Default of Credit Card Clients

Group Leader: Taha Bin Amer

Group Member: Pranav Vinod

University of Massachusetts Dartmouth

CIS 490 - Machine Learning

Supervisor: Professor (Julia) Hua Fang

03/16/2022

Contents

Abstract

The aim of this project is to analyze the credit card client's dataset and gain meaningful insights from it. We shall employ various data analysis techniques to explore the dataset for any ambiguities. Moreover, we will be practicing logistic regression as well as classification trees to predict whether a client would default on his/her credit card payment or not. Lastly, but certainly not least we will be analyzing the outcomes from both these techniques and then compare and contrast the output generated by both models.

Description of the Dataset

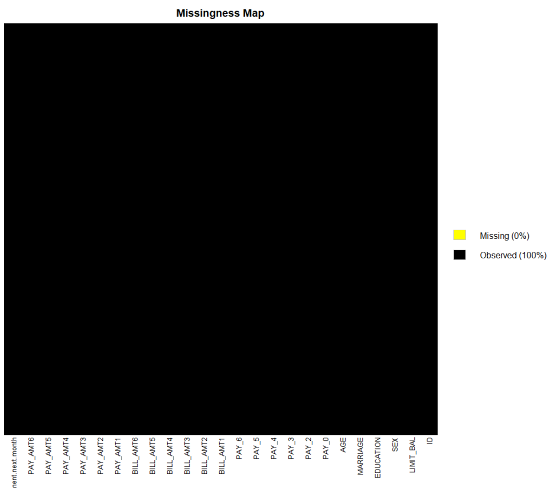
This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. There are 25 attributes in total with **default.payment.next.month** being our target variable (Y). Please refer to the table below for a detailed explanation of all the attributes.

Attribute Name	Details
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	(1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
MARRIAGE	Marital status (1=married, 2=single, 3=others)
AGE	Age in years
PAY_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)
PAY_2	Repayment status in August, 2005 (scale same as above)
PAY_3	Repayment status in July, 2005 (scale same as above)
PAY_4	Repayment status in June, 2005 (scale same as above)
PAY_5	Repayment status in May, 2005 (scale same as above)
PAY_6	Repayment status in April, 2005 (scale same as above)

BILL_AMT1	Amount of bill statement in September, 2005 (NT dollar)
BILL_AMT2	Amount of bill statement in August 2005 (NT dollar)
BILL_AMT3	Amount of bill statement in July 2005 (NT dollar)
BILL_AMT4	Amount of bill statement in June 2005 (NT dollar)
BILL_AMT5	Amount of bill statement in May 2005 (NT dollar)
BILL_AMT6	Amount of bill statement in April 2005 (NT dollar)
PAY_AMT1	Amount of previous payment in September 2005 (NT dollar)
PAY_AMT2	Amount of previous payment in August 2005 (NT dollar)
PAY_AMT3	Amount of previous payment in July, 2005 (NT dollar)
PAY_AMT4	Amount of previous payment in June, 2005 (NT dollar)
PAY_AMT5	Amount of previous payment in May, 2005 (NT dollar)
PAY_AMT6	Amount of previous payment in April, 2005 (NT dollar)
default.payment.next.month	Default payment (1=yes, 0=no)

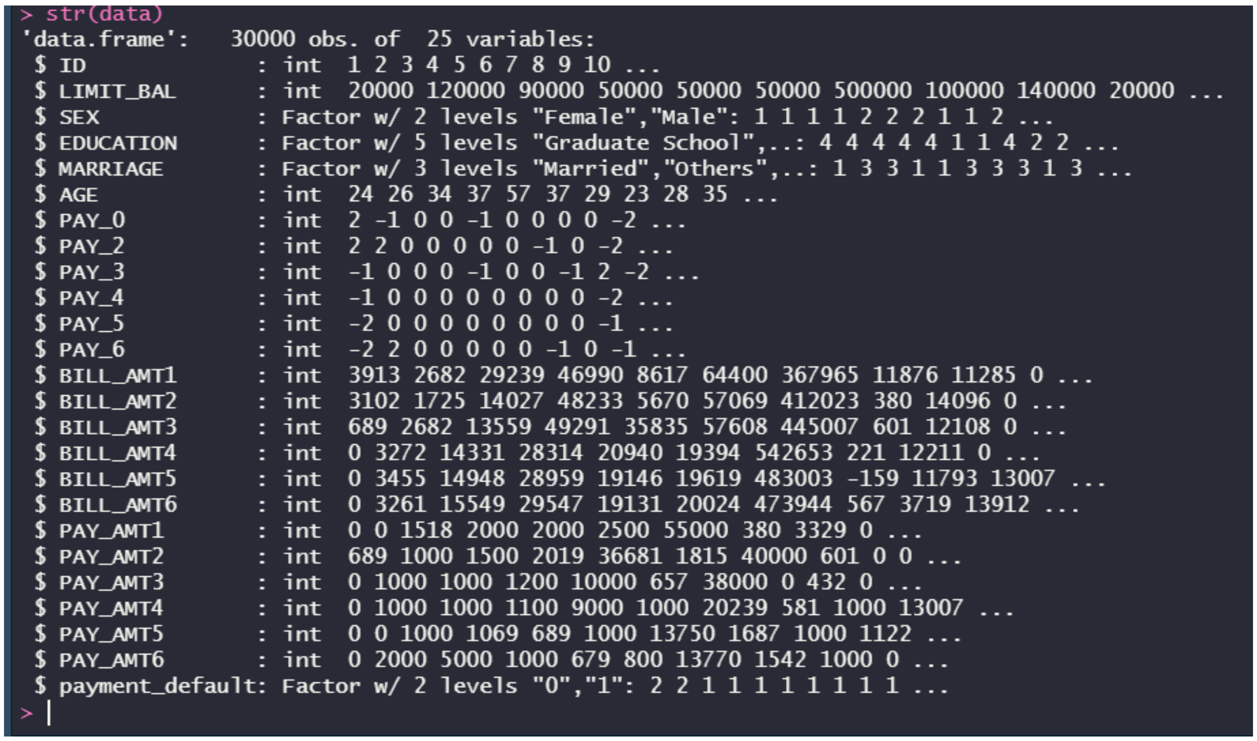
Data Exploration

```
# Missing values in data
missmap(data, col = c('yellow', 'black'), y.at = 1, y.labels = '', legend = TRUE)
```



The above plot clearly shows that the data is free from NA's.

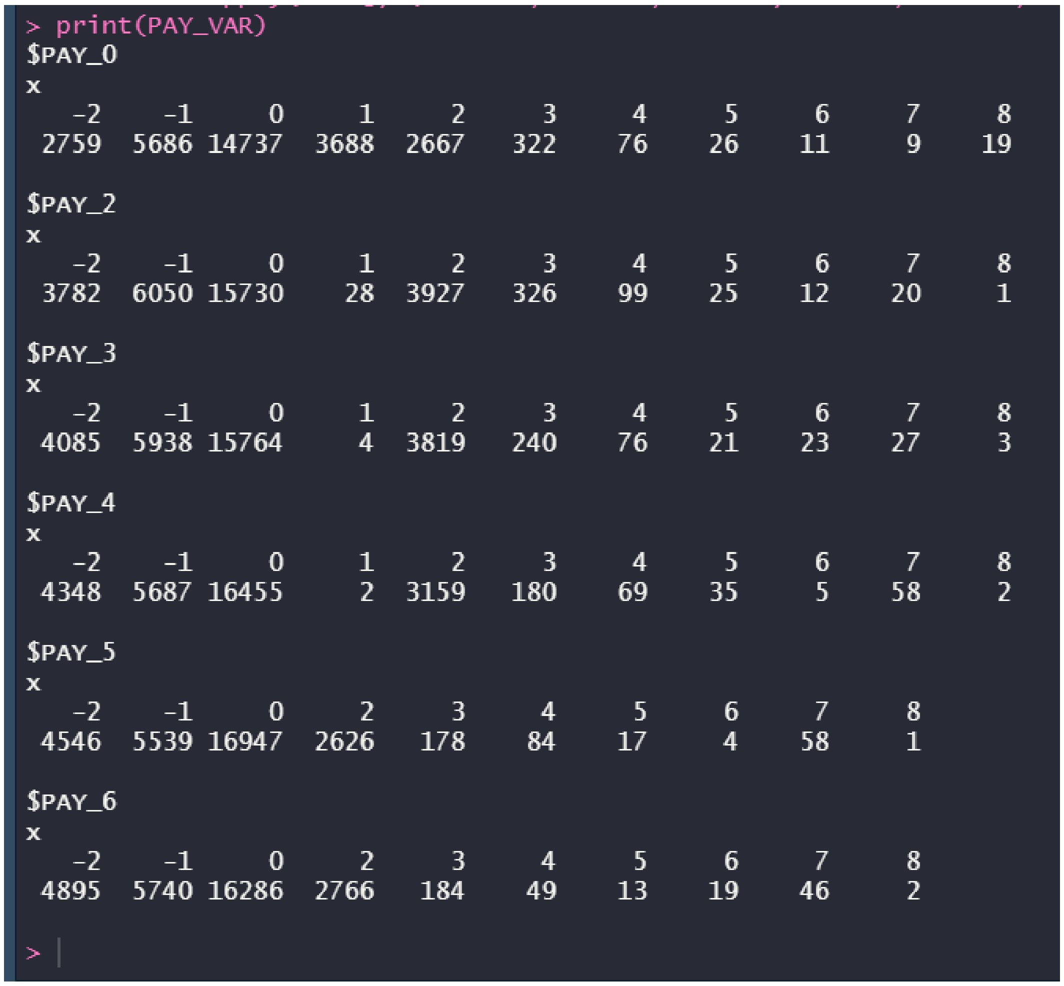
```
# Change target variable name to payment_default and set as factor variable
colnames(data)[25] <- "payment_default"
data$payment_default<-as.factor(data$payment_default)

# Change SEX, EDUCATION, and MARRIAGE to factor
data$SEX<-ifelse(data$SEX==1,"Male","Female")

data$EDUCATION<-ifelse(data$EDUCATION==1,"Graduate School",
                    ifelse(data$EDUCATION==2,"University",
                        (ifelse(data$EDUCATION==3,"High School",
                            ifelse(data$EDUCATION==4,"Others","Unknown")))))

data$MARRIAGE<-ifelse(data$MARRIAGE==1,"Married",
                      ifelse(data$MARRIAGE==2,"Single","Others"))

names<-c("SEX","EDUCATION","MARRIAGE")
data[names]<-lapply(data[names],as.factor)
```



```
> str(data)
'data.frame':   30000 obs. of  25 variables:
 $ ID             : int  1 2 3 4 5 6 7 8 9 10 ...
 $ LIMIT_BAL      : int  20000 120000 90000 50000 50000 50000 500000 100000 140000 20000 ...
 $ SEX            : Factor w/ 2 levels "Female","Male": 1 1 1 1 2 2 2 1 1 2 ...
 $ EDUCATION      : Factor w/ 5 levels "Graduate School",..: 4 4 4 4 4 1 1 4 2 2 ...
 $ MARRIAGE       : Factor w/ 3 levels "Married","Others",..: 1 3 3 1 1 3 3 3 1 3 ...
 $ 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      : int  3913 2682 29239 46990 8617 64400 367965 11876 11285 0 ...
 $ BILL_AMT2      : int  3102 1725 14027 48233 5670 57069 412023 380 14096 0 ...
 $ BILL_AMT3      : int  689 2682 13559 49291 35835 57608 445007 601 12108 0 ...
 $ BILL_AMT4      : int  0 3272 14331 28314 20940 19394 542653 221 12211 0 ...
 $ BILL_AMT5      : int  0 3455 14948 28959 19146 19619 483003 -159 11793 13007 ...
 $ BILL_AMT6      : int  0 3261 15549 29547 19131 20024 473944 567 3719 13912 ...
 $ PAY_AMT1       : int  0 0 1518 2000 2000 2500 55000 380 3329 0 ...
 $ PAY_AMT2       : int  689 1000 1500 2019 36681 1815 40000 601 0 0 ...
 $ PAY_AMT3       : int  0 1000 1000 1200 10000 657 38000 0 432 0 ...
 $ PAY_AMT4       : int  0 1000 1000 1100 9000 1000 20239 581 1000 13007 ...
 $ PAY_AMT5       : int  0 0 1000 1069 689 1000 13750 1687 1000 1122 ...
 $ PAY_AMT6       : int  0 2000 5000 1000 679 800 13770 1542 1000 0 ...
 $ payment_default: Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
> |
```

There are 6 monthly variables for the repayment status which needs to be coded properly as per the data dictionary for better analysis. Further they need to be converted to categorical variables for the modeling. Please refer to the attribute table above for number coding of these attributes. But before proceeding let us first check the number of counts under each category.

```
# Check number of counts under each repayment category
PAY_VAR<-lapply(data[,c("PAY_0","PAY_2","PAY_3","PAY_4","PAY_5","PAY_6")],
function(x) table(x))
print(PAY_VAR)
```



```
> print(PAY_VAR)
$PAY_0
x
   -2    -1     0     1     2     3     4     5     6     7     8
 2759  5686 14737  3688  2667   322    76    26    11     9    19

$PAY_2
x
   -2    -1     0     1     2     3     4     5     6     7     8
 3782  6050 15730    28  3927   326    99    25    12    20     1

$PAY_3
x
   -2    -1     0     1     2     3     4     5     6     7     8
 4085  5938 15764     4  3819   240    76    21    23    27     3

$PAY_4
x
   -2    -1     0     1     2     3     4     5     6     7     8
 4348  5687 16455     2  3159   180    69    35     5    58     2

$PAY_5
x
   -2    -1     0     2     3     4     5     6     7     8
 4546  5539 16947  2626   178    84    17     4    58     1

$PAY_6
x
   -2    -1     0     2     3     4     5     6     7     8
 4895  5740 16286  2766   184    49    13    19    46     2

> |
```

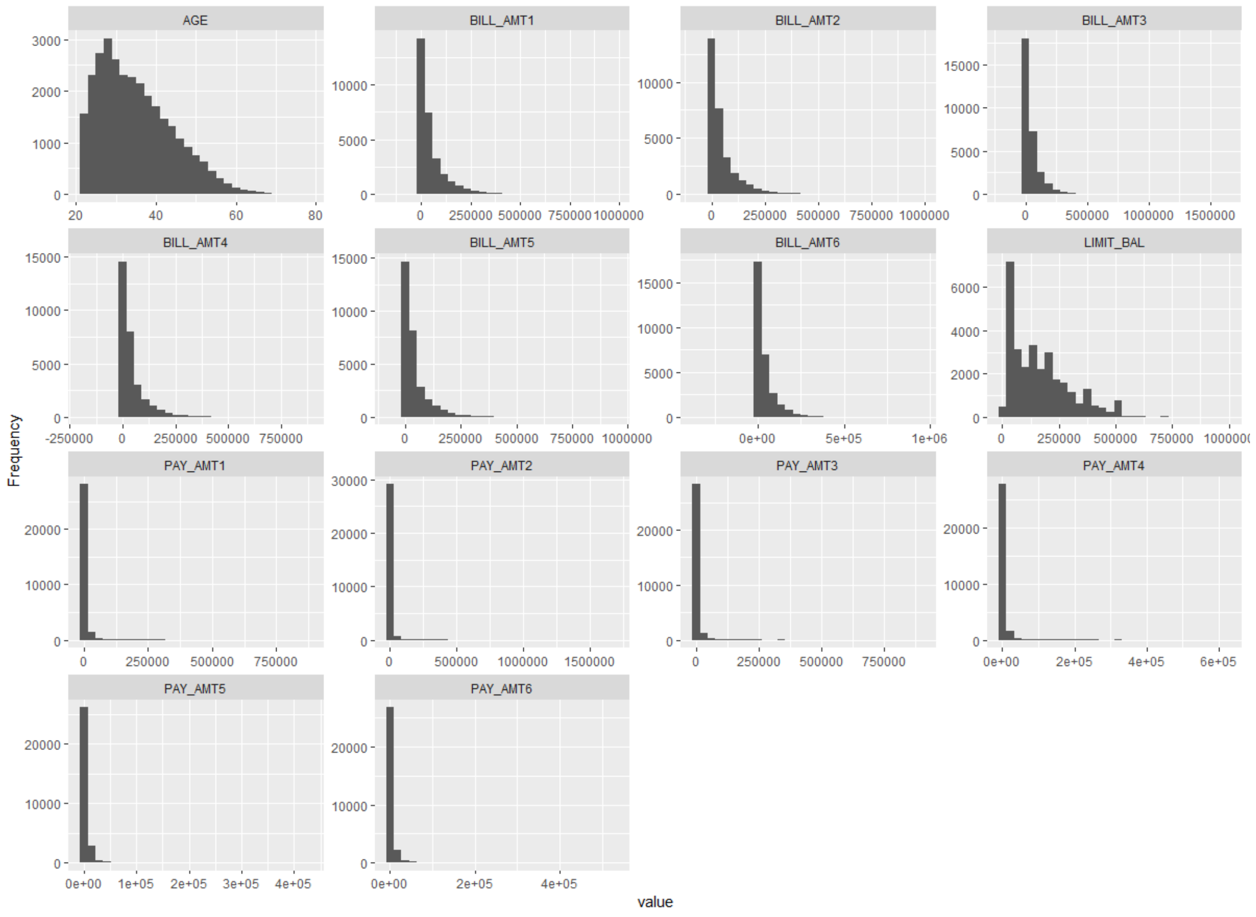
The above results show that the number of customers who have delayed payments by 5 months or more is very minimal. Therefore we can group them under one category.

```
# Change payment attributes to categorical and group 5 months or more delayed
together
data$PAY_0<-ifelse(data$PAY_0 < 1 ,"Paid Duly",
                   ifelse(data$PAY_0==1,"1 month delay",
                        ifelse(data$PAY_0==2,"2 month
delay",ifelse(data$PAY_0==3,"3 month delay",
                            ifelse(data$PAY_0==4,"4 month delay","5 month    or
more delay")))))
```

```
data$PAY_2<-ifelse(data$PAY_2 <1 ,"Paid Duly",
                   ifelse(data$PAY_2==1,"1 month delay",
                        ifelse(data$PAY_2==2,"2 month delay",
                            ifelse(data$PAY_2==3,"3 month delay",
                               ifelse(data$PAY_2==4,"4 month delay","5 month or
more delay")))))

data$PAY_3<-ifelse(data$PAY_3 <1 ,"Paid Duly",
                   ifelse(data$PAY_3==1,"1 month delay",
                      ifelse(data$PAY_3==2,"2 month delay",
                          ifelse(data$PAY_3==3,"3 month delay",
                              ifelse(data$PAY_3==4,"4 month delay","5 month or more
delay")))))

data$PAY_4<-ifelse(data$PAY_4 <1 ,"Paid Duly",
                   ifelse(data$PAY_4==1,"1 month delay",
                      ifelse(data$PAY_4==2,"2 month delay",
                          ifelse(data$PAY_4==3,"3 month delay",
                              ifelse(data$PAY_4==4,"4 month delay","5 month or more
delay")))))

data$PAY_5<-ifelse(data$PAY_5 <1 ,"Paid Duly",
                   ifelse(data$PAY_5==1,"1 month delay",
                       ifelse(data$PAY_5==2,"2 month delay",
                              ifelse(data$PAY_5==3,"3 month delay",
                                  ifelse(data$PAY_5==4,"4 month delay","5 month or
more delay")))))

data$PAY_6<-ifelse(data$PAY_6 <1 ,"Paid Duly",
                 ifelse(data$PAY_6==1,"1 month delay",
                    ifelse(data$PAY_6==2,"2 month delay",
                        ifelse(data$PAY_6==3,"3 month delay",
                            ifelse(data$PAY_6==4,"4 month delay","5 month or more
delay")))))

names<-c("PAY_0","PAY_2","PAY_3","PAY_4","PAY_5","PAY_6")
data[names]<-lapply(data[names],as.factor)
```

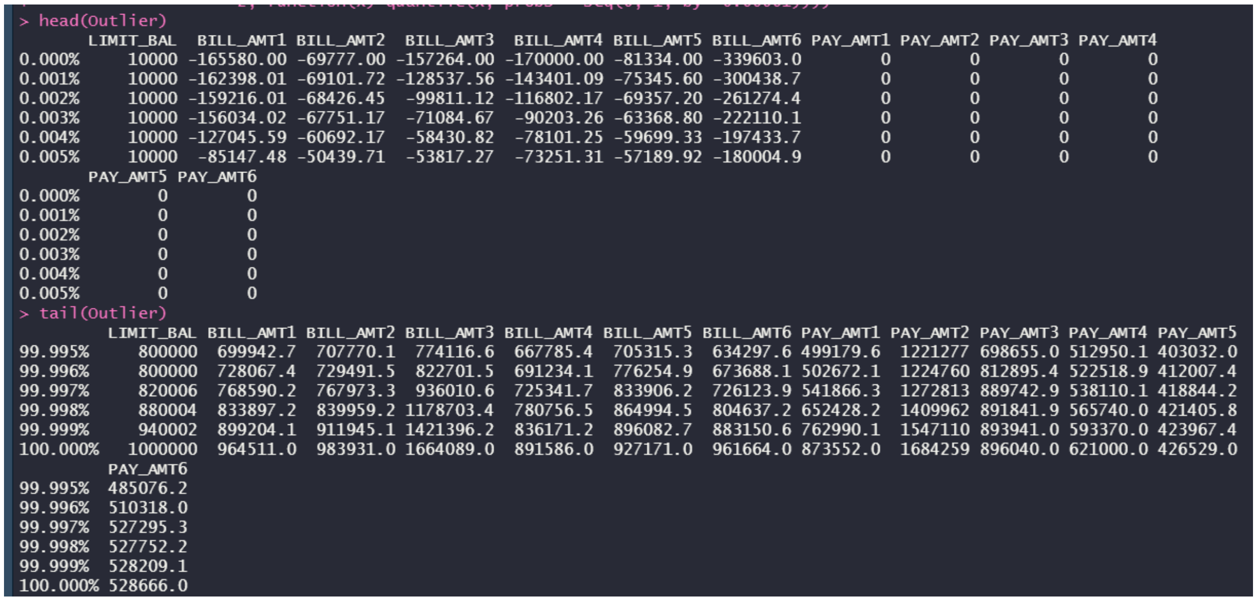
Let's analyze the continuous variable now to check for any potential outliers.

```
# Analysis of continuous variables
plot_histogram(data[,-1])
```



All of the variables look skewed and have some outliers which will require treatment. So, lets identify and remove potential outliers.

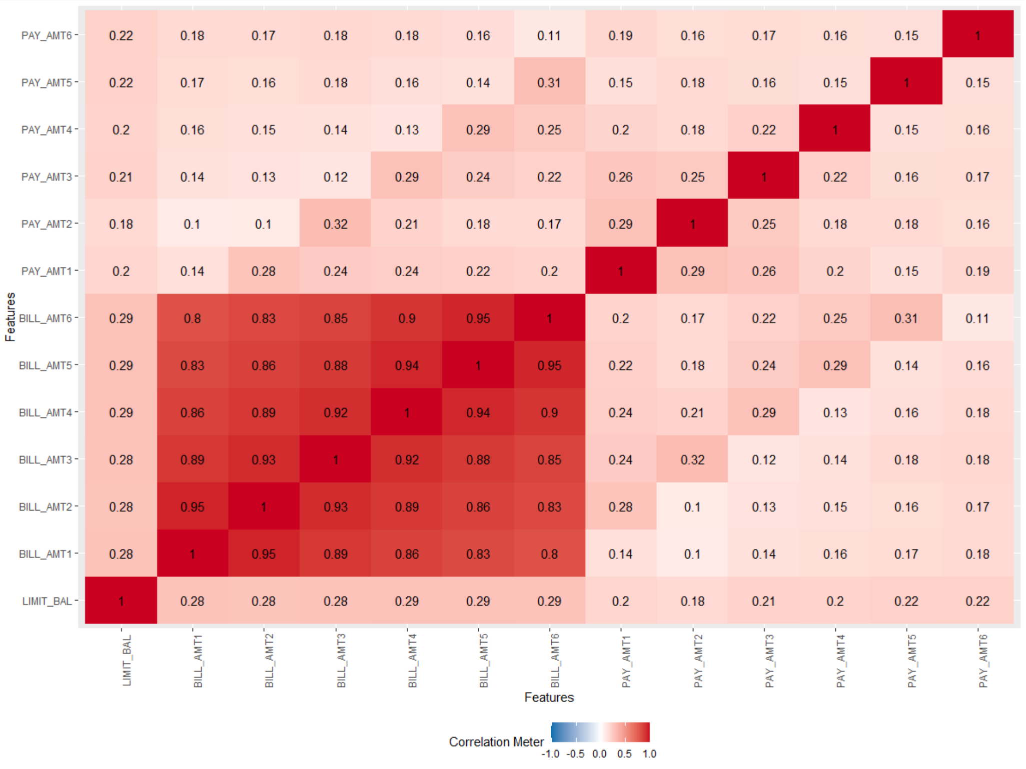
```
# Check and remove outliers
Outlier<-data.frame(apply(data[,c("LIMIT_BAL","BILL_AMT1","BILL_AMT2",
"BILL _AMT3","BILL_AMT4","BILL_AMT5", "BILL_AMT6","PAY_AMT1","PAY_AMT2",
"PAY_AMT3","PAY_AMT4","PAY_AMT5","PAY_AMT6")],
2, function(x) quantile(x, probs = seq(0, 1, by= 0.00001))))
head(Outlier)
tail(Outlier)

data<-subset(data, !(data$LIMIT_BAL> quantile(data$LIMIT_BAL, 0.99999) |
                   data$BILL_AMT1< quantile(data$BILL_AMT1, 0.00001)))
```



```
> head(Outlier)
        LIMIT_BAL  BILL_AMT1 BILL_AMT2  BILL_AMT3  BILL_AMT4 BILL_AMT5 BILL_AMT6 PAY_AMT1 PAY_AMT2 PAY_AMT3 PAY_AMT4
0.000%      10000 -165580.00 -69777.00 -157264.00 -170000.00 -81334.00 -339603.0        0        0        0        0
0.001%      10000 -162398.01 -69101.72 -128537.56 -143401.09 -75345.60 -300438.7        0        0        0        0
0.002%      10000 -159216.01 -68426.45  -99811.12 -116802.17 -69357.20 -261274.4        0        0        0        0
0.003%      10000 -156034.02 -67751.17  -71084.67  -90203.26 -63368.80 -222110.1        0        0        0        0
0.004%      10000 -127045.59 -60692.17  -58430.82  -78101.25 -59699.33 -197433.7        0        0        0        0
0.005%      10000  -85147.48 -50439.71  -53817.27  -73251.31 -57189.92 -180004.9        0        0        0        0
        PAY_AMT5 PAY_AMT6
0.000%         0        0
0.001%         0        0
0.002%         0        0
0.003%         0        0
0.004%         0        0
0.005%         0        0
> tail(Outlier)
          LIMIT_BAL BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6 PAY_AMT1 PAY_AMT2 PAY_AMT3 PAY_AMT4 PAY_AMT5
99.995%      800000  699942.7  707770.1  774116.6  667785.4  705315.3  634297.6 499179.6  1221277 698655.0 512950.1 403032.0
99.996%      800000  728067.4  729491.5  822701.5  691234.1  776254.9  673688.1 502672.1  1224760 812895.4 522518.9 412007.4
99.997%      820006  768590.2  767973.3  936010.6  725341.7  833906.2  726123.9 541866.3  1272813 889742.9 538110.1 418844.2
99.998%      880004  833897.2  839959.2 1178703.4  780756.5  864994.5  804637.2 652428.2  1409962 891841.9 565740.0 421405.8
99.999%      940002  899204.1  911945.1 1421396.2  836171.2  896082.7  883150.6 762990.1  1547110 893941.0 593370.0 423967.4
100.000%    1000000  964511.0  983931.0 1664089.0  891586.0  927171.0  961664.0 873552.0  1684259 896040.0 621000.0 426529.0
          PAY_AMT6
99.995%   485076.2
99.996%   510318.0
99.997%   527295.3
99.998%   527752.2
99.999%   528209.1
100.000%  528666.0
```

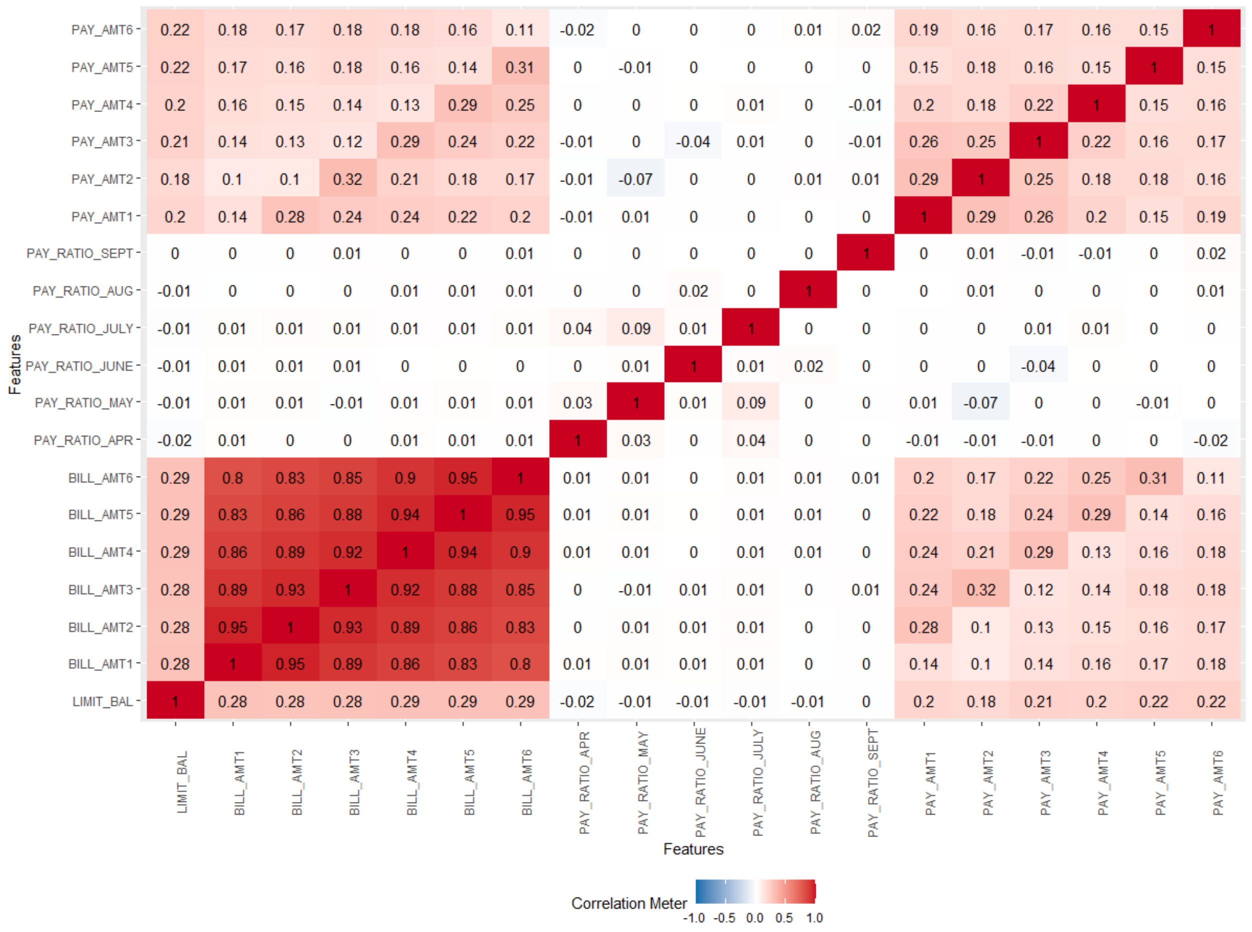
We notice two major outliers one at each end which would be removed.

```
# Collinearity among variables
numeric_fields<-c("LIMIT_BAL","BILL_AMT1","BILL_AMT2","BILL_AMT3",
"BILL_AMT4","BILL_AMT5","BILL_AMT6",
"PAY_AMT1","PAY_AMT2","PAY_AMT3","PAY_AMT4","PAY_AMT5","PAY_AMT6")
df_numeric<-subset(data, select=numeric_fields)
plot_correlation(df_numeric)
```



There is high collinearity among the 6 variables corresponding to BILL AMOUNT and therefore will not be suitable for modeling. We would be excluding them from our models as well. Instead, we'll generate monthly ratio variables for the amount of payment made for the bill amounts corresponding to each month.

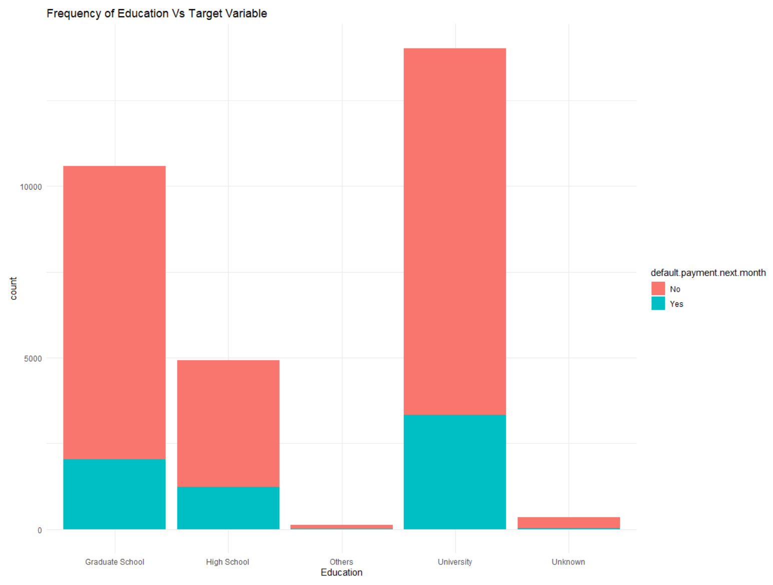
```
data$PAY_RATIO_APR<-ifelse(is.nan(data$PAY_AMT1/data$BILL_AMT1),0,
ifelse(is.infinite(data$PAY_AMT1/data$BILL_AMT1),0,
round(data$PAY_AMT1/data$BILL_AMT1,2)))

data$PAY_RATIO_MAY<-ifelse(is.nan(data$PAY_AMT2/data$BILL_AMT2),0,
ifelse(is.infinite(data$PAY_AMT2/data$BILL_AMT2),0,
round(data$PAY_AMT2/data$BILL_AMT2,2)))

data$PAY_RATIO_JUNE<-ifelse(is.nan(data$PAY_AMT3/data$BILL_AMT3),0,
ifelse(is.infinite(data$PAY_AMT3/data$BILL_AMT3),0,
round(data$PAY_AMT3/data$BILL_AMT3,2)))

data$PAY_RATIO_JULY<-ifelse(is.nan(data$PAY_AMT4/data$BILL_AMT4),0,
ifelse(is.infinite(data$PAY_AMT4/data$BILL_AMT4),0,
round(data$PAY_AMT4/data$BILL_AMT4,2)))

data$PAY_RATIO_AUG<-ifelse(is.nan(data$PAY_AMT5/data$BILL_AMT5),0,
ifelse(is.infinite(data$PAY_AMT5/data$BILL_AMT5),0,
round(data$PAY_AMT5/data$BILL_AMT5,2)))

data$PAY_RATIO_SEPT<-ifelse(is.nan(data$PAY_AMT6/data$BILL_AMT6),0,
ifelse(is.infinite(data$PAY_AMT6/data$BILL_AMT6),0,
round(data$PAY_AMT6/data$BILL_AMT6,2)))

numeric_fields<-c("LIMIT_BAL","BILL_AMT1","BILL_AMT2","BILL_AMT3","BILL_AMT4","BILL
_AMT5",
"BILL_AMT6","PAY_RATIO_APR","PAY_RATIO_MAY","PAY_RATIO_JUNE",
"PAY_RATIO_JULY","PAY_RATIO_AUG","PAY_RATIO_SEPT",
"PAY_AMT1","PAY_AMT2","PAY_AMT3","PAY_AMT4","PAY_AMT5","PAY_AMT6")

df_numeric<-subset(data, select=numeric_fields)
plot_correlation(df_numeric)
```

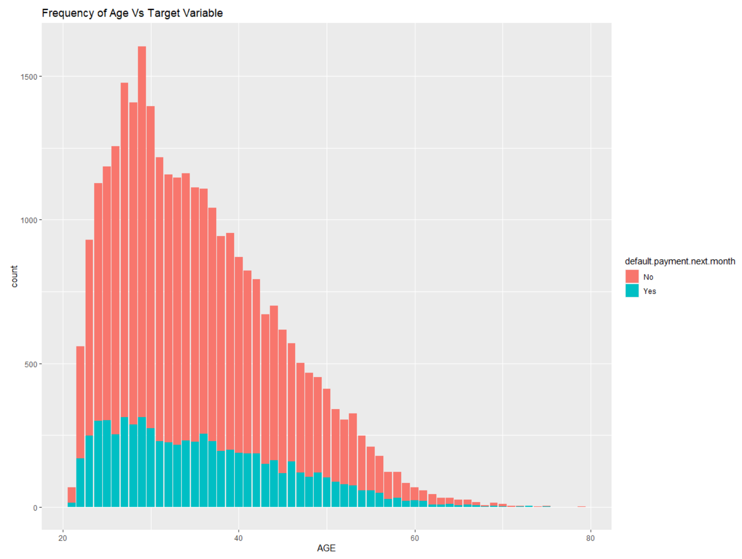


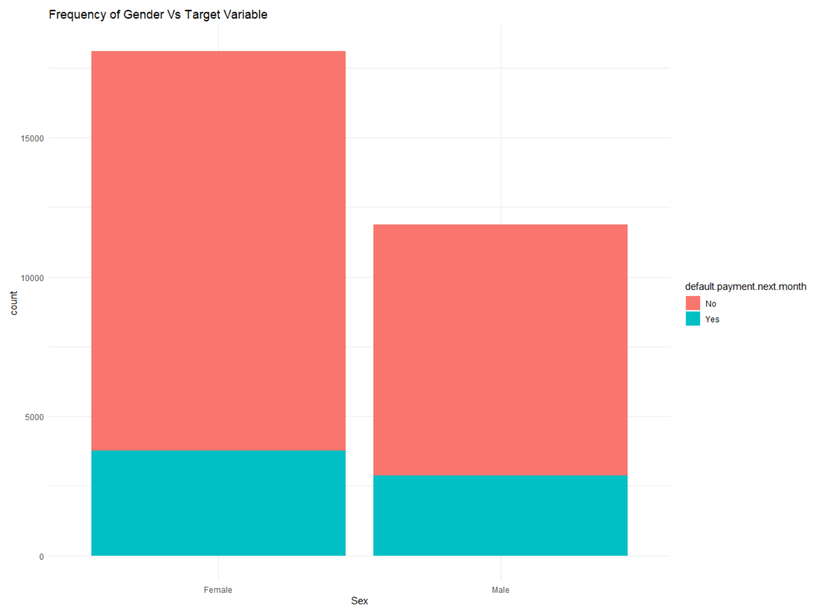
We can now see that there is much less collinearity among the variables and so now are suitable for modeling.

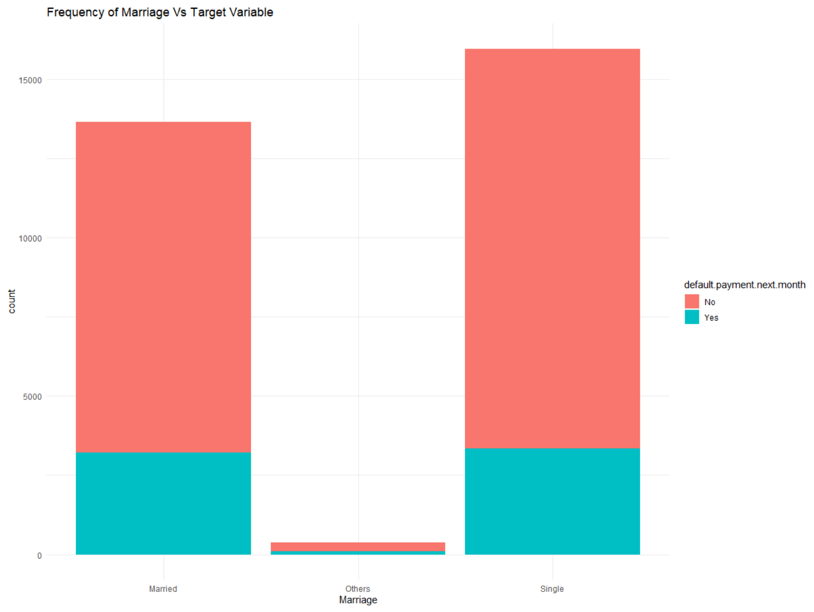
```
# Frequency Plot Sex, Education, Marriage, Age Vs Target Variable
ggplot(data, aes(x = EDUCATION, fill = default.payment.next.month)) +
  geom_bar() +
  ggtitle("Frequency of Education Vs Target Variable") +
  labs(x = 'Education') +
  theme_minimal()

ggplot(data, aes(x = AGE, fill = default.payment.next.month)) +
  geom_bar() +
  ggtitle("Frequency of Age Vs Target Variable")
  labs(x = 'Age') +
  theme_minimal()

ggplot(data, aes(x = SEX, fill = default.payment.next.month)) +
  geom_bar() +
  ggtitle("Frequency of Gender Vs Target Variable") +
labs(x = 'Sex') +
  theme_minimal()
```

```
ggplot(data, aes(x = MARRIAGE, fill = default.payment.next.month)) +
  geom_bar() +
  ggtitle("Frequency of Marriage Vs Target Variable") +
  labs(x = 'Marriage') +
  theme_minimal()
```









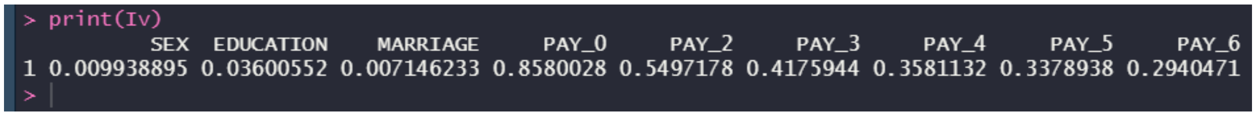
The above plots show that majority of the clients did not default on their credit card payments. We also observe that majority of the clients have an education level of University followed by Graduate School. Our dataset is skewed to the right with age attribute with the majority of the people lying in the 23 to 30 age bracket. There are a greater number of females than males in this dataset. And lastly, we see an almost equal distribution of the relationship status of clients.

Logistic Regression

```
##################### Logistic Regression################################
Random.seed <- c('Mersenne-Twister', 1)
set.seed(490)

indices <-sample(1:nrow(data), 0.8 * nrow(data), replace = FALSE)
train <-data[indices,]
test <-data[-indices,]
```

Splitting the dataset into train and test. 80% train, 20% test with seed value as 490 for reproducibility. Lets now observe the information value of categorical variables to see if there are any that can be omitted.

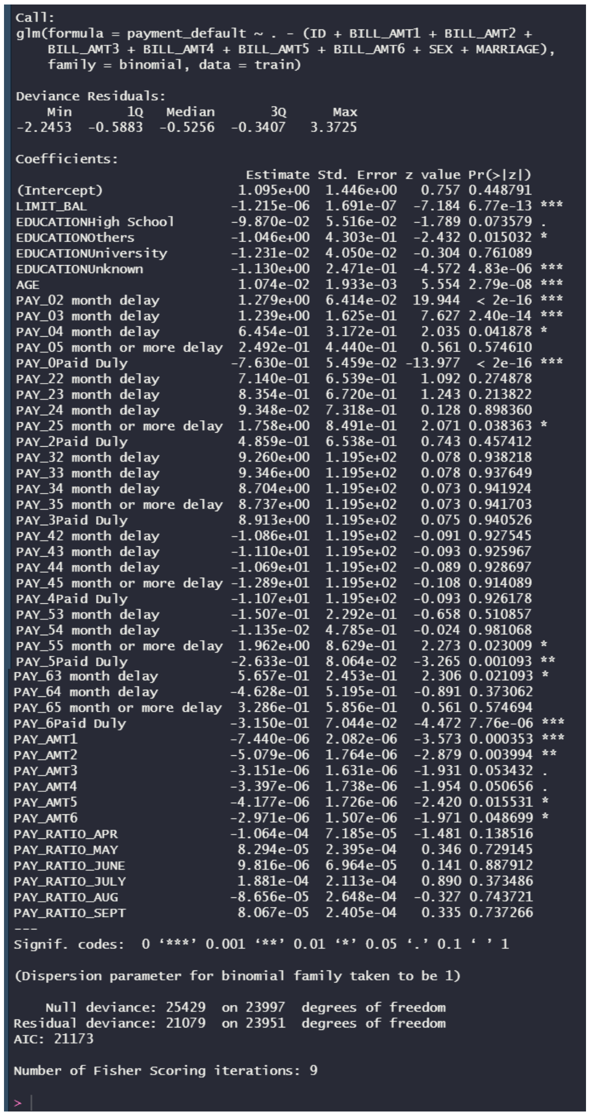
```
# Check information value of categorical variables to understand if they can be
# omitted.

SEX<-data.frame("SEX"=IV(train$SEX,train$payment_default))
EDUCATION<-data.frame("EDUCATION"=IV(train$EDUCATION,train$payment_default))
MARRIAGE<-data.frame("MARRIAGE"=IV(train$MARRIAGE,train$payment_default))
PAY_0<-data.frame("PAY_0"=IV(train$PAY_0,train$payment_default))
PAY_2<-data.frame("PAY_2"=IV(train$PAY_2,train$payment_default))
PAY_3<-data.frame("PAY_3"=IV(train$PAY_3,train$payment_default))
PAY_4<-data.frame("PAY_4"=IV(train$PAY_4,train$payment_default))
PAY_5<-data.frame("PAY_5"=IV(train$PAY_5,train$payment_default))
PAY_6<-data.frame("PAY_6"=IV(train$PAY_6,train$payment_default))

Iv<-cbind(SEX,EDUCATION,MARRIAGE,PAY_0,PAY_2,PAY_3,PAY_4,PAY_5,PAY_6)
print(Iv)
```



```
> print(Iv)
          SEX  EDUCATION    MARRIAGE     PAY_0     PAY_2     PAY_3     PAY_4     PAY_5     PAY_6
1 0.009938895 0.03600552 0.007146233 0.8580028 0.5497178 0.4175944 0.3581132 0.3378938 0.2940471
> 
```

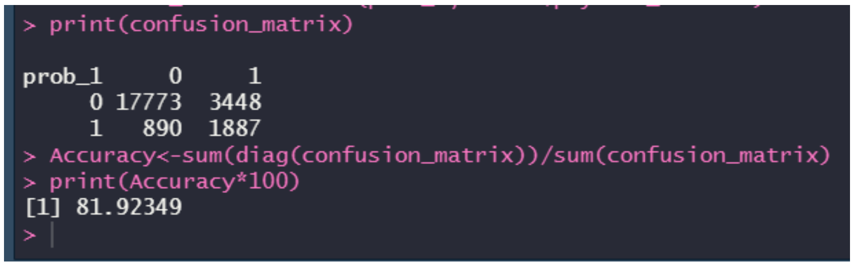
The above results show that SEX ad MARRIAGE are very weak predictors. Therefore we shall omit them for logistic regression.

```
mod_1<-glm(payment_default~.-(ID+BILL_AMT1+BILL_AMT2+BILL_AMT3+
BILL_AMT4+BILL_AMT5+BILL_AMT6+SEX+MARRIAGE), train, family=binomial)
summary(mod_1)
```



```
Call:
glm(formula = payment_default ~ . - (ID + BILL_AMT1 + BILL_AMT2 +
    BILL_AMT3 + BILL_AMT4 + BILL_AMT5 + BILL_AMT6 + SEX + MARRIAGE),
    family = binomial, data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2453  -0.5883  -0.5256  -0.3407   3.3725

Coefficients:
                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                 1.095e+00  1.446e+00   0.757 0.448791
LIMIT_BAL                  -1.215e-06  1.691e-07  -7.184 6.77e-13 ***
EDUCATIONHigh School       -9.870e-02  5.516e-02  -1.789 0.073579 .
EDUCATIONOthers            -1.046e+00  4.303e-01  -2.432 0.015032 *
EDUCATIONUniversity        -1.231e-02  4.050e-02  -0.304 0.761089
EDUCATIONUnknown           -1.130e+00  2.471e-01  -4.572 4.83e-06 ***
AGE                         1.074e-02  1.933e-03   5.554 2.79e-08 ***
PAY_02 month delay          1.279e+00  6.414e-02  19.944  < 2e-16 ***
PAY_03 month delay          1.239e+00  1.625e-01   7.627 2.40e-14 ***
PAY_04 month delay          6.454e-01  3.172e-01   2.035 0.041878 *
PAY_05 month or more delay  2.492e-01  4.440e-01   0.561 0.574610
PAY_0Paid Duly             -7.630e-01  5.459e-02 -13.977  < 2e-16 ***
PAY_22 month delay          7.140e-01  6.539e-01   1.092 0.274878
PAY_23 month delay          8.354e-01  6.720e-01   1.243 0.213822
PAY_24 month delay          9.348e-02  7.318e-01   0.128 0.898360
PAY_25 month or more delay  1.758e+00  8.491e-01   2.071 0.038363 *
PAY_2Paid Duly              4.859e-01  6.538e-01   0.743 0.457412
PAY_32 month delay          9.260e+00  1.195e+02   0.078 0.938218
PAY_33 month delay          9.346e+00  1.195e+02   0.078 0.937649
PAY_34 month delay          8.704e+00  1.195e+02   0.073 0.941924
PAY_35 month or more delay  8.737e+00  1.195e+02   0.073 0.941703
PAY_3Paid Duly              8.913e+00  1.195e+02   0.075 0.940526
PAY_42 month delay         -1.086e+01  1.195e+02  -0.091 0.927545
PAY_43 month delay         -1.110e+01  1.195e+02  -0.093 0.925967
PAY_44 month delay         -1.069e+01  1.195e+02  -0.089 0.928697
PAY_45 month or more delay -1.289e+01  1.195e+02  -0.108 0.914089
PAY_4Paid Duly             -1.107e+01  1.195e+02  -0.093 0.926178
PAY_53 month delay         -1.507e-01  2.292e-01  -0.658 0.510857
PAY_54 month delay         -1.135e-02  4.785e-01  -0.024 0.981068
PAY_55 month or more delay  1.962e+00  8.629e-01   2.273 0.023009 *
PAY_5Paid Duly             -2.633e-01  8.064e-02  -3.265 0.001093 **
PAY_63 month delay          5.657e-01  2.453e-01   2.306 0.021093 *
PAY_64 month delay         -4.628e-01  5.195e-01  -0.891 0.373062
PAY_65 month or more delay  3.286e-01  5.856e-01   0.561 0.574694
PAY_6Paid Duly             -3.150e-01  7.044e-02  -4.472 7.76e-06 ***
PAY_AMT1                   -7.440e-06  2.082e-06  -3.573 0.000353 ***
PAY_AMT2                   -5.079e-06  1.764e-06  -2.879 0.003994 **
PAY_AMT3                   -3.151e-06  1.631e-06  -1.931 0.053432 .
PAY_AMT4                   -3.397e-06  1.738e-06  -1.954 0.050656 .
PAY_AMT5                   -4.177e-06  1.726e-06  -2.420 0.015531 *
PAY_AMT6                   -2.971e-06  1.507e-06  -1.971 0.048699 *
PAY_RATIO_APR              -1.064e-04  7.185e-05  -1.481 0.138516
PAY_RATIO_MAY               8.294e-05  2.395e-04   0.346 0.729145
PAY_RATIO_JUNE              9.816e-06  6.964e-05   0.141 0.887912
PAY_RATIO_JULY              1.881e-04  2.113e-04   0.890 0.373486
PAY_RATIO_AUG              -8.656e-05  2.648e-04  -0.327 0.743721
PAY_RATIO_SEPT              8.067e-05  2.405e-04   0.335 0.737266
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 25429  on 23997  degrees of freedom
Residual deviance: 21079  on 23951  degrees of freedom
AIC: 21173

Number of Fisher Scoring iterations: 9

>
```

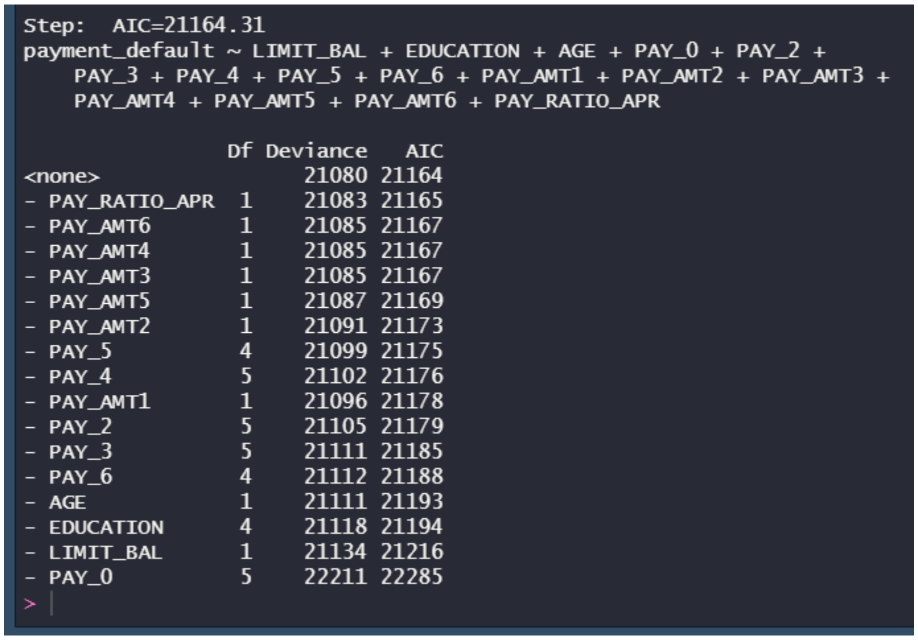
```r
# Check accuracy and confusion matrix of our model
predict_1<-predict(mod_1, train, type='response')
prob_1<-ifelse( predict_1 > 0.5,1,0)

confusion_matrix<-table(prob_1, train$payment_default)
print(confusion_matrix)

Accuracy<-sum(diag(confusion_matrix))/sum(confusion_matrix)
print(Accuracy*100)
```



```
> print(confusion_matrix)

prob_1     0     1
     0 17773  3448
     1   890  1887
> Accuracy<-sum(diag(confusion_matrix))/sum(confusion_matrix)
> print(Accuracy*100)
[1] 81.92349
>
```

Our model accuracy turned out to be 81.92% which is fairly good. Let's use the Step AIC methodology to find the best subset of attributes.

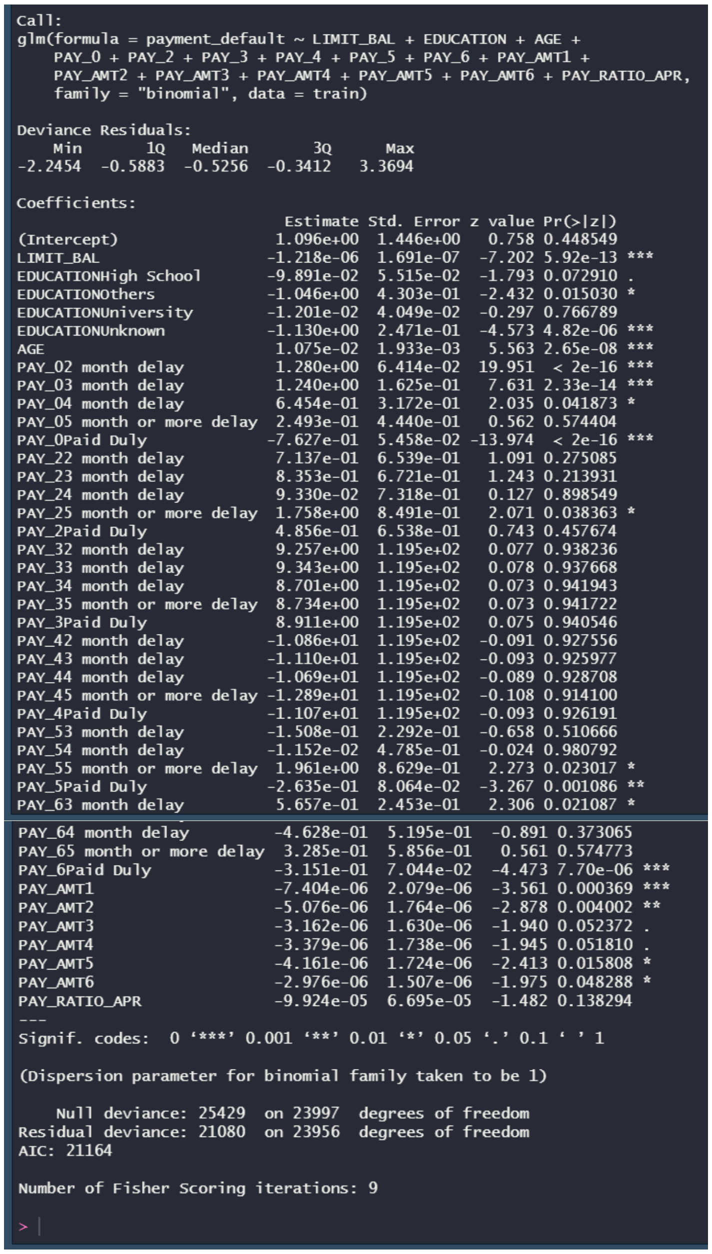
```
# Employ StepAIC to find best subset of values
step_AIC<-stepAIC(mod_1,direction='backward')
```



```
Step:  AIC=21164.31
payment_default ~ LIMIT_BAL + EDUCATION + AGE + PAY_0 + PAY_2 +
    PAY_3 + PAY_4 + PAY_5 + PAY_6 + PAY_AMT1 + PAY_AMT2 + PAY_AMT3 +
    PAY_AMT4 + PAY_AMT5 + PAY_AMT6 + PAY_RATIO_APR

                Df Deviance   AIC
<none>                21080 21164
- PAY_RATIO_APR  1    21083 21165
- PAY_AMT6       1    21085 21167
- PAY_AMT4       1    21085 21167
- PAY_AMT3       1    21085 21167
- PAY_AMT5       1    21087 21169
- PAY_AMT2       1    21091 21173
- PAY_5          4    21099 21175
- PAY_4          5    21102 21176
- PAY_AMT1       1    21096 21178
- PAY_2          5    21105 21179
- PAY_3          5    21111 21185
- PAY_6          4    21112 21188
- AGE            1    21111 21193
- EDUCATION      4    21118 21194
- LIMIT_BAL      1    21134 21216
- PAY_0          5    22211 22285
>
```

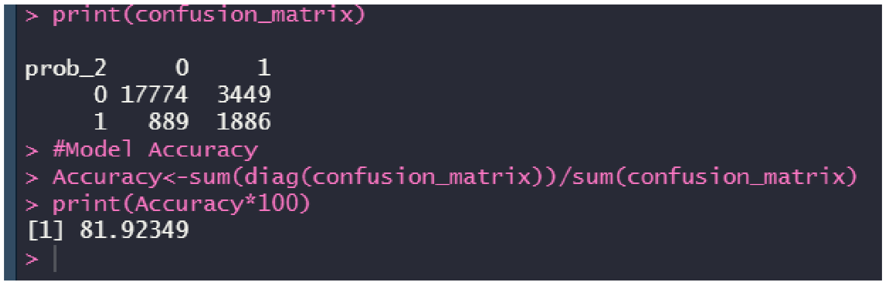
Lets rerun the model with the best subset obtained.

```
mod_2<-glm(payment_default ~ LIMIT_BAL + EDUCATION + AGE + PAY_0 + PAY_2 +
PAY_3 + PAY_4 + PAY_5 + PAY_6 + PAY_AMT1 + PAY_AMT2 + PAY_AMT3 + PAY_AMT4 +
PAY_AMT5 + PAY_AMT6 + PAY_RATIO_APR, train, family='binomial')

summary(mod_2)
```



```
Call:
glm(formula = payment_default ~ LIMIT_BAL + EDUCATION + AGE +
    PAY_0 + PAY_2 + PAY_3 + PAY_4 + PAY_5 + PAY_6 + PAY_AMT1 +
    PAY_AMT2 + PAY_AMT3 + PAY_AMT4 + PAY_AMT5 + PAY_AMT6 + PAY_RATIO_APR,
    family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2454  -0.5883  -0.5256  -0.3412   3.3694

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
(Intercept)                1.096e+00  1.446e+00   0.758 0.448549
LIMIT_BAL                 -1.218e-06  1.691e-07  -7.202 5.92e-13 ***
EDUCATIONHigh School      -9.891e-02  5.515e-02  -1.793 0.072910 .
EDUCATIONOthers           -1.046e+00  4.303e-01  -2.432 0.015030 *
EDUCATIONUniversity       -1.201e-02  4.049e-02  -0.297 0.766789
EDUCATIONUnknown          -1.130e+00  2.471e-01  -4.573 4.82e-06 ***
AGE                        1.075e-02  1.933e-03   5.563 2.65e-08 ***
PAY_02 month delay         1.280e+00  6.414e-02  19.951  < 2e-16 ***
PAY_03 month delay         1.240e+00  1.625e-01   7.631 2.33e-14 ***
PAY_04 month delay         6.454e-01  3.172e-01   2.035 0.041873 *
PAY_05 month or more delay 2.493e-01  4.440e-01   0.562 0.574404
PAY_0Paid Duly            -7.627e-01  5.458e-02 -13.974  < 2e-16 ***
PAY_22 month delay         7.137e-01  6.539e-01   1.091 0.275085
PAY_23 month delay         8.353e-01  6.721e-01   1.243 0.213931
PAY_24 month delay         9.330e-02  7.318e-01   0.127 0.898549
PAY_25 month or more delay 1.758e+00  8.491e-01   2.071 0.038363 *
PAY_2Paid Duly             4.856e-01  6.538e-01   0.743 0.457674
PAY_32 month delay         9.257e+00  1.195e+02   0.077 0.938236
PAY_33 month delay         9.343e+00  1.195e+02   0.078 0.937668
PAY_34 month delay         8.701e+00  1.195e+02   0.073 0.941943
PAY_35 month or more delay 8.734e+00  1.195e+02   0.073 0.941722
PAY_3Paid Duly             8.911e+00  1.195e+02   0.075 0.940546
PAY_42 month delay        -1.086e+01  1.195e+02  -0.091 0.927556
PAY_43 month delay        -1.110e+01  1.195e+02  -0.093 0.925977
PAY_44 month delay        -1.069e+01  1.195e+02  -0.089 0.928708
PAY_45 month or more delay -1.289e+01  1.195e+02  -0.108 0.914100
PAY_4Paid Duly            -1.107e+01  1.195e+02  -0.093 0.926191
PAY_53 month delay        -1.508e-01  2.292e-01  -0.658 0.510666
PAY_54 month delay        -1.152e-02  4.785e-01  -0.024 0.980792
PAY_55 month or more delay 1.961e+00  8.629e-01   2.273 0.023017 *
PAY_5Paid Duly            -2.635e-01  8.064e-02  -3.267 0.001086 **
PAY_63 month delay         5.657e-01  2.453e-01   2.306 0.021087 *
PAY_64 month delay        -4.628e-01  5.195e-01  -0.891 0.373065
PAY_65 month or more delay 3.285e-01  5.856e-01   0.561 0.574773
PAY_6Paid Duly            -3.151e-01  7.044e-02  -4.473 7.70e-06 ***
PAY_AMT1                  -7.404e-06  2.079e-06  -3.561 0.000369 ***
PAY_AMT2                  -5.076e-06  1.764e-06  -2.878 0.004002 **
PAY_AMT3                  -3.162e-06  1.630e-06  -1.940 0.052372 .
PAY_AMT4                  -3.379e-06  1.738e-06  -1.945 0.051810 .
PAY_AMT5                  -4.161e-06  1.724e-06  -2.413 0.015808 *
PAY_AMT6                  -2.976e-06  1.507e-06  -1.975 0.048288 *
PAY_RATIO_APR             -9.924e-05  6.695e-05  -1.482 0.138294
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 25429  on 23997  degrees of freedom
Residual deviance: 21080  on 23956  degrees of freedom
AIC: 21164

Number of Fisher Scoring iterations: 9

> |
```

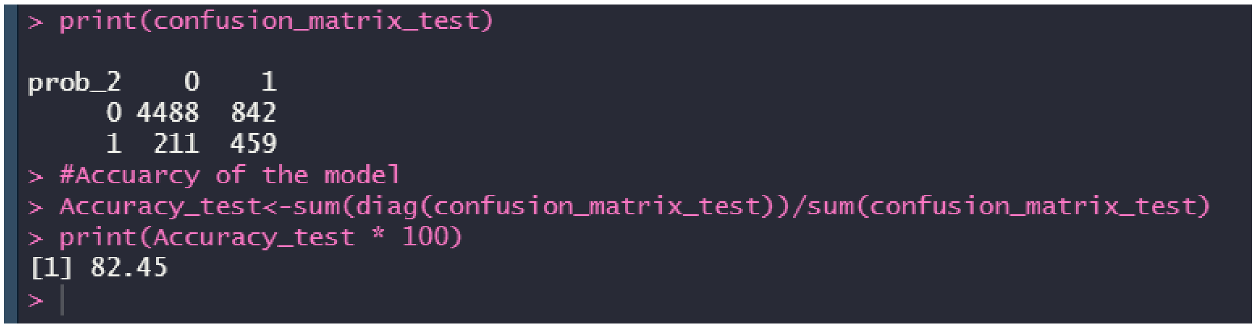
```
predict_2<-predict(mod_2,train,type='response')
prob_2<-ifelse(predict_2>0.5,1,0)

#Confusion Matrix
confusion_matrix<-table(prob_2, train$payment_default)
print(confusion_matrix)

#Model Accuracy
Accuracy<-sum(diag(confusion_matrix))/sum(confusion_matrix)
print(Accuracy*100)
```



```
> print(confusion_matrix)

prob_2     0     1
     0 17774  3449
     1   889  1886
> #Model Accuracy
> Accuracy<-sum(diag(confusion_matrix))/sum(confusion_matrix)
> print(Accuracy*100)
[1] 81.92349
> 
```

Our model accuracy is the same, therefore these are the best subset of attributes for logistic regression. Lets run the model now on test data set.

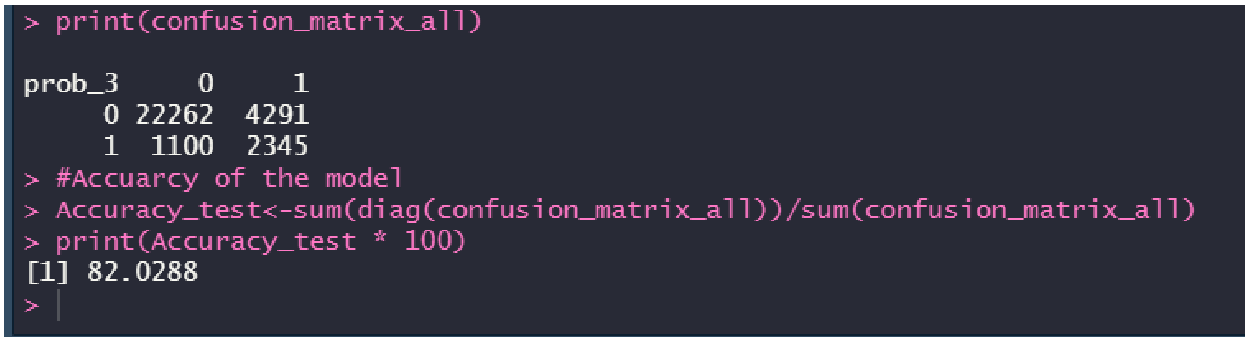
```
predict_2<-predict(mod_2,test,type='response')
prob_2<-ifelse(predict_2>0.5,1,0)

#Confusion matrix for test data
confusion_matrix_test<-table(prob_2,test$payment_default)

#Accuarcy of the model
Accuracy_test<-sum(diag(confusion_matrix_test))/sum(confusion_matrix_test)
print(Accuracy_test * 100)
```



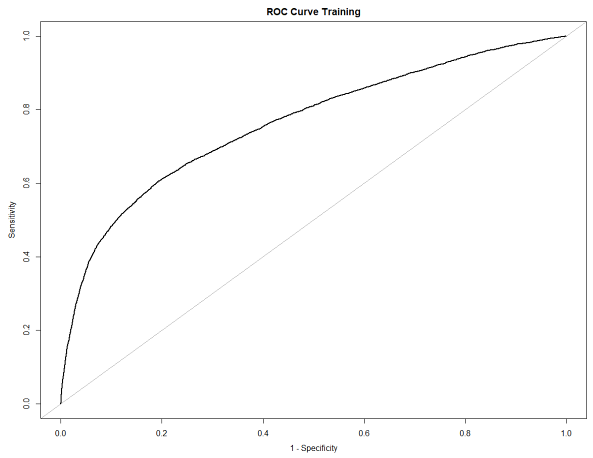
```
> print(confusion_matrix_test)

prob_2    0    1
     0 4488  842
     1  211  459
> #Accuarcy of the model
> Accuracy_test<-sum(diag(confusion_matrix_test))/sum(confusion_matrix_test)
> print(Accuracy_test * 100)
[1] 82.45
> 
```

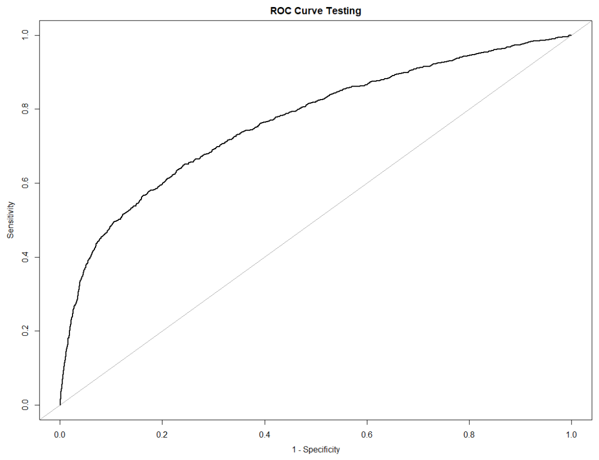
Our logistic regression model gave a slightly better accuracy for testing dataset than train which is a good sign. Lets run the model on complete dataset.

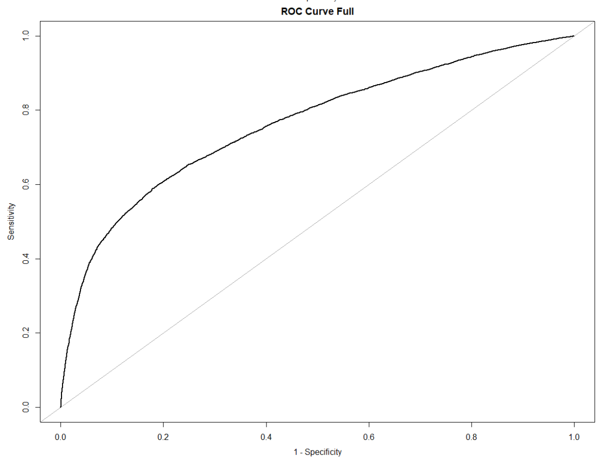
```
predict_3<-predict(mod_2,data,type='response')
prob_3<-ifelse(predict_3>0.5,1,0)

#Confusion matrix for all data
confusion_matrix_all<-table(prob_3,data$payment_default)

#Accuarcy of the model
Accuracy_all<-sum(diag(confusion_matrix_all))/sum(confusion_matrix_all)
print(Accuracy_all * 100)
```



```
> print(confusion_matrix_all)

prob_3     0     1
     0 22262  4291
     1  1100  2345
> #Accuarcy of the model
> Accuracy_test<-sum(diag(confusion_matrix_all))/sum(confusion_matrix_all)
> print(Accuracy_test * 100)
[1] 82.0288
> 
```



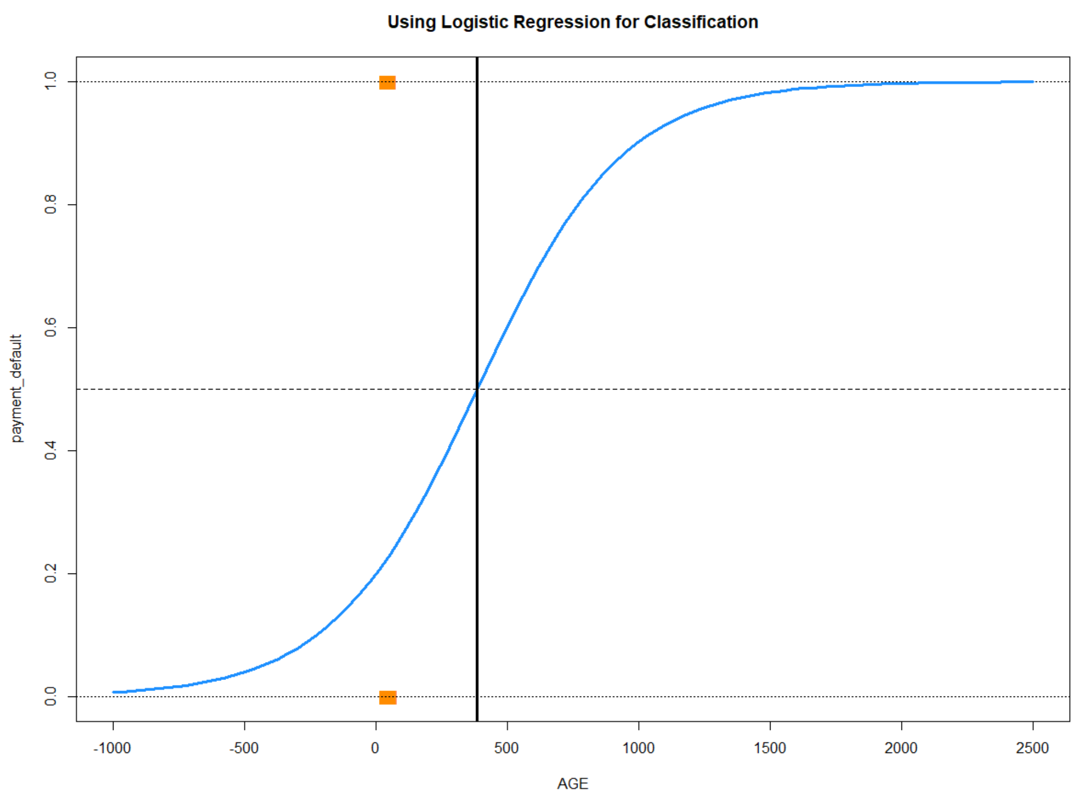




The Receiver Operating Character Curve (ROC) measures how well the logistic regression can predict if we change the discrimination threshold. The area under the ROC (AUC) serves as another useful metric for evaluating the predictive accuracy of the logistic regression, generally classification methods. The AUC for all three sets can be found in the summary table below. The baseline AUC is 0.5 and we got around 0.76 for all three models so overall we can conclude that our logistic regression model is a good fit!

The S-curve plots the logistic curve against the data. This plot contains a wealth of information. The orange bars represent the data attribute which in our case we chose AGE. The blue curve is the predicted probabilities given by the fitted logistic regression. The solid vertical black line represents the decision boundary, the midpoint of the S-curve where the AGE obtains a predicted probability of 0.5.

```
# s curve
gre_mdl <- glm(payment_default ~ AGE, data = data, family = "binomial")
data$payment_default <- as.numeric(data$payment_default)-1
plot(payment_default ~ AGE, data = data,
     col = "darkorange", pch = "|", xlim = c(-1000, 2500), ylim = c(0, 1),
     main = "Using Logistic Regression for Classification")
abline(h = 0, lty = 3)
abline(h = 1, lty = 3)
abline(h = 0.5, lty = 2)
curve(predict(gre_mdl, data.frame(AGE = x), type = "response"), add = TRUE, lwd =
3, col = "dodgerblue")
abline(v = -coef(gre_mdl)[1] / coef(gre_mdl)[2], lwd = 3)
```



Final logistic regression model equation:

$$Logit(payment\ default) = 1.096 - 1.218 * 10^{-6}\ LIMIT\_BAL - 1.13\ EDUCATION + 1.075 * 10^{-2}\ AGE - 7.627 * 10^{-1}\ PAY\_0 + 1.28\ PAY\_2 + 1.24 PAY\_3 + 6.540 * 10^{-1}\ PAY\_4 - 2.635 * 10^{-1}\ PAY\_5 - 3.151 * 10^{-1}\ PAY\_6 - 7.404 * 10^{-6}\ PAY\_AMT1 - 5.076 * 10^{-6}\ PAY\_AMT2 - 3.162 * 10^{-6}\ PAY\_AMT3 - 3.379 * 10^{-6}\ PAY\_AMT4 - 4.161 * 10^{-6}\ PAY\_AMT5 - 2.176 * 10^{-6}\ PAY\_AMT6 - 9.924 * 10^{-5}\ PAY\_RATIO\_APR$$

Logistic Regression Summary Table			
Metric	**Testing**	**Training**	**Full**
Accuracy	82.45	81.92	82.03
Sensitivity	0.95	0.95	0.95
Specificity	0.35	0.35	0.35
PPV	0.84	0.83	0.83
NPV	0.68	0.67	0.68
AUC	0.7656	0.762	0.7627

Classification Trees

Classification trees are a type of decision tree based model used for regression and classification. These involve segmenting the predictor space into a number of smaller and simpler regions and using the most commonly occurring class of observations in a region to make predictions about that region.

The following are the steps in constructing a classification tree:

1. Using recursive binary splitting, build a large tree. Stop only when each terminal node has fewer than a certain number of observations or on splitting further the error does not improve above a certain threshold limit.
2. Apply cost complexity/weakest link pruning in conjunction with 10 fold cross validation to obtain the optimal α that minimizes the average error.
3. Using that value of α, we shall choose the most optimal subtree.

Now let us go through each of the steps mentioned.

Recursive Binary Splitting

To stratify our predictor space into simpler regions, we must first choose a criterion to make binary splits.
As opposed to Regression Trees, we cannot choose RSS to be the splitting criterion.
Other possible splitting criteria are classification error rate, Gini index and Entropy/Deviance.
Among the three possible criterions, we choose the Gini index to construct our tree.

Gini index is given by:

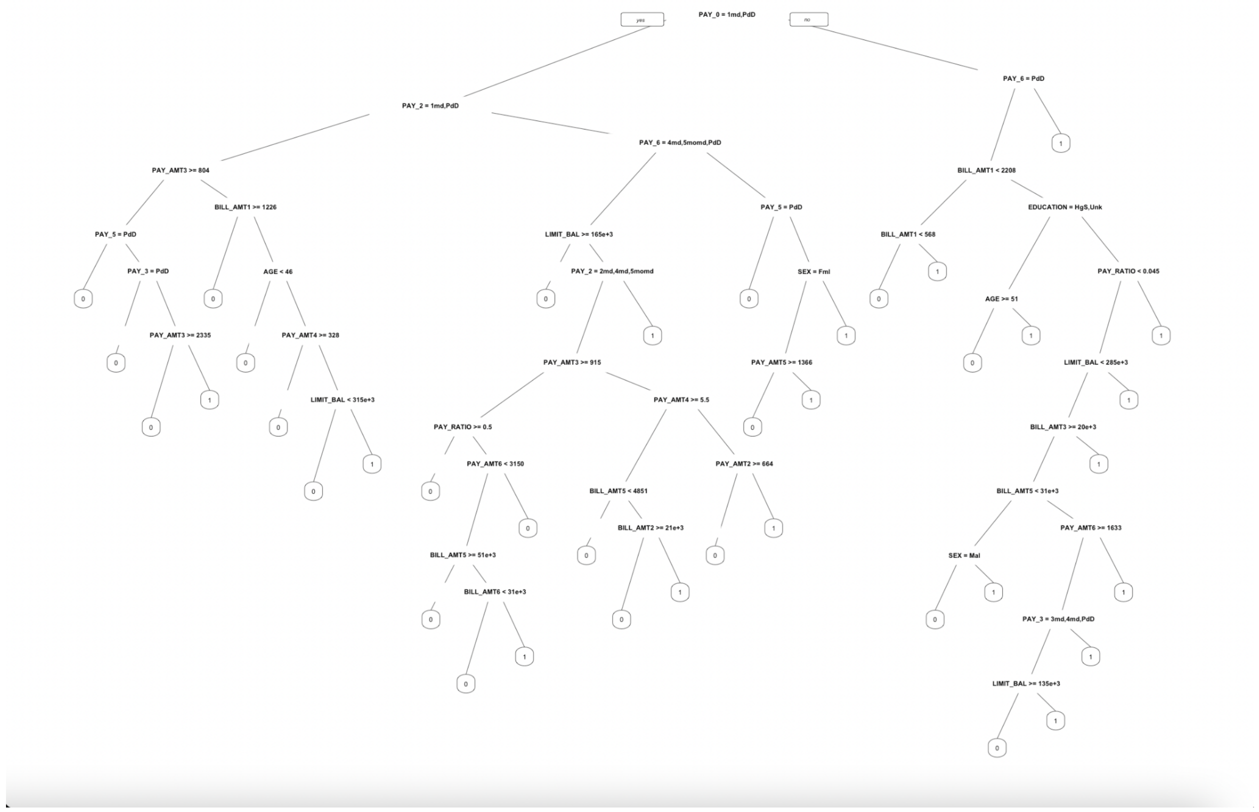
$$G = \sum_{k=1} p_{mk}(1 - p_{mk})$$

Here, p_{mk} represents the proportion of training observations in the mth region that are from kth class.

If a node contains observations from mostly one class, the gini index would have a small value. For this reason, it is also a measure of node purity. To build a tree, we choose the next split as one that results in the smallest G.

In our tree, each prediction is either 1 or 0, depending on whether people are likely to default on the next payment (prediction = 1) or not default on the next payment (prediction = 0).

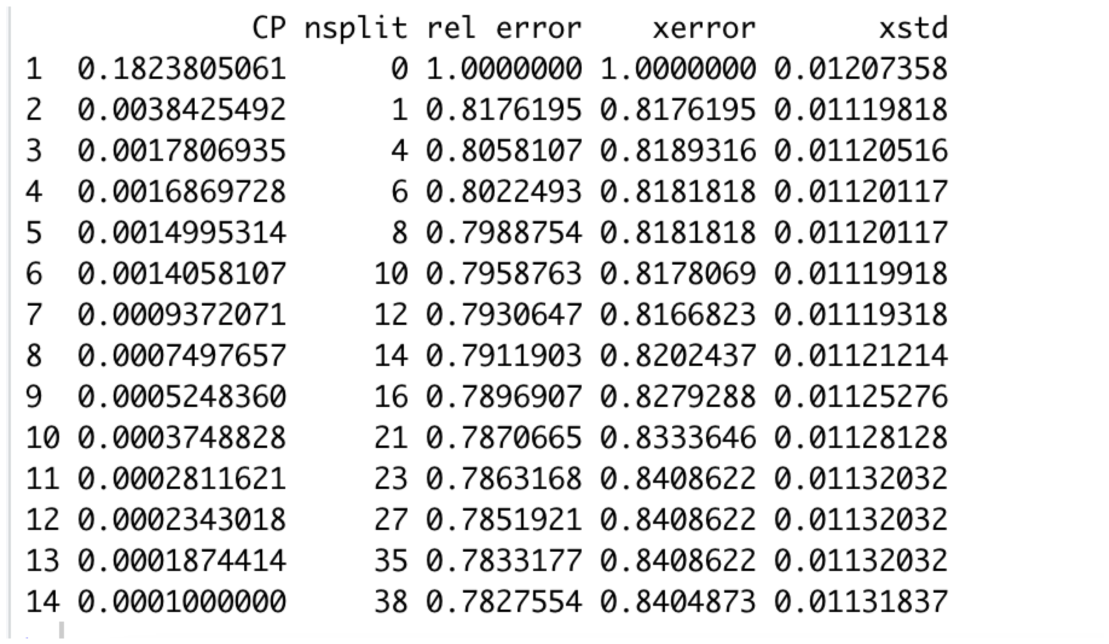
```
set.seed(490)

# split into training and testing
credit.split <- sample(1:nrow(credit), size = nrow(credit)*0.8)
train_set <- credit[credit.split,]
test_set <- credit[-credit.split,]

class.cart2 <- rpart(formula = payment_default ~ .-ID, data = train_set, method =
"class", control = rpart.control(minbucket = 60,minsplit = 30, xval = 10,
cp=0.0001))
prp(class.cart2, roundint = FALSE)
```



Here we can see the structure of the tree.
The most important criteria when it comes to predicting if people will default on their next payment is PAY_0 (repayment status in September 2005). If a person had duly paid their credit amount (PAY_0 = -2,-1,0) or if there was a payment delay of only 1 month (PAY_0 = 1), then the next most important criteria was the repayment status in August 2005 (PAY_2). For all other delays in September 2005, the next most important criteria was repayment status in July 2005 (PAY_3).

Some of the nodes of the unpruned tree are factors like PAY_6 (repayment status in April 2005). If people had duly paid their dues in April (PAY_6 =-1) then they would not default on payment in October 2005. Else they would default on the payment in October.
Similarly, if the bill amount in April was greater or equal to $6680 then people would not default in the bill payment else they would.

```
cp.class.param <- class.cart2$cptable
cp.class.param
```



```
              CP nsplit rel error    xerror       xstd
1  0.1823805061      0 1.0000000 1.0000000 0.01207358
2  0.0038425492      1 0.8176195 0.8176195 0.01119818
3  0.0017806935      4 0.8058107 0.8189316 0.01120516
4  0.0016869728      6 0.8022493 0.8181818 0.01120117
5  0.0014995314      8 0.7988754 0.8181818 0.01120117
6  0.0014058107     10 0.7958763 0.8178069 0.01119918
7  0.0009372071     12 0.7930647 0.8166823 0.01119318
8  0.0007497657     14 0.7911903 0.8202437 0.01121214
9  0.0005248360     16 0.7896907 0.8279288 0.01125276
10 0.0003748828     21 0.7870665 0.8333646 0.01128128
11 0.0002811621     23 0.7863168 0.8408622 0.01132032
12 0.0002343018     27 0.7851921 0.8408622 0.01132032
13 0.0001874414     35 0.7833177 0.8408622 0.01132032
14 0.0001000000     38 0.7827554 0.8404873 0.01131837
```

The above tree has resulted in a tree with 38 splits. Such a large number of splits makes it difficult to interpret the results and could be a result of overfitting. To reduce overfitting and obtain a tree that makes interpretation easier, we will prune the tree to reach a smaller tree.

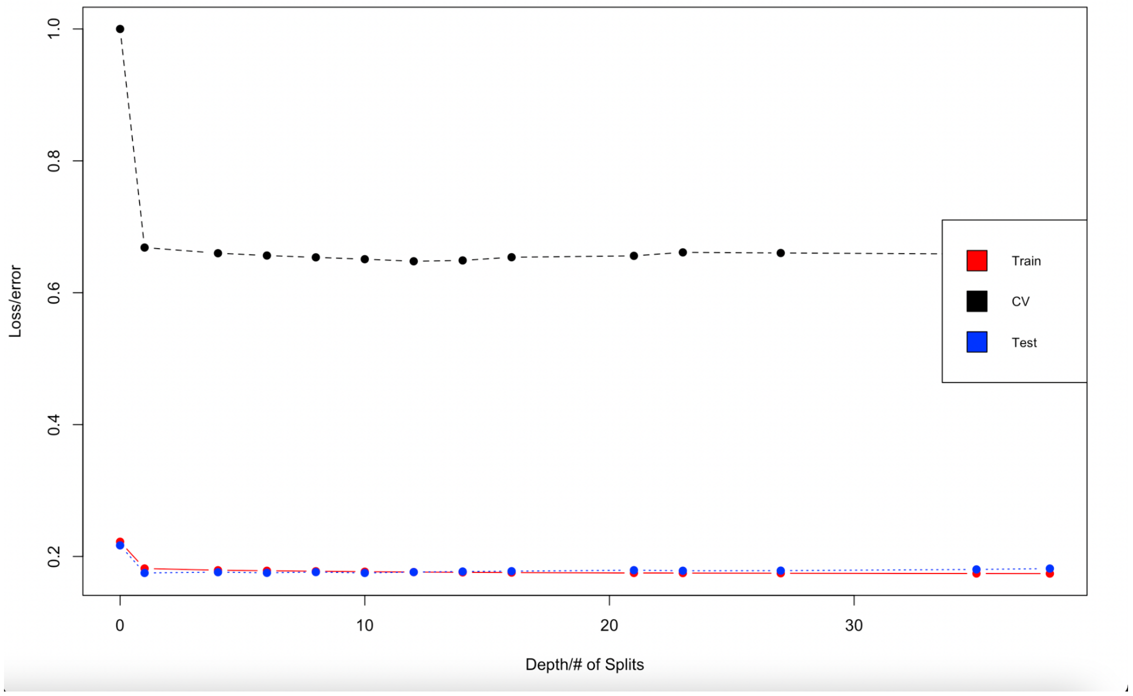
Cost Complexity Pruning

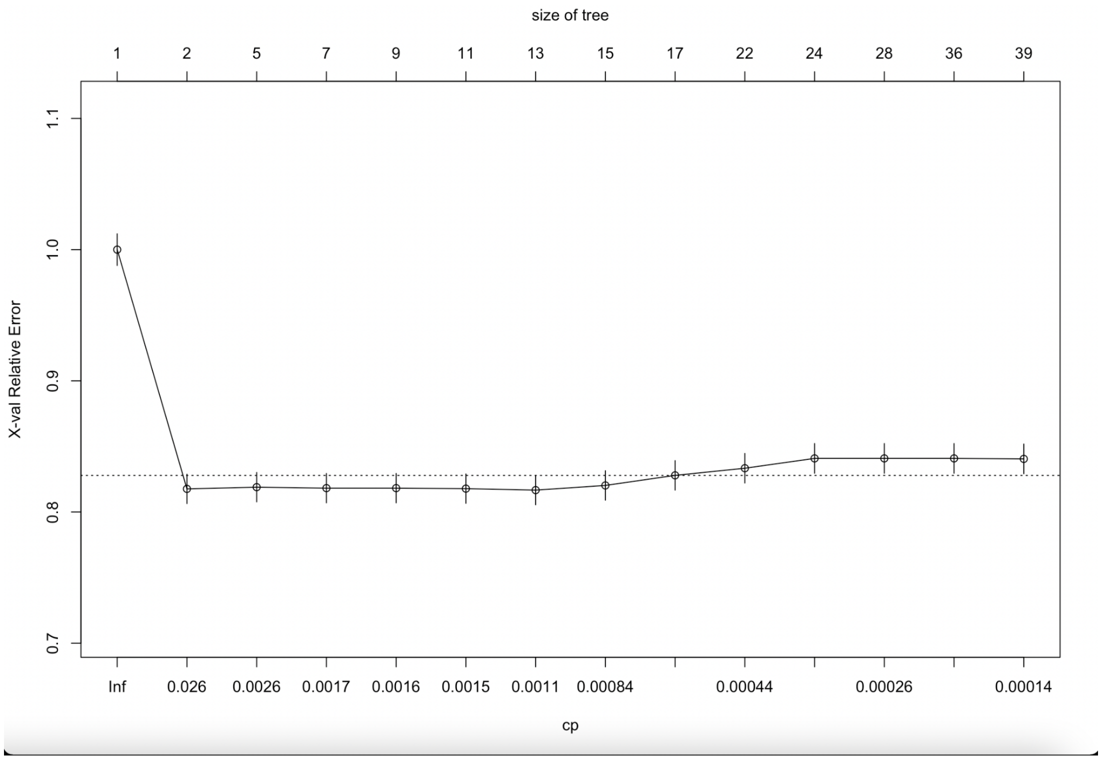
To prune a tree and obtain an optimal subtree, we use classification error rate as our pruning criteria. It is given by,

$$E = 1 - max(p_{mk}) + \alpha|T|$$

It is simply the ratio of training observations in a region that do not belong to the most common class. Here α is the tuning parameter and $|T|$ indicates the number of terminal nodes of tree T.
When α = 0, then the tree is T_0 because the equation measures error rate. But as α increases, there is a cost to having multiple nodes. So the tree will tend to be minimized.
Now, we obtain a sequence of subtrees as a function of α and then use 10 fold cross validation to obtain the optimal value of the tuning parameter and obtain the subtree corresponding to this α.

```r
# Cost Complexity Pruning

train.err <- double(14)
cv.err <- double(14)
test.err <- double(14)

for (i in 1:nrow(cp.class.param)) {
  alpha <- cp.class.param[i, 'CP']
  train.cm <- table(train_set$payment_default, predict(prune(class.cart2,
cp=alpha), newdata = train_set, type='class'))
  train.err[i] <- 1-sum(diag(train.cm))/sum(train.cm)
  cv.err[i] <- cp.class.param[i, 'xerror'] * cp.class.param[i, 'rel error']
  test.cm <- table(test_set$payment_default, predict(prune(class.cart2, cp=alpha),
newdata = test_set, type='class'))
  test.err[i] <- 1-sum(diag(test.cm))/sum(test.cm)
}

# Print classification error (1 - accuracy) values
train.err
test.err

# Plot training, CV and testing errors at # of Splits/depth

matplot(cp.class.param[,'nsplit'], cbind(train.err, cv.err, test.err), pch=19,
col=c("red", "black", "blue"), type="b", ylab="Loss/error", xlab="Depth/# of
Splits")
legend("right", c('Train', 'CV', 'Test') ,col=seq_len(3),cex=0.8,fill=c("red",
"black", "blue"))

plotcp(class.cart2)
```



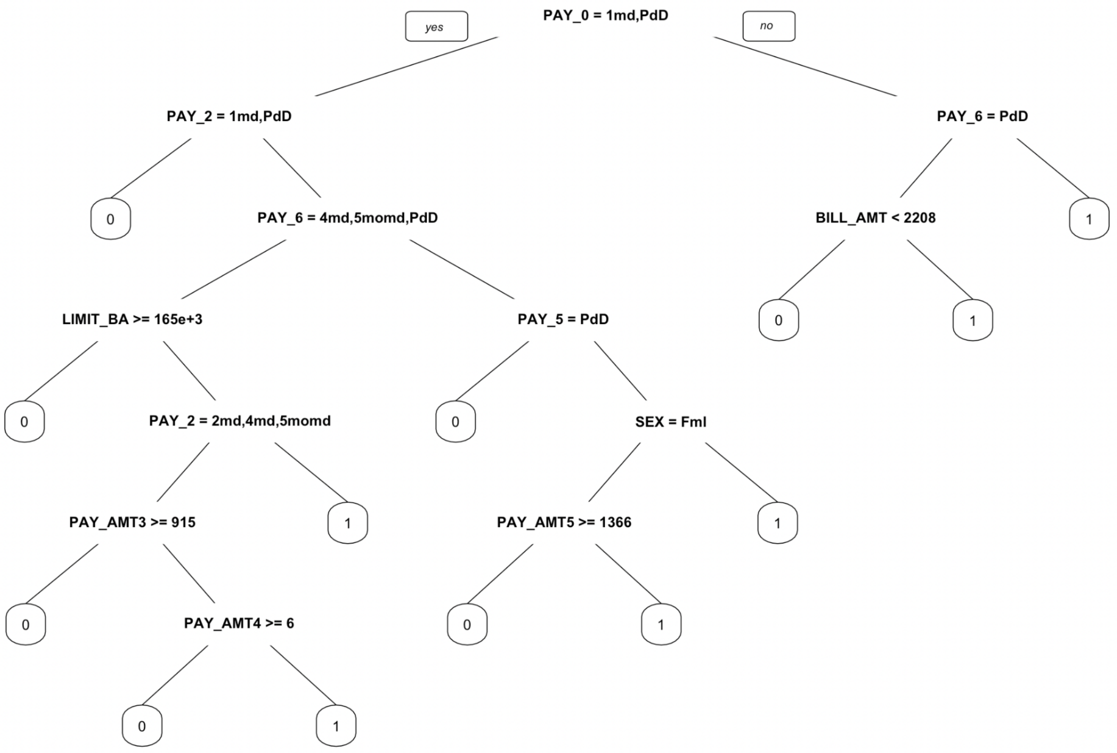


We can see that for a tree of size 8, the error is minimum. So we chose the complexity/tuning parameter that corresponds to that tree size and build the resulting subtree.
The corresponding α is 0.0009372071.

Final Output

Using 0.0009372071 as the value of tuning parameter, we prune the tree at the value of α.

```
# Check CP table, when size of tree =13, the nsplit =12 and CP =
0.0009372071
# Prune the tree at nsplit =12 defined by the complexity parameter
prune.class.trees <- prune(class.cart2, cp=cp.class.param[7,'CP'])
prp(prune.class.trees)
```



We can see that pruning the tree has resulted in a much smaller tree which can be easily interpreted. It has 12 splits and 13 terminal leaves.

The most important factor in deciding whether people will default on their next payment remains PAY_0. It is followed by PAY_2 and PAY_6. Bill amount and limit balance also pays a role in predicting if people will default or not.
The right subtree can be explained as people not having either fully paid their dues or 1 month behind their payments in September and defaulted on their payment for April are predicted to default in the month of October. On the other hand, people who duly paid their dues in April and have a bill amount less than $2208 would not default in October's payment. But the ones who have bill amount more than $2208 are predicted to default in October's payment.
This tree can be interpreted much more easily than the full tree and now we shall evaluate the model to find out its accuracy and other evaluation parameters.

Model Evaluation

		Unpruned Tree	Pruned Tree
Tree Size		39	13
Accuracy	Train	0.825	0.823
	Test	0.818	0.823
	All (Cross-validated)	0.342	0.352

We can see we do not experience any losses in terms of train or testing errors on pruning our tree. In fact, we experience a slight increase on our testing accuracy on pruned tree as compared to a large tree with 39 terminal nodes.

Next, using confusion tables for train set, test set and All cases, we shall compute classification accuracy, sensitivity, specificity, PPV and NPV.

```
# Creating confusion tables

# For testing data
cftable <- table(test_set$payment_default, predict(prune.class.trees, type =
'class', newdata = test_set))
cftable

# Calculate Accuracy
accuracy <- sum(diag(cftable))/sum(cftable)
accuracy

# Calculate Sensitivity
sensitivity<-cftable[1]/(cftable[1] + cftable[2])
sensitivity
```

```
# Calculate Specificity
specificity <- cftable[4]/(cftable[3] + cftable[4])
specificity

# Calculate Positive Predictive Value
ppv <- cftable[1]/(cftable[1] + cftable[3])
ppv

# Calculate Negative Predictive Value
npv <- cftable[4]/(cftable[2] + cftable[4])
npv
```

Classification Tree Summary Table

Metric	Testing	Training	Full
Accuracy (%)	82.3	82.3	82.3
Sensitivity	0.84	0.84	0.84
Specificity	0.67	0.70	0.69
PPV	0.95	0.95	0.95
NPV	0.36	0.37	0.37
AUC	0.65	0.66	0.66

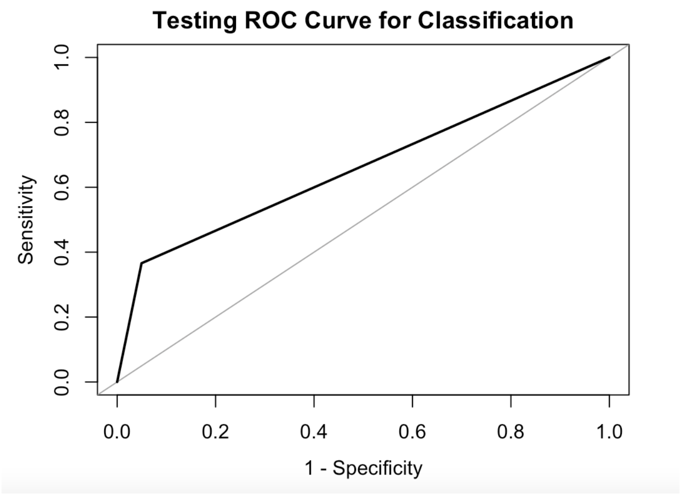
Throughout the three cases, the classification tree model has a near constant accuracy of 82.2%. Sensitivity, Specificity, PPV and NPV also do not show much variation. This means that the model performs equally well across all the three datasets.

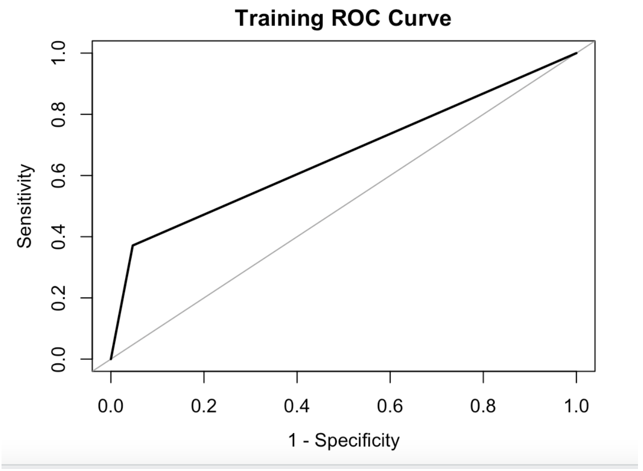
A sensitivity of 0.84 implies that 84% of time the model correctly predicts people who will not default on their payment and a specificity of 67% implies that 67% of the time, the model will correctly predict the people who will default on their payment.

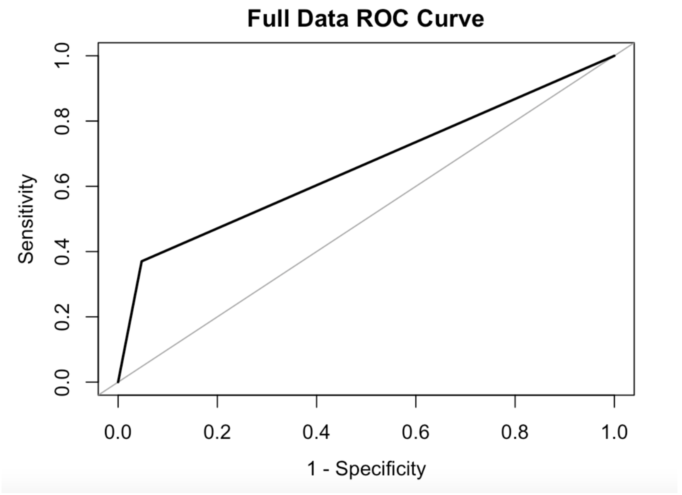
ROC Curve

```
# test data
test.prob <- predict(prune.class.trees, newdata = test_set, type = "class")
str(test.prob)
table(test.prob)
summary(test_set)

test.prob <- as.numeric(test.prob)
test.roc = roc(test_set$payment_default, test.prob)
plot.roc(test.roc, col=par("fg"),plot=TRUE,print.auc = FALSE, legacy.axes = TRUE,
asp =NA, main = "Testing ROC Curve for Classification")
test.roc
```

```
# train data
train.prob <- predict(prune.class.trees, newdata = train_set, type = "class")
str(train.prob)
table(train.prob)
summary(train_set)

train.prob <- as.numeric(train.prob)
train.roc = roc(train_set$payment_default, train.prob)
plot.roc(train.roc, col=par("fg"),plot=TRUE,print.auc = FALSE, legacy.axes = TRUE,
asp =NA, main = "Training ROC Curve")
train.roc

# Full data
full.prob <- predict(prune.class.trees, newdata = credit, type = "class")
str(full.prob)
table(full.prob)
summary(credit)

full.prob <- as.numeric(full.prob)
full.roc = roc(credit$payment_default, full.prob)
plot.roc(full.roc, col=par("fg"),plot=TRUE,print.auc = FALSE, legacy.axes = TRUE,
asp =NA, main = "Full Data ROC Curve")
full.roc
```


Testing ROC Curve for Classification
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1 - Specificity


Training ROC Curve
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1 - Specificity


Full Data ROC Curve
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1 - Specificity

On looking at the ROC curves, we can see that none of them have the ideal shape of ROC curves that hug the top left corner of the graph. Even the average area under the curve is 0.66 which is not very good.

One can say that Classification tree is not a good model to make predictions for this dataset.

Comparison Between Models

On the basis of Accuracy and the parameters computed using confusion tables we can compare Logistic Regression and Classification tree to find out which model better predicts people who will default in the next payment or not.
We will use the testing set for comparison.

Metric	Classification Tree	Logistic Regression
Accuracy (%)	82.3	82.45
Sensitivity	0.84	0.95
Specificity	0.67	0.35
PPV	0.95	0.84
NPV	0.36	0.68
AUC	0.65	0.7656

Both of the models have similar accuracy. But the area under the ROC curve for both models differs by a significant amount. For the logistic regression model, the area under the curve is 0.76 whereas for the classification tree model, the area is 0.65. This indicates that overall the logistic regression model is a better model for classification than the tree model.

Logistic regression is also better at predicting people who will not default on their payments with a sensitivity of 0.95 as compared to 0.84.
Whereas, the tree based model outperforms the logistic regression model while predicting people who would default on their payments.

References

1.An Introduction To Statistical Learning With Applications in R by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani.

2. https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset