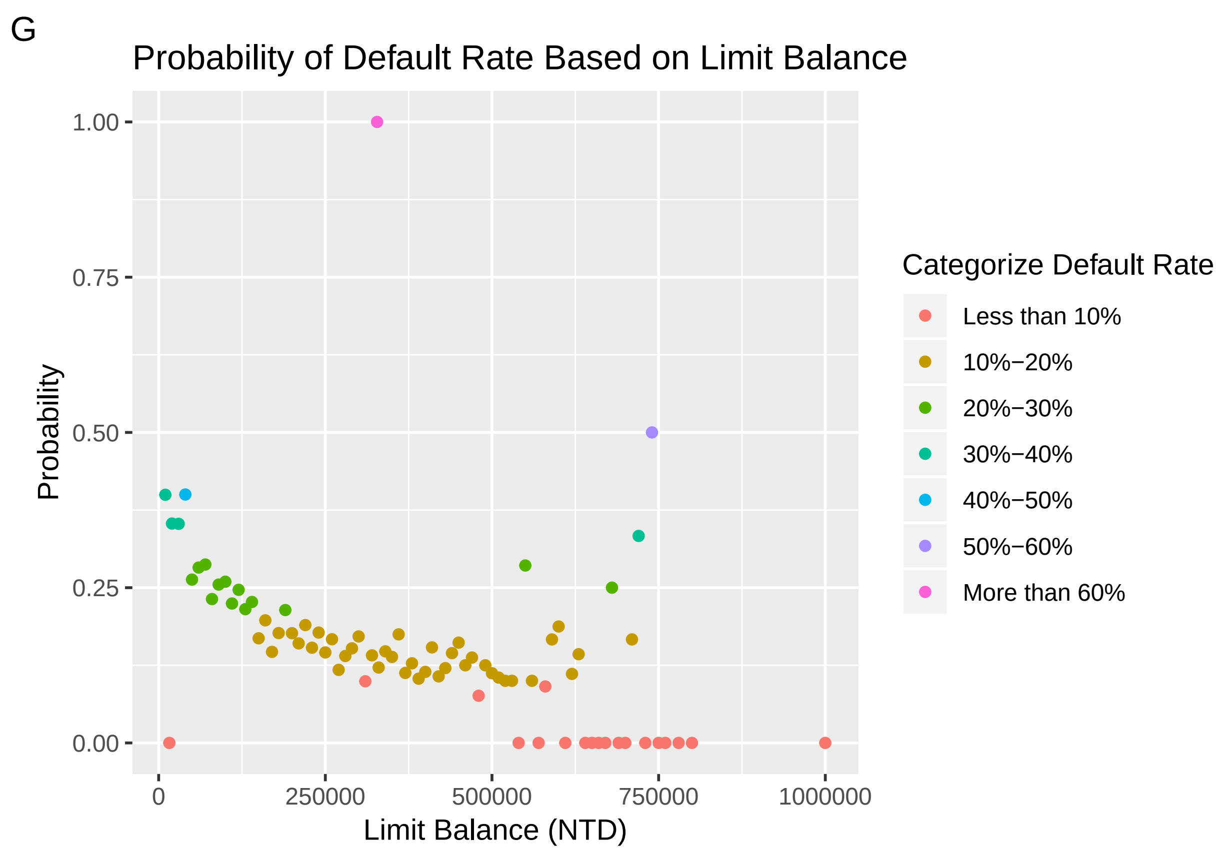
```
# Get default data
default = CreditCard[CreditCard$default.payment.next.month == 1,]
summary(default)
```

```
##        ID           LIMIT_BAL           SEX          EDUCATION
##  Min.   :    1   Min.   : 10000   Min.   :1.000   Min.   :1.000
##  1st Qu.: 7408   1st Qu.: 50000   1st Qu.:1.000   1st Qu.:1.000
##  Median :14758   Median : 90000   Median :2.000   Median :2.000
##  Mean   :14774   Mean   :130110   Mean   :1.567   Mean   :1.895
##  3rd Qu.:21832   3rd Qu.:200000   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :30000   Max.   :740000   Max.   :2.000   Max.   :6.000
##     MARRIAGE          AGE            PAY_0              PAY_2
##  Min.   :0.000   Min.   :21.00   Min.   :-2.0000   Min.   :-2.0000
##  1st Qu.:1.000   1st Qu.:28.00   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median :2.000   Median :34.00   Median : 1.0000   Median : 0.0000
##  Mean   :1.528   Mean   :35.73   Mean   : 0.6682   Mean   : 0.4583
##  3rd Qu.:2.000   3rd Qu.:42.00   3rd Qu.: 2.0000   3rd Qu.: 2.0000
##  Max.   :3.000   Max.   :75.00   Max.   : 8.0000   Max.   : 7.0000
##      PAY_3              PAY_4              PAY_5              PAY_6
##  Min.   :-2.0000   Min.   :-2.0000   Min.   :-2.0000   Min.   :-2.0000
##  1st Qu.:-1.0000   1st Qu.:-1.0000   1st Qu.:-1.0000   1st Qu.:-1.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.3621   Mean   : 0.2545   Mean   : 0.1679   Mean   : 0.1121
##  3rd Qu.: 2.0000   3rd Qu.: 2.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   : 8.0000   Max.   : 8.0000   Max.   : 8.0000   Max.   : 8.0000
```

```
##    BILL_AMT1        BILL_AMT2        BILL_AMT3        BILL_AMT4
##  Min.   : -6676   Min.   :-17710   Min.   :-61506   Min.   :-65167
##  1st Qu.:  2988   1st Qu.:  2694   1st Qu.:  2500   1st Qu.:  2142
##  Median : 20185   Median : 20300   Median : 19834   Median : 19120
##  Mean   : 48509   Mean   : 47284   Mean   : 45182   Mean   : 42037
##  3rd Qu.: 59626   3rd Qu.: 57920   3rd Qu.: 54734   3rd Qu.: 50176
##  Max.   :613860   Max.   :581775   Max.   :578971   Max.   :548020
##    BILL_AMT5        BILL_AMT6         PAY_AMT1         PAY_AMT2
##  Min.   :-53007   Min.   :-339603   Min.   :     0   Min.   :     0
##  1st Qu.:  1503   1st Qu.:   1150   1st Qu.:     0   1st Qu.:     0
##  Median : 18478   Median :  18028   Median :  1636   Median :  1534
##  Mean   : 39540   Mean   :  38271   Mean   :  3397   Mean   :  3389
##  3rd Qu.: 47853   3rd Qu.:  47424   3rd Qu.:  3478   3rd Qu.:  3310
##  Max.   :547880   Max.   : 514975   Max.   :300000   Max.   :358689
##     PAY_AMT3         PAY_AMT4         PAY_AMT5         PAY_AMT6
##  Min.   :     0   Min.   :     0   Min.   :     0   Min.   :     0
##  1st Qu.:     0   1st Qu.:     0   1st Qu.:     0   1st Qu.:     0
##  Median :  1222   Median :  1000   Median :  1000   Median :  1000
##  Mean   :  3367   Mean   :  3156   Mean   :  3219   Mean   :  3442
##  3rd Qu.:  3000   3rd Qu.:  2939   3rd Qu.:  3000   3rd Qu.:  2974
##  Max.   :508229   Max.   :432130   Max.   :332000   Max.   :345293
##  default.payment.next.month
##  Min.   :1
##  1st Qu.:1
##  Median :1
##  Mean   :1
##  3rd Qu.:1
##  Max.   :1
```

```
# Define function to calculate default rate based on one factor
DefaultRate = function(tab){

  names = colnames(tab) # Get column names of table
  N = 2:6 # Get LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE
  DefaultRateList = list() # Initialize

  for ( i in 1:length(N)){

    factor = names[N[i]]
    fre_cc = as.data.frame(table(CreditCard[factor]))
    fre_de = as.data.frame(table(default[factor]))

    # Left join
    fre_table = merge(fre_cc, fre_de, by='Var1', all.x=TRUE)

    fre_table[is.na(fre_table)] = 0 # Replace NA as 0
    colnames(fre_table) = c(factor, 'AllData', 'Default')

    # Get the default rate and count of no default
    fre_table$NoDefault = fre_table$AllData - fre_table$Default
    fre_table$Rate = fre_table$Default / fre_table$AllData
    DefaultRateList[[i]] = as.matrix(fre_table)

  }
```

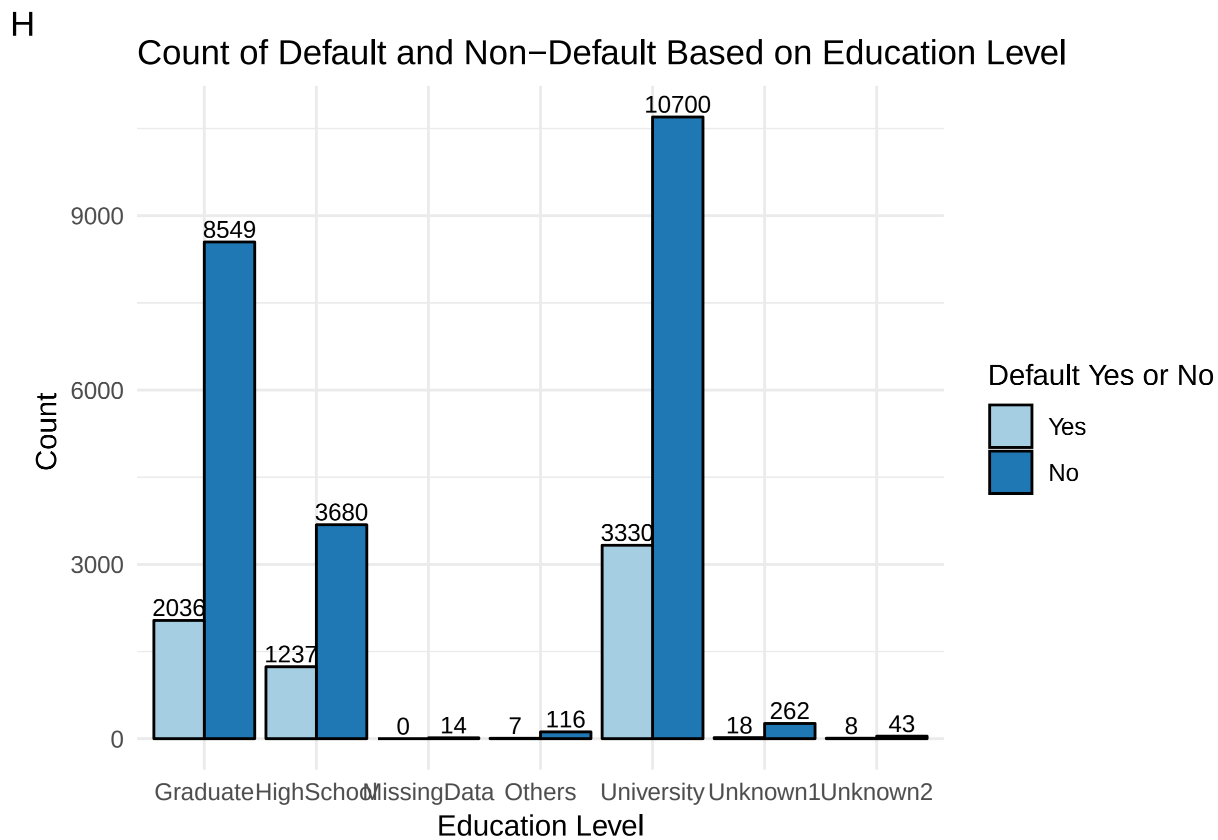
```r
  return(DefaultRateList)
}

DefaultRateMat = DefaultRate(CreditCard)

# Plot 7 ---------------------------------------------------------------------
# Extract LIMIT_D_R default rate
LimitBal_D_R = as.data.frame(DefaultRateMat[[1]])
LimitBal_D_R$LIM_cut = cut(as.numeric(as.character(LimitBal_D_R$LIMIT_BAL)),
                           c((0:8)*130000), right = FALSE,
                           labels = c("0-130K", "130K-260K", "260K-390K",
                                      "390K-520K", "520K-650K", "650K-780K",
                                      "780K-910K", "910K-1040K")) # Categorize LIMIT_BAL
LimitBal_D_R$Rate_cut = cut(as.numeric(as.character(LimitBal_D_R$Rate)),
                            c(seq(0,1.1,0.1)), right = FALSE) # Categorize Defualt Rate

# Visualize the default of LIMIT_BAL
LimitBal_D_R$Rate = as.numeric(as.character(LimitBal_D_R$Rate))
LimitBal_D_R$LIMIT_BAL = as.numeric(as.character(LimitBal_D_R$LIMIT_BAL))
ggplot(data=LimitBal_D_R, aes(x=LIMIT_BAL, y=Rate, color=Rate_cut)) +
  geom_point() + labs(title = "Probability of Default Rate Based on Limit Balance",
                      x = "Limit Balance (NTD)", y = "Probability", tag = "G") +
  scale_colour_discrete(name="Categorize Default Rate",
                        labels=c("Less than 10%", "10%-20%", "20%-30%",
                                 "30%-40%", "40%-50%", "50%-60%","More than 60%"))
```



```
## Comment: The average of default rate is relatively low
##          when the balance limit is between 250,000 and 500,000.
##          The volatility of default rate is relatively high
##          when the balance limit is between 500,000 and 750,000.

# Plot 8 -------------------------------------------------------------------------
Edu_D_R = as.data.frame(DefaultRateMat[[3]])
Edu_D_R$Rate = as.numeric(as.character(Edu_D_R$Rate))
Edu_D_R$EDUCATION = c("MissingData", "Graduate", "University",
                                  "HighSchool", "Others", "Unknown1", "Unknown2")

Edu_D_R_new = Edu_D_R[,c(1,3,4,5)] %>% gather(DefaultYN, Count, Default:NoDefault)
```

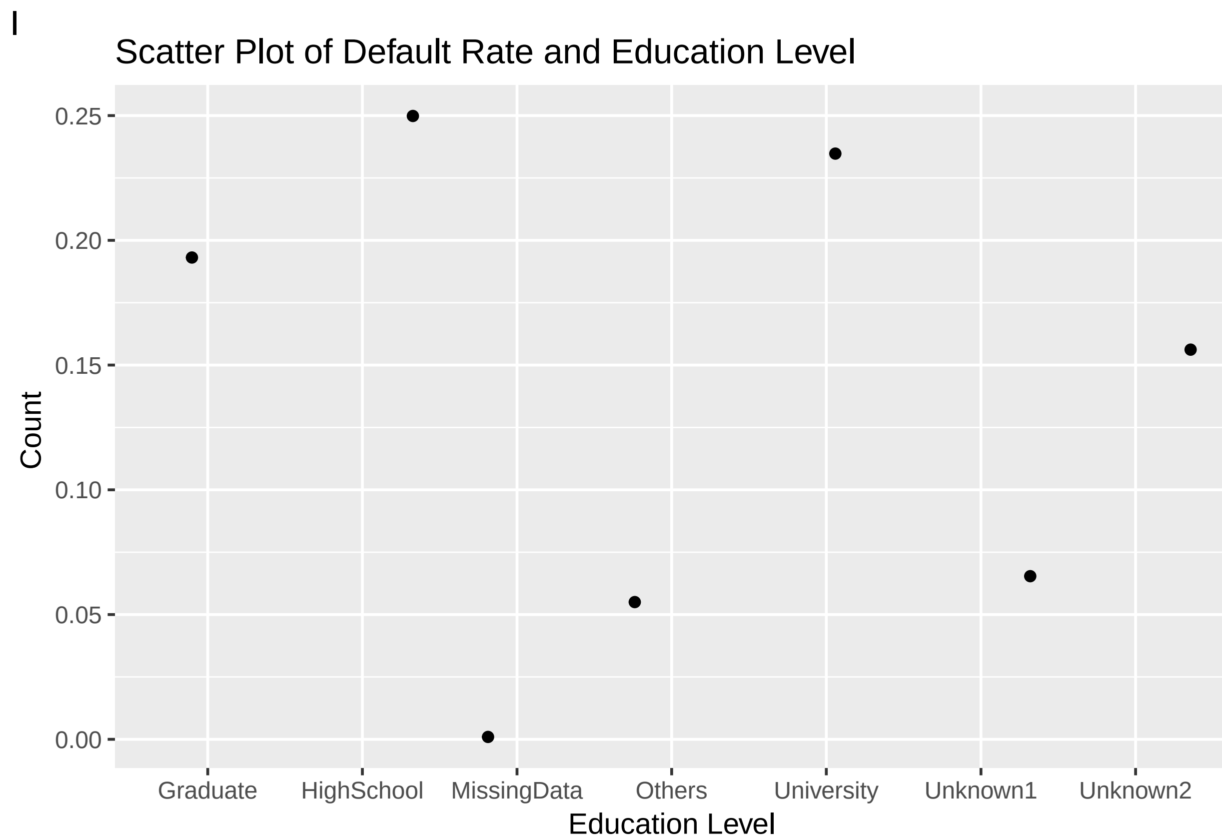
```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

```
Edu_D_R_new$Count = as.numeric(as.character(Edu_D_R_new$Count))
ggplot(data=Edu_D_R_new, aes(x=EDUCATION, y=Count, fill=DefaultYN)) +
  geom_bar(colour="black", stat="identity", position=position_dodge()) +
  geom_text(aes(label=Count), position=position_dodge(width=0.9), vjust=-0.25, size=3) +
  labs(title = "Count of Default and Non-Default Based on Education Level",
       x = "Education Level", y = "Count ", tag = "H") + theme_minimal() +
  scale_fill_brewer(palette="Paired", name="Default Yes or No", labels=c("Yes", "No"))
```



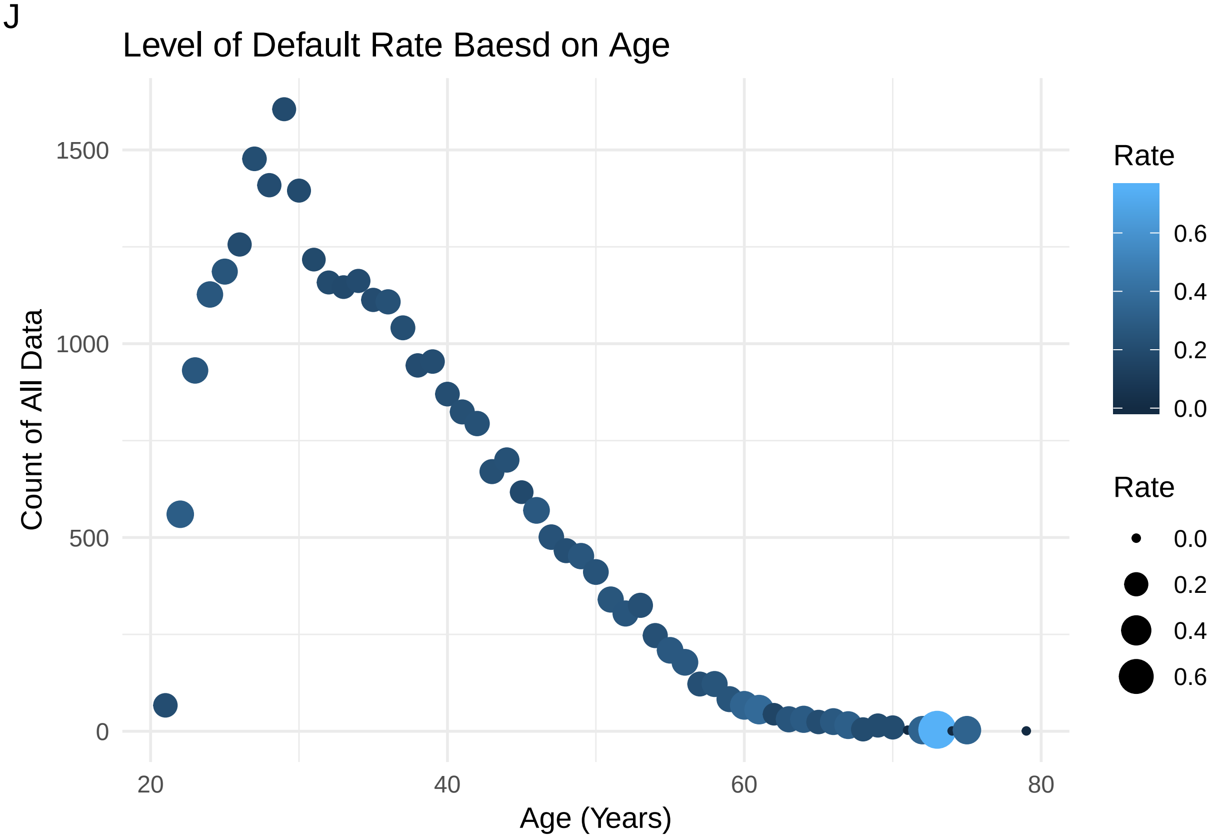

```
## Comment: Most of the cardholders have bachelor's degree or master's degree.

ggplot(data=Edu_D_R, aes(x=EDUCATION, y=Rate)) + geom_jitter(aes(x=EDUCATION, y=Rate)) +
```

```
  labs(title = "Scatter Plot of Default Rate and Education Level",
       x = "Education Level", y = "Count ", tag = "I")
```




```
## Comment: Clients with higher education level has lower default rate.


# Plot 9 -------------------------------------------------------------------------
Age_D_R = as.data.frame(DefaultRateMat[[5]])

# Convert format to numeric
for (i in 1:length(Age_D_R)){
  Age_D_R[,i] = as.numeric(as.character(Age_D_R[,i]))
}
ggplot(Age_D_R, aes(x=AGE, y=AllData, color=Rate)) +
  geom_point(aes(size=Rate)) +
  labs(title = "Level of Default Rate Baesd on Age",
       x = "Age (Years)", y = "Count of All Data", tag = "J") + theme_minimal()
```

J




```
## Comment: More people have credit card between age 22 and age 40.
##          The variance is higher for clients older than 60.
```

```
ftable(CreditCard$SEX, CreditCard$EDUCATION, CreditCard$MARRIAGE)
```

```
##          0    1    2    3
##
## 1 0      0    2    6    0
##   1      1 1690 2633   30
##   2      1 2370 2940   63
##   3     12 1048  894   36
##   4      0   18   23    1
##   5      0   48   46    1
##   6      0   14   11    0
## 2 0      0    2    4    0
##   1      3 2032 4176   20
##   2      5 4472 4080   99
##   3     32 1813 1015   67
##   4      0   34   45    2
##   5      0  102   81    2
##   6      0   14   10    2
```

```
## We can see some unknown data and 0 in this data set
##### Note. can do predict for those missing data set as well,
##### but have not included in this project.

# Data Cleaning
```

```r
# Remove rows with Education=0,5,6 and MARRIAGE=0,3 and LIMIT_BAL,SEX,AGE=0
without0 = apply(CreditCard,1, function(x) all(x[2:6]!=0) && x[4]!=5 && x[4]!=6 && x[5]!=3)
CreditCard = CreditCard[without0,]
```

There are 24 factors against amount of given credit. In order to aviod overfitting, I selected the most important factors using forward stepwise selection.

Algorithm: Forward Stepwise Selection a. Let $\mathcal{M}_{\prime}$ denote the null model, which contains no predictors. b. For $k = 0, 1, \ldots, p-1$; p is the number of predictors b.1 Consider all $p-k$ models that augment the predictors in $\mathcal{M}_{\|}$ with one additional predictor b.2 Choose the best among these $p-k$ models, and call it $\mathcal{M}_{\|-\infty}$. Here best is defined as having the smallest RSS, or equivalently largest R^2. c. Select a single best model fromamong $\mathcal{M}_{\prime}, \mathcal{M}_{\infty}, \ldots, \mathcal{M}_{\checkmark}$ using cross-validated prediction error, Cp, AIC, BIC, or adjusted R^2.

At the step c, I chose Bayesian Information Criterion (BIC) for determining the cross-validated prediction error. The Bayesian Information Criterion (BIC) gives unnecessary variable much greater penalty, so it can more efficient to aviod overfitting.

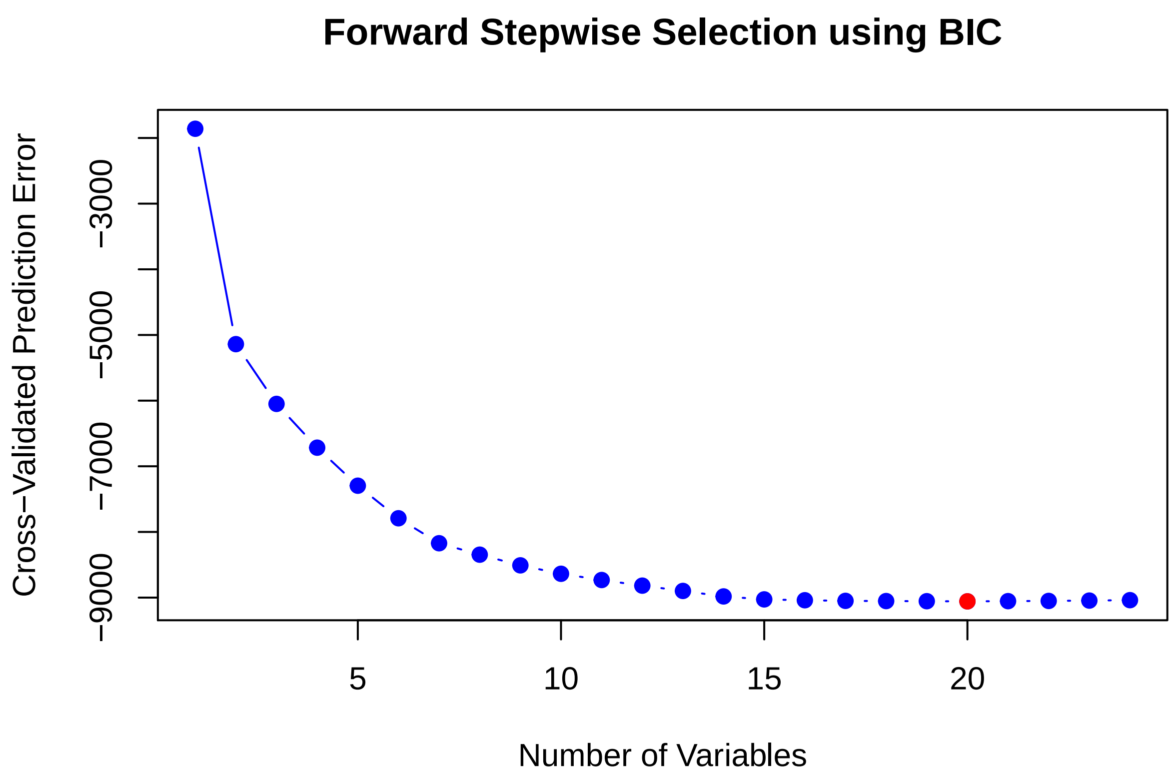
Criteria: Bayesian Information Criterion For the least squares model with d predictors up to irrelevant constants

$$BIC = \frac{1}{n}(RSS + d\hat{\sigma}^2 \log n)$$

Since $\log n > 2$ for $n > 7$, the BIC places a heavier penalty on models with many variables.

$\mathcal{M}_{\prime}$ ## 2. Quantitative factors as responses Statistic Learning on LIMIT_BAL against other factors

```r
# Set up the 70% for train set and the rest of 30% for test set
train = round(nrow(CreditCard) * 0.7,0)
train = sample(nrow(CreditCard) ,train)
CreditCard.train = CreditCard[train, ]
CreditCard.test = CreditCard[-train, ]

# Determine which parameters are more important using forward selection
Forward = regsubsets(LIMIT_BAL ~., data = CreditCard.train, method="forward",
                     nvmax=length(CreditCard)-1)
Forward.summary = summary(Forward)

# Determine how many paramenters will be used using **Bayesian Information Criterion (BIC)**
plot(Forward.summary$bic,type='b',col="blue", pch=19, xlab = "Number of Variables",
     ylab = "Cross-Validated Prediction Error",
     main = "Forward Stepwise Selection using BIC")
points(which.min(Forward.summary$bic), Forward.summary$bic[which.min(Forward.summary$bic)],
       col="red", pch=19)
```




```
# List all improtant parameters using BIC
GetColumn = t(Forward.summary$which)[,which.min(Forward.summary$bic)]
## Remove LIMIT_BAL from column names because LIMIT_BAL is the response of regression
NameCol = names(CreditCard)[-2]
NameCol[GetColumn[2:length(CreditCard)]]
```

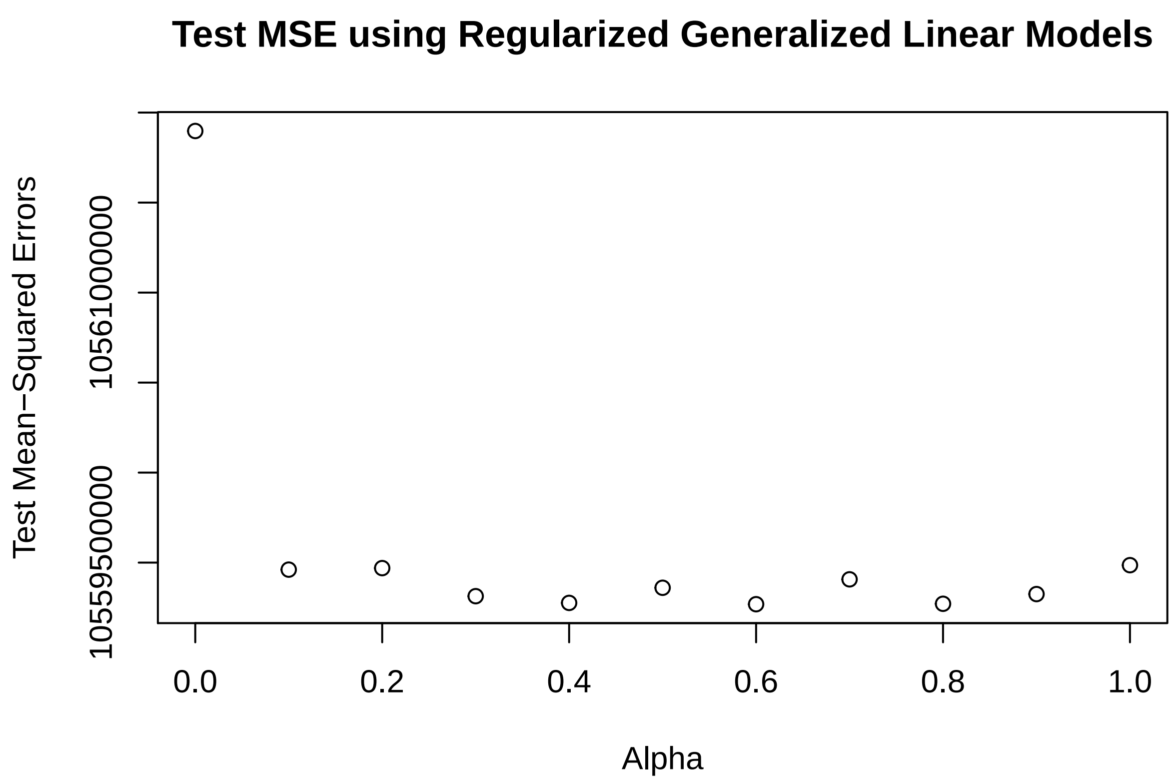
```
##  [1] "SEX"       "EDUCATION" "MARRIAGE"  "AGE"       "PAY_0"
##  [6] "PAY_2"     "PAY_3"     "PAY_5"     "PAY_6"     "BILL_AMT1"
## [11] "BILL_AMT2" "BILL_AMT3" "BILL_AMT4" "BILL_AMT5" "PAY_AMT1"
## [16] "PAY_AMT2"  "PAY_AMT3"  "PAY_AMT4"  "PAY_AMT5"  "PAY_AMT6"
```

```
## I pick 20 parameters, which has the minimum error using bic
```

```
# Set the formula for part 2 of this project
formulaQ2 =
  as.formula(LIMIT_BAL ~ .-ID-PAY_4-BILL_AMT6-default.payment.next.month)

# Logistic Regression
fit.lm = lm(formulaQ2, data = CreditCard.train)
# Predict
yhat.lm = predict(fit.lm, CreditCard.test)

# Test MSE
mse_lm = round(mean((yhat.lm - CreditCard.test$LIMIT_BAL)^2), 4)
paste("The test MSE using linear regession is", mse_lm)
```

```
## [1] "The test MSE using linear regession is 10560901025.4155"
```

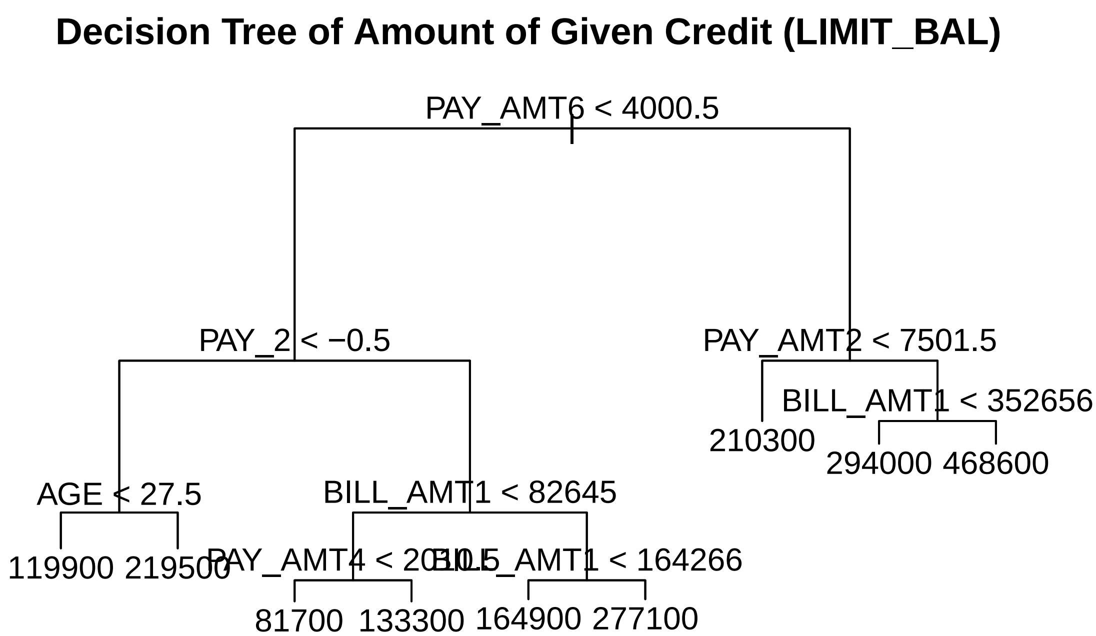
```r
####################################################################################
# Lasso and Elastic-Net Regularized Generalized Linear Models
tryalpha = seq(0,1,0.1)
x.train = model.matrix(formulaQ2, data = CreditCard.train)
x.test = model.matrix(formulaQ2, data = CreditCard.test)
y = CreditCard.train$LIMIT_BAL

mse_glmnet = rep(NA, length(tryalpha))
for (i in 1:length(tryalpha)){

  fit.glmnet = glmnet(x.train, y, alpha = tryalpha[i])
  pred.glmnet = predict(fit.glmnet, s = 0.01, newx = x.test)
  mse_glmnet[i] = round(mean((pred.glmnet - CreditCard.test$LIMIT_BAL)^2), 4)

}
plot(tryalpha, mse_glmnet, xlab = "Alpha", ylab = "Test Mean-Squared Errors",
     main = "Test MSE using Regularized Generalized Linear Models")
```




```r
# Test MSE
# Lasso and Elastic-Net Regularized Generalized Linear Models (glmnet)
paste("The lowest test MSE using glmnet is",
      min(mse_glmnet), "with alpha =",
      tryalpha[which.min(mse_glmnet)], "as alpha is in [0, 1]")
```

```
## [1] "The lowest test MSE using glmnet is 10559269010.7383 with alpha = 0.6 as alpha is in [0, 1]"
```

```r
###################################################################################
# Tree
# Fit a regression tree to the training set
fit.tree = tree(formulaQ2, data = CreditCard.train)

# Plot the tree
plot(fit.tree)
text(fit.tree, pretty = 0)
title("Decision Tree of Amount of Given Credit (LIMIT_BAL)")
```

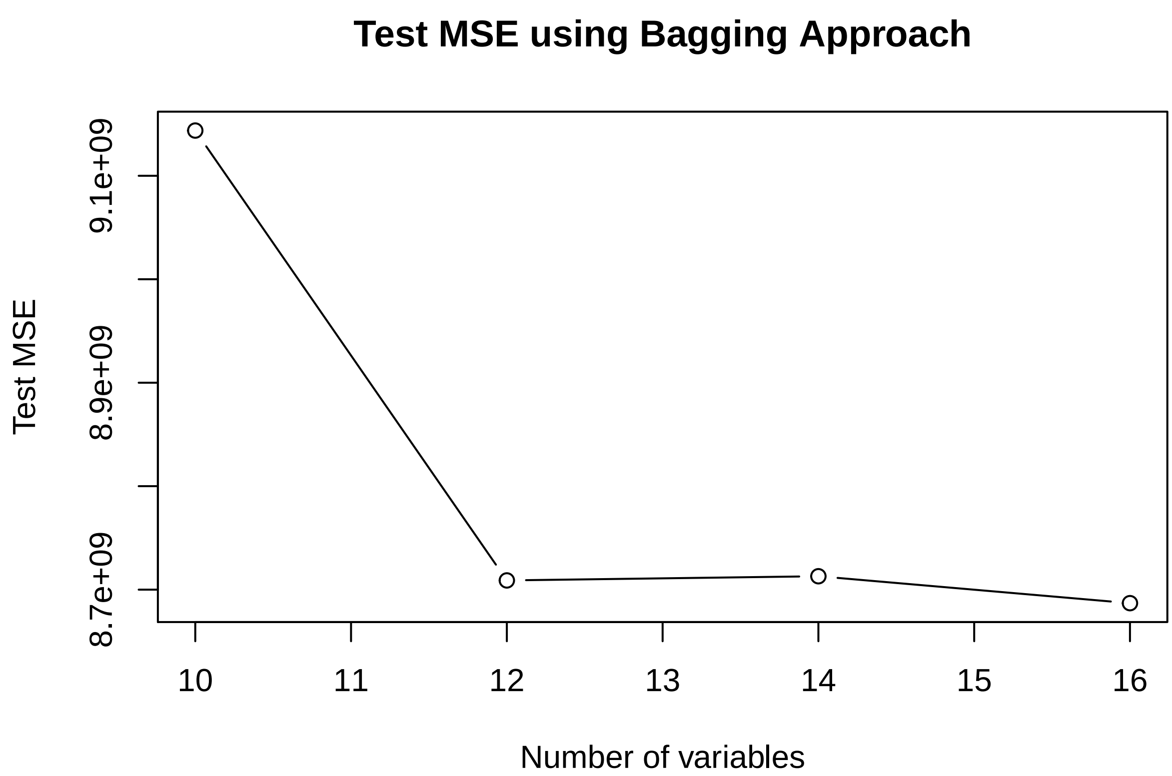



```r
# Test MSE
tree.pred = predict(fit.tree, newdata = CreditCard.test)
mse.tree = mean((tree.pred - CreditCard.test$LIMIT_BAL)^2)
print (paste("The test MSE using tree is", round(mse.tree,4) ))
```

```
## [1] "The test MSE using tree is 11370064101.2229"
```

```r
###################################################################################
# Random Forest
# Use the bagging approach
trymtry = seq(10,16,2) # (which.min(Forward.summary$bic)-1)

mse.bag = rep(NA, length(trymtry))
for (i in 1:length(trymtry)){

  fit.bag = randomForest(formulaQ2, data = CreditCard.train, mtry = i,
                         ntree = 500, importance = TRUE)
```

```r
  yhat.bag = predict(fit.bag, newdata = CreditCard.test)
  mse.bag[i] = round(mean((yhat.bag - CreditCard.test$LIMIT_BAL)^2),4)

}

plot(trymtry, mse.bag, type = "b", xlab = "Number of variables", ylab = "Test MSE",
     main = "Test MSE using Bagging Approach")
```

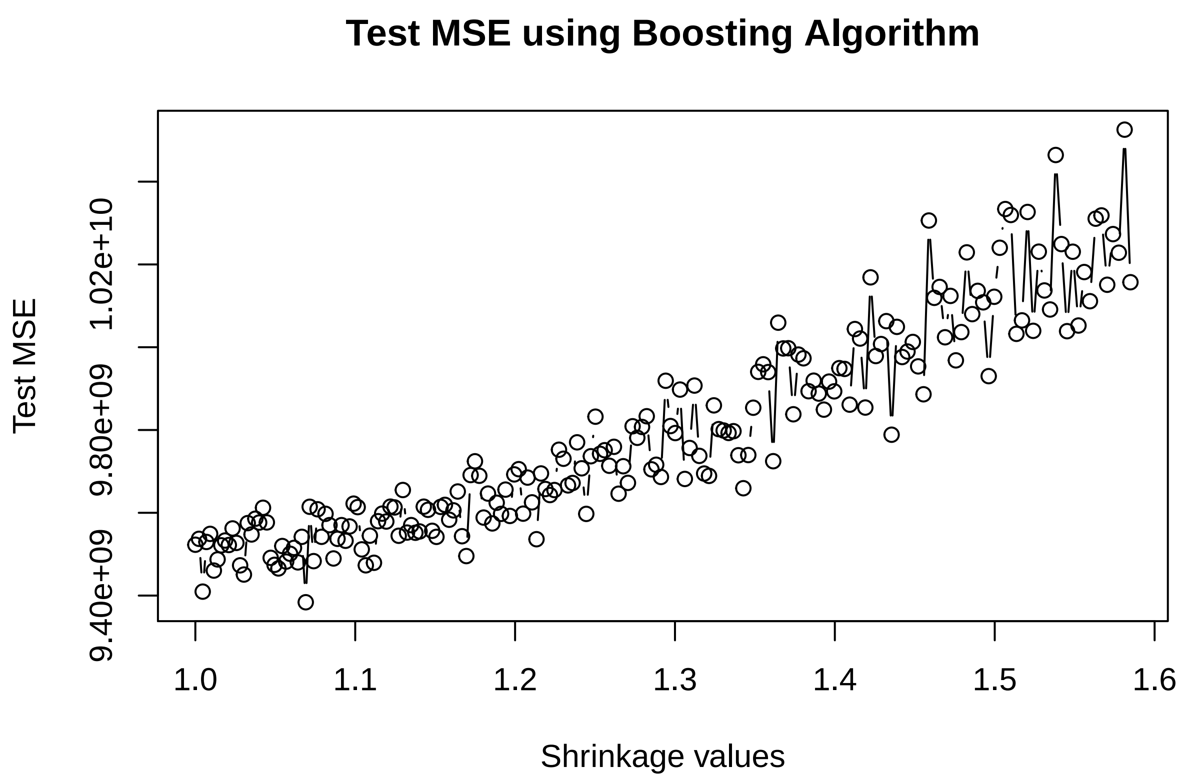



```r
paste("The lowest test MSE using bagging is",  min(mse.bag))
```

```
## [1] "The lowest test MSE using bagging is 8686941487.6679"
```

```r
##################################################################################
# Generalized Boosted Regression Models
# Give shrinkage parameters 'lambda'
lambdas = 10^seq(0, 0.2, by = 0.001)
test.err = rep(NA, length(lambdas))
for (i in 1:length(lambdas)) {

    boost.Limitbal = gbm(formulaQ2, data = CreditCard.train, distribution = "gaussian",
                         n.trees = 1000, shrinkage = lambdas[i])
    yhat = predict(boost.Limitbal, CreditCard.test, n.trees = 1000)
    test.err[i] = mean((yhat - CreditCard.test$LIMIT_BAL)^2)
}

# Plot with different shrinkage parameters on the x-axis
# and the corresponding training set MSE on the y-axis
```

```r
plot(lambdas, test.err, type = "b", xlab = "Shrinkage values", ylab = "Test MSE",
     main = "Test MSE using Boosting Algorithm")
```




```r
mse_boosting = round(min(test.err), 4)
paste("The test MSE using boosting is", mse_boosting)
```

```
## [1] "The test MSE using boosting is 9383846760.229"
```

```r
P2_accuracy = data.frame("Test MSE"=c( mse_lm, min(mse_glmnet),
                                      mse.tree, min(mse.bag), mse_boosting))
rownames(P2_accuracy) = c("lm", "glmnet", "tree", "bag", "boosting")
P2_accuracy
```

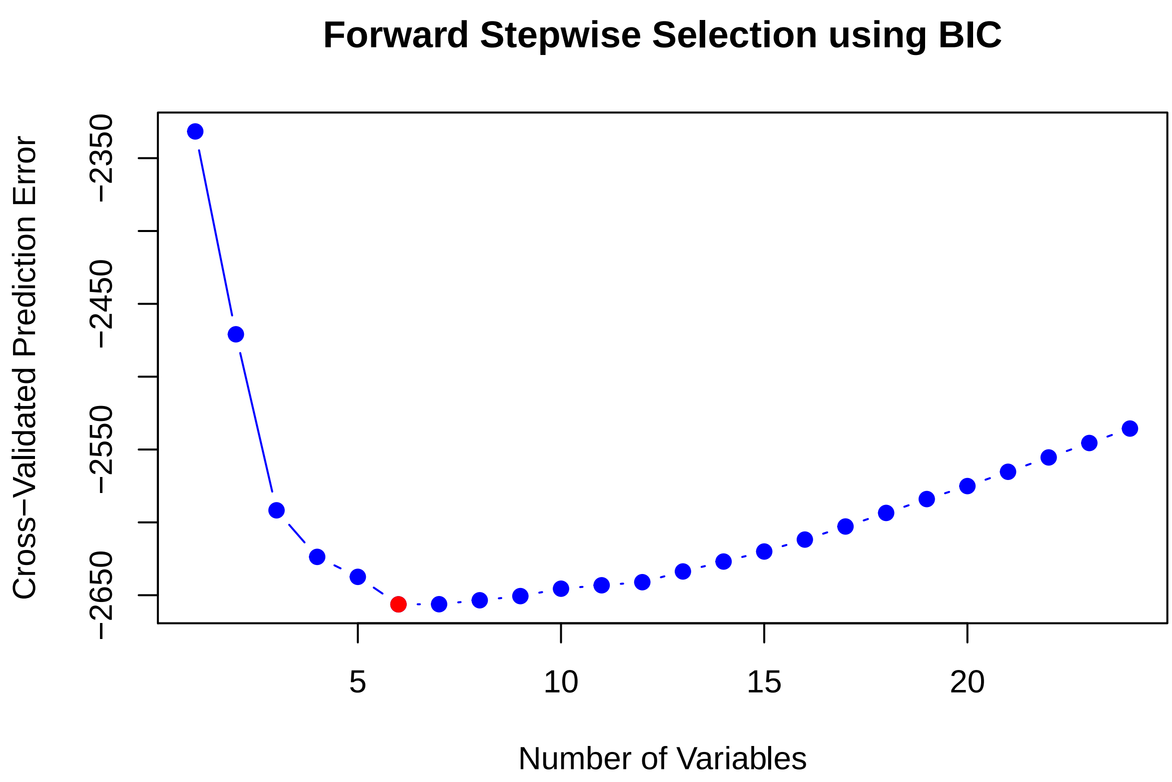
```
##             Test.MSE
## lm       10560901025
## glmnet   10559269011
## tree     11370064101
## bag       8686941488
## boosting  9383846760
```

I did machine leanring on this credit card dataset using five algorithms, including linear regression (lm), lasso and elastic-net regularized generalized linear models (glmnet), classification tree, bagging, and boosting. From the test MSE table, we can see linear regression is not a good fit for our credit card dataset to predict limit balance, even shrinking the coefficients α from 0 to 1. Among these five algorithms, bagging approach has the lowest test MSE. It is because bagging approach (Bootstrap aggregation) can reduce the variance and hence decrease the prediction mean-squared errors of a statistical learning method. Also, it takes many training sets from the population and build a separate prediction model using each training set. Then we average the resulting predictions. Thus, the test MSE is lower than the test MSE using a single.

3. Qualitative factors as responses

And then, I did the same process on whether clients default payment next month (default.payment.next.month) against others factors.

```
# Determine which parameters are more important using forward selection
Forward = regsubsets(default.payment.next.month ~., data = CreditCard.train,
                     method="forward", nvmax=length(CreditCard)-1)
Forward.summary = summary(Forward)

# Determine how many paramenters will be used using **Bayesian Information Criterion (BIC)**
plot(Forward.summary$bic,type='b',col="blue", pch=19,
     xlab = "Number of Variables",
     ylab = "Cross-Validated Prediction Error",
     main = "Forward Stepwise Selection using BIC")
points(which.min(Forward.summary$bic),
       Forward.summary$bic[which.min(Forward.summary$bic)],
       col="red", pch=19)
```




```
# List all improtant parameters using BIC
names(CreditCard) [t(Forward.summary$which)[,which.min(Forward.summary$bic)]
                   [2:length(CreditCard)]]
```

```
## [1] "MARRIAGE"  "PAY_0"     "PAY_2"     "PAY_3"     "BILL_AMT1" "PAY_AMT1"
```

```
## I picked 6 parameters, which has the minimum error using bic
```

```r
# Set formula for part 3 of this project
formulaQ3 =
  as.formula(default.payment.next.month~MARRIAGE+PAY_0+PAY_2+PAY_3+
                 BILL_AMT1+PAY_AMT1)

########################################################################################
# Generalized Linear Model with Logistic Regression
fit.glm = glm(formulaQ3, data = CreditCard.train, family = binomial)

fit.glm = glm(formulaQ3, data = CreditCard.train, family = binomial)

# Predict
pred.prob = predict(fit.glm, CreditCard.test, type = "response")
pred.glm = rep(0, length(pred.prob))
pred.glm[pred.prob > 0.5] = 1
pred.table = table(pred.glm, CreditCard.test$default.payment.next.month)
pred.table
```

```
##
## pred.glm    0    1
##        0 6604 1500
##        1  210  471
```

```r
# Sensitivity: TP/P = TPR
Sensitivity = pred.table[1,1] / sum(pred.table[,1])
# Specificity: TN/N = TNR
Specificity = pred.table[2,2] / sum(pred.table[,2])
# Accuracy: (TP + TN)/(P + N)
Accuracy = sum(pred.table[1,1], pred.table[2,2]) / sum(pred.table[,])
# Total Error Rate: (FP + FN)/(P + N)
TotalError = sum(pred.table[1,2],pred.table[2,1]) / sum(pred.table[,])

glm.Confusion = round(data.frame(Sensitivity, Specificity, Accuracy, TotalError),4)
row.names(glm.Confusion) = "GLM"

paste("The accuracy using Logistic regression is", Accuracy,4)
```

```
## [1] "The accuracy using Logistic regression is 0.805350028457598 4"
```

```r
########################################################################################
# Linear Discriminant Analysis (LDA)
fit.lda = lda(formulaQ3, data = CreditCard.train)

# Predict default.payment.next.month in tesing data set
pred.prob.lda = predict(fit.lda, CreditCard.test)

# Predict table
pred.table.lda = table(pred.prob.lda$class, CreditCard.test$default.payment.next.month)
pred.table.lda
```

```
##
##        0    1
##   0 6570 1457
##   1  244  514
```

```r
# Sensitivity: TP/P = TPR
Sensitivity = pred.table.lda[1,1] / sum(pred.table.lda[,1])
# Specificity: TN/N = TNR
Specificity = pred.table.lda[2,2] / sum(pred.table.lda[,2])
# Accuracy: (TP + TN)/(P + N)
Accuracy = sum(pred.table.lda[1,1],pred.table.lda[2,2]) / sum(pred.table.lda[,])
# Total Error Rate: (FP + FN)/(P + N)
TotalError = sum(pred.table.lda[1,2],pred.table.lda[2,1]) / sum(pred.table.lda[,])

lda.Confusion = round(data.frame(Sensitivity, Specificity, Accuracy, TotalError),4)
row.names(lda.Confusion) = "LDA"

paste("The accuracy using LDA is", Accuracy)
```

```
## [1] "The accuracy using LDA is 0.806374501992032"
```

```r
##########################################################################################
# Quadratic discriminant analysis (QDA)
fit.qda = qda(formulaQ3, data = CreditCard.train)

# Predict default.payment.next.month in tesing data set
pred.prob.qda = predict(fit.qda, CreditCard.test)

# Predict table
pred.table.qda = table(pred.prob.qda$class, CreditCard.test$default.payment.next.month)
pred.table.qda
```

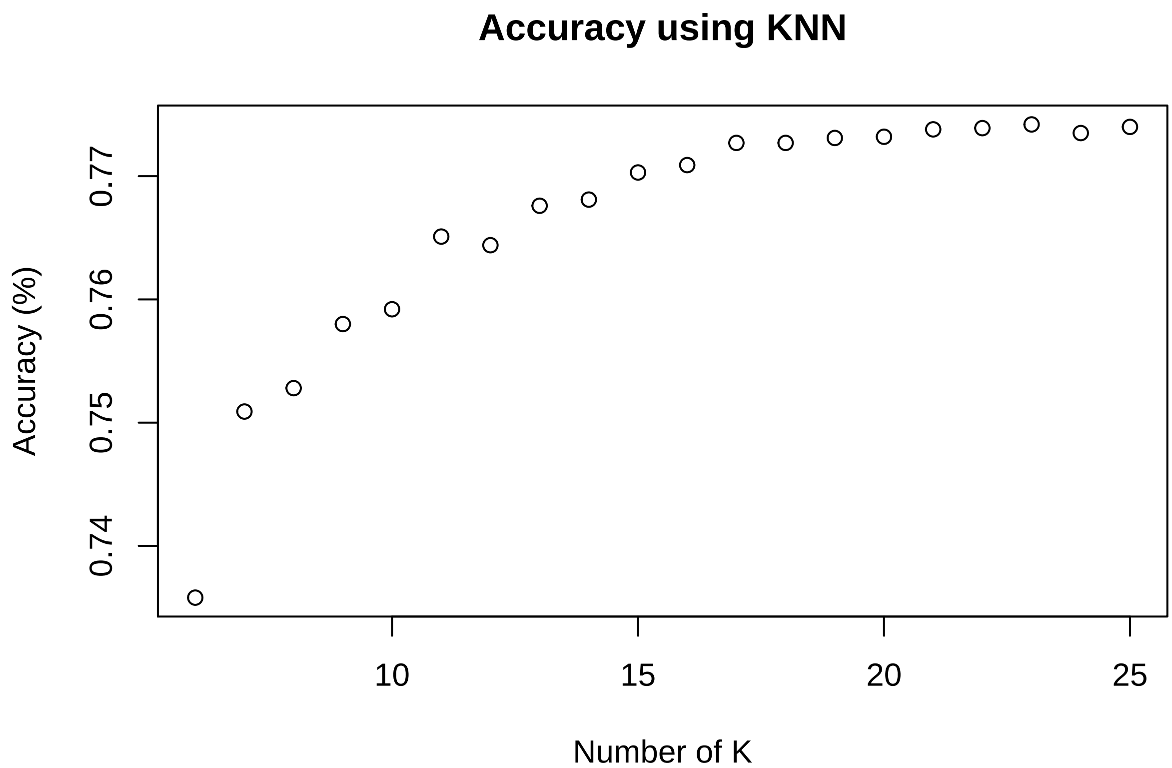
```
##
##         0    1
##    0 5574  841
##    1 1240 1130
```

```r
# Sensitivity: TP/P = TPR
Sensitivity = round(pred.table.qda[1,1] / sum(pred.table.qda[,1]),4)
# Specificity: TN/N = TNR
Specificity = round(pred.table.qda[2,2] / sum(pred.table.qda[,2]),4)
# Accuracy: (TP + TN)/(P + N)
Accuracy = round(sum(pred.table.qda[1,1],
                     pred.table.qda[2,2]) / sum(pred.table.qda[,]),4)
# Total Error Rate: (FP + FN)/(P + N)
TotalError = round(sum(pred.table.qda[1,2],
                       pred.table.qda[2,1]) / sum(pred.table.qda[,]),4)

qda.Confusion = data.frame(Sensitivity, Specificity, Accuracy, TotalError)
row.names(qda.Confusion) = "QDA"

paste("The accuracy using QDA is", Accuracy)
```

```
## [1] "The accuracy using QDA is 0.7631"
```

```r
##########################################################################################
# K-nearest neighbors algorithm
CreditCard.bic = dplyr::select(CreditCard, MARRIAGE, AGE, PAY_0, PAY_2,
                               PAY_3, BILL_AMT1, PAY_AMT1, PAY_AMT2)

# Set up the 70% for train set and the rest of 30% for test set
```

```r
train = round(nrow(CreditCard) * 0.7,0)
train = sample(nrow(CreditCard) ,train)
CreditCard.bic.train = CreditCard.bic[train, ]
CreditCard.bic.test = CreditCard.bic[-train, ]

pred.tables.knn = table(NULL)
Accuracy.knn.table = table(NULL)
for(K in 6:25){

  pred.knn = knn(CreditCard.bic.train, CreditCard.bic.test,
                 CreditCard.train$default.payment.next.month, k = K)
  pred.table.knn = table(pred.knn, CreditCard.test$default.payment.next.month)
  Accuracy = round(sum( pred.table.knn[1,1],
                        pred.table.knn[2,2] ) / sum(pred.table.knn[,]),4)

  # rbind Accuracy table and Confusion table for K=1,2,3
  Accuracy.knn.table = rbind(Accuracy.knn.table, Accuracy)
  pred.tables.knn = rbind(pred.tables.knn, pred.table.knn)

}

# Convert pred.tables.knn from **matrix** into **data.table**
predicts = rownames(pred.tables.knn)
pred.tables.knn = data.table(pred.tables.knn)

# Create two columns for the predictions and K, respectively
pred.tables.knn$Predicts = predicts
pred.tables.knn$K = rep(6:25, each=2)

# Swith the order of columns
pred.tables.knn = pred.tables.knn[,c(4,3, 1:2)]

# Rename the rows and columns of the Accuracy table
rownames(Accuracy.knn.table) = c("K=6", "K=7", "K=8", "K=9", "K=10", "K=11", "K=12",
                                 "K=13", "K=14", "K=15", "K=16", "K=17", "K=18",
                                 "K=19", "K=20", "K=21", "K=22", "K=23", "K=24", "K=25")
colnames(Accuracy.knn.table) = "Accurancy"

# Plot the accuracy as K = [6,25]
plot(x=seq(6,25), y=Accuracy.knn.table, xlab = "Number of K",
     ylab = "Accuracy (%)", main = "Accuracy using KNN")
```




```r
paste("The highest accuracy using KNN is", max(Accuracy.knn.table), " with K =",
      which.max(Accuracy.knn.table))
```

```
## [1] "The highest accuracy using KNN is 0.7742  with K = 18"
```

```r
#####################################################################################
# Generalized Additive Modelz
fit.gam = gam(formulaQ3, data = CreditCard.train)

# Predict the out-of-state tuition using test set
pred.prob.gam = predict(fit.gam, CreditCard.test)
pred.gam = rep(0, length(pred.prob.gam))
pred.gam[pred.gam > 0.5] = 1
pred.table.gam = table(pred.gam, CreditCard.test$default.payment.next.month)
pred.table.gam
```

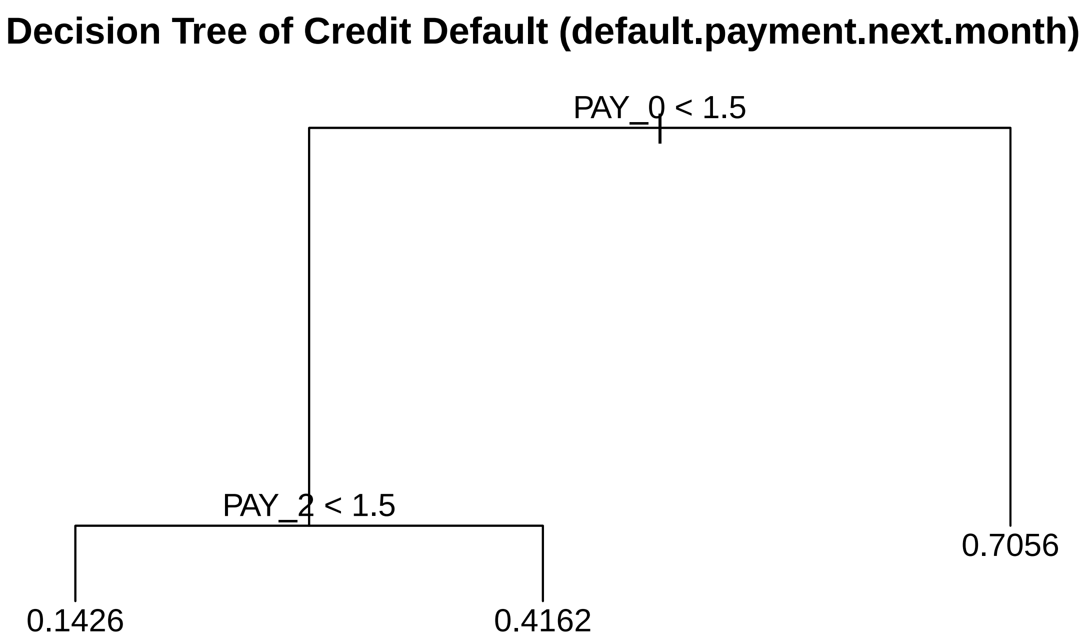
```
##
## pred.gam    0    1
##        0 6814 1971
```

```r
pred.table.gam.Accuracy = round(pred.table.gam[1] / length(pred.prob.gam),4)
paste("The accuracy of generalized additive model is", pred.table.gam.Accuracy)
```

```
## [1] "The accuracy of generalized additive model is 0.7756"
```

```r
#####################################################################################
# Tree
# Fit a regression tree to the training set
```

```r
fit.tree = tree(formulaQ3, data = CreditCard.train)

# Plot the tree
plot(fit.tree)
text(fit.tree, pretty = 0)
title("Decision Tree of Credit Default (default.payment.next.month)")
```




```r
# Predict default.payment.next.month in tesing data set
pred.prob.tree = predict(fit.tree, CreditCard.test)

# Use default.payment.next.month to test the model accuracy
pred.tree = rep(0, length(pred.prob.tree))
pred.tree[pred.prob.tree > 0.5] = 1
pred.table.tree = table(pred.tree, CreditCard.test$default.payment.next.month)
pred.table.tree
```

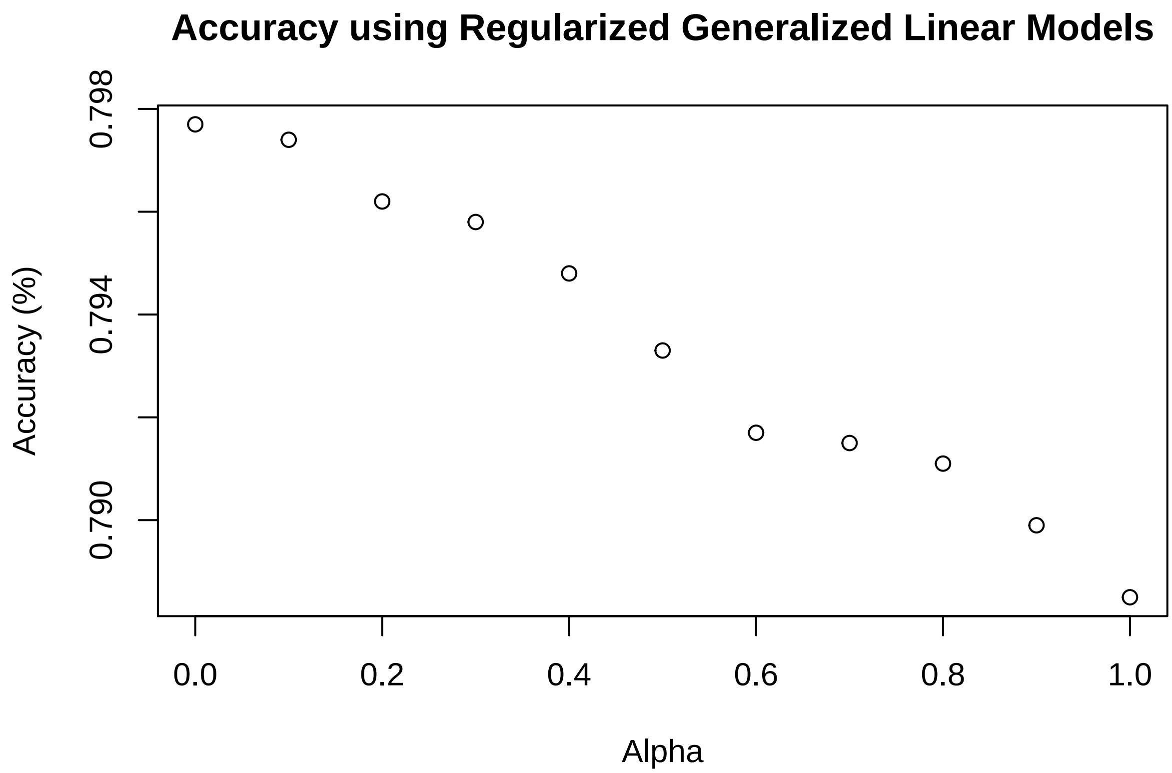
```
##
## pred.tree    0    1
##         0 6528 1333
##         1  286  638
```

```r
# Sensitivity: TP/P = TPR
Sensitivity = pred.table.tree[1,1] / sum(pred.table.tree)
# Specificity: TN/N = TNR
Specificity = pred.table.tree[2,2] / sum(pred.table.tree[,2])
# Accuracy: (TP + TN)/(P + N)
Accuracy = sum(pred.table.tree[1,1], pred.table.tree[2,2]) / sum(pred.table.tree)
```

```r
# Total Error Rate: (FP + FN)/(P + N)
TotalError = sum(pred.table.tree[1,2], pred.table.tree[2,1]) / sum(pred.table.tree[,])

tree.Confusion = round(data.frame(Sensitivity, Specificity, Accuracy, TotalError),4)
row.names(tree.Confusion) = "Tree"

paste("The accuracy of Tree is", Accuracy)
```

```
## [1] "The accuracy of Tree is 0.81570859419465"
```

```r
######################################################################################
# Random Forest:
# In random forests data is resampled from the the train set for as many trees as in the forest
# (default is 500 in R).
# Since the respons are only have two unique values,
# it is not enough for the random forest to create unique trees.
# Thus, I won't use Random Forest to do prediction.

######################################################################################
# Lasso and Elastic-Net Regularized Generalized Linear Models
# Fit a regression tree to the training set
x.train = model.matrix(formulaQ3, data = CreditCard.train)
x.test = model.matrix(formulaQ3, data = CreditCard.test)

tryalpha = seq(0,1,0.1)
acc_glmnet = rep(NA, length(tryalpha))
for (i in 1:length(tryalpha)){

  fit.glmnet = glmnet(x.train, CreditCard.train$default.payment.next.month, alpha = tryalpha[i])
  pred.prob.glmnet = predict(fit.glmnet, s = 0.01, newx = x.test)

  # Use default.payment.next.month to test the model accuracy
  pred.glmnet = rep(0, length(pred.prob.glmnet))
  pred.glmnet[pred.prob.glmnet > 0.5] = 1
  pred.table.glmnet = table(pred.glmnet, CreditCard.test$default.payment.next.month)
  pred.table.glmnet

  # Sensitivity: TP/P = TPR
  Sensitivity = pred.table.glmnet[1,1] / sum(pred.table.glmnet)
  # Specificity: TN/N = TNR
  Specificity = pred.table.glmnet[2,2] / sum(pred.table.glmnet[,2])
  # Accuracy: (TP + TN)/(P + N)
  Accuracy = sum(pred.table.glmnet[1,1], pred.table.glmnet[2,2]) / sum(pred.table.glmnet[,])
  # Total Error Rate: (FP + FN)/(P + N)
  TotalError = sum(pred.table.glmnet[1,2], pred.table.glmnet[2,1]) / sum(pred.table.glmnet[,])

  glmnet.Confusion = data.frame(Sensitivity, Specificity, Accuracy, TotalError)
  acc_glmnet[i] = round(glmnet.Confusion$Accuracy,4)
}
plot(tryalpha, acc_glmnet, xlab = "Alpha", ylab = "Accuracy (%)",
     main = "Accuracy using Regularized Generalized Linear Models")
```




```r
paste("The highest accuracy using glmnet is",
      max(acc_glmnet), "with alpha =",
      tryalpha[which.max(acc_glmnet)], "as alpha is in [0, 1]")
```

```
## [1] "The highest accuracy using glmnet is 0.7977 with alpha = 0 as alpha is in [0, 1]"
```

```r
P3_accuracy = data.frame("Accuracy"= c( glm.Confusion$Accuracy, lda.Confusion$Accuracy,
                              qda.Confusion$Accuracy, max(Accuracy.knn.table),
                              pred.table.gam.Accuracy, tree.Confusion$Accuracy,
                              max(acc_glmnet)))
rownames(P3_accuracy) = c("glm", "lda", "qda", "KNN", "gam", "tree", "glmnet" )
P3_accuracy
```

```
##        Accuracy
## glm      0.8054
## lda      0.8064
## qda      0.7631
## KNN      0.7742
## gam      0.7756
## tree     0.8157
## glmnet   0.7977
```

```r
# From these seven algorithms, Tree has the highest accuracy.
```

In part 3, I did machine leanring on this credit card dataset using seven algorithms, including generalized linear model (glm), linear and quadratic discriminant analysis, k-nearest neighbors, generalized additive model, classification tree, and Lasso and elastic-net regularized generalized linear models. From the accuracy table, the possibility of credit card default next month against other factors is near linear relation based on high accuracy of generalized linear model. Additionally, it has clear feature on repayment status of the previous two months. Thus, the accuracy of classification tree is the highest, and lda is the second highest.

Conclusion

1. More credit card defualt for limit balance about 10000. It might mean that credit card might be too easy to be issued for people who have low credit scores. The variance of the default rate for limit balance over 500,000 NTD is higher than other range of limit balance.
2. It is lower default rate for cardholders have higher education level. Moreover, the default rate for clients whose age over 60 was higher than mid age and young people.
3. The best fit algorithm for predicting limit balance is bagging approach.
4. The best fit algorithm for predicting whether a client default next month is classification tree.

Reference

1. https://bradzzz.gitbooks.io/ga-dsi-seattle/content/dsi/dsi_05_classification_databases/2.1-lesson/assets/datasets/DefaultCreditCardClients_yeh_2009.pdf
2. https://gerardnico.com/data_mining/stepwise_regression
3. http://www-math.mit.edu/~rmd/650/bic.pdf