CDSA1010-Lab2-Group3

Yrysguli Kaireden, Amina Sheik-Ahmed, Lily Ye

24/04/2020

Abstract: In this assignment, a dataset containing default credit card payments samples from customers in Taiwan will be used. The goal of this assignment is to accurately predict the likelihood of a customer defaulting in payment. The data was carefully explored, adjusted and transformed before proceeding to modelling. The methods used to solve this problem are the following classification algorithms: logistic regression, decision tree, random forest and KNN. To improve the prediction accuracy of the models, missing values and skewness were addressed. Additional improvement could have been made with more attributes in the dataset, such as income. The models generated from this dataset and project can be deployed to be used by bank institution executives or risk analysts to foresee and prevent credit card debt.

Introduction

Credit cards in today's day and age, are the most widely promoted and used products amongst financial products available. Proper credit card usage is advantageous for customers, financial institution's performance and the economy. Poor credit card management such as default negatively impacts the consumer's credit and poses a greater challenge to banks as such may result in the loss of large sums of money, adversely impacting the institution's financial performance. Banks can greatly reduce their risk of loss by employing techniques in predicting delinquent credit card users. The absence of such practices can be seen in the 2006 cash and credit crisis in Taiwan which was expected to peak in the third quarter. In an attempt to increase market shares, card-issuing banks in Taiwan did not take any measures in assessing potential risk, instead, an over-issuance of cash and credit cards was approved to unqualified applicants (Yeh & Lien, 2009). Also, credit card users did overuse credit card consumption, regardless of their ability to repay, this resulted in a heavy credit and cash card debts.

For many years, banks managed to reduce uncertainty by employing risk prediction methods whereby financial information such as a repayment record and transaction history is used to determine a customer's credit risk (Yeh & Lien, 2009). However, such practices alone are not sufficient in managing risk and improving the financial performance of banks. Banks should employ techniques where a customer's probability of default can be predicted.

Data

Background and Objective

The dataset used in this assignment is retrieved from UCI machine learning repository. It contains 25 features which include: the ID of each client, the amount of credit given in NT dollars, gender, education level, marital status, age, repayment status in 2005 for the months of April-September, amount of bill statement in 2005 for the months of April - September, amount of previous payment in 2005 for the months April - September, and default payment.

The objective of this study is to provide an algorithm that can predict the likelihood of a customer defaulting payment based on payment history and customer characteristics. The target variable "default payments" is binary and responses are coded as Yes=1 or No=0; as such, a number of models can be used to predict customer reliability, logistic regression, decision tree, random forest and KNN will be used. Models will be evaluated by several common metrics including accuracy, precision, recall and F-measure etc..

In the next sections, missing values and outliers will be handled. After that, numeric features transformation, reduction and selection steps will be described to explain how the final set of features were received. The final dataset is then passed to classification algorithms, out of which the best performing model is selected based on the set of metrics.

Data understanding

The credit card dataset has 30,000 observations and 25 columns.

```
##   ID LIMIT_BAL SEX EDUCATION MARRIAGE AGE PAY_0 PAY_2 PAY_3 PAY_4 PAY_5 PAY_6
## 1  1     20000   2         2        1  24     2     2    -1    -1    -2    -2
## 2  2    120000   2         2        2  26    -1     2     0     0     0     2
## 3  3     90000   2         2        2  34     0     0     0     0     0     0
## 4  4     50000   2         2        1  37     0     0     0     0     0     0
## 5  5     50000   1         2        1  57    -1     0    -1     0     0     0
## 6  6     50000   1         1        2  37     0     0     0     0     0     0
##   BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6 PAY_AMT1 PAY_AMT2
## 1      3913      3102       689         0         0         0        0      689
## 2      2682      1725      2682      3272      3455      3261        0     1000
## 3     29239     14027     13559     14331     14948     15549     1518     1500
## 4     46990     48233     49291     28314     28959     29547     2000     2019
## 5      8617      5670     35835     20940     19146     19131     2000    36681
## 6     64400     57069     57608     19394     19619     20024     2500     1815
##   PAY_AMT3 PAY_AMT4 PAY_AMT5 PAY_AMT6 Y
## 1        0        0        0        0 1
## 2     1000     1000        0     2000 1
## 3     1000     1000     1000     5000 0
## 4     1200     1100     1069     1000 0
## 5    10000     9000      689      679 0
## 6      657     1000     1000      800 0
```

```
## '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      : int  2 2 2 2 1 1 1 2 2 1 ...
##  $ EDUCATION: int  2 2 2 2 2 1 1 2 3 3 ...
##  $ MARRIAGE : int  1 2 2 1 1 2 2 2 1 2 ...
##  $ AGE      : int  24 26 34 37 57 37 29 23 28 35 ...
##  $ PAY_0    : int  2 -1 0 0 -1 0 0 0 0 -2 ...
##  $ PAY_2    : int  2 2 0 0 0 0 0 -1 0 -2 ...
##  $ PAY_3    : int  -1 0 0 0 -1 0 0 -1 2 -2 ...
##  $ PAY_4    : int  -1 0 0 0 0 0 0 0 0 -2 ...
##  $ PAY_5    : int  -2 0 0 0 0 0 0 0 0 -1 ...
##  $ PAY_6    : int  -2 2 0 0 0 0 0 -1 0 -1 ...
##  $ BILL_AMT1: 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 ...
##  $ Y        : int  1 1 0 0 0 0 0 0 0 0 ...
```

The first column shows the credit card users' ID. Second column shows the credit card maximum limit that the credit card user can use per month. Third column shows the gender of the credit card user, where 1 is male and 2 is female. Fourth column shows the education level, where 1 is graduate school, 2 is university, 3 is high school, and 4 is others.

In addition to the usual demographic information of the credit card users, the dataset also has information on the credit card users' payment history over the period of April to September 2005. Therefore, columns 7-24 represent each month of September, August, July, June, May, and April 2005, and the credit card users' repayment status, bill amount, and payment amount. For instance, the columns 7-12 represent repayment status in September, August, July, June, May, and April, and the columns 13-18 represent amount of bill statement for the same 6 months, and the columns 19-24 represent amount of payment also for the same months also.

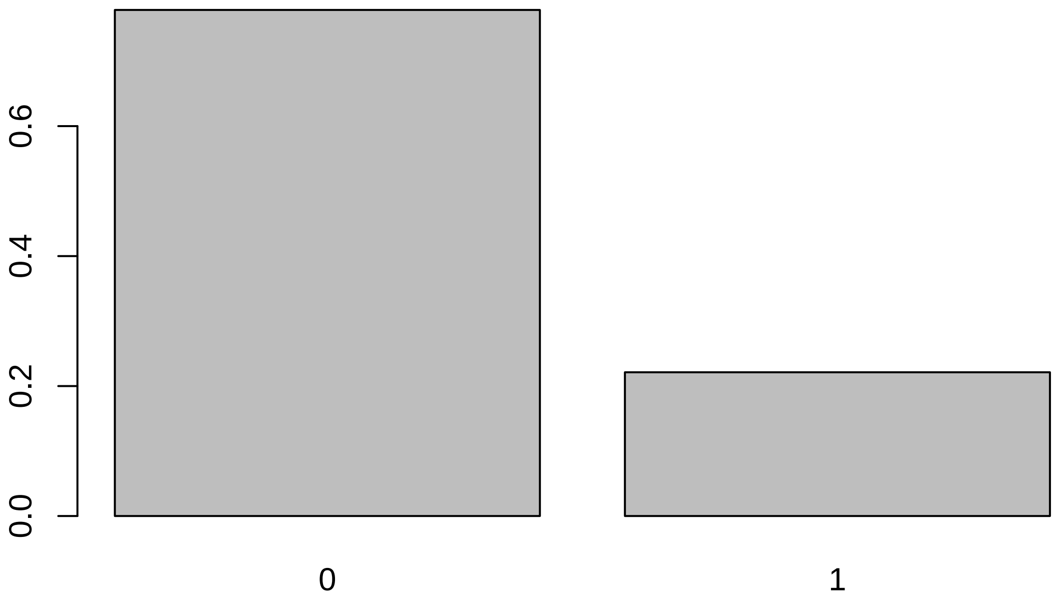
While the columns 13-24 show the exact numeric value of bill statements and payment amount, the columns 7-12 for repayment status have 11 values with different meanings. "-2" means that a credit card user did not owe anything. "-1" means that a credit card user paid duly. "0" means that a credit card user was not a customer in the first place. This can be observed from the bill statements all showing $0 for credit users with "0" value in their repayment status. 1 means that the credit card user's payment was delayed for 1 month. 2 means that the payment was delayed for 2 months, and so on, until 9, which means that the payment was delayed for 9 months and above.

Finally, the last column, 25th column, is our target variable of credit card user's default payment status, where 0 means no default payment, and 1 means default payment.

Correlation and Relationship of the Variables

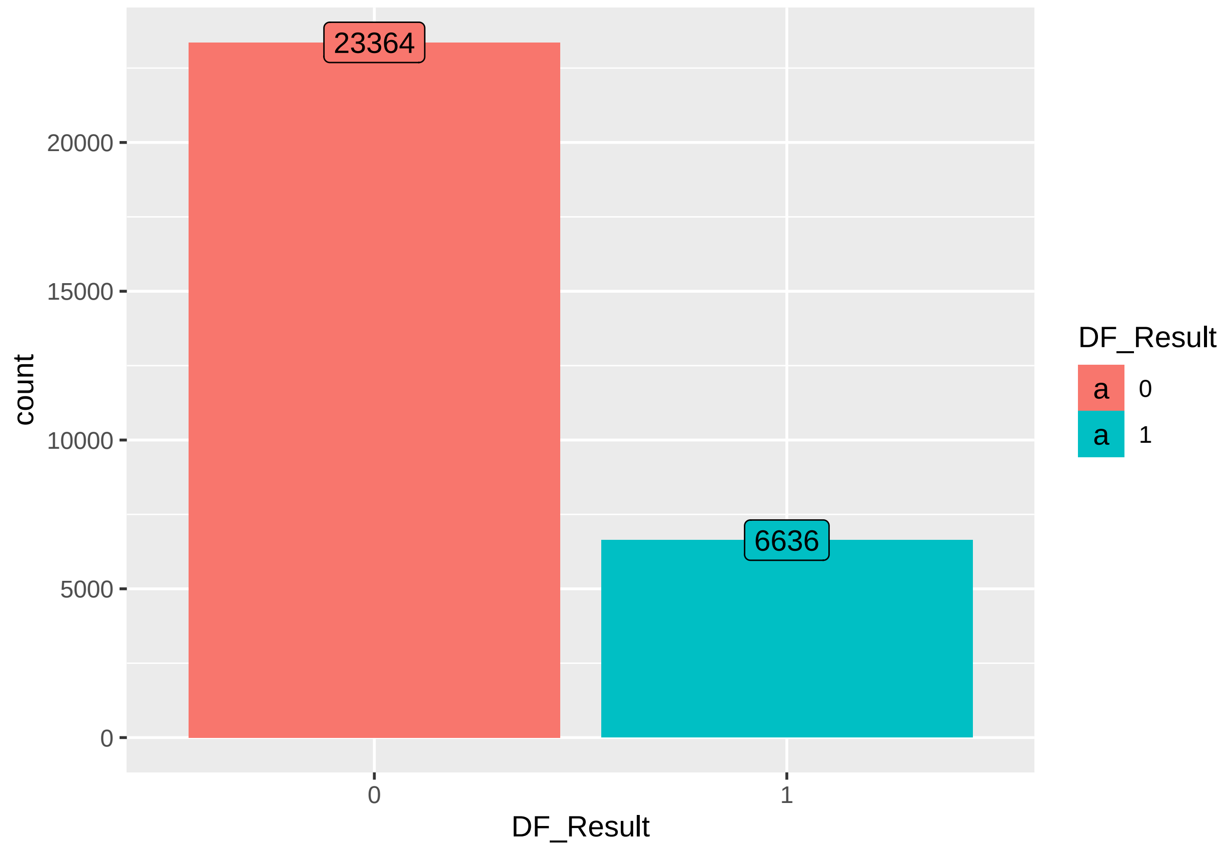
Looking closely at the target variable, it can be observed that there are disproportionately large incidents of "0's" (23,364 incidents) compared to "1's" (6,636 incidents), where 0 means no default payment, and 1 meants default payment.

```
##        ID LIMIT_BAL       SEX EDUCATION  MARRIAGE       AGE     PAY_1     PAY_2
##     30000        81         2         4         3        56        11        11
##     PAY_3     PAY_4     PAY_5     PAY_6 BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4
##        11        11        10        10     22723     22346     22026     21548
## BILL_AMT5 BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6
##     21010     20604      7943      7899      7518      6937      6897      6939
## DF_Result    GENDER
##         2         2
```



To show further details, the table and the graph below was plotted to display the exact percentage and count distribution of the target variable.

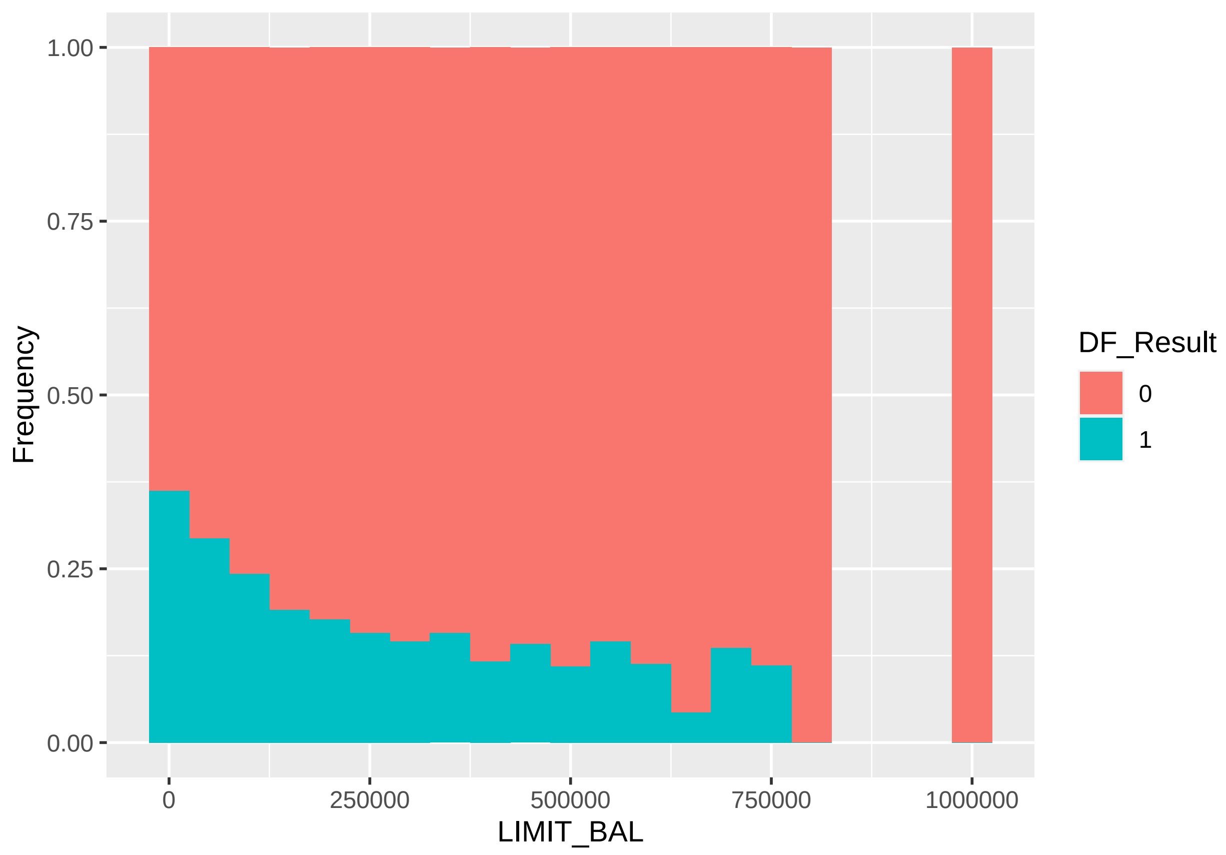
```
##
##      0      1
## 0.7788 0.2212
```



There are 78% of 0's and 22% of 1's, and this indicates that the dataset is unevenly balanced. This unbalance will be addressed later when splitting the dataset into test and train.

Now, features will be observed in detail one by one.

```
## Warning: Removed 6 rows containing missing values (geom_bar).
```



Looking at each selected feature individually, starting with the credit limit attribute, the correlation trend in this graph shows that credit card holders with a higher credit limit are less likely to have credit default. This seems logical, because banks would normally approve high credit limits to people with a stable background, such as high income, and good financial track record that gives banks the confidence of the person's payment ability.

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

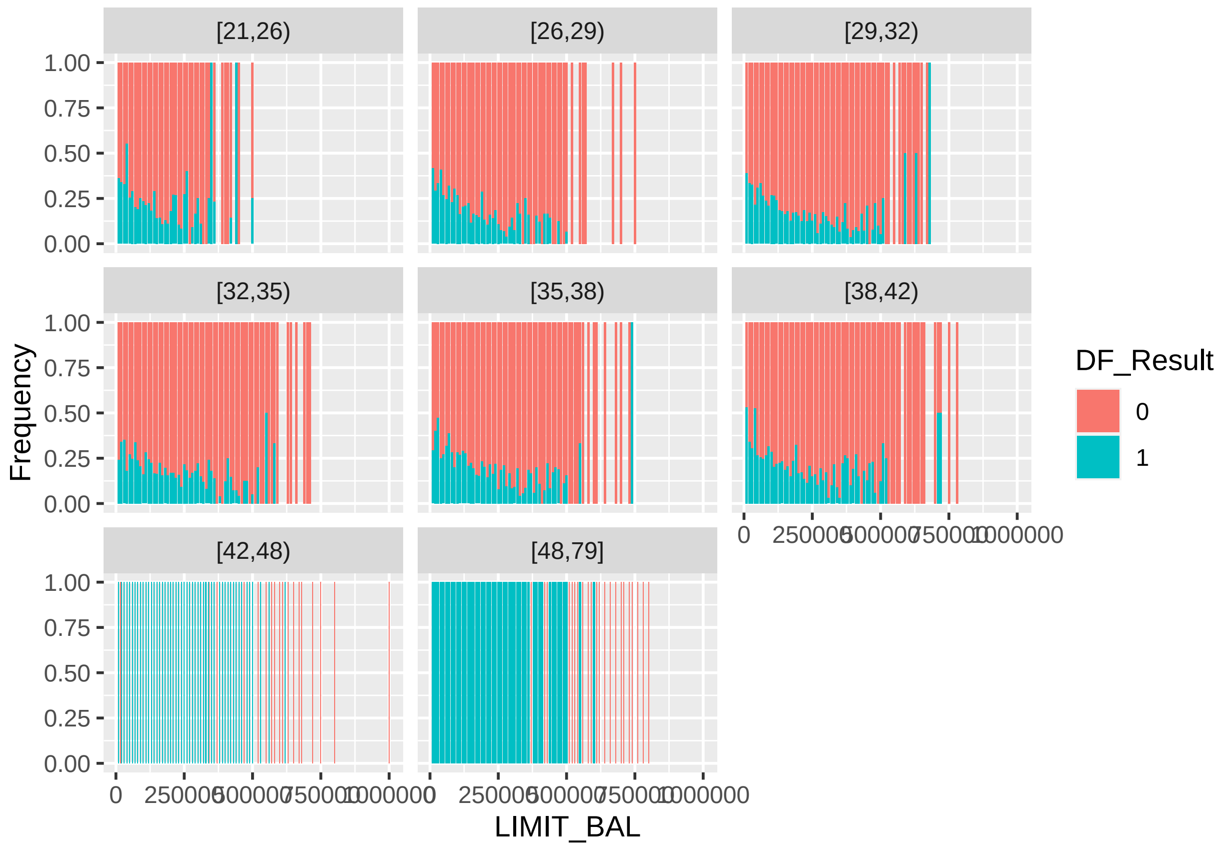
```
## Loading required package: Formula
```

```
## 
## Attaching package: 'Hmisc'
```

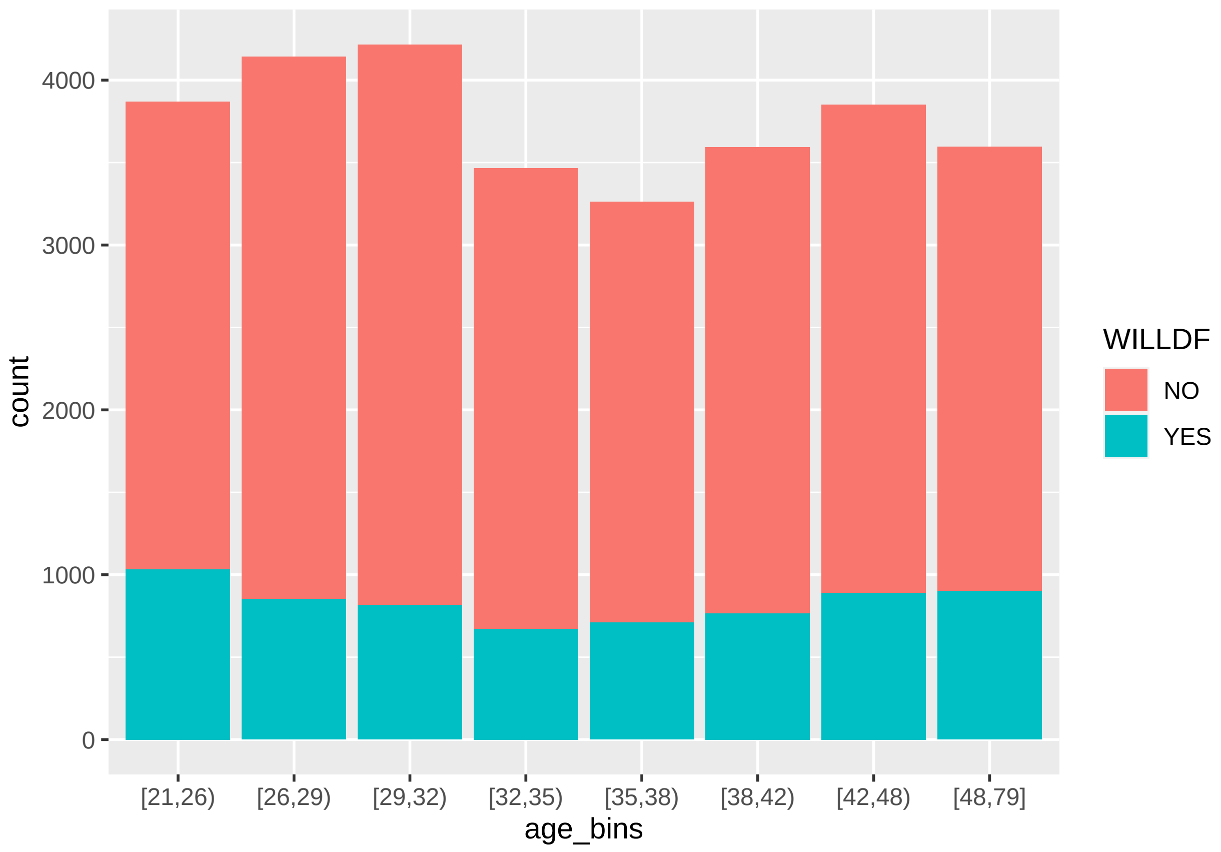
```
## The following objects are masked from 'package:base':
## 
##     format.pval, units
```

```
## Warning: position_stack requires non-overlapping x intervals
```

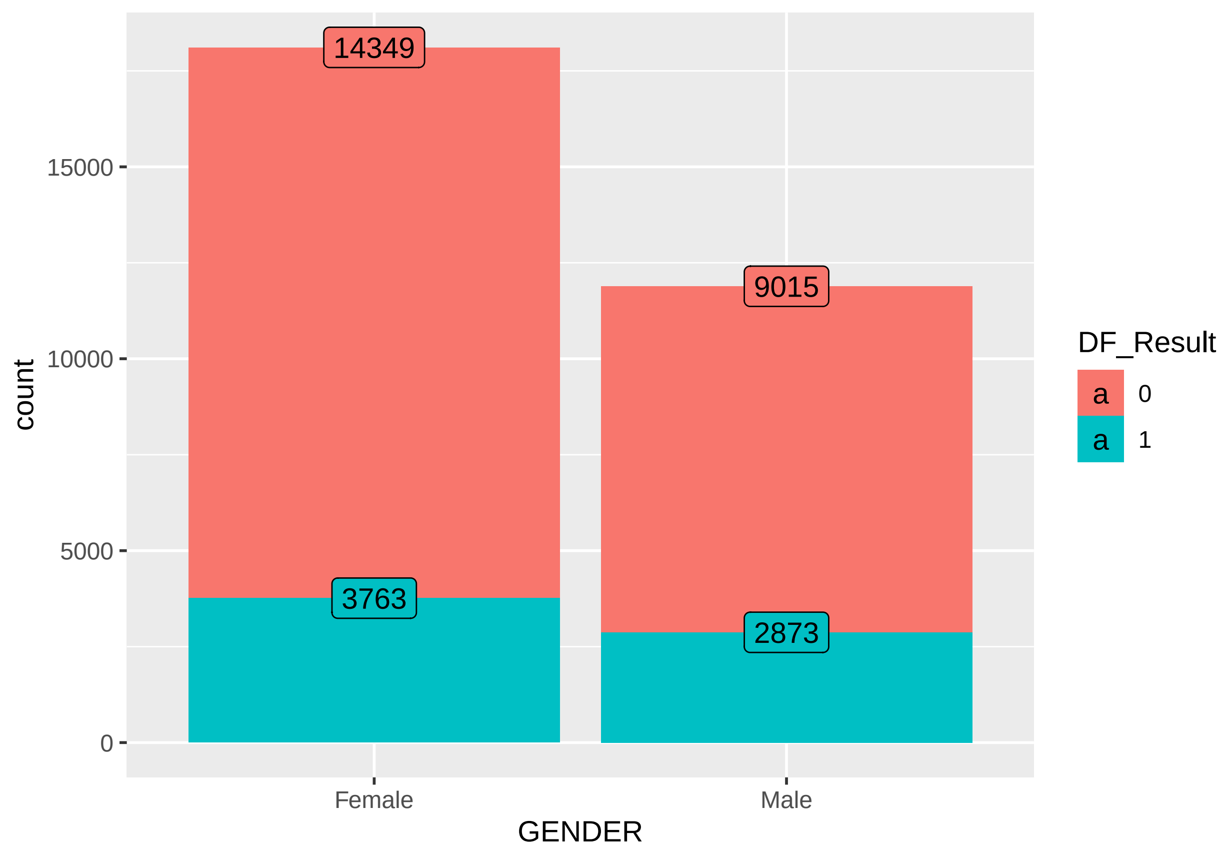
```
## Warning: position_stack requires non-overlapping x intervals
```



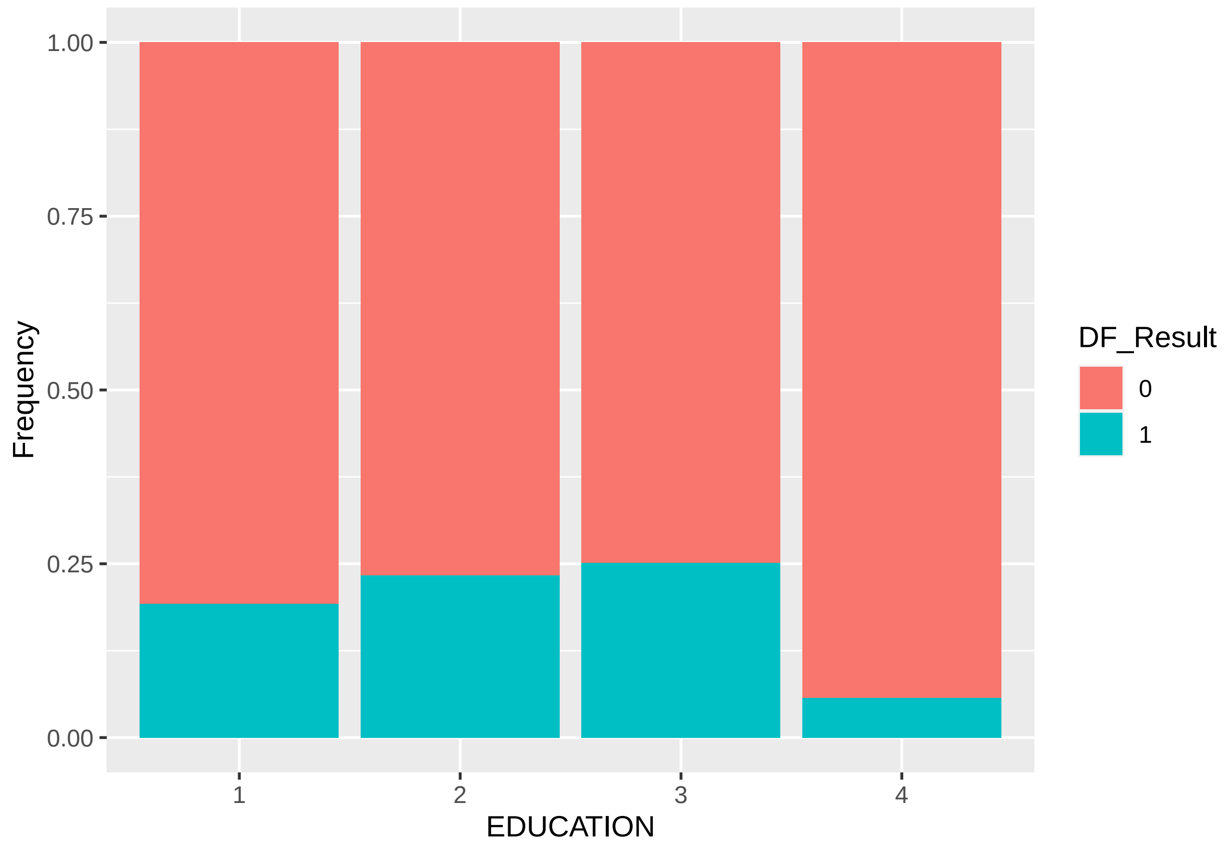
The dataset would replicate one of the common trends in credit card limit: credit card banks issue a lower credit limit to younger people, and older people tend to have a higher credit card limit. Thus, credit limit was plotted against the target variable, and was divided into 8 plots of different age groups. The graph confirms that younger people indeed do have a lower credit limit, and a high credit limit is usually held by credit card users in the higher age group.



Also, it was interesting to see that the proportion of the credit card users with credit default goes down from 20's to early 30's then rises again. The highest proportion of users with credit default can be seen among the group of people with age 21-26. This seems logical, because people in their early twenties are usually studying in universities or just at the beginning stage of their career. The default rate decreases by a user reaches early 30s i.e. by the time career is stabilized. Then, the proportion of credit card holders with default credit increases from mid-thirties and onward, and this also seems logical, as it can be explained with the trend where people getting married, starting a family, and having much more added responsibilities, such as bills and housing payments.



Looking at the gender attribute, it can be observed that men are more likely to have default payment than women.

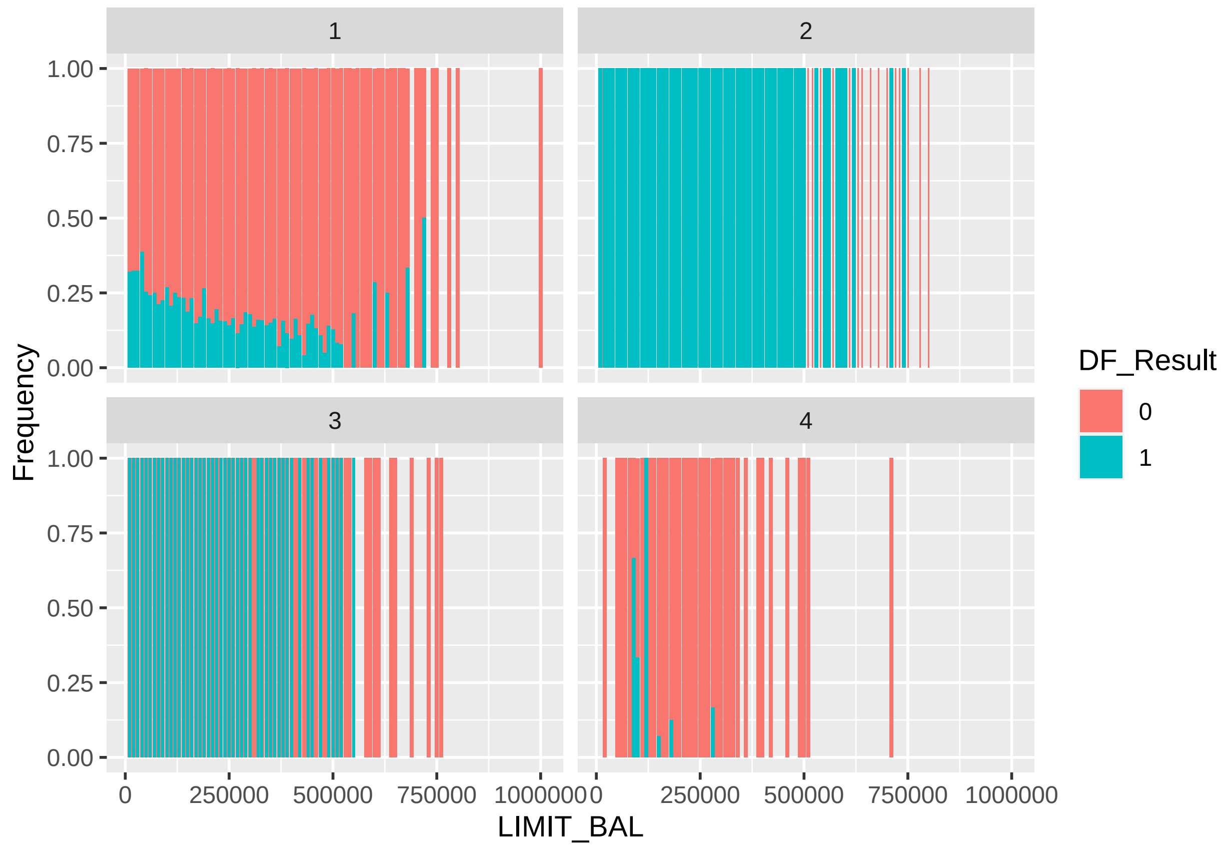


Looking at the education attribute from the above graph, it can be observed that the clients who have obtained only the university or high-school education tend to default more often than the clients with graduate degrees."Other" category is not defined, and it had the lowest proportion of credit card users with default payment. This could be due to the "Other" category combining clients with very high education, such as PhD or professional degrees, and customers with no education.

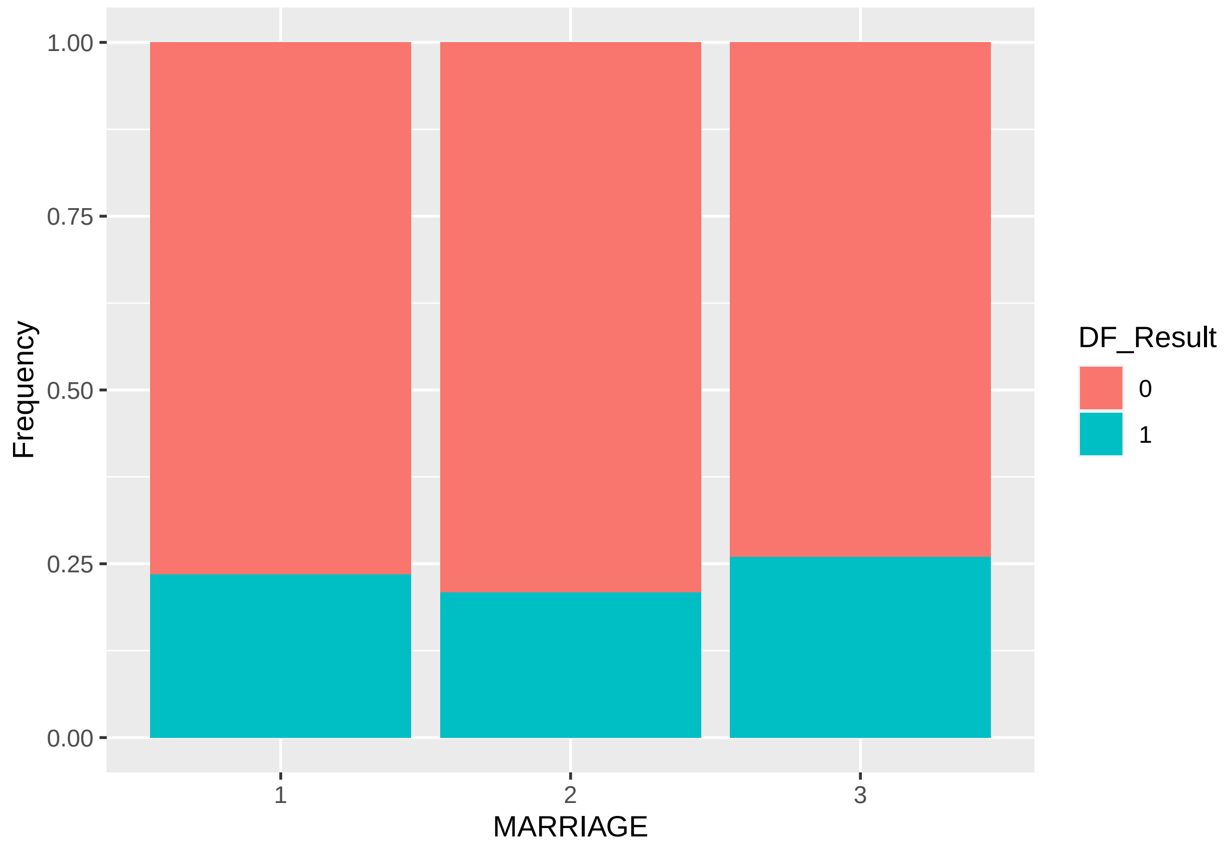
To see whether our assumption that the "Other" category in education has clients with very high education, such as PhD, the relationship between education and credit limit was observed.

```
## Warning: position_stack requires non-overlapping x intervals

## Warning: position_stack requires non-overlapping x intervals
```

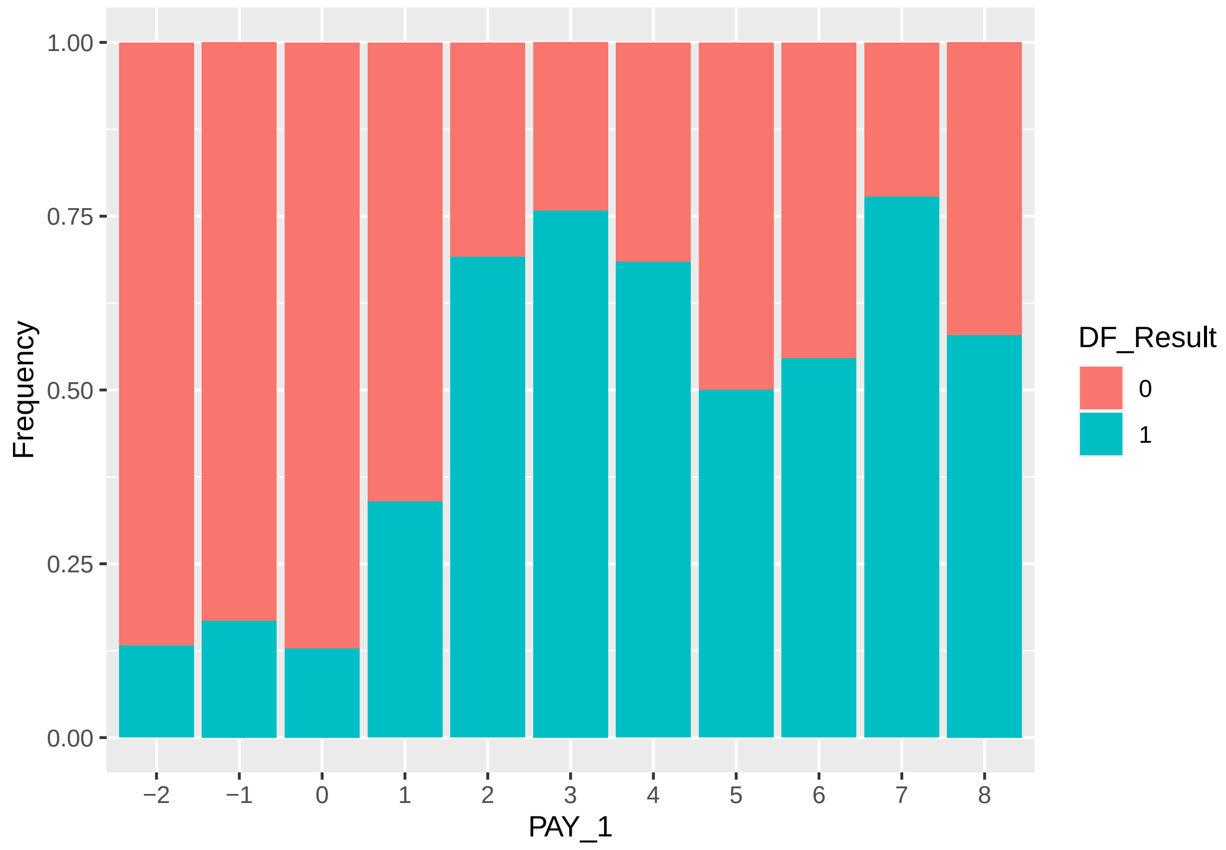


The above graph plots credit limit amount and the target variable, and divides into 4 categories, where “1” is graduate degree, “2” is university degree, “3” is high school, and “4” is other. In here, we can see that trends between “1” and “4” are similar, and between “2” and “3” are similar. Essentially, plots indicate that people with university or other education are more likely to pay credit card bills, and avoid credit default, regardless of credit limit, while people with university and high school degrees are especially likely to default when they have low credit limit as well. Due to the similarity of the “Other” category to the “Graduate degree”, but with even lower likelihood of credit default, the assumption can be made with certain confidence that “Other” category in education is likely to have clients with education degrees that are higher and more advanced than the graduate degrees.

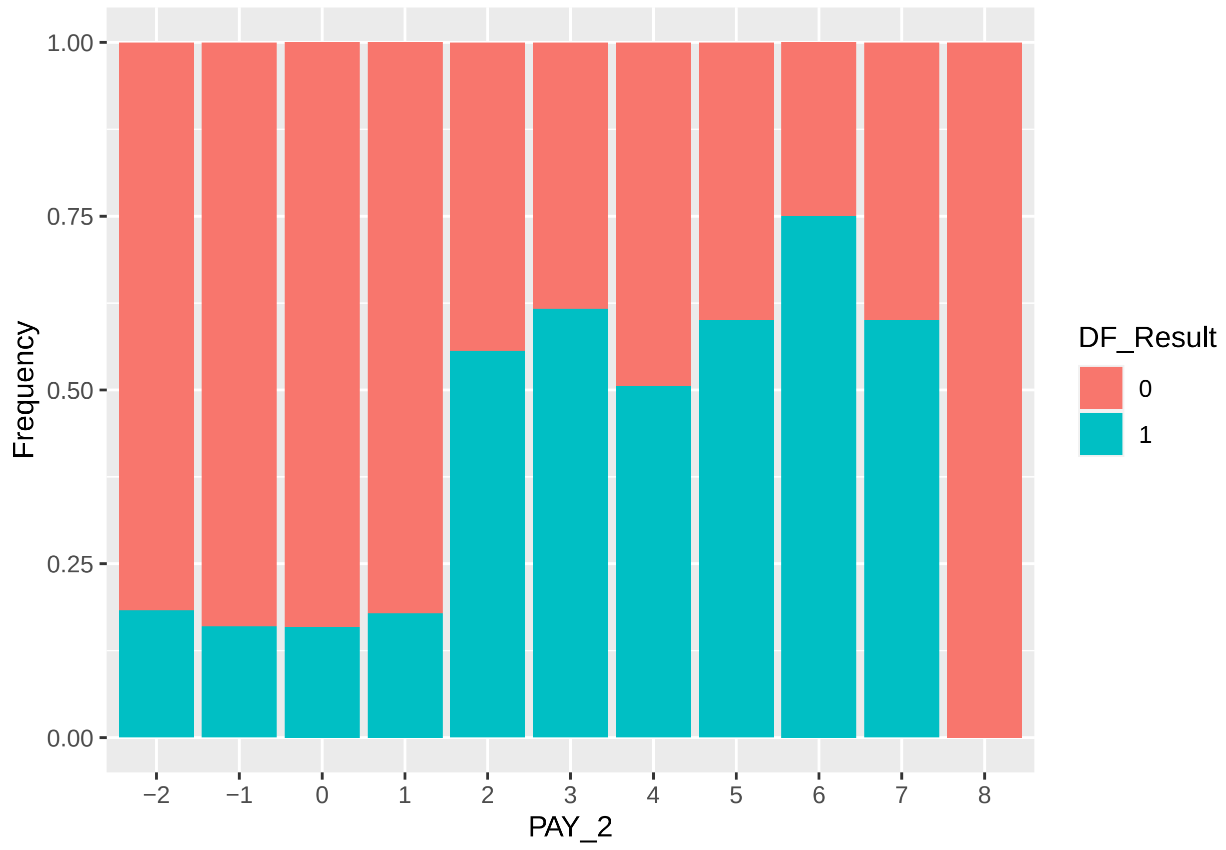


Looking at the marital status plot against the target variable, married people tend to default a little more often than single. This may be due to married people having more responsibilities and bills than single people. The "Other" category had the highest default rate. This category might represent those who are divorced or widowed, and who may have the similar responsibilities and bills of married people, but do not have partners to share the bill.

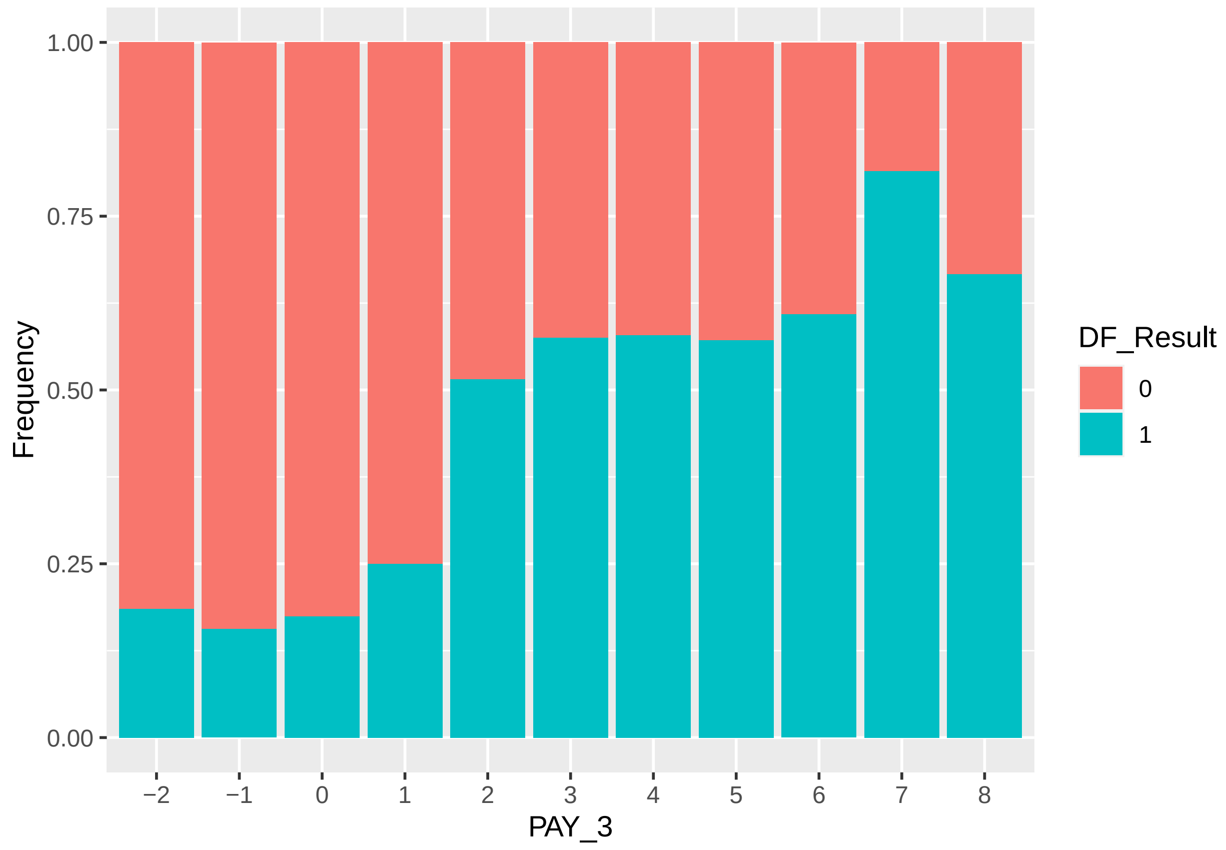
In addition, greater delays in credit card bill repayment (PAY_1~PAY_6 columns) was positively correlated with the target variable, the credit default. For this visual, new column was created to pull the averages from the 6 columns, from PAY_1 to PAY_6, and that new column was plotted against the target variable.

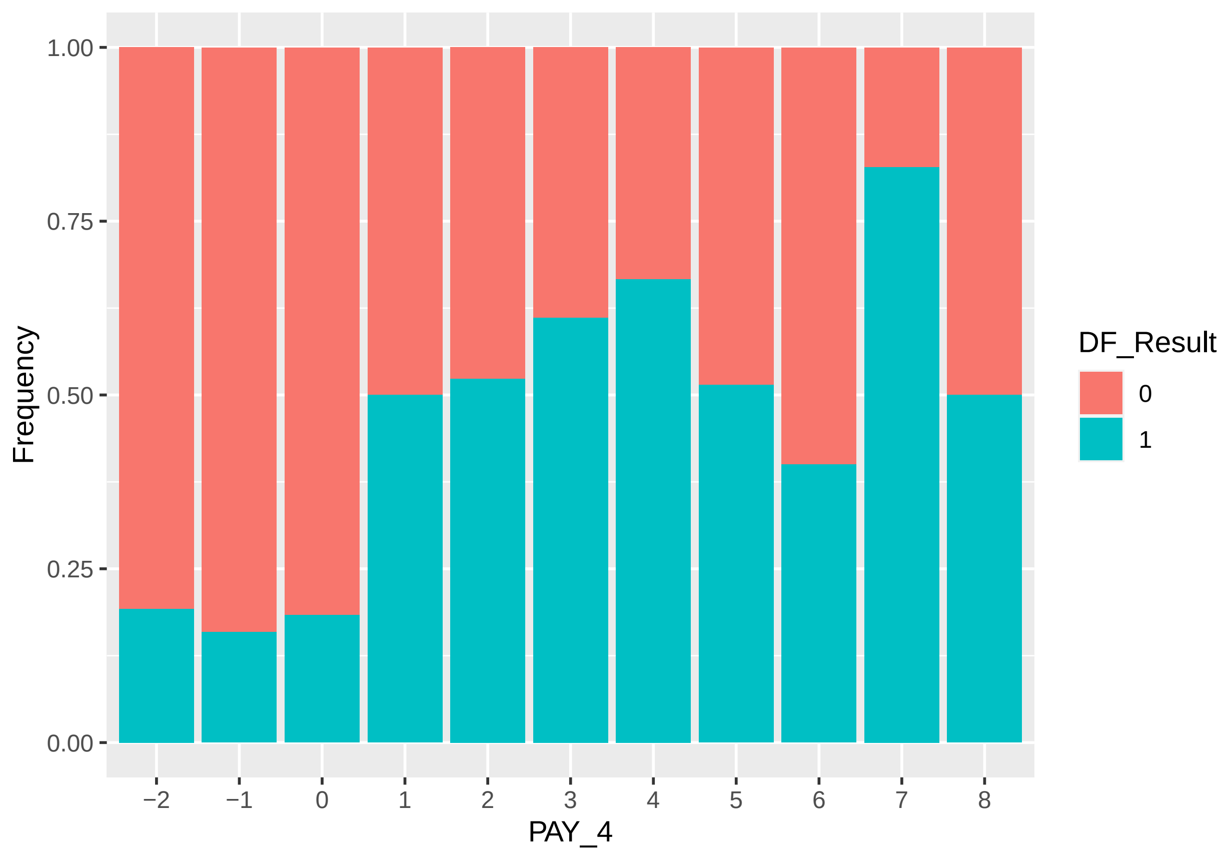


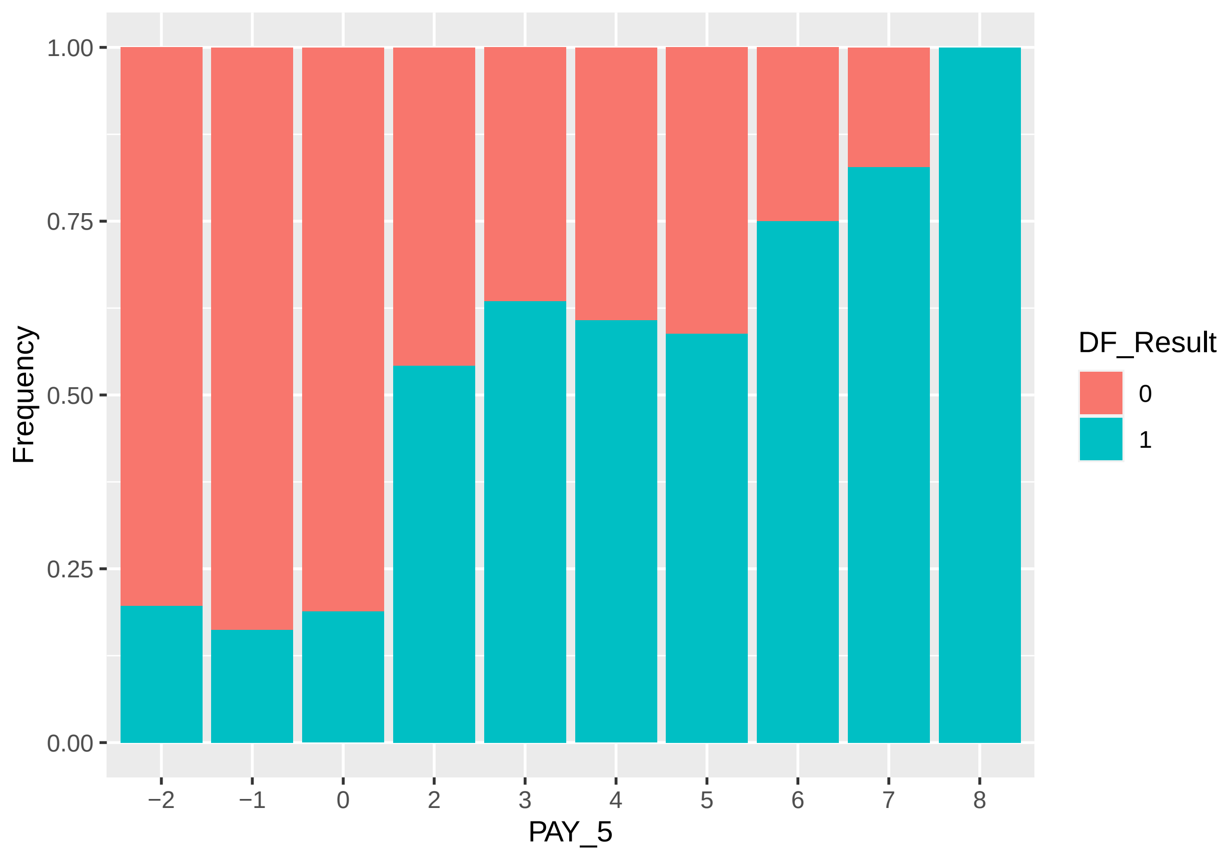
1.00
0.75
0.50
0.25
0.00
Frequency
−2
−1
0
1
2
3
4
5
6
7
8
PAY_1
DF_Result
0
1

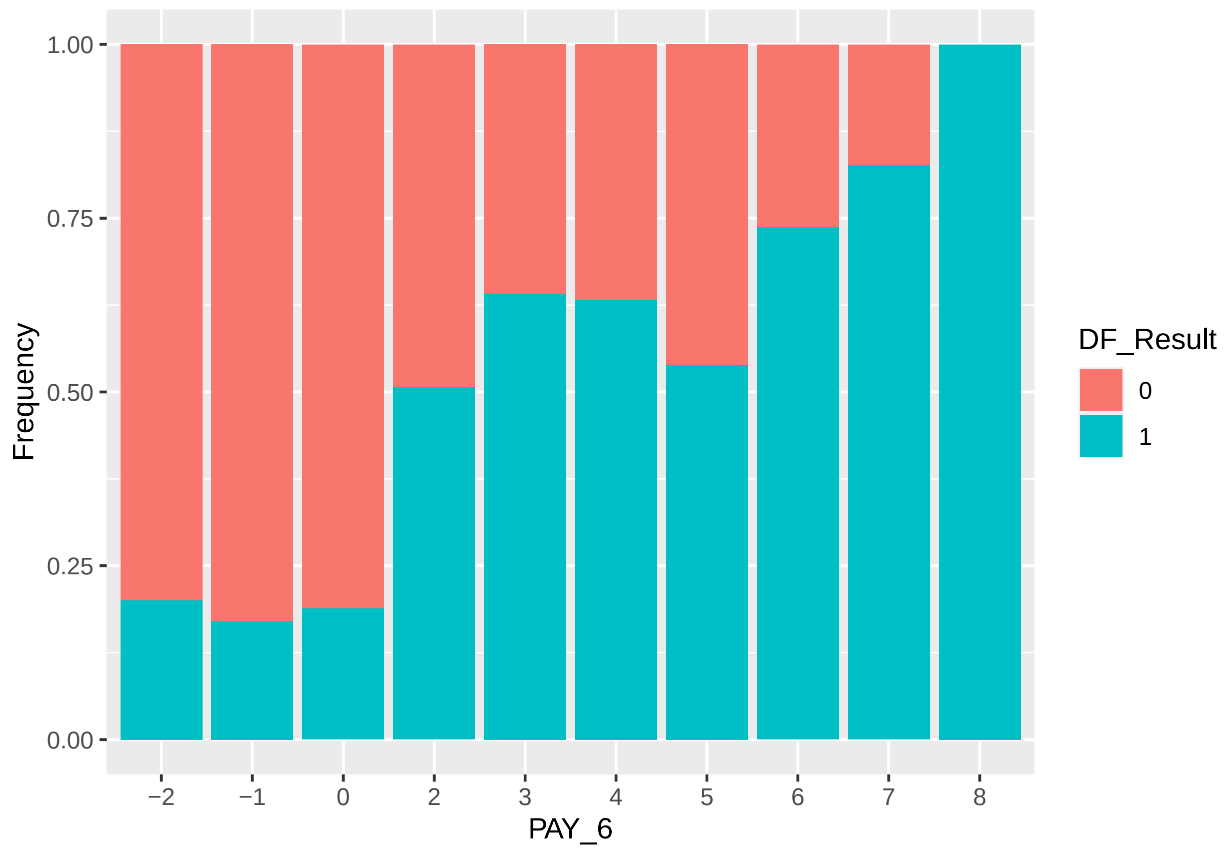


1.00
0.75
0.50
0.25
0.00
Frequency
−2
−1
0
1
2
3
4
5
6
7
8
PAY_2
DF_Result
0
1


1.00
0.75
0.50
0.25
0.00
Frequency
−2
−1
0
1
2
3
4
5
6
7
8
PAY_3
DF_Result
0
1


1.00
0.75
0.50
0.25
0.00
Frequency
−2
−1
0
1
2
3
4
5
6
7
8
PAY_4
DF_Result
0
1

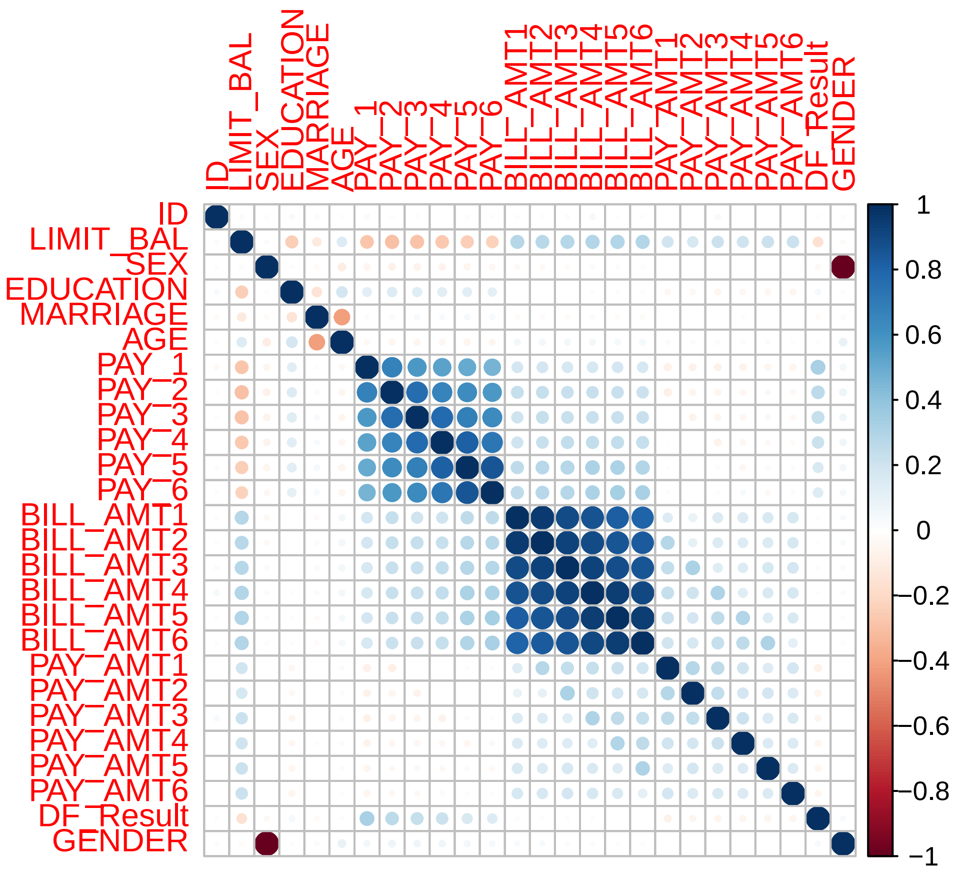

Frequency
1.00
0.75
0.50
0.25
0.00
−2
−1
0
2
3
4
5
6
7
8
PAY_5
DF_Result
0
1



Thus far, correlation of the variables were observed against the target variable.

To observe the relationship of variables to each other, a correlation plot was used.

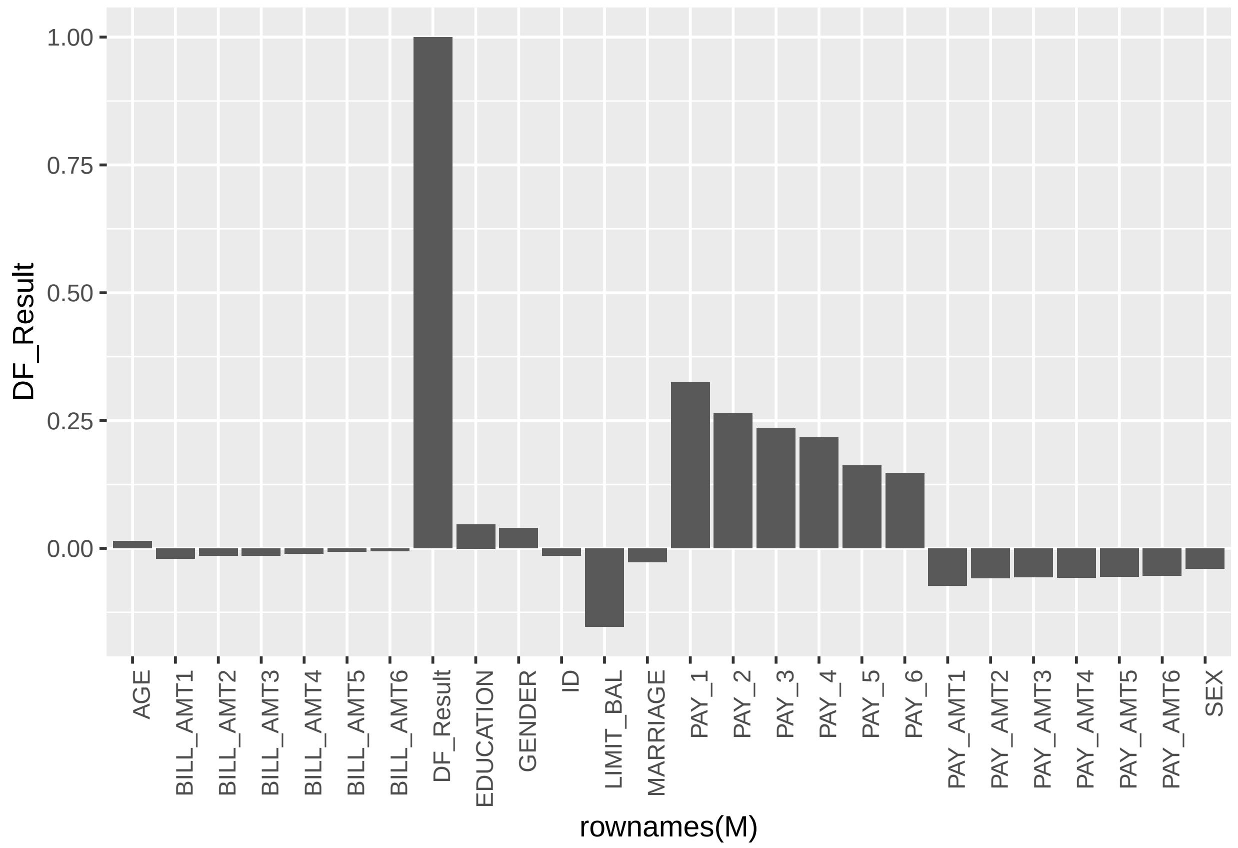
```
## corrplot 0.84 loaded
```



Looking at the graph, it looks like there is a strong correlation between the bill amount statuses in different months. The graph indicates that if a credit card user has had a high bill statement amount, the user is likely to have a high bill statement amount in the next months, which is explained by a credit card user's consistent spending behavior. Also, the graph shows that a credit card user who paid late is likely to pay late again in the next months, which also supports a consistent payment delay habit of credit card users. The next strongest correlation in the plot seems to be between gender and marital status. By the slight red hue, it indicates negative correlation, meaning the higher the credit card users' age, the more likely they are to be married ("1" meant married, while "2" meant single).

Feature Selection

Correlation values were used to identify important features.



```
##    Attributes Correlation.coefficient
## 1   DF_Result                   1.000
## 2       PAY_1                   0.325
## 3       PAY_2                   0.264
## 4       PAY_3                   0.235
## 5       PAY_4                   0.217
## 6       PAY_5                   0.162
## 7       PAY_6                   0.148
## 8   EDUCATION                   0.047
## 9      GENDER                   0.040
## 10        AGE                   0.014
## 11  BILL_AMT6                  -0.005
## 12  BILL_AMT5                  -0.007
## 13  BILL_AMT4                  -0.010
## 14  BILL_AMT3                  -0.014
## 15  BILL_AMT2                  -0.014
## 16  BILL_AMT1                  -0.020
## 17   MARRIAGE                  -0.027
## 18   PAY_AMT6                  -0.053
## 19   PAY_AMT5                  -0.055
## 20   PAY_AMT3                  -0.056
## 21   PAY_AMT4                  -0.057
## 22   PAY_AMT2                  -0.059
## 23   PAY_AMT1                  -0.073
## 24  LIMIT_BAL                  -0.154
```

The correlation values were obtained by using cor() function. From observing the correlation table, features with a very weak correlation strength (between -0.02 and 0.02) were identified. Other than all the variables shown in this table, there were no other features that could be added or created from existing features. It was decided that any features beyond -0.02 and 0.02 correlation value will not be included in the final dataset.

To summarize, the following features were selected to be put into the prediction models later on. * Credit limit * Gender * Eduation level * Marital status * Repayment status in September 2005 * Repayment status in August 2005 * Repayment status in July 2005 * Repayment status in June 2005 * Repayment status in May 2005 * Repayment status in April 2005 * Bill statement amount in September 2005 * Bill payment amount in September 2005 * Bill payment amount in August 2005 * Bill payment amount in July 2005 * Bill payment amount in June 2005 * Bill payment amount in May 2005 * Bill payment amount in April 2005

Preparation

Missing values

The dataset had no NA value or blank value.

```
dataraw <- read.csv(file = 'creditcarddata.csv',stringsAsFactors = FALSE)
colSums(is.na(dataraw))
```

```
##        ID LIMIT_BAL       SEX EDUCATION  MARRIAGE       AGE     PAY_0     PAY_2
##         0         0         0         0         0         0         0         0
##     PAY_3     PAY_4     PAY_5     PAY_6 BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4
##         0         0         0         0         0         0         0         0
## BILL_AMT5 BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6
##         0         0         0         0         0         0         0         0
##         Y
##         0
```

```
colSums(dataraw=="")
```

```
##        ID LIMIT_BAL       SEX EDUCATION  MARRIAGE       AGE     PAY_0     PAY_2
##         0         0         0         0         0         0         0         0
##     PAY_3     PAY_4     PAY_5     PAY_6 BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4
##         0         0         0         0         0         0         0         0
## BILL_AMT5 BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6
##         0         0         0         0         0         0         0         0
##         Y
##         0
```

However, the dataset was not very clean as there were many unexplained data embedded throughout the dataset.

Yeh & Lien 2009 paper mentioned that the education attribute had 4 values, from 1 being a graduate degree, 2 being university, 3 being high school, and 4 being others. However, the education column had 7 values equal to 0, 5, and 6, description of which is missing in the source.

```
colSums(dataraw["EDUCATION"]==0 | dataraw["EDUCATION"]==5 | dataraw["EDUCATION"]==6)
```

```
## EDUCATION
##       345
```

Marital status column had the same issue. 1 meant married, 2 meant single, and 3 meant "Others" according to Yeh & Lien 2009 paper. However, there was a value "0" in the column, which was not explained.

```
colSums(dataraw["MARRIAGE"]==0)
```

```
## MARRIAGE
##       54
```

Therefore, rows containing values equal to 0, 5 and 6 for Education column as well as rows containing values equal to 0 in Marriage Column will be replaced by the median of the column correspondingly

```
datadump <- read.csv(file = 'creditcarddata.csv',stringsAsFactors = FALSE)

datadump$EDUCATION[datadump$EDUCATION == 5 |
                     datadump$EDUCATION == 6 |
                     datadump$EDUCATION == 0] <-median(datadump$EDUCATION)
datadump$MARRIAGE[datadump$MARRIAGE == 0] <- median(datadump$MARRIAGE)
```

There were minor changes applied to the initial dataset. Column PAY_0 was renamed to PAY_1 in order to keep the consistency with other repayment columns. Also, the target column was renamed from "default.payment.nextmonth" to "DF_Result" for convenience while coding.

```
colnames(datadump)[colnames(datadump) == "PAY_0"] = "PAY_1"
colnames(datadump)[colnames((datadump)) == "Y"] = "DF_Result"
datadump$GENDER = ifelse(datadump$SEX == 1, "Male", "Female")
```

Moreover, discrete variables that have specific set of values were categorized

```
#Change datatypes of particular columns to factor
apply(datadump,2, function(x) length(unique(x)))
```

```
##        ID LIMIT_BAL       SEX EDUCATION  MARRIAGE       AGE     PAY_1     PAY_2
##     30000        81         2         4         3        56        11        11
##     PAY_3     PAY_4     PAY_5     PAY_6 BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4
##        11        11        10        10     22723     22346     22026     21548
## BILL_AMT5 BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6
##     21010     20604      7943      7899      7518      6937      6897      6939
## DF_Result    GENDER
##         2         2
```

```
cols<-c("SEX","EDUCATION","MARRIAGE","PAY_1","PAY_2","PAY_3","PAY_4","PAY_5","PAY_6","DF_Result","GENDE
for (i in cols){
  datadump[,i] <- as.factor(datadump[,i])
}
```

Outliers

The numeric variables as credit limit, bill statement amount, and bill payment amount were observed for outliers by calculating IQR for each of them and identifying lower and upper bound.

First, the bill statement amount in September 2005 had 2,400 outliers.

```
#Removing insignificant columns
datadump=datadump[,-c(1,6,14,15,16,17,18,26)]

# Outlier calculation for bill statement amount in September 2005
Q1 = quantile(datadump$BILL_AMT1, 0.25, na.rm=TRUE)
Q3 = quantile(datadump$BILL_AMT1, 0.75, na.rm=TRUE)
IQR=IQR(datadump$BILL_AMT1, na.rm=TRUE)
UL=Q3+1.5*IQR
LL=Q1-1.5*IQR
off_vals<-datadump$BILL_AMT1[datadump$BILL_AMT1>=UL | datadump$BILL_AMT1<=LL]
summary(datadump$BILL_AMT1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -165580    3559   22382   51223   67091  964511
```

The bill payment amount in September, August, July, June, May, and April had 2745, 2714, 2598, 2994, 2946, and 2958, respectively.

```
# Outlier calculation for bill payment in September
Q1 = quantile(datadump$PAY_AMT1, 0.25, na.rm=TRUE)
Q3 = quantile(datadump$PAY_AMT1, 0.75, na.rm=TRUE)
IQR=IQR(datadump$PAY_AMT1, na.rm=TRUE)
UL=Q3+1.5*IQR
LL=Q1-1.5*IQR
off_vals<-datadump$PAY_AMT1[datadump$PAY_AMT1>=UL | datadump$PAY_AMT1<=LL]
summary(datadump$PAY_AMT1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0    1000    2100    5664    5006  873552
```

```
# Outlier calculation for bill payment in August
Q1 = quantile(datadump$PAY_AMT2, 0.25, na.rm=TRUE)
Q3 = quantile(datadump$PAY_AMT2, 0.75, na.rm=TRUE)
IQR=IQR(datadump$PAY_AMT2, na.rm=TRUE)
UL=Q3+1.5*IQR
LL=Q1-1.5*IQR
off_vals<-datadump$PAY_AMT2[datadump$PAY_AMT2>=UL | datadump$PAY_AMT2<=LL]
summary(datadump$PAY_AMT2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0     833    2009    5921    5000 1684259
```

```
# Outlier calculation for bill payment in July
Q1 = quantile(datadump$PAY_AMT3, 0.25, na.rm=TRUE)
Q3 = quantile(datadump$PAY_AMT3, 0.75, na.rm=TRUE)
IQR=IQR(datadump$PAY_AMT3, na.rm=TRUE)
UL=Q3+1.5*IQR
LL=Q1-1.5*IQR
off_vals<-datadump$PAY_AMT3[datadump$PAY_AMT3>=UL | datadump$PAY_AMT3<=LL]
summary(datadump$PAY_AMT3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0     390    1800    5226    4505  896040
```

```
# Outlier calculation for bill payment in June
Q1 = quantile(datadump$PAY_AMT4, 0.25, na.rm=TRUE)
Q3 = quantile(datadump$PAY_AMT4, 0.75, na.rm=TRUE)
IQR=IQR(datadump$PAY_AMT4, na.rm=TRUE)
UL=Q3+1.5*IQR
LL=Q1-1.5*IQR
off_vals<-datadump$PAY_AMT4[datadump$PAY_AMT4>=UL | datadump$PAY_AMT4<=LL]
summary(datadump$PAY_AMT4)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0     296    1500    4826    4013  621000
```

```
# Outlier calculation for bill payment in May
Q1 = quantile(datadump$PAY_AMT5, 0.25, na.rm=TRUE)
Q3 = quantile(datadump$PAY_AMT5, 0.75, na.rm=TRUE)
IQR=IQR(datadump$PAY_AMT5, na.rm=TRUE)
UL=Q3+1.5*IQR
LL=Q1-1.5*IQR
off_vals<-datadump$PAY_AMT5[datadump$PAY_AMT5>=UL | datadump$PAY_AMT5<=LL]
summary(datadump$PAY_AMT5)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##      0.0    252.5   1500.0   4799.4   4031.5 426529.0
```

```
# Outlier calculation for bill payment in April
Q1 = quantile(datadump$PAY_AMT6, 0.25, na.rm=TRUE)
Q3 = quantile(datadump$PAY_AMT6, 0.75, na.rm=TRUE)
IQR=IQR(datadump$PAY_AMT6, na.rm=TRUE)
UL=Q3+1.5*IQR
LL=Q1-1.5*IQR
off_vals<-datadump$PAY_AMT6[datadump$PAY_AMT6>=UL | datadump$PAY_AMT6<=LL]
summary(datadump$PAY_AMT6)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##      0.0    117.8   1500.0   5215.5   4000.0 528666.0
```

The limit balance had 167 outliers.

```
# Outlier calculation for credit card limit
Q1 = quantile(datadump$LIMIT_BAL, 0.25, na.rm=TRUE)
Q3 = quantile(datadump$LIMIT_BAL, 0.75, na.rm=TRUE)
IQR=IQR(datadump$LIMIT_BAL, na.rm=TRUE)
UL=Q3+1.5*IQR
LL=Q1-1.5*IQR
off_vals<-datadump$LIMIT_BAL[datadump$LIMIT_BAL>=UL | datadump$LIMIT_BAL<=LL]
summary(datadump$LIMIT_BAL)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10000   50000  140000  167484  240000 1000000
```

In total, the outliers account for at least 10% of the whole dataset, and removing this much amount of data would have a negative effect on model performance and accuracy. It was decided that the disadvantage of taking out outliers would be greater than the disadvantage of keeping the outliers, and therefore, the outliers were kept in the dataset. Another reason why the outliers were kept was that they are considered to be natural, meaning credit limit, bill statements and pay amounts do vary from 10,000 to 1,700,000.

Transformation

The following table summarizes the range, mode, mean, median, standard deviation, skewness distribution, and kurtosis of the final dataset.

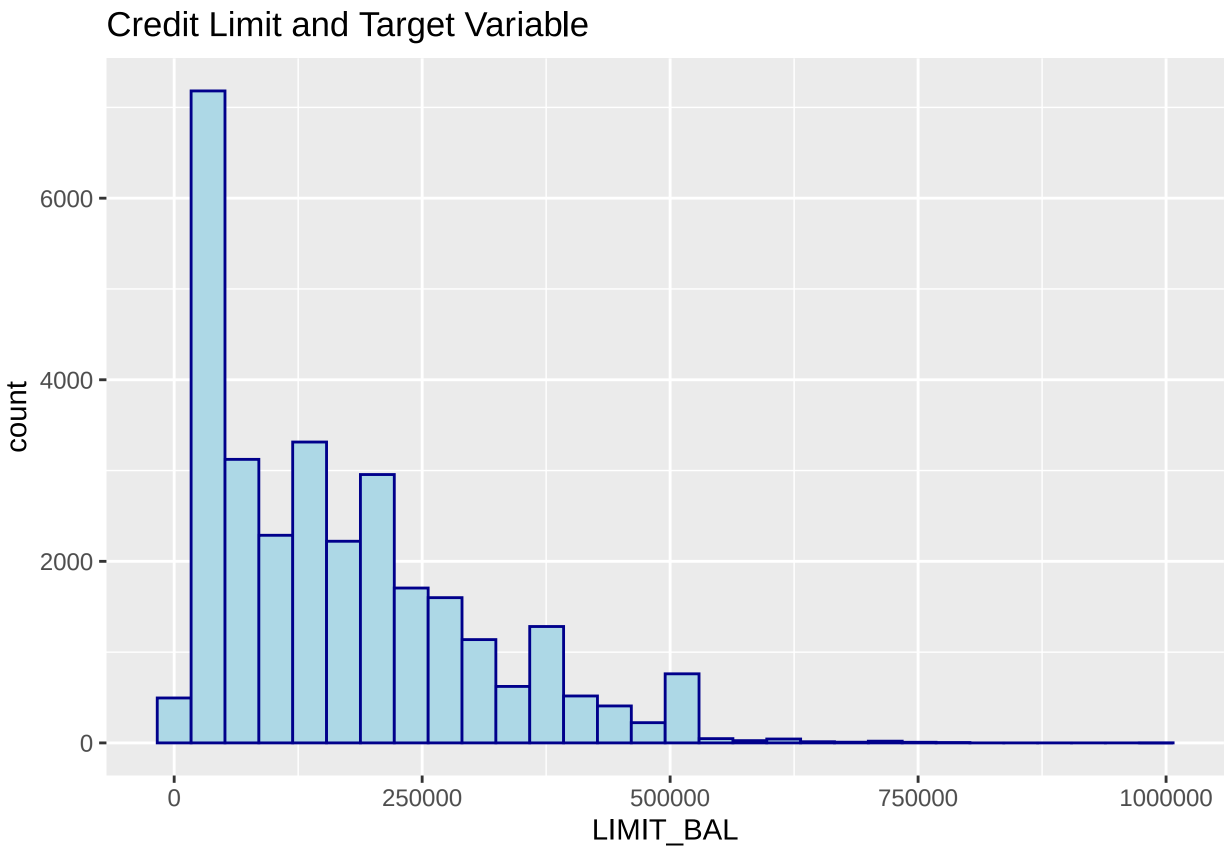
```
describe(datadump)
```

```
## datadump
##
##  18  Variables      30000  Observations
## ---------------------------------------------------------------------------
## LIMIT_BAL
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    30000        0       81    0.998   167484   141783    20000    30000
##      .25      .50      .75      .90      .95
##    50000   140000   240000   360000   430000
##
## lowest :   10000   16000   20000   30000   40000
## highest:  750000  760000  780000  800000 1000000
## ---------------------------------------------------------------------------
## SEX
##        n  missing distinct
##    30000        0        2
##
## Value          1     2
## Frequency  11888 18112
## Proportion 0.396 0.604
## ---------------------------------------------------------------------------
## EDUCATION
##        n  missing distinct
##    30000        0        4
##
## Value          1     2     3     4
## Frequency  10585 14375  4917   123
## Proportion 0.353 0.479 0.164 0.004
## ---------------------------------------------------------------------------
## MARRIAGE
##        n  missing distinct
##    30000        0        3
##
## Value          1     2     3
## Frequency  13659 16018   323
## Proportion 0.455 0.534 0.011
## ---------------------------------------------------------------------------
## PAY_1
##        n  missing distinct
##    30000        0       11
```

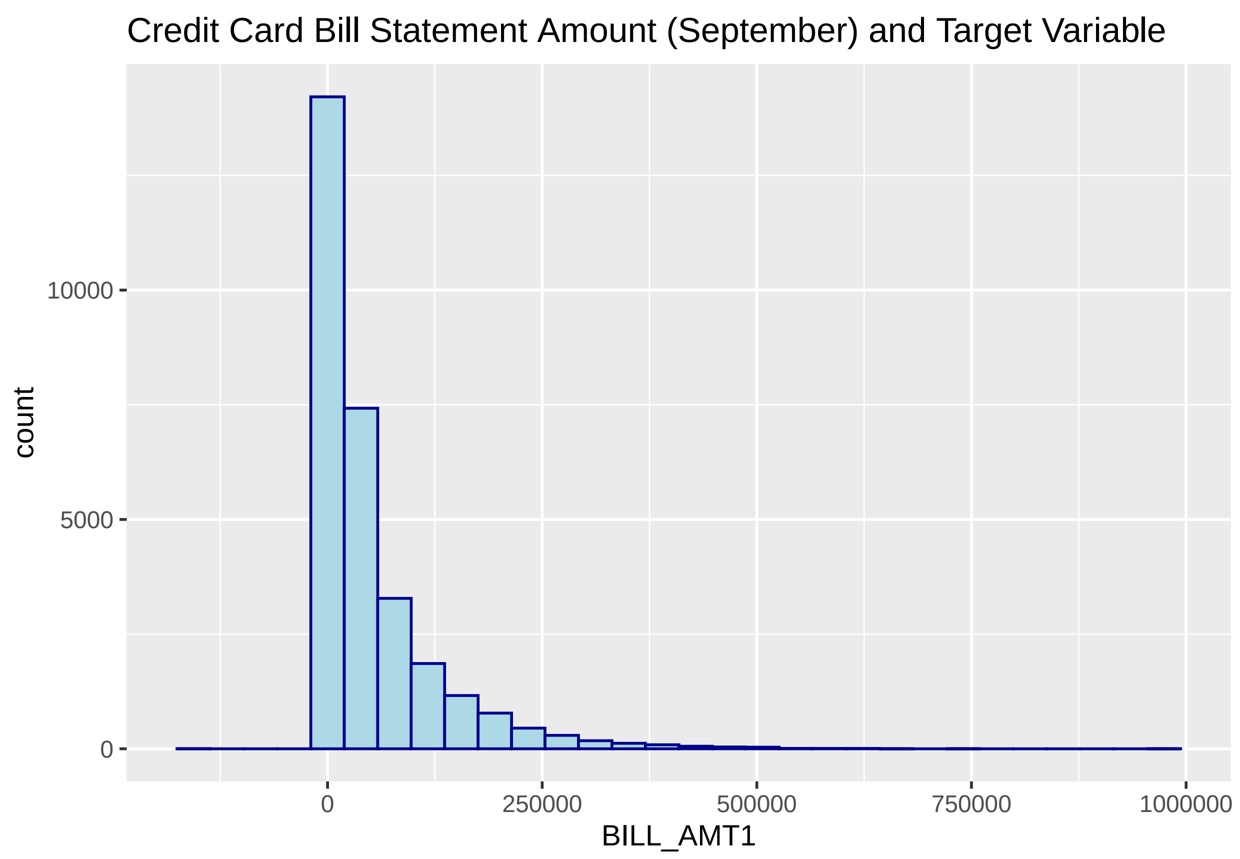
```
##
## lowest : -2 -1 0  1  2 , highest: 4  5  6  7  8
##
## Value         -2    -1     0     1     2     3     4     5     6     7     8
## Frequency   2759  5686 14737  3688  2667   322    76    26    11     9    19
## Proportion 0.092 0.190 0.491 0.123 0.089 0.011 0.003 0.001 0.000 0.000 0.001
## ---------------------------------------------------------------------------
## PAY_2
##        n  missing distinct
##    30000        0       11
##
## lowest : -2 -1 0  1  2 , highest: 4  5  6  7  8
##
## Value         -2    -1     0     1     2     3     4     5     6     7     8
## Frequency   3782  6050 15730    28  3927   326    99    25    12    20     1
## Proportion 0.126 0.202 0.524 0.001 0.131 0.011 0.003 0.001 0.000 0.001 0.000
## ---------------------------------------------------------------------------
## PAY_3
##        n  missing distinct
##    30000        0       11
##
## lowest : -2 -1 0  1  2 , highest: 4  5  6  7  8
##
## Value         -2    -1     0     1     2     3     4     5     6     7     8
## Frequency   4085  5938 15764     4  3819   240    76    21    23    27     3
## Proportion 0.136 0.198 0.525 0.000 0.127 0.008 0.003 0.001 0.001 0.001 0.000
## ---------------------------------------------------------------------------
## PAY_4
##        n  missing distinct
##    30000        0       11
##
## lowest : -2 -1 0  1  2 , highest: 4  5  6  7  8
##
## Value         -2    -1     0     1     2     3     4     5     6     7     8
## Frequency   4348  5687 16455     2  3159   180    69    35     5    58     2
## Proportion 0.145 0.190 0.548 0.000 0.105 0.006 0.002 0.001 0.000 0.002 0.000
## ---------------------------------------------------------------------------
## PAY_5
##        n  missing distinct
##    30000        0       10
##
## lowest : -2 -1 0  2  3 , highest: 4  5  6  7  8
##
## Value         -2    -1     0     2     3     4     5     6     7     8
## Frequency   4546  5539 16947  2626   178    84    17     4    58     1
## Proportion 0.152 0.185 0.565 0.088 0.006 0.003 0.001 0.000 0.002 0.000
## ---------------------------------------------------------------------------
## PAY_6
##        n  missing distinct
##    30000        0       10
##
## lowest : -2 -1 0  2  3 , highest: 4  5  6  7  8
##
## Value         -2    -1     0     2     3     4     5     6     7     8
## Frequency   4895  5740 16286  2766   184    49    13    19    46     2
## Proportion 0.163 0.191 0.543 0.092 0.006 0.002 0.000 0.001 0.002 0.000
## ---------------------------------------------------------------------------
## BILL_AMT1
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    30000        0    22723        1    51223    66644      0.0    278.9
##      .25      .50      .75      .90      .95
##   3558.8  22381.5  67091.0 142133.7 201203.0
##
## lowest : -165580 -154973  -15308  -14386  -11545
## highest:  626648  630458  653062  746814  964511
## ---------------------------------------------------------------------------
## PAY_AMT1
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    30000        0     7943    0.994     5664     7939        0        0
##      .25      .50      .75      .90      .95
##     1000     2100     5006    10300    18428
##
## lowest :      0      1      2      3      4, highest: 405016 423903 493358 505000 873552
## ---------------------------------------------------------------------------
## PAY_AMT2
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    30000        0     7899    0.994     5921     8602        0        0
##      .25      .50      .75      .90      .95
##      833     2009     5000    10401    19004
##
## lowest :       0       1       2       3       4
## highest:  580464 1024516 1215471 1227082 1684259
## ---------------------------------------------------------------------------
## PAY_AMT3
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    30000        0     7518    0.992     5226     7760        0        0
##      .25      .50      .75      .90      .95
##      390     1800     4505    10000    17589
##
## lowest :      0      1      2      3      4, highest: 400972 417588 508229 889043 896040
## ---------------------------------------------------------------------------
## PAY_AMT4
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    30000        0     6937     0.99     4826     7300        0        0
##      .25      .50      .75      .90      .95
##      296     1500     4013     9571    16015
##
## lowest :      0      1      2      3      4, highest: 400046 432130 497000 528897 621000
## ---------------------------------------------------------------------------
## PAY_AMT5
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    30000        0     6897    0.989     4799     7246      0.0      0.0
##      .25      .50      .75      .90      .95
##    252.5   1500.0   4031.5   9500.0  16000.0
##
## lowest :      0      1      2      3      4, highest: 332000 379267 388071 417990 426529
## ---------------------------------------------------------------------------
## PAY_AMT6
##         n  missing distinct     Info     Mean      Gmd      .05      .10
##     30000        0     6939    0.986     5216     8114      0.0      0.0
##       .25      .50      .75      .90      .95
##     117.8   1500.0   4000.0   9600.0  17343.8
##
## lowest :      0      1      2      3      4, highest: 403500 422000 443001 527143 528666
## --------------------------------------------------------------------------------
## DF_Result
##         n  missing distinct
##     30000        0        2
##
## Value           0     1
## Frequency  23364  6636
## Proportion 0.779 0.221
## --------------------------------------------------------------------------------
```

The following histograms confirm that the shapes are positively skewed for limit balance, bill statement amount in September, and bill payment amounts in April~September.

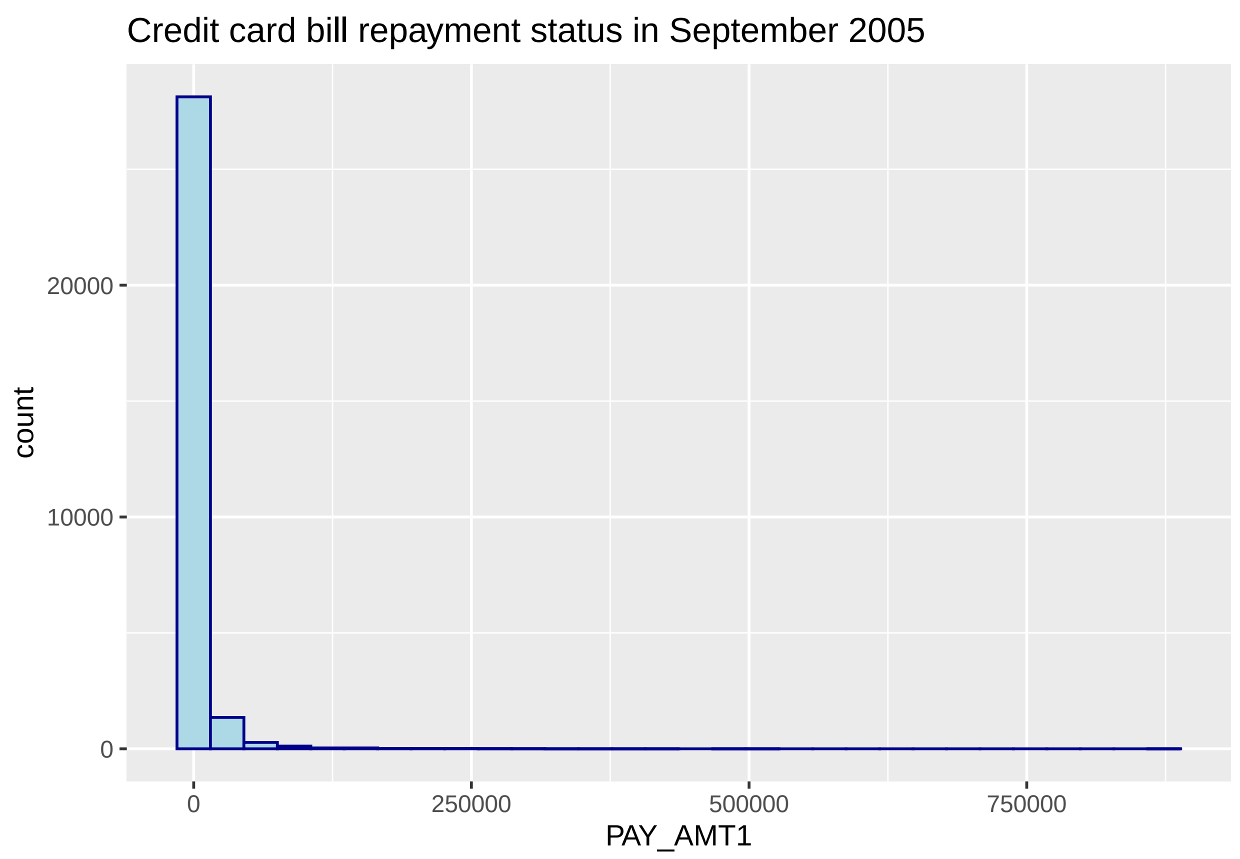
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



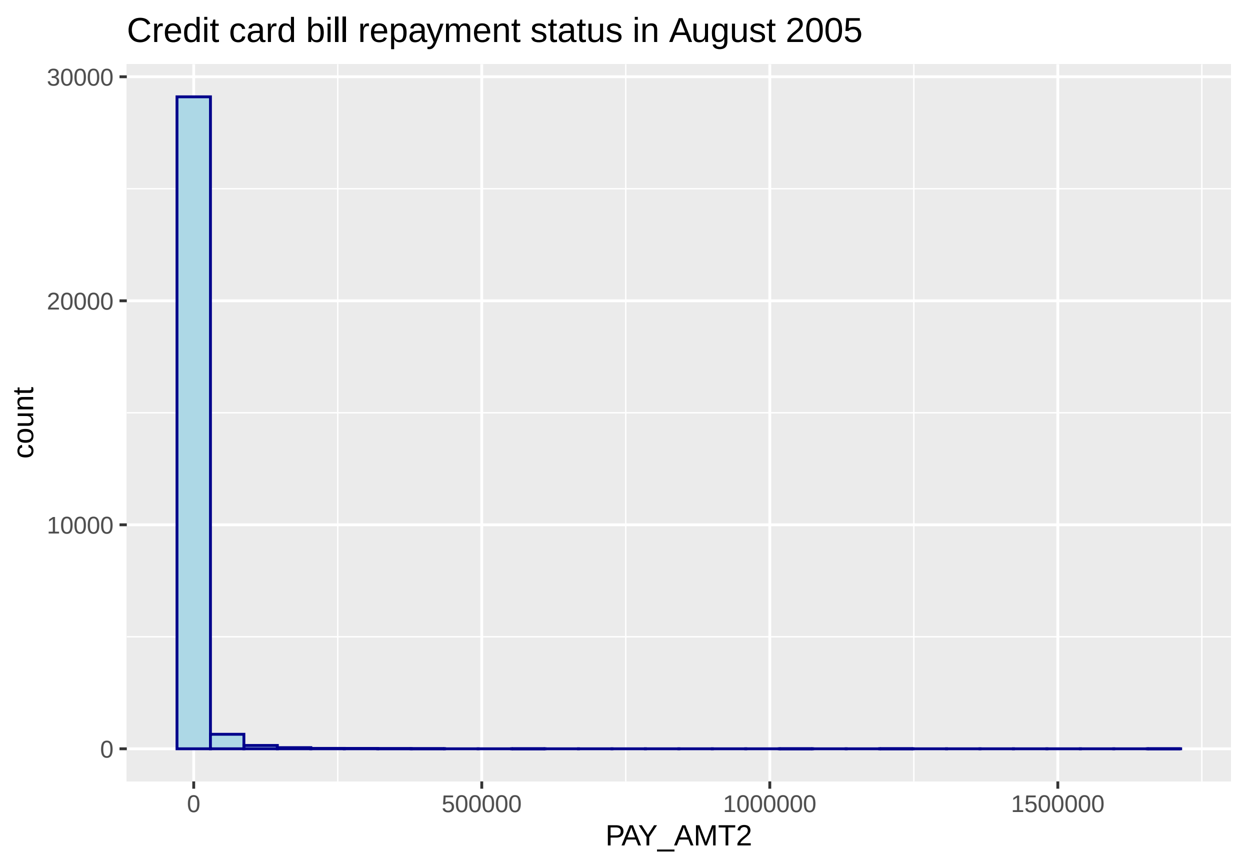
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



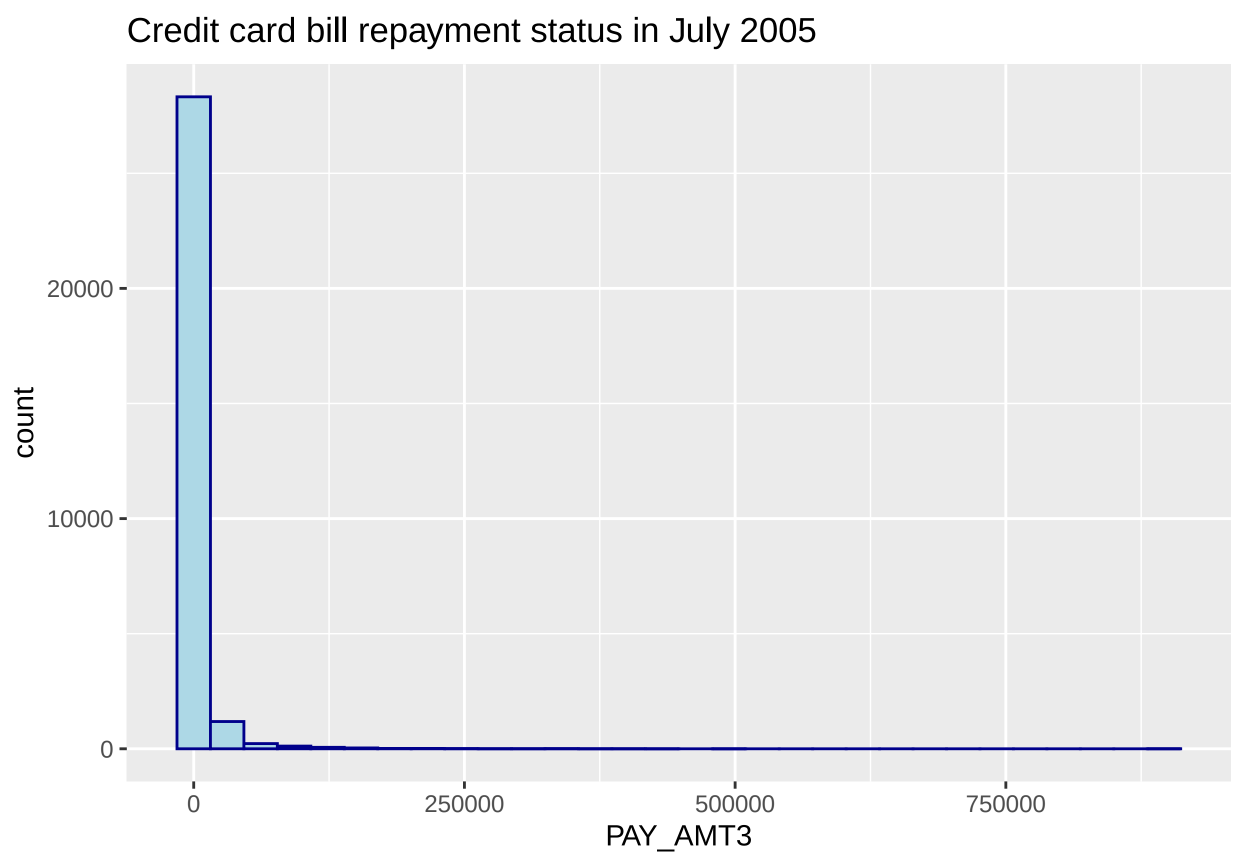
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



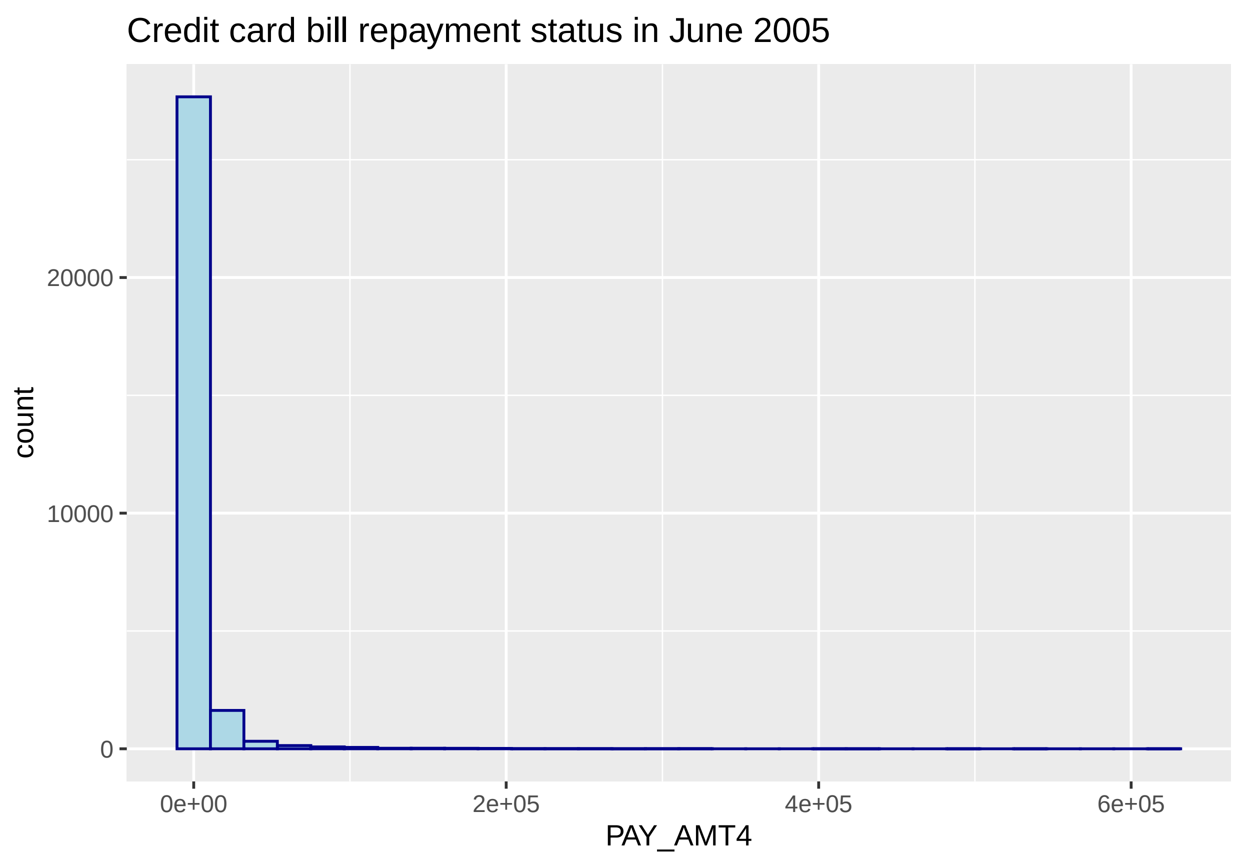
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



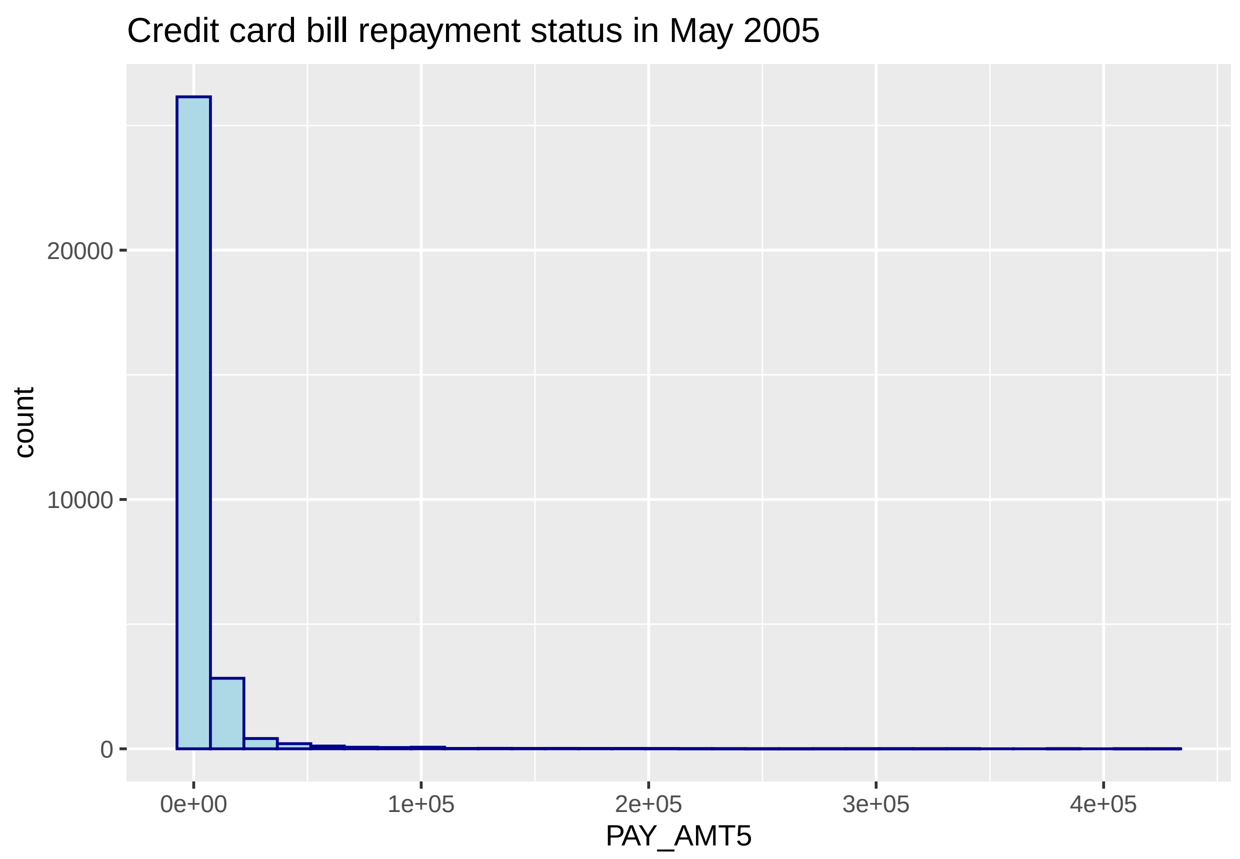
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



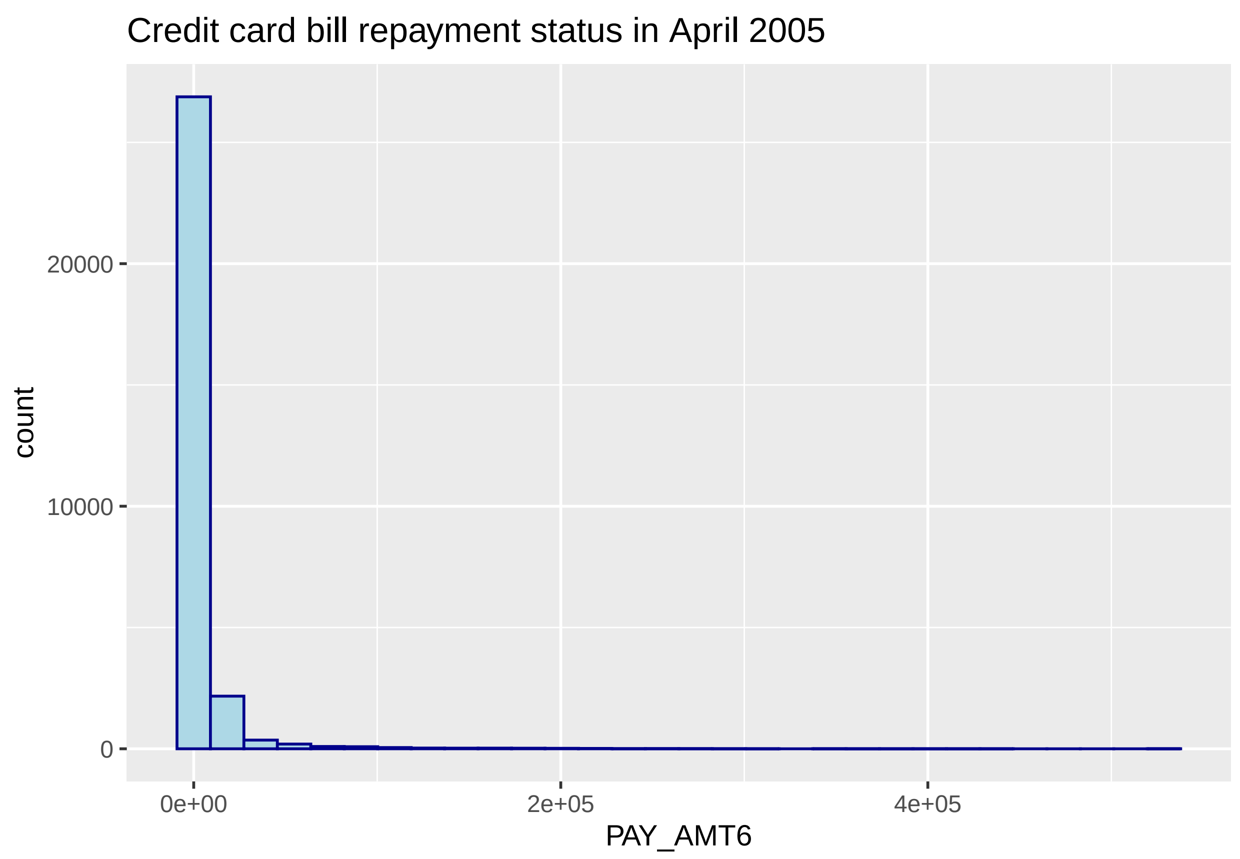
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Before putting these values into models, the skewness was corrected for all these attributes by applying log function. The rationale behind selecting a log transformation method was because it is a recommended algorithm for transforming data that is highly right skewed (Hashmat, 2020).

```
datadump$LIMIT_BAL=log(as.numeric(datadump$LIMIT_BAL)+1)
BILL1_median=median(datadump$BILL_AMT1)
datadump$BILL_AMT1=log(as.numeric(datadump$BILL_AMT1)+1)
```

```
## Warning in log(as.numeric(datadump$BILL_AMT1) + 1): NaNs produced
```

```
datadump$PAY_AMT1=log(as.numeric(datadump$PAY_AMT1)+1)
datadump$PAY_AMT2=log(as.numeric(datadump$PAY_AMT2)+1)
datadump$PAY_AMT3=log(as.numeric(datadump$PAY_AMT3)+1)
datadump$PAY_AMT4=log(as.numeric(datadump$PAY_AMT4)+1)
datadump$PAY_AMT5=log(as.numeric(datadump$PAY_AMT5)+1)
datadump$PAY_AMT6=log(as.numeric(datadump$PAY_AMT6)+1)
```

However, the transformation resulted with the total of 590 Nan and -Inf values for the bill statement amount in September (BILL_AMT1) due to the negative values in the column.

```
colSums(datadump=="NaN"|datadump=="-Inf")
```

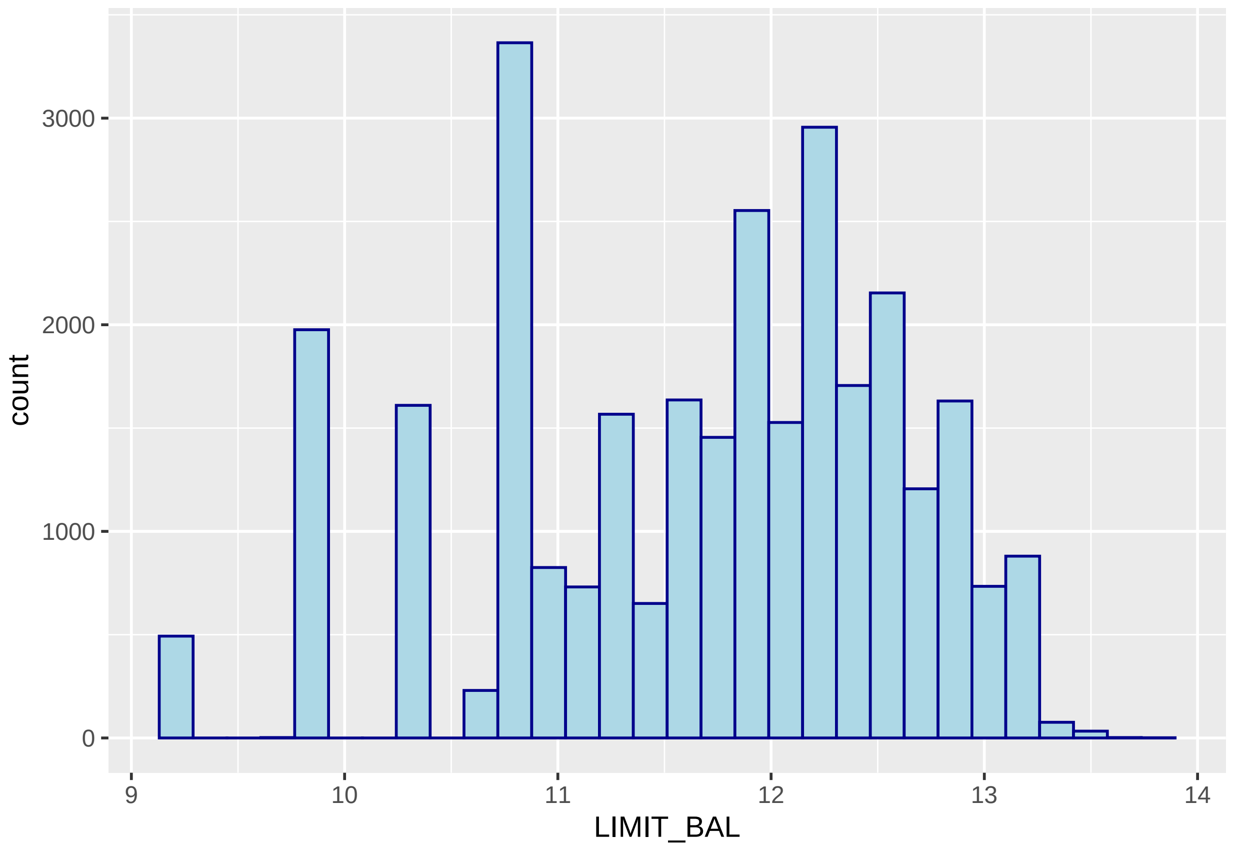
```
## LIMIT_BAL       SEX EDUCATION  MARRIAGE     PAY_1     PAY_2     PAY_3     PAY_4
##         0         0         0         0         0         0         0         0
##      PAY_5      PAY_6 BILL_AMT1  PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5
##          0          0       590         0         0         0         0         0
##   PAY_AMT6 DF_Result
##          0         0
```

Therefore, it was decided that NaN and -Inf values to be replaced with the log of the median of the column.

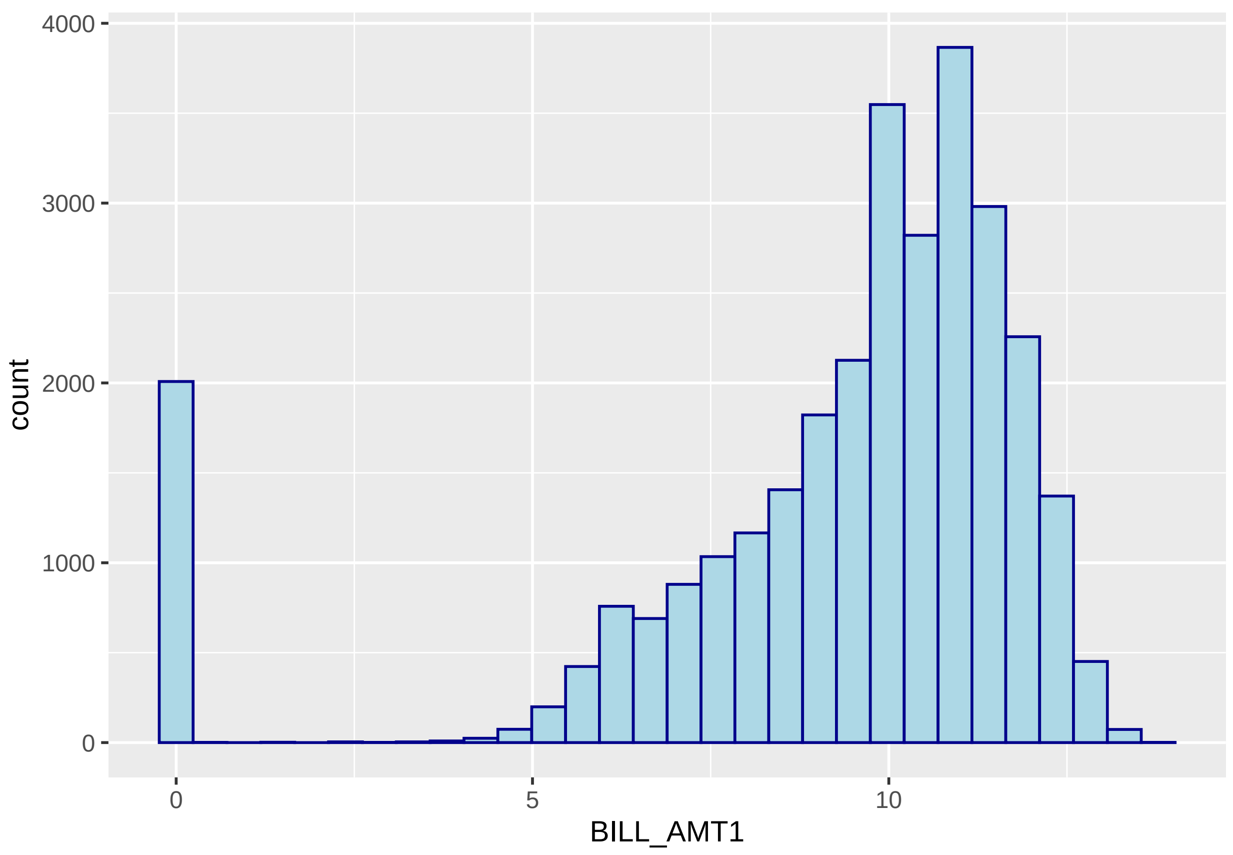
```
datadump$BILL_AMT1[datadump$BILL_AMT1 == "-Inf" |
                     datadump$BILL_AMT1 == "NaN"] <- log(BILL1_median+1)
```

This removed all the Nan and -Inf values. Then, the histograms were generated again, and the graphs now show more even distribution across the attributes.

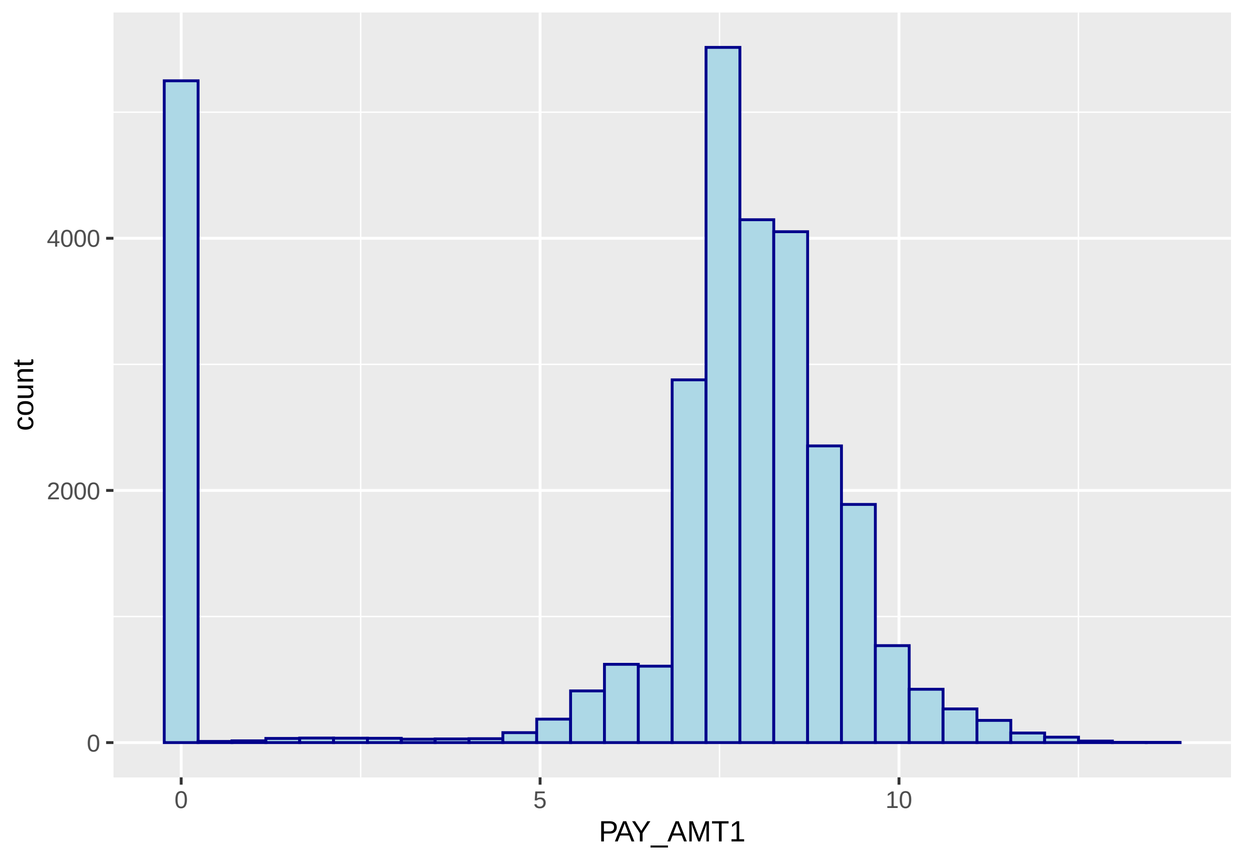
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



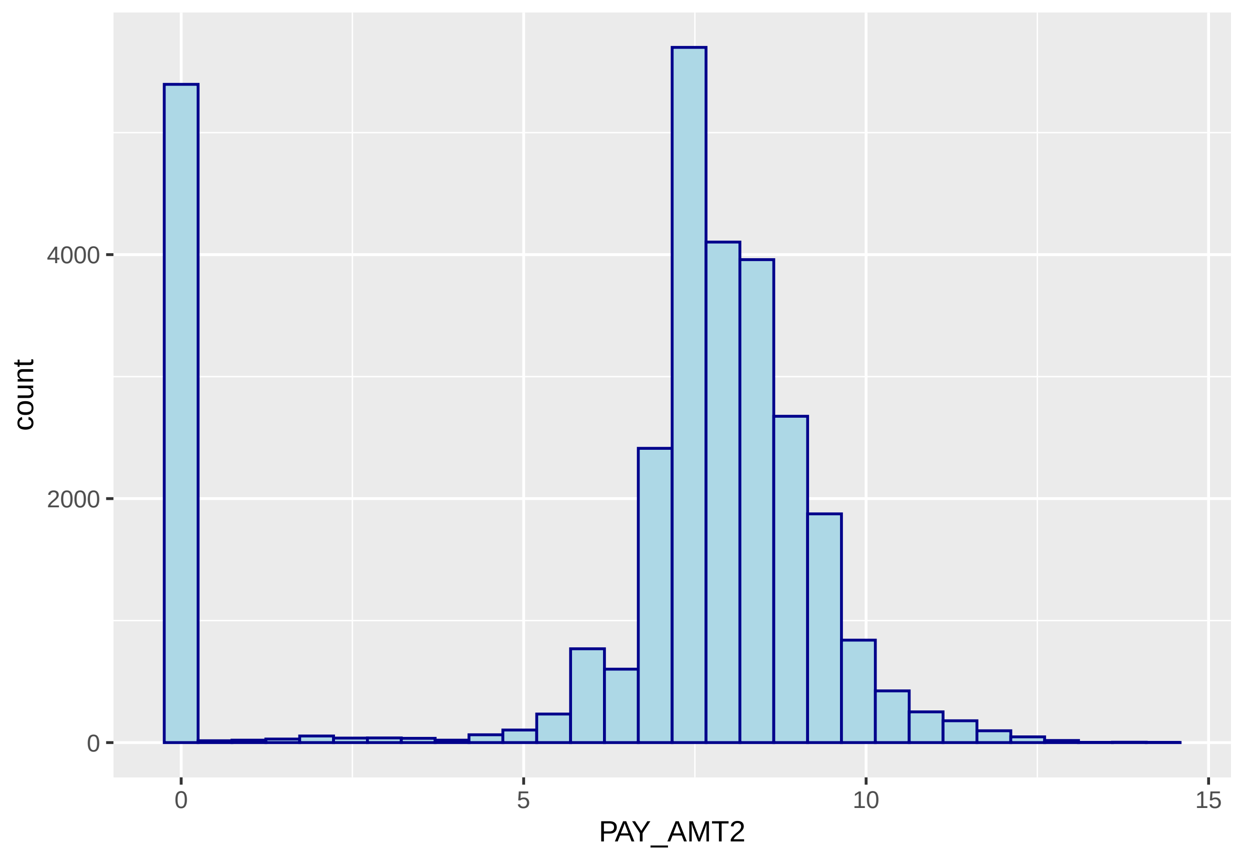
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



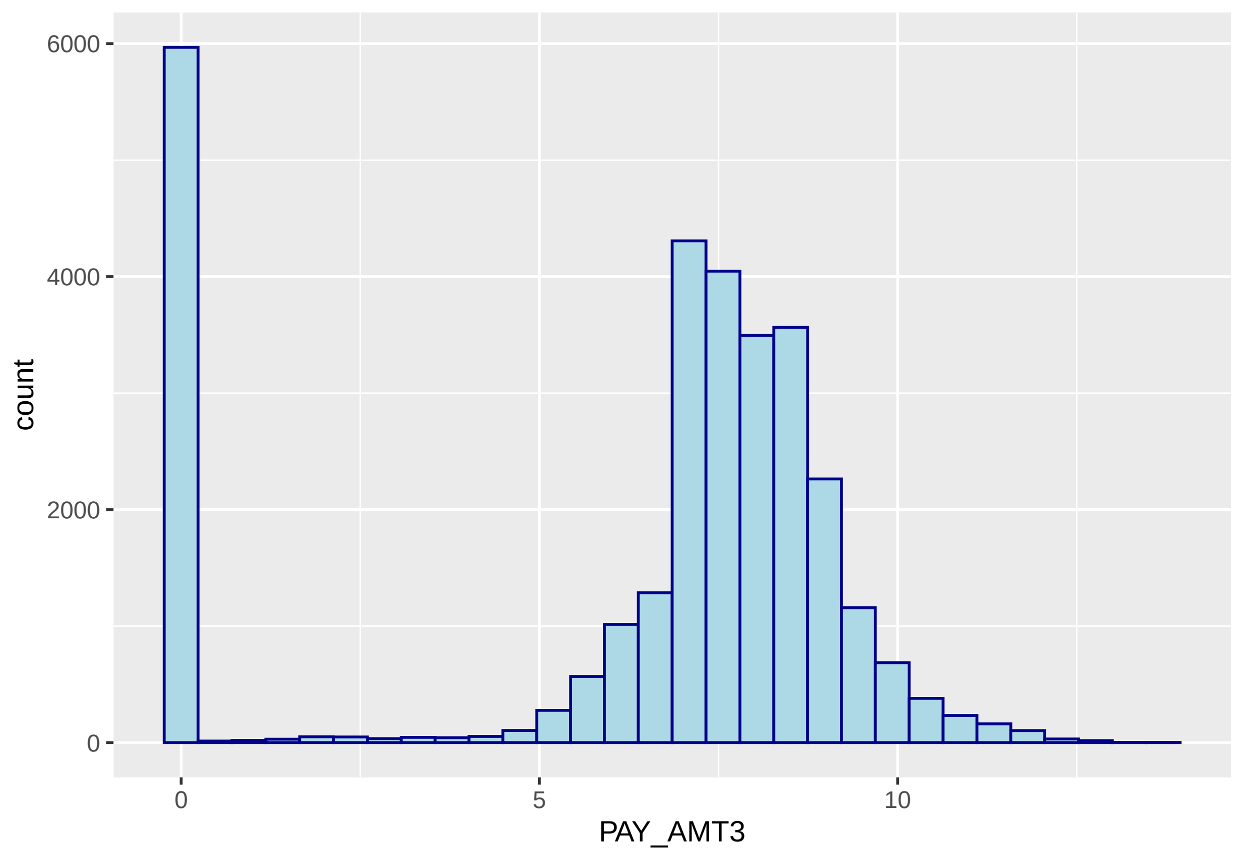
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



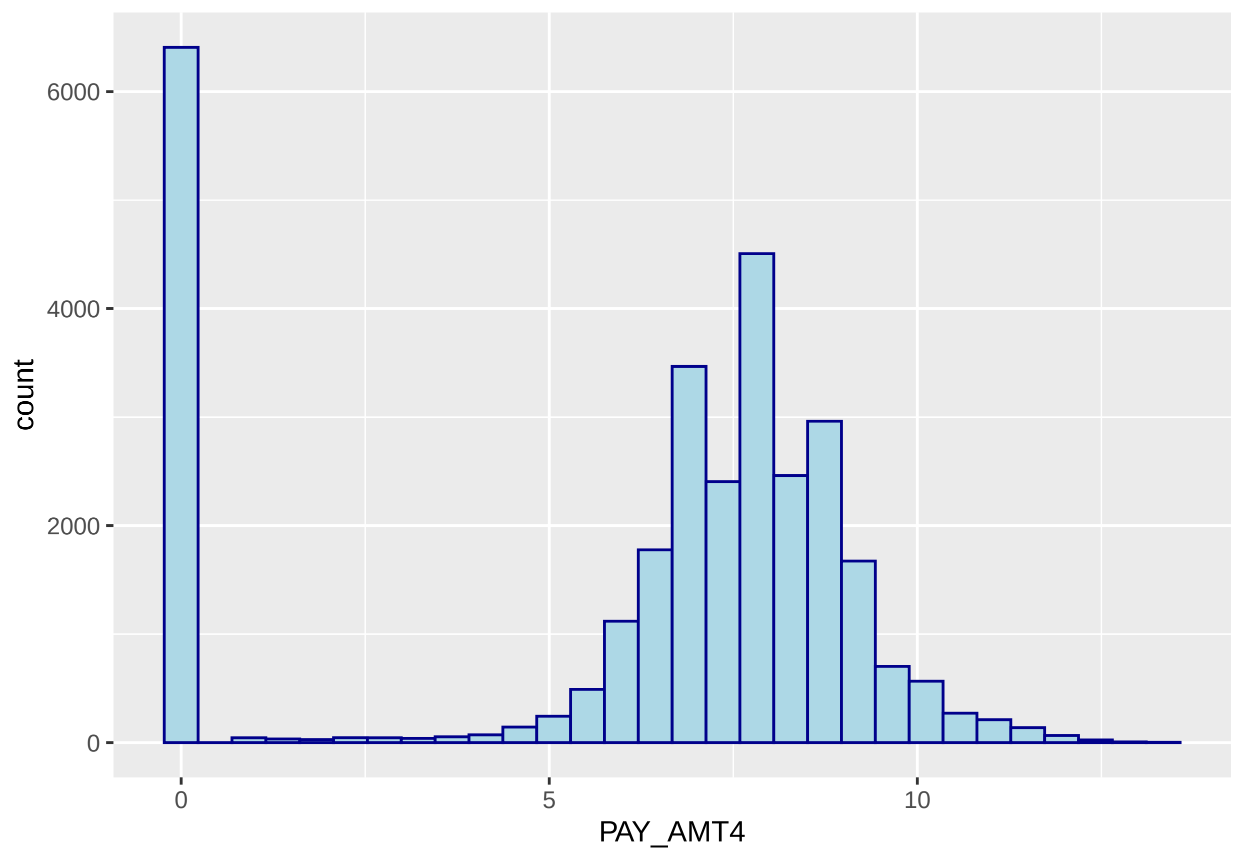
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



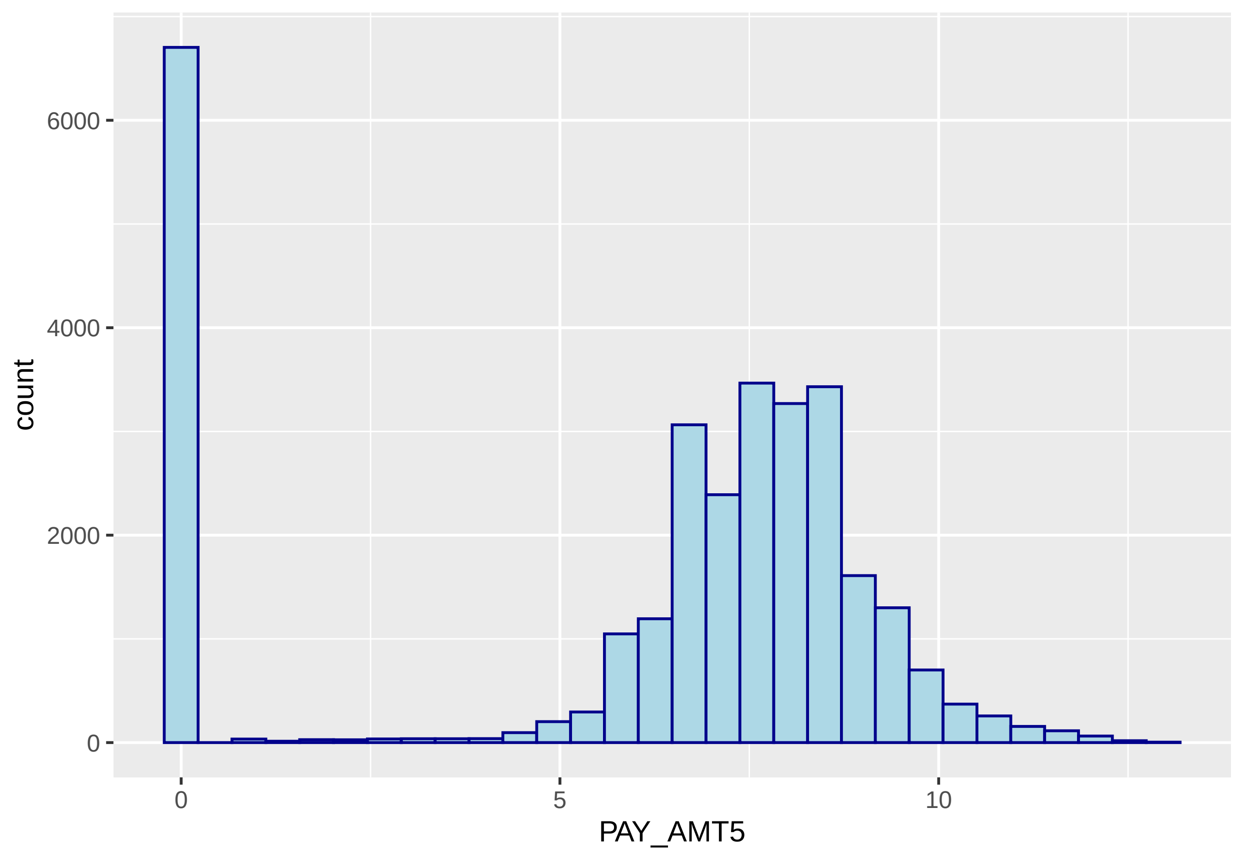
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



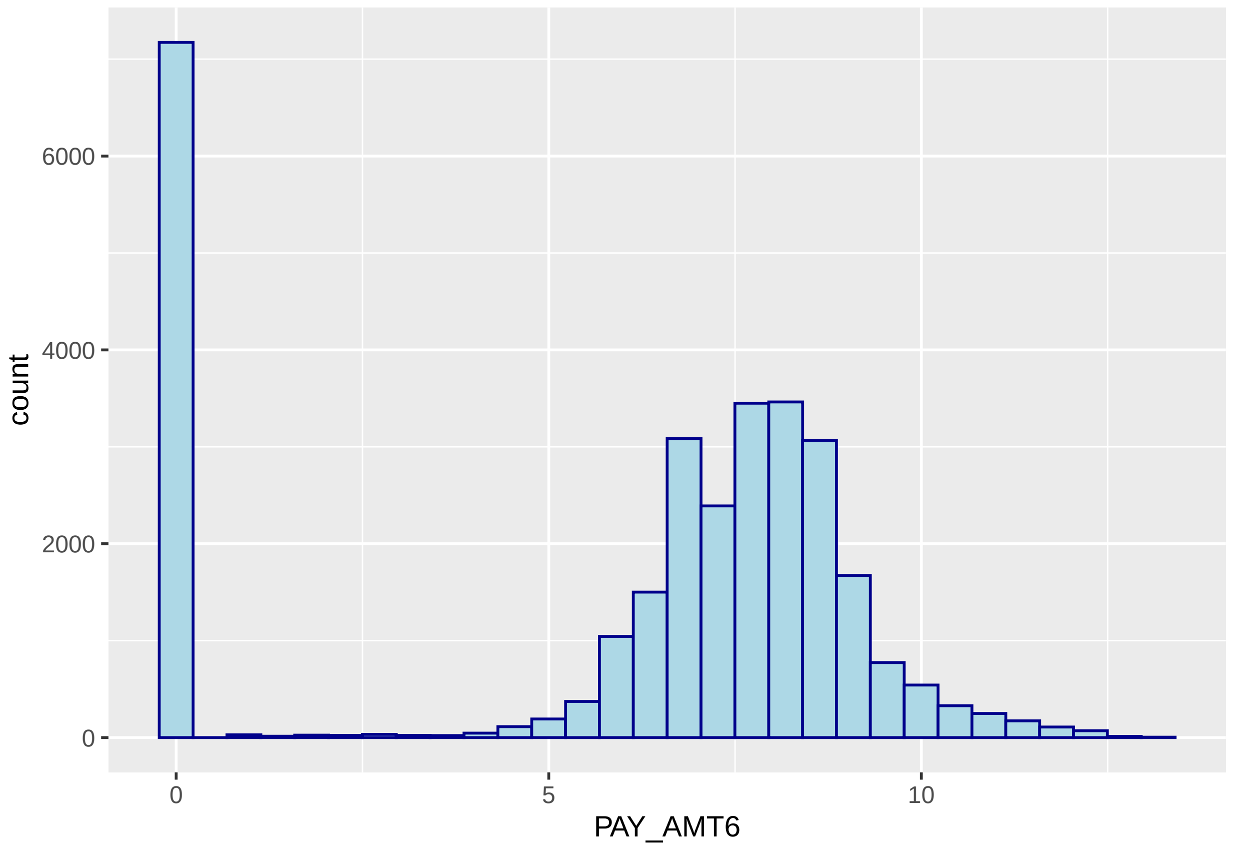
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
colSums(datadump=="NaN"|datadump=="-Inf")
```

```
## LIMIT_BAL       SEX EDUCATION  MARRIAGE     PAY_1     PAY_2     PAY_3     PAY_4
##         0         0         0         0         0         0         0         0
##     PAY_5     PAY_6 BILL_AMT1  PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5
##         0         0         0         0         0         0         0         0
##  PAY_AMT6 DF_Result
##         0         0
```

Data Analysis Methods

The final dataset was split into Train, Validate and Test datasets. Because of the unbalanced distribution of target variables, it was necessary to apply a stratified method when splitting. Thus, createDataPartition() function was used, since the function implies the stratified method (Kuhn, 2019). In addition, the proportion of target variables in all three datasets were double checked to ensure it is the same across all.

```
library(caret)
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:survival':
##
##     cluster
```

```r
set.seed(1992)
firstcut = createDataPartition(datadump$DF_Result, p = 0.9, list=FALSE)
bigtrain_d = datadump[firstcut,]
test_d = datadump[-firstcut,]
secondcut=createDataPartition(bigtrain_d$DF_Result, p = 0.9, list=FALSE)
train_d=bigtrain_d[secondcut,]
val_d=bigtrain_d[-secondcut,]
```

Logistic Regression Model

A logistic regression is a machine learning algorithm that models the probabilities for a classification problem by giving two probable outcomes (Molnar, 202). Logistic regression models (LRM) the probabilities for classification problems with two possible outcomes. LRM is a continuation of the linear regression model as it takes classification problems one step further (Molnar, 2020) When compared to other other models can be quite advantageous is that they provide probilities and can determine the final classification. LRM are not without disadvantage as they perform poorly against other models when predicting performance. Also, LRM poses the challenge of being unable to further train in the event where there is a feature that could separate two classes (Molnar, 2020)

Outcome

```r
library(caret)
set.seed(1992)
firstcut = createDataPartition(datadump$DF_Result, p = 0.9, list=FALSE)
bigtrain_d = datadump[firstcut,]
test_d = datadump[-firstcut,]
secondcut=createDataPartition(bigtrain_d$DF_Result, p = 0.9, list=FALSE)
train_d=bigtrain_d[secondcut,]
val_d=bigtrain_d[-secondcut,]

#metrics in one table
metrics<-data.frame("LogisticRegression"=numeric(6),"DecisionTree"=numeric(6),
                    "RandomForest"=numeric(6),"KNN"=numeric(6))
rownames(metrics)<-c("Precision","Recall","Accuracy","Specificity","F1","AUC")

model_lr <- glm(DF_Result ~.,family=binomial(link='logit'),data=train_d)

#Running the model on Validation Set
pred.val_lr <- predict(model_lr,val_d)
pred.val_lr <- ifelse(pred.val_lr > 0.5,1,0)
mean(pred.val_lr==val_d$DF_Result)
```

```
## [1] 0.8136347
```

```r
tlr_1<-table(pred.val_lr,val_d$DF_Result)
# Metrics of the model on the Validation Set
#precision
presicion_lr_val<- tlr_1[1,1]/(sum(tlr_1[1,]))
#recall or sensitivity
recall_lr_val<- tlr_1[1,1]/(sum(tlr_1[,1]))
```

```
#accuracy
accuracy_lr_val<-(tlr_1[1,1]+tlr_1[2,2])/sum(tlr_1)
#specificity or selectivity
specificity_lr_val<- tlr_1[2,2]/(sum(tlr_1[,2]))
# F1 score
F1_lr_val<- 2*presicion_lr_val*recall_lr_val/(presicion_lr_val+recall_lr_val)
# Presicion and recall of the model on the Test Set
pred.test_lr <- predict(model_lr,test_d)
pred.test_lr <- ifelse(pred.test_lr > 0.5,1,0)
mean(pred.test_lr==test_d$DF_Result)
```

```
## [1] 0.8162721
```

```
tlr_2<-table(pred.test_lr,test_d$DF_Result)

# Metrics of the model on the Test Set
#precision
metrics["Precision","LogisticRegression"]= tlr_2[1,1]/(sum(tlr_2[1,]))
#recall or sensitivity
metrics["Recall","LogisticRegression"]= tlr_2[1,1]/(sum(tlr_2[,1]))
#accuracy
metrics["Accuracy","LogisticRegression"]=(tlr_2[1,1]+tlr_2[2,2])/sum(tlr_2)
#specificity or selectivity
metrics["Specificity","LogisticRegression"]= tlr_2[2,2]/(sum(tlr_2[,2]))

# F1 score
metrics["F1","LogisticRegression"]=
  2*metrics["Precision","LogisticRegression"]*
  metrics["Recall","LogisticRegression"]/
  (metrics["Precision","LogisticRegression"]+
     metrics["Recall","LogisticRegression"])
```

The mean score of achieving the equal target variable for logistic regression was 0.816. Precision score was 0.821, and recall was 0.977. The F1 score was 0.892.

The following graph shows the plot between the false positive rate, and the true positive rate.

```
library(ggplot2)
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

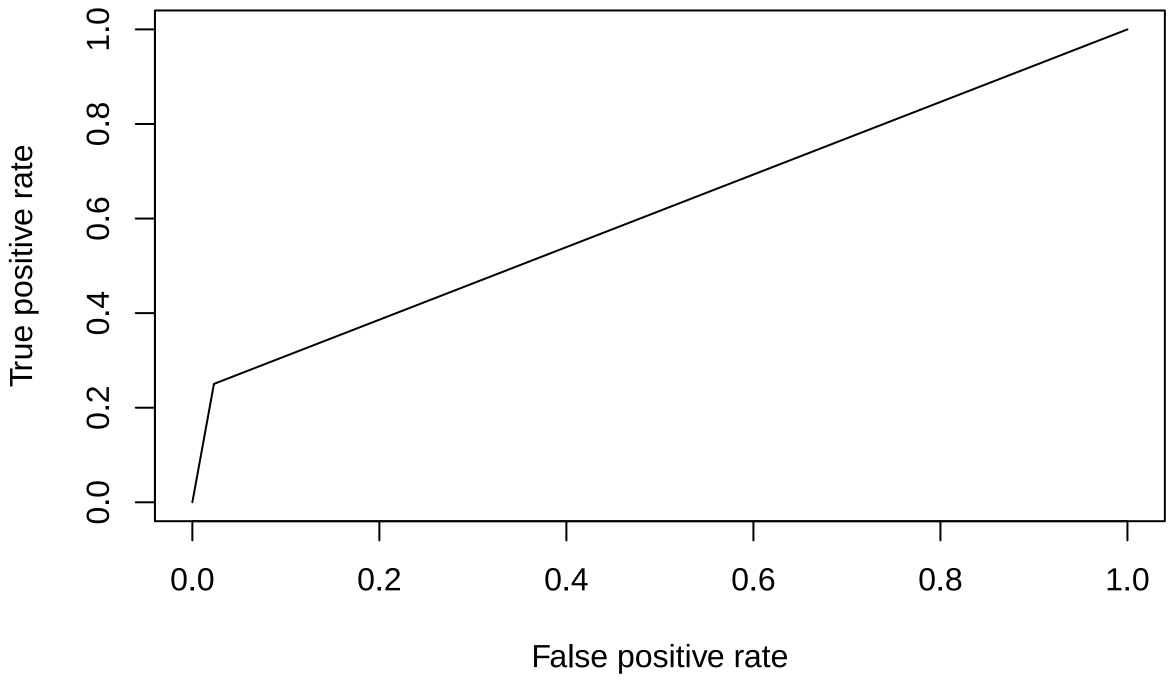
```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(Metrics)
```

```
##
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
#ROC and AUC
pr_lr <- prediction(pred.test_lr,test_d$DF_Result)
perf_lr <- performance(pr_lr,measure = "tpr",x.measure = "fpr")
plot(perf_lr,color="red")
```



```
auc_lr<-auc(test_d$DF_Result,pred.test_lr)
metrics["AUC","LogisticRegression"]=auc_lr
```

Decision Tree Model

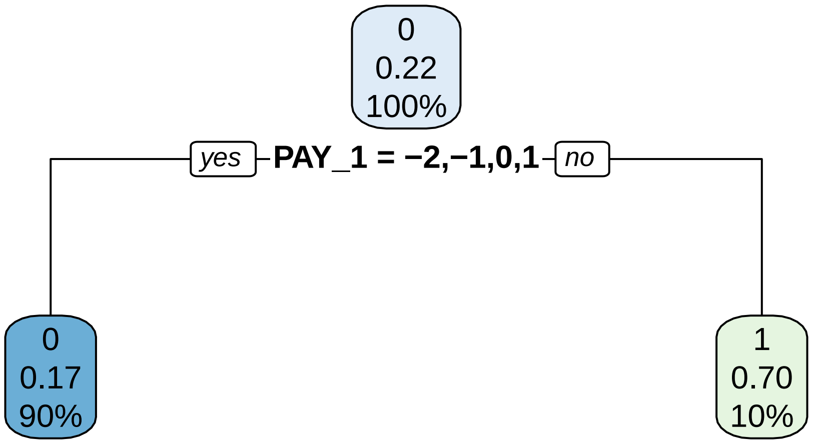
A decision tree can be used for classification and regression problems (Molnar, 2020). This model is able to demonstrate interactions between varying features within the data. Decision trees allow for the data to be represented in specific groups, this allows for easy data interpretation. Another major advantage of decision trees is the model's ability in providing a clear explanation (Molnar, 2020). Decision tree models are unable to handle linear relationships, this is because the existing relationship between a feature and an outcome is approximated by splits, resulting in a step function (Molnar, 2020). This process is not efficient for this model. Another disadvantage in such a model can prove to be unsteady in that minor change made to the training dataset results in another tree.

Outcome

```
library(rpart.plot)
```

```
## Loading required package: rpart
```

```
model_dt<- rpart(DF_Result ~.,data=train_d, method="class")
rpart.plot(model_dt)
```



```
#Running the model on Validation Set
pred.val_d.dt <- predict(model_dt,val_d,type = "class")
mean(pred.val_d.dt==val_d$DF_Result)
```

```
## [1] 0.8177103
```

```
tdt_1<-table(pred.val_d.dt,val_d$DF_Result)
# Metrics of the model on the Validation Set
#precision
presicion_dt_val<- tdt_1[1,1]/(sum(tdt_1[1,]))
#recall or sensitivity
recall_dt_val<- tdt_1[1,1]/(sum(tdt_1[,1]))
#accuracy
accuracy_dt_val<-(tdt_1[1,1]+tdt_1[2,2])/sum(tdt_1)
```

```
#specificity or selectivity
specificity_dt_val<- tdt_1[2,2]/(sum(tdt_1[,2]))
#F1 Score
F1_dt_val<- 2*presicion_dt_val*recall_dt_val/(presicion_dt_val+recall_dt_val)
#Running the same model on Test Set
pred.test_d.dt <- predict(model_dt,test_d,type="class")
mean(pred.test_d.dt==test_d$DF_Result)
```

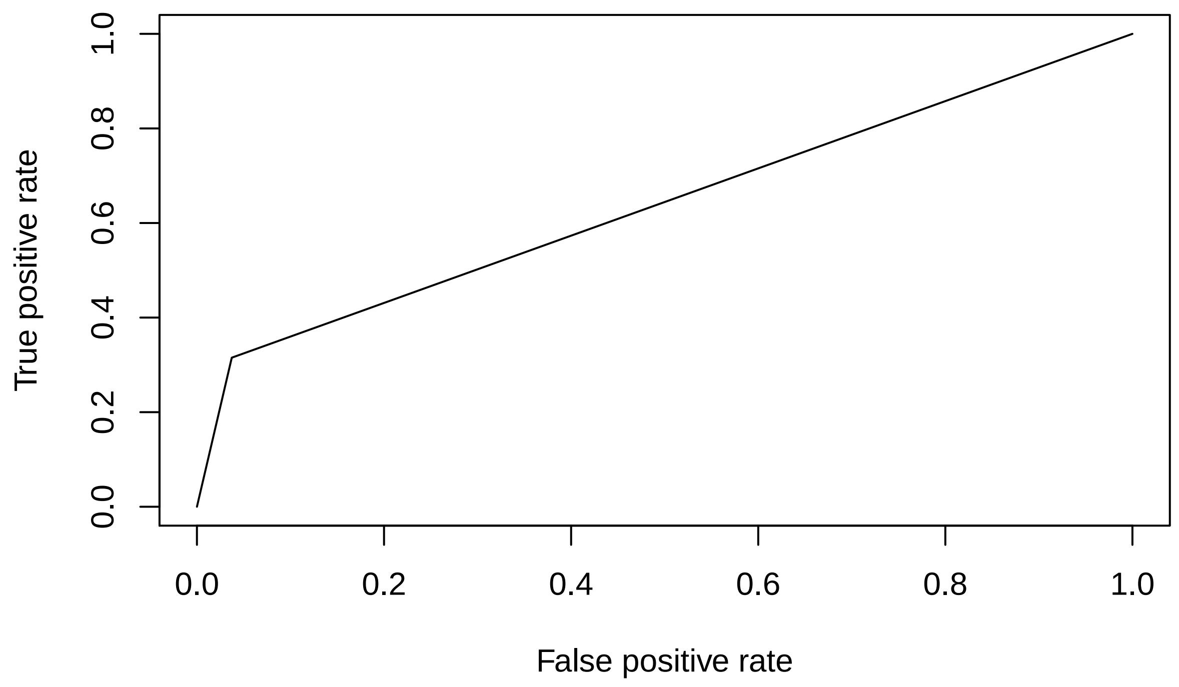
```
## [1] 0.8196065
```

```
tdt_2<-table(pred.test_d.dt,test_d$DF_Result)
# Metrics of the model on the Test Set
#precision
metrics["Precision","DecisionTree"]=tdt_2[1,1]/(sum(tdt_2[1,]))
#recall or sensitivity
metrics["Recall","DecisionTree"]=tdt_2[1,1]/(sum(tdt_2[,1]))
#accuracy
metrics["Accuracy","DecisionTree"]=(tdt_2[1,1]+tdt_2[2,2])/sum(tdt_2)
#specificity or selectivity
metrics["Specificity","DecisionTree"]=tdt_2[2,2]/(sum(tdt_2[,2]))
#F1 Score
metrics["F1","DecisionTree"]=
  2*metrics["Precision","DecisionTree"]*
  metrics["Recall","DecisionTree"]/
  (metrics["Precision","DecisionTree"]+metrics["Recall","DecisionTree"])
```

The mean score of achieving the equal target variable for decision tree model was 0.820. Precision score was 0.832, and recall was 0.963. The F1 score was 0.893.

The following graph shows the plot between the false positive rate, and the true positive rate.

```
#ROC and AUC
pr_dt <- prediction(as.numeric(pred.test_d.dt),test_d$DF_Result)
perf_dt <- performance(pr_dt,measure = "tpr",x.measure = "fpr")
plot(perf_dt,color="red")
```



```
metrics["AUC","DecisionTree"]=auc(test_d$DF_Result,pred.test_d.dt)
```

Random Forest Model

Random forest models are made of decision trees that have no correlation between them. Such a model can determine the significance of a feature as well as the interaction between the varying features ("Random Forest", n.d.). Features do not have to be selected and dimensions do not have to be reduced, this is because such a model can give high dimensional data ("Random Forest", n.d.). The more trees that are present, this model decreases the tendency to overfit. Lastly, this model is very easy to train and implement. One of the major disadvantages of random forest is that it tends to fit specific noisy classification or regression problems ("Random Forest", n.d.).

Outcome

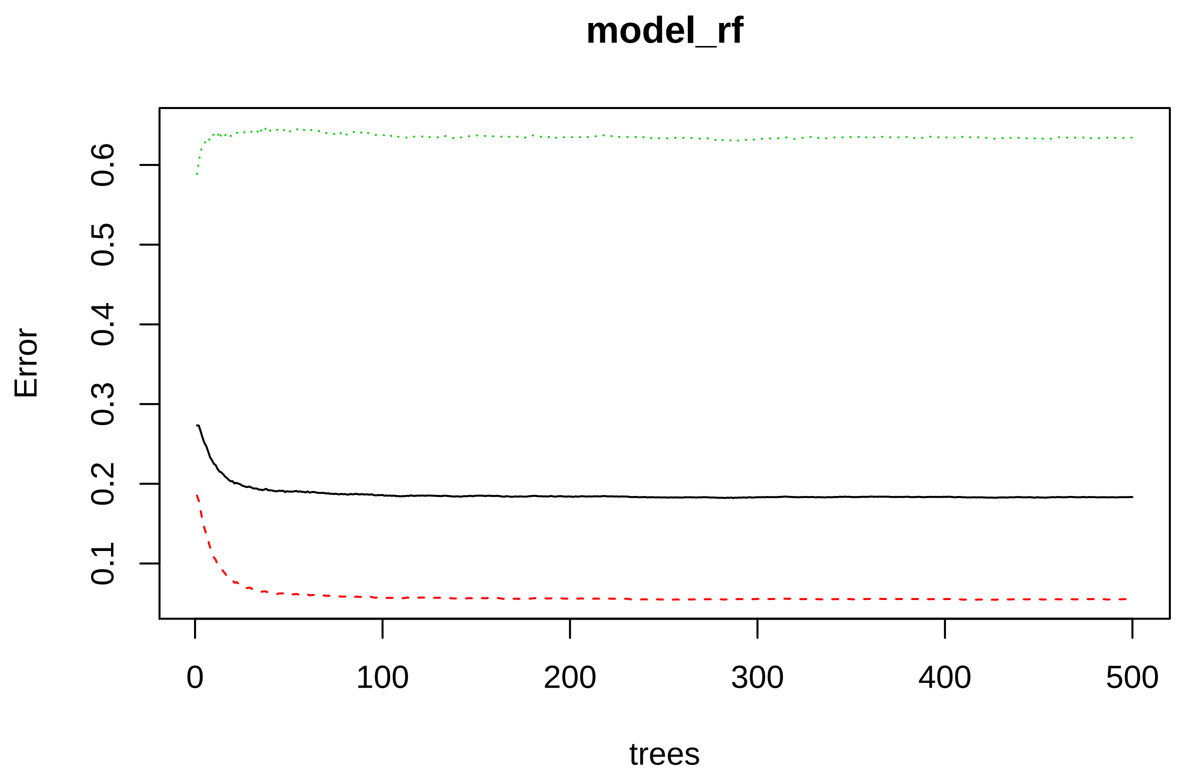
```
library(randomForest)
```

```
## 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
```

```
model_rf<-randomForest(DF_Result~.,data=train_d)
#the error
plot(model_rf)
```



```
#Running the model on Validation Set
pred.val_d.rf <- predict(model_rf,val_d)
mean(pred.val_d.rf==val_d$DF_Result)
```

```
## [1] 0.8117821
```

```
trf_1<-table(pred.val_d.rf,val_d$DF_Result)
# Metrics of the model on the Validation Set
#precision
presicion_rf_val<- trf_1[1,1]/(sum(trf_1[1,]))
#recall or sensitivity
recall_rf_val<- trf_1[1,1]/(sum(trf_1[,1]))
#accuracy
accuracy_rf_val<-(trf_1[1,1]+trf_1[2,2])/sum(trf_1)
#specificity or selectivity
specificity_rf_val<- trf_1[2,2]/(sum(trf_1[,2]))
```

```
#F1 Score
F1_rf_val<- 2*presicion_rf_val*recall_rf_val/(presicion_rf_val+recall_rf_val)
#Running the same model on Test Set
pred.test_d.rf <- predict(model_rf,test_d)
mean(pred.test_d.rf==test_d$DF_Result)
```

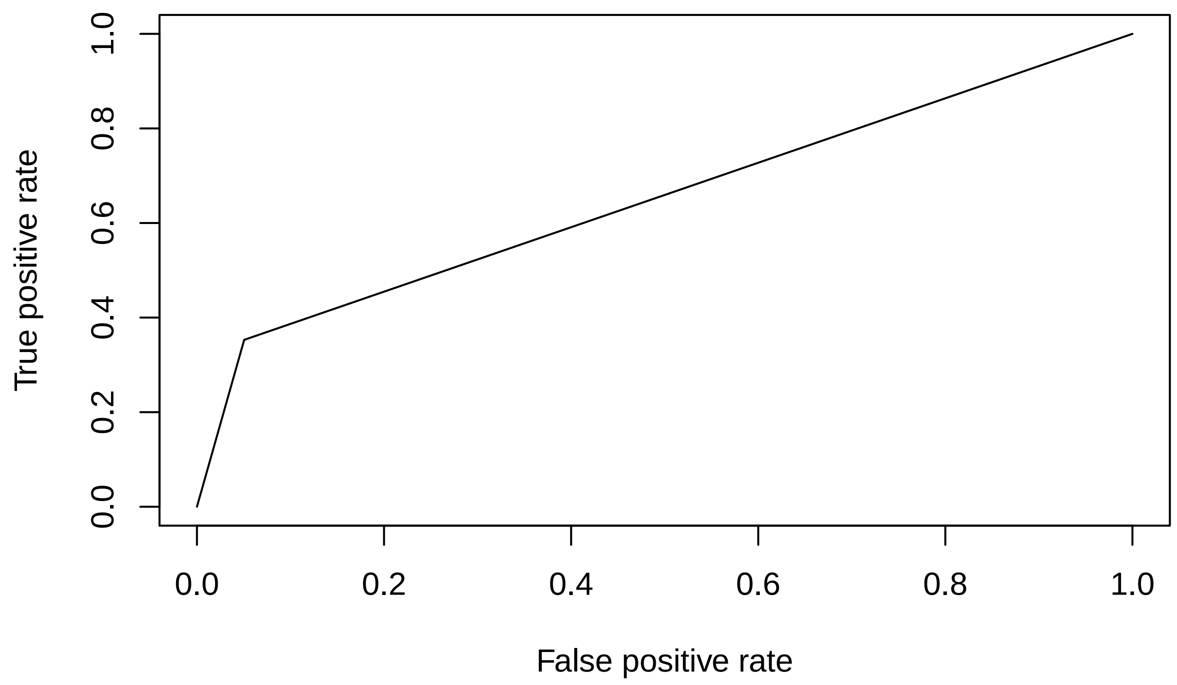
```
## [1] 0.8176059
```

```
trf_2<-table(pred.test_d.rf,test_d$DF_Result)
# Metrics of the model on the Test Set
#precision
metrics["Precision","RandomForest"]=trf_2[1,1]/(sum(trf_2[1,]))
#recall or sensitivity
metrics["Recall","RandomForest"]=trf_2[1,1]/(sum(trf_2[,1]))
#accuracy
metrics["Accuracy","RandomForest"]=(trf_2[1,1]+trf_2[2,2])/sum(trf_2)
#specificity or selectivity
metrics["Specificity","RandomForest"]= trf_2[2,2]/(sum(trf_2[,2]))
#F1 Score
metrics["F1","RandomForest"]=
  2*metrics["Precision","RandomForest"]*
  metrics["Recall","RandomForest"]/
  (metrics["Precision","RandomForest"]+metrics["Recall","RandomForest"])
```

The mean score of achieving the equal target variable for random forest model was 0.818. Precision score was 0.838, and recall was 0.950. The F1 score was 0.890.

The following graph shows the plot between the false positive rate, and the true positive rate.

```
#ROC and AUC
pr_rf <- prediction(as.numeric(pred.test_d.rf),test_d$DF_Result)
perf_rf <- performance(pr_rf,measure = "tpr",x.measure = "fpr")
plot(perf_rf,color="red")
```



```
metrics["AUC","RandomForest"]=auc(test_d$DF_Result,pred.test_d.rf)
```

K-Means Nearest Neighbor (KNN) Model

The K-Nearest Neighbor also referred to as k-NN, is an algorithm that is easy to understand and implement. Although it is said to be lazy, it proves to be quite a versatile algorithm. k-NNt can be used to solve classification and regression problems (“What is the K-nearest”, n.d.). k-NNcan also be used for non-linear classification. Interestingly, kNN models make no assumptions about the data, they demonstrate high accuracy and are not sensitive to outliers (“What is the K-nearest”, n.d.). One of the many challenges this model presents is that it requires a lot of memory. Also, it deploys an ineffective sample balance, in that the number of samples in certain categories can be large, while the number of other samples are small. Additionally, in order to obtain optimal k value selection, the k value size must be combined with k-fold cross validation. Lastly, if a sample is unbalanced, k-NN gives a large prediction bias (“What is the K-nearest”, n.d.)

Outcome

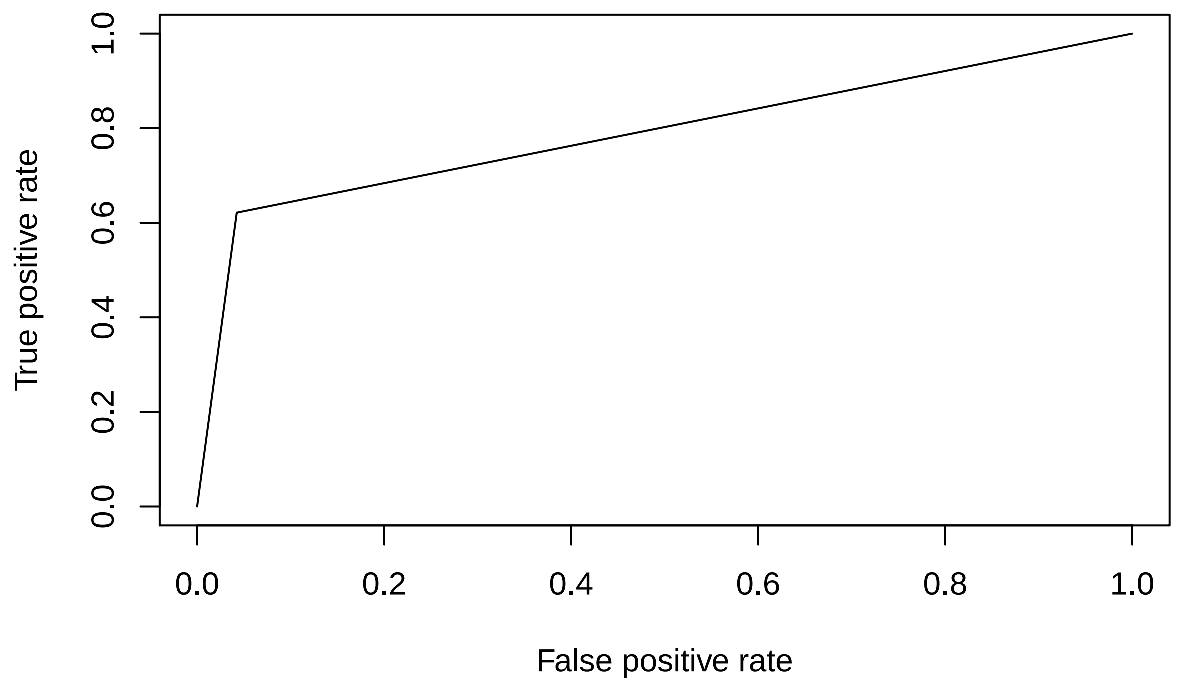
```
library(class)
train_labels<-train_d[,18]
test_labels<-test_d[,18]
val_labels<-val_d[,18]
knumber=sqrt(dim(train_d)[2])
#Running Model on Validation set
```

```r
model_KNN <- knn(train=train_d,test=val_d,cl=train_labels,k=4)
tKNN_1=table(model_KNN,val_labels)
# Metrics of the model on the Vaildation Set
#precision
precision_knn_val=tKNN_1[1,1]/(sum(tKNN_1[1,]))
#recall or sensitivity
recall_knn_val=tKNN_1[1,1]/(sum(tKNN_1[,1]))
#accuracy
accuracy_knn_val=(tKNN_1[1,1]+tKNN_1[2,2])/sum(tKNN_1)
#specificity or selectivity
specificity_knn_val=tKNN_1[2,2]/(sum(tKNN_1[,2]))
#F1 Score
f1_knn_val=2*precision_knn_val*recall_knn_val/(precision_knn_val+recall_knn_val)
#Running KNN on the Test Set
model_KNN <- knn(train=train_d,test=test_d,cl=train_labels,k=4)
tKNN_2=table(model_KNN,test_labels)
# Metrics of the model on the Test Set
#precision
metrics["Precision","KNN"]=tKNN_2[1,1]/(sum(tKNN_2[1,]))
#recall or sensitivity
metrics["Recall","KNN"]=tKNN_2[1,1]/(sum(tKNN_2[,1]))
#accuracy
metrics["Accuracy","KNN"]=(tKNN_2[1,1]+tKNN_2[2,2])/sum(tKNN_2)
#specificity or selectivity
metrics["Specificity","KNN"]=tKNN_1[2,2]/(sum(tKNN_2[,2]))
#F1 Score
metrics["F1","KNN"]=
  2*metrics["Precision","KNN"]*metrics["Recall","KNN"]/
  (metrics["Precision","KNN"]+metrics["Recall","KNN"])
```

KNN model's precision score was 0.897, and recall score was 0.958. The F1 score was 0.926.

The following graph shows the plot between the false positive rate, and the true positive rate.

```r
#ROC and AUC
pr_knn <- prediction(as.numeric(model_KNN),test_d$DF_Result)
perf_knn <- performance(pr_knn,measure = "tpr",x.measure = "fpr")
plot(perf_knn,color="red")
```



```r
metrics["AUC","KNN"]=auc(test_d$DF_Result,model_KNN)
```

Discussion

Analysis

Model Evaluation Metrics

The models used in this assignment were evaluated using F1 score and AUC. The F1 score determines a test's accuracy by using precision and recall of the test. Precision is simply the correctly identified positive cases from all the predicted positive cases, while recall is the correctly identified positive cases from all the actual positive cases (Huilgol, 2019). The formula used to compute F1 score is demonstrated as:

$F1 = 2 * [(Precision * Recall)/(Precision + Recall)]$

Ultimately, the F1 score is the means of precision and recall (Huilgol, P. 2019).

The Area under the ROC Curve, AUC, measures classification problems at different threshold points. The ROC is a probability curve, whereas AUC determines separability. This evaluation metric demonstrace the model's ability to differentiate between classes (Narkhede, 2018).

Since there was uneven class distribution along with a need to fix the balance between precision and recall, the F1 score seemed to be a good fit for evaluating the models used in this assignment (Shung, 2019). Additionally AUC seemed to be a good fit in evaluating the models, this is because the models used were identify characteristics of the classes to help determine if this influenced default payments.

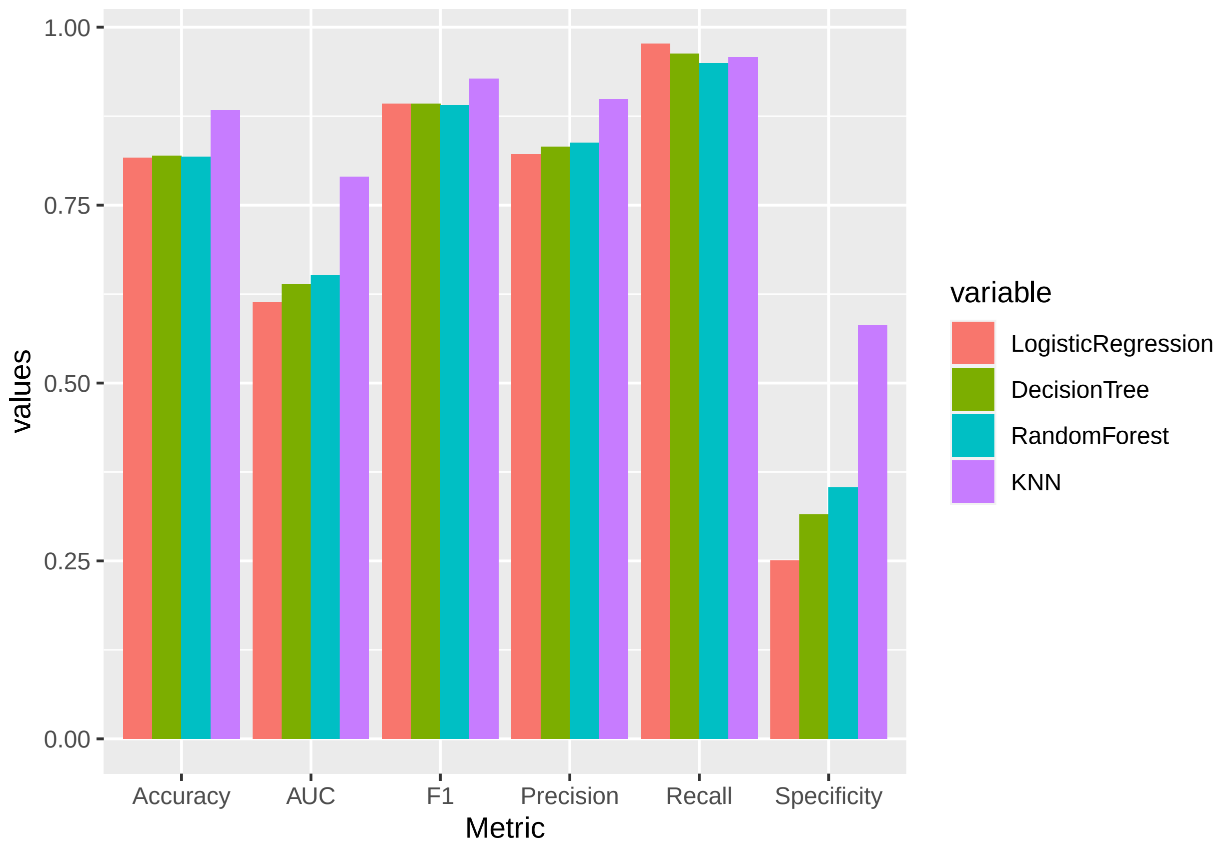
Model Comparison and Effectiveness

```
##             LogisticRegression DecisionTree RandomForest       KNN
## Precision            0.8211587    0.8320385    0.8379297 0.8991158
## Recall               0.9768836    0.9627568    0.9494863 0.9576199
## Accuracy             0.8162721    0.8196065    0.8176059 0.8832944
## Specificity          0.2503771    0.3152338    0.3529412 0.5806938
## F1                   0.8922776    0.8926374    0.8902268 0.9274461
## AUC                  0.6136303    0.6389953    0.6512137 0.7895188
```

The above table shows the evaluation metrics collected for all four models used in this project.

```
##
## Attaching package: 'reshape'

## The following object is masked from 'package:class':
##
##     condense
```



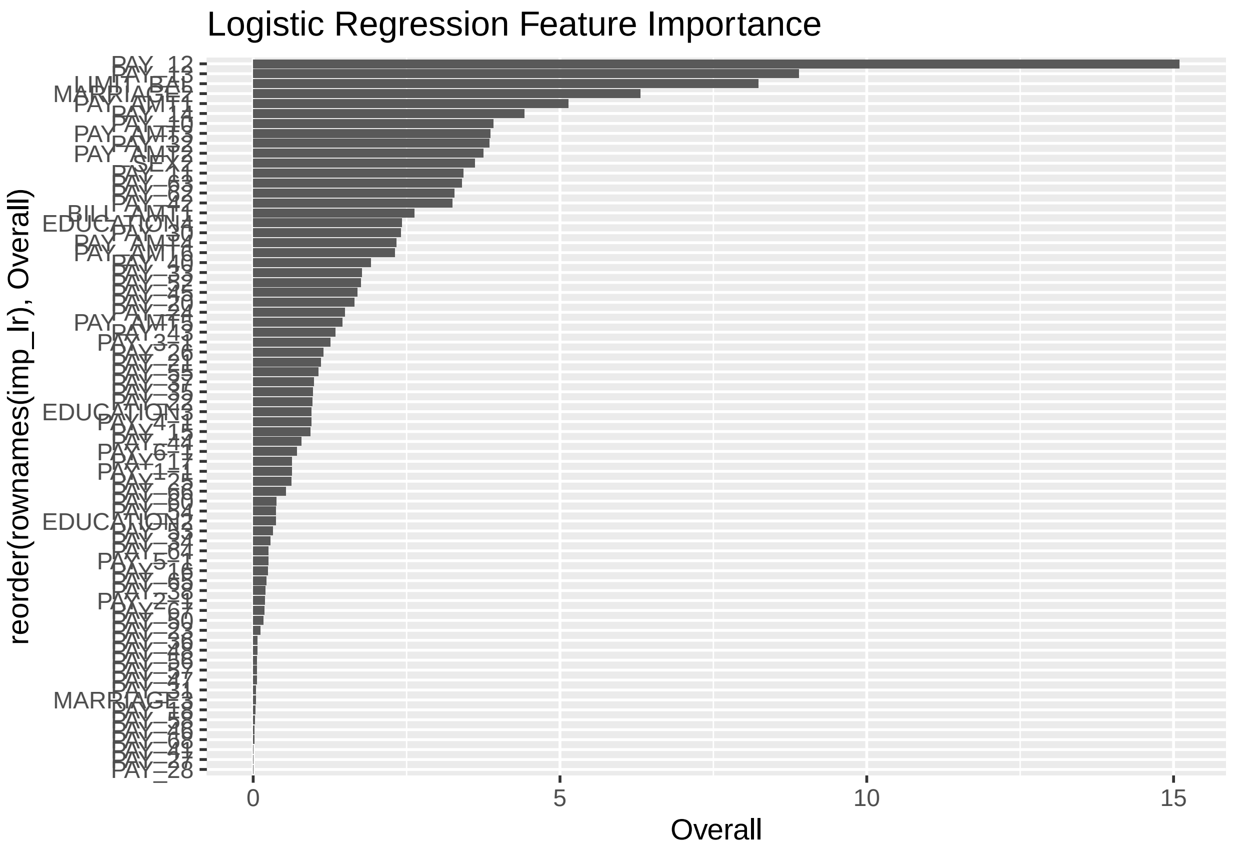
The above graph shows the same metrics shown in the table, but in a visual format.

The k-NN model appears to be the best model in our analysis. It has an F1 score of 0.926 and an AUC of 0.79, compared to our other models in which random forest had an F1 score 0.890 and an AUC = 0.65, decision tree had an f1 score 0.893 and an AUC = 0.63, linear regression had an F1 score = 0.892 and an AUC = 0.61.

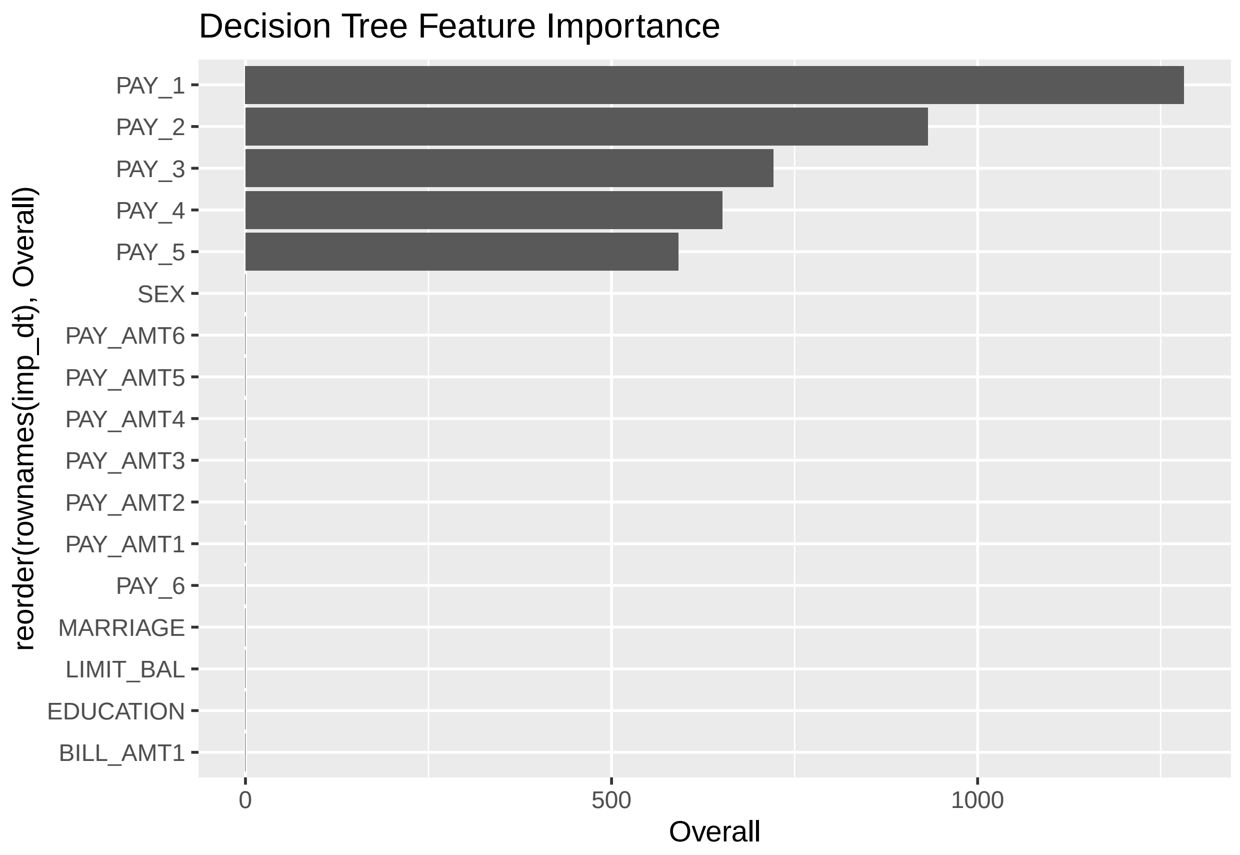
Based on F1 score, random forest had the worst performance, then logistic regression, and the decision tree model came second after KNN, although all F1 scores were quite close to one another.

Feature Importance

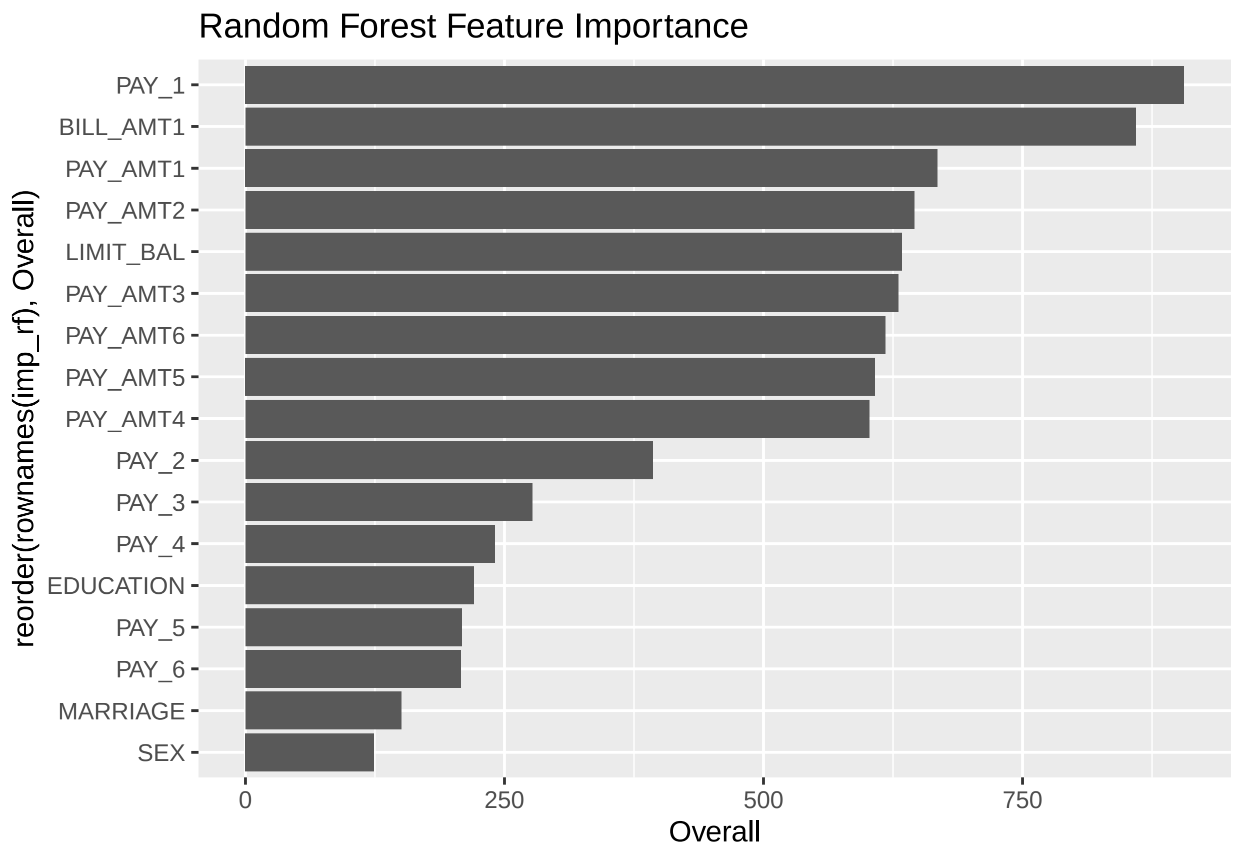
Although all models have been employed and there is a clear difference in their performance, it is still interesting to see a level contribution of each feature to model behavior. Thus caret varImp() function was used to calculate feature importance for models like Logistic Regression, Decision Tree and Random Forest.



From this plot, it seemse that PAY_1 (repayment status in September 2005) seems to be the most important.



From this plot, which shows the feature importance graph for the decision tree model, it also seems that PAY_1 is the most important feature.

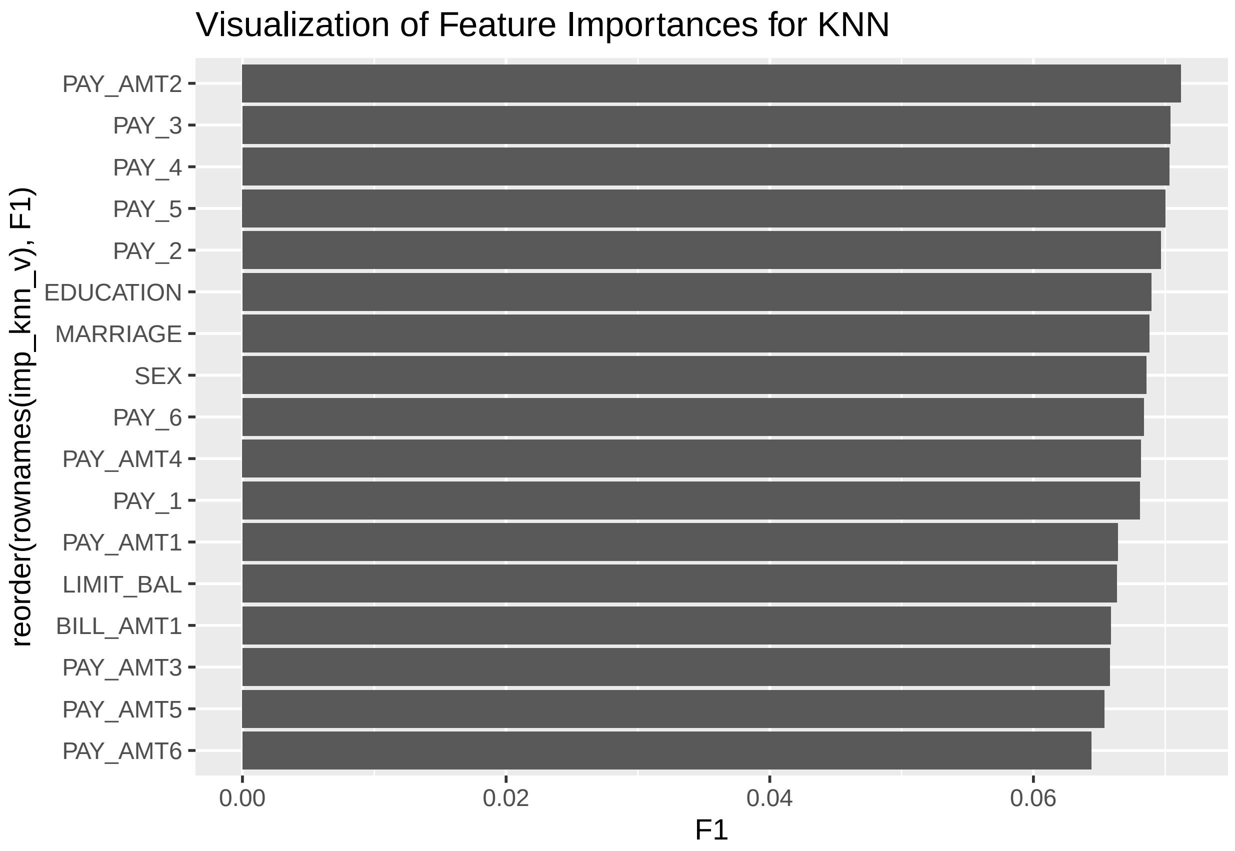


From this plot, which shows the feature importance graph for the random forest model, it also replicated the same results from the previous two graphs that PAY_1 is the most important feature.

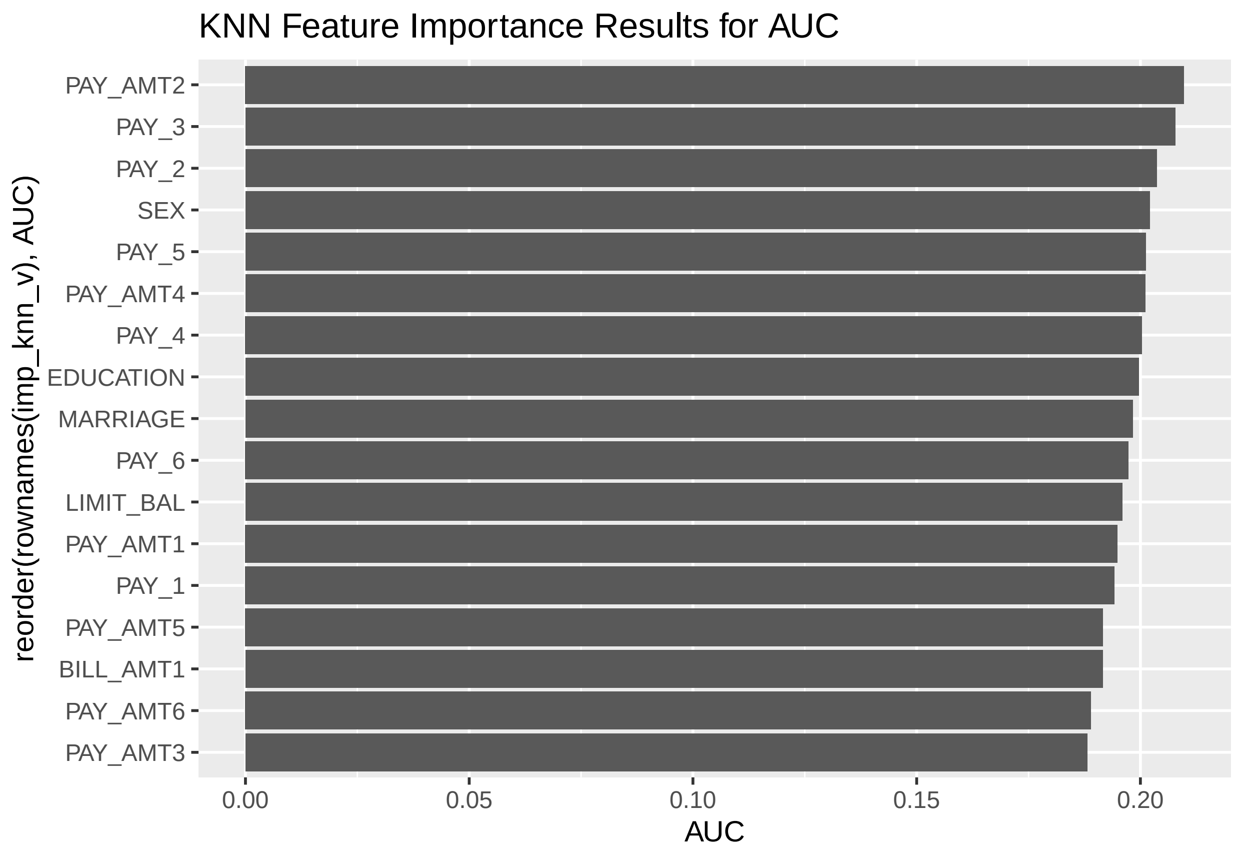
It was discovered that in contrast to other models, R does not provide any built-in functions to evaluate feature importance for the k-NN model. Therefore, it was decided to build a custom code that would calculate metrics of k-NN model which is run by removing a predictor variable at a time. So, the k-NN model 17 times with missing one of 17 predictor variables every time. While the metrics of every run is recorded in the corresponding row of the removed variable name, it is possible to see how metrics change after removing a particular variable.

```
##           Precision    Recall  Accuracy Specificity        F1       AUC
## LIMIT_BAL 0.9051447 0.9640411 0.8932978   0.6440422 0.9336650 0.8040417
## SEX       0.9024488 0.9623288 0.8896299   0.6334842 0.9314274 0.7979065
## EDUCATION 0.9040323 0.9597603 0.8892964   0.6410256 0.9310631 0.8003930
## MARRIAGE  0.9047235 0.9593322 0.8896299   0.6440422 0.9312279 0.8016872
## PAY_1     0.9068449 0.9584760 0.8909637   0.6530920 0.9319459 0.8057840
## PAY_2     0.9019293 0.9606164 0.8879627   0.6319759 0.9303483 0.7962962
## PAY_3     0.8998397 0.9614726 0.8866289   0.6229261 0.9296358 0.7921993
## PAY_4     0.9041262 0.9567637 0.8872958   0.6425339 0.9297005 0.7996488
## PAY_5     0.9035123 0.9580479 0.8876292   0.6395173 0.9299813 0.7987826
## PAY_6     0.9051272 0.9597603 0.8902968   0.6455505 0.9316435 0.8026554
## BILL_AMT1 0.9075495 0.9623288 0.8942981   0.6546003 0.9341367 0.8084645
## PAY_AMT1  0.9057971 0.9631849 0.8932978   0.6470588 0.9336100 0.8051219
## PAY_AMT2  0.8990385 0.9606164 0.8852951   0.6199095 0.9288079 0.7902630
## PAY_AMT3  0.9095702 0.9601884 0.8946315   0.6636501 0.9341941 0.8119192
## PAY_AMT4  0.9028503 0.9627568 0.8902968   0.6349925 0.9318417 0.7988747
## PAY_AMT5  0.9072954 0.9636130 0.8949650   0.6530920 0.9346066 0.8083525
## PAY_AMT6  0.9084677 0.9644692 0.8966322   0.6576169 0.9356312 0.8110430
```

When some features were removed, the model seemed to improve even more. For example, the removal of PAY_AMT6 led to a higher F1 and AUC for the model.



The above graph shows the F1 comparison of the features in the KNN model. According to this graph, it shows that PAY_AMT2 is the most important feature, while PAY_AMT6 is the least important.



The above graph shows the AUC comparison of the features in the KNN model. According to this graph, it shows that PAY_AMT2 is the most important feature, while PAY_AMT3 is the least important.

Limitations

One of the striking limitations of the dataset was the short time period of data collection (April to September 2005). The data could have been more robust if bill history was collected over a longer period of time, such as 2 years, instead of 6 months. The short period of time might not accurately capture the credit card users' behaviour. For instance, in this dataset, credit history was collected mostly during the summer season. Credit cardholders' spending and payment behaviour might be different in other seasons. Also, if the data was collected over a very long period of time, such as 10 years, more interesting insights might be obtained, similar to those observed during the 2008 financial crisis.

Other limitations include a fairly low amount of demographic data collected. Only age, marital status, gender, and education level were used to gain insight on the credit card users as individuals. Other data that should have been collected include income, previous loans, dependants. This additional data would have provided more depth into understanding which factors that strongly influenced default payment and thereby result in more robust models.

There were quite a number of outliers in credit limit amount, bill statement amount, and bill payment amount. This is understandable, because data that are related to money, such as housing and income, are generally more susceptible to having outliers, than data like an age. If the dataset was collected over a longer period of time (e.g. 10 years), and if it had more background and demographic information about the credit card users, then outliers could have been removed, since there is a sufficient amount of other data that can help after removing a significant chunk of outliers.

Deployment

This model proves to be useful for financial institutions that issue loans to their customers. Predicting the probability of default payments will provide financial institutions with a more sophisticated risk management approach. For instance, banks can use this model to determine to freeze or cut lines of credit that are likely to go into default. This will save banks hundred of millions of dollars annually (Chen et al., 2016).

The prediction model can be deployed to a user interface through Shinyapp (https://lily-ye.shinyapps.io/test2/). The theory behind the interface is that stakeholders would interact with the panels and widgets in the user interface, such as education level and previous bill history, and obtain an output value of a default credit card payment prediction. This deployment interface can be used by relevant stakeholders in reports and meetings to cast important forecasts of credit card users' repayment ability, and thereby inform their decisions for risk management for credit card debt and operational strategies to minimize risks.

Conclusion

Credit card data was obtained from UCI Machine Learning Repository, and the dataset originated from the Yeh & Lien 2009 study. Information about credit card users was collected in Taiwan over a period of April~September 2005, and the dataset had 30,000 observations and 25 columns. The dataset was utilized to build a predictive model to determine the credit card payment default of credit cardholders based on 17 features that were selected from the correlation strength.

Logistic regression, decision tree, random forest, and KNN models were built, tested, compared, and evaluated. All four models had high scores that were similar to each other, but k-NN model stood out as the best model with the highest score (F1 = 0.93 and AUC = 0.79).

This type of dataset and prediction model can be deployed to be used by bank institutions for risk management. If bank institutions can predict when and when not to issue credit card based on the person's likelihood of having default credit payment, it may help to prevent a large scale debt crisis, similar to the one that occurred in Taiwan in 2006 from over-issuing a credit card (Yeh & Lien, 2009).

Bibliography

Chou, M. (2006). Cash and credit card crisis in Taiwan. Business Weekly, 24–27.

Hashmat, R. (2020). Week 4 Lecture. York University.

Huilgol, P. 2019. Accuracy vs. F1-Score - Analytics Vidhya - Medium. Retrieved April 24, 2020, from https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2

Kuhn, M. (2019). 4 Data Splitting | The caret Package. http://topepo.github.io/caret/data-splitting.html

Molnar, C. (2020.). 4.4 Decision Tree | Interpretable Machine Learning. Retrieved April 24, 2020, from https://christophm.github.io/interpretable-ml-book/tree.html

Narkhede, S. 2018. Understanding AUC - ROC Curve - Towards Data Science. Retrieved April 24, 2020, from https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Read Random Forest-Random Forest (4 implementation steps + 10 advantages and disadvantages). (n.d.). Retrieved April 24, 2020, from https://easyai.tech/en/ai-definition/random-forest/

Shung, K. P. (2019). Accuracy, Precision, Recall or F1? - Towards Data Science. Retrieved April 24, 2020, from https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

UCI Machine Learning Repository. (2009). UCI Machine Learning Repository: Default of credit card clients Data Set. https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

What is the K-nearest neighbors|KNN? - Product Manager's Artificial Intelligence Learning Library. (n.d.). Retrieved April 24, 2020, from https://easyai.tech/en/ai-definition/k-nearest-neighbors/

Yeh, I.-C., & Lien, C. (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. https://doi.org/10.1016/j.eswa.2007.12.020