

Project 2

Nerurkar Jui

1/31/2019

Introduction:

Credit card data is quite insightful and provides a considerable amount of information about the consumers, which can help in preventing crimes against consumers as well as merchants. However, in addition to the losses suffered by consumers and merchants, banks have to suffer the highest amount of losses due to the credit card frauds and defaulters. Hence, it is very crucial to develop a method to solve this pressing problem by predicting fraudulent behavior or default in payment using the records from past transactions.

Data:

This dataset (downloaded from the UCI Machine Learning Repository) contains information on default payments, demographic factors, payment history, and bill statements from April 2005 to September 2005 of credit card holders in Taiwan. This dataset uses 23 variables as explanatory variables/predictors, out of which 9 are categorical and the remaining are numerical.

It is an imbalanced dataset with only 6636 observations belonging to default payment = 1 category, i.e only 22% of the data.

The data description given by the UCI website is as follows: A binary variable, default payment is the response variable where Default = 1 and No Default = 0.

Amount of the given credit (in NT dollar) includes both the individual consumer credit and his/her family (supplementary) credit.

Gender: 1 = male; 2 = female.

Education: 1 = graduate school; 2 = university; 3 = high school; 4 = others.

Marital status: 1 = married; 2 = single; 3 = others.

Age: Age in years.

History of past payments:

PAY_0: the repayment status in September, 2005;

PAY_2 = the repayment status in August, 2005; . . .

PAY_6 = the repayment status in April, 2005.

The measurement scale for the repayment status is:

-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.

Amount of bill statement (NT dollar):

BILL_AMT1 = amount of bill statement in September, 2005;

BILL_AMT2 = amount of bill statement in August, 2005; . . .;

BILL_AMT6= amount of bill statement in April, 2005.

Amount of previous payment (NT dollar):

PAY_AMT1 = amount paid in September, 2005;

PAY_AMT2 = amount paid in August, 2005; . . .;

PAY_AMT6 = amount paid in April, 2005.

It is important to note that the dataset does not provide enough information about the response variable i.e what conditions need to be satisfied to say that a credit card holder has defaulted in making the credit card bill payments? On moneycontrol.com, default is defined as “When payments are not made in time and according to the agreement signed by the card holder, the account is said to be in default.”

For the purposes of this analysis, I have assumed that every payment which is not made before 3 months is termed as default.

Analysis:

Checking the data for missing values:

The repayment status columns have 0 and -2 values which are undocumented and hence, can be treated as missing values. Upon analyzing the data to understand the allocation of 0 and -2 to the repayment status, I realized that there are 25,939 rows with repayment status values of 0 and -2 and hence, removing them would be a significant loss of data and information.

Transforming the data:

Deleted the first row which was the ID column.

All repayment status values in the dataset - credit less than 3 were replaced with 0 (non-defaulters) and more than or equal to 2 were replaced with 1. Thereafter, I calculated the total number of defaults for every observation and created a new variable “Number_of_Defaults” to store those values.

Plotted the histograms for the numerical variables. Except limit balance and age, all the other numerical variables are heavily skewed and hence, I used the log transformations on those variables. However, those variables have negative values. Since, log of negative values do not exist and need a much more detailed evaluation, I restricted the said variables only to positive values. Finally, I had a dataset of 15,049 observations with 2106 observations, i.e approximately 14% belonging to the default payment = 1 category.

After the log transformations, I checked the numeric range of the different features which varies quite substantially. Hence, I normalized the numerical variables in the given dataset.

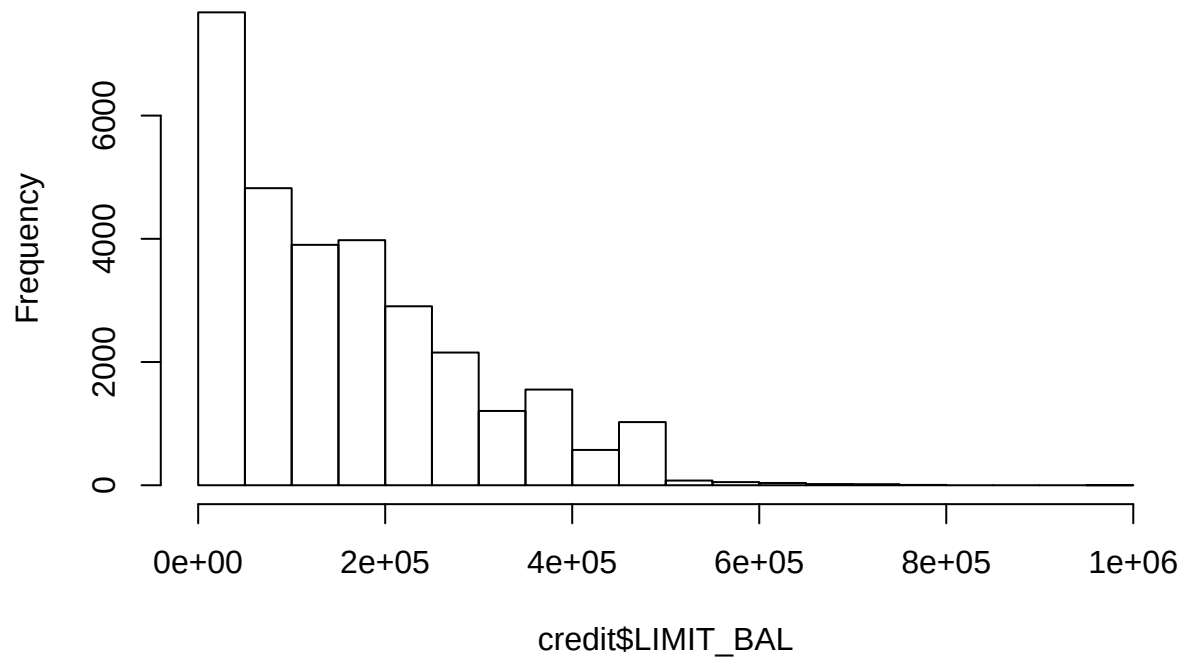
The dataset obtained after these transformations shall be referred to as the transformed dataset- credit.

```
credit <- read.csv("Credit_Card_default.csv")
credit <- credit[,-1]

##Calculate Total Number of defaults for every observation
credit <- credit %>% mutate(PAY_0 = ifelse(PAY_0 < 3, 0, 1))
credit <- credit %>% mutate(PAY_2 = ifelse(PAY_2 < 3, 0, 1))
credit <- credit %>% mutate(PAY_3 = ifelse(PAY_3 < 3, 0, 1))
credit <- credit %>% mutate(PAY_4 = ifelse(PAY_4 < 3, 0, 1))
credit <- credit %>% mutate(PAY_5 = ifelse(PAY_5 < 3, 0, 1))
credit <- credit %>% mutate(PAY_6 = ifelse(PAY_6 < 3, 0, 1))
for (i in 1:nrow(credit)){
  credit$Number_of_defaults[i] <- (credit$PAY_0[i] + credit$PAY_2[i] + credit$PAY_3[i] + credit$PAY_4[i] + credit$PAY_5[i] + credit$PAY_6[i])
}

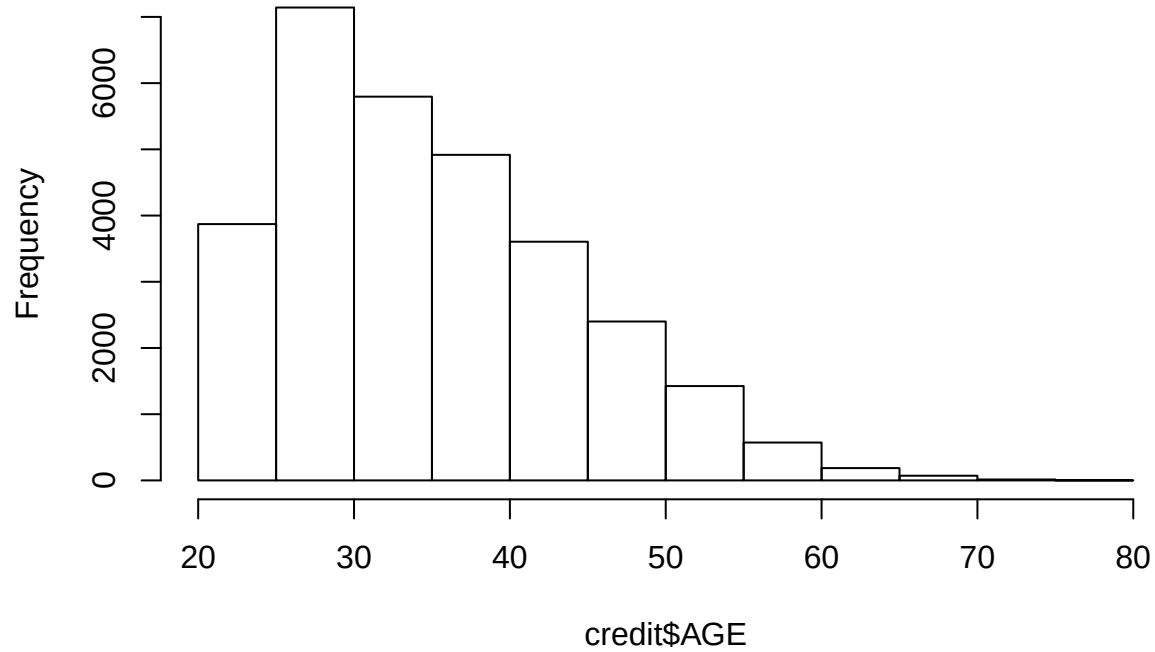
#Univariate plots for variables
hist(credit$LIMIT_BAL, main = "Histogram - Limit Balance")
```


Histogram – Limit Balance



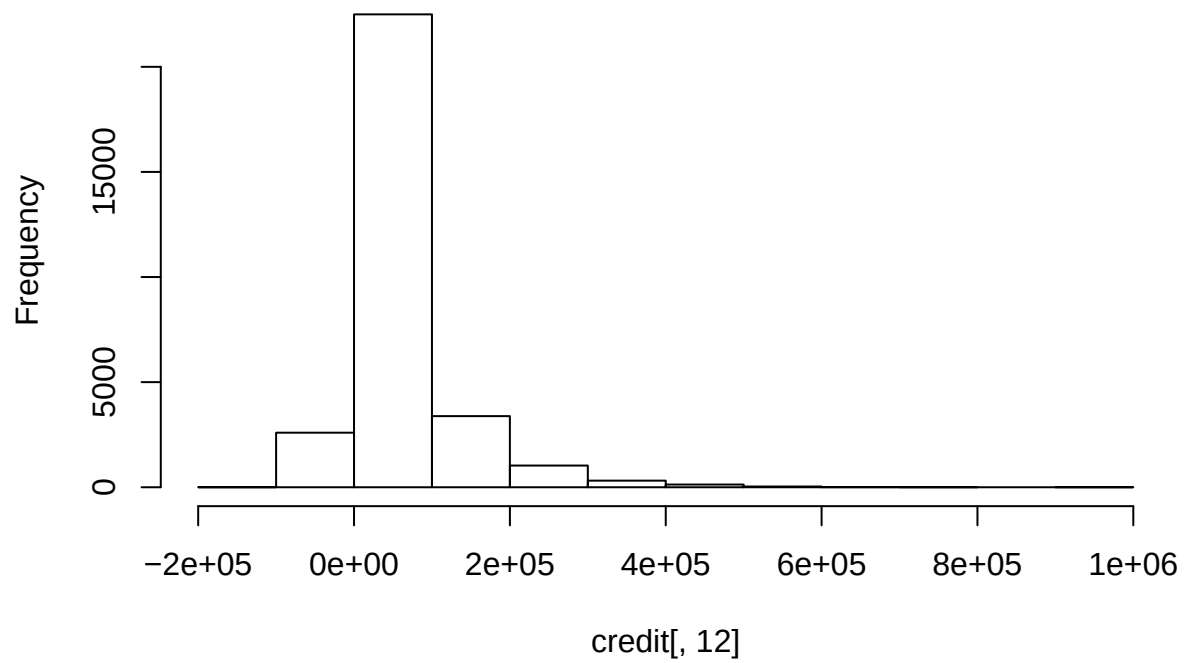
```
hist(credit$AGE, main = "Histogram - Age")
```

Histogram – Age



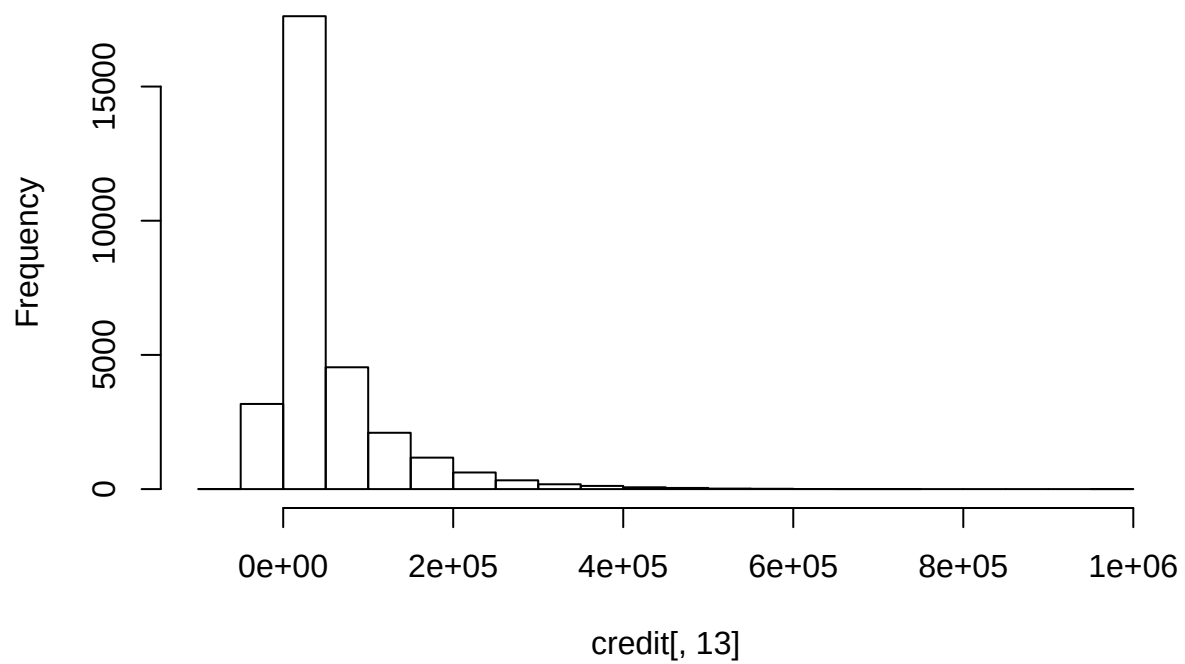
```
hist(credit[,12], main = "Histogram for BILL_AMT1")
```

Histogram for BILL_AMT1



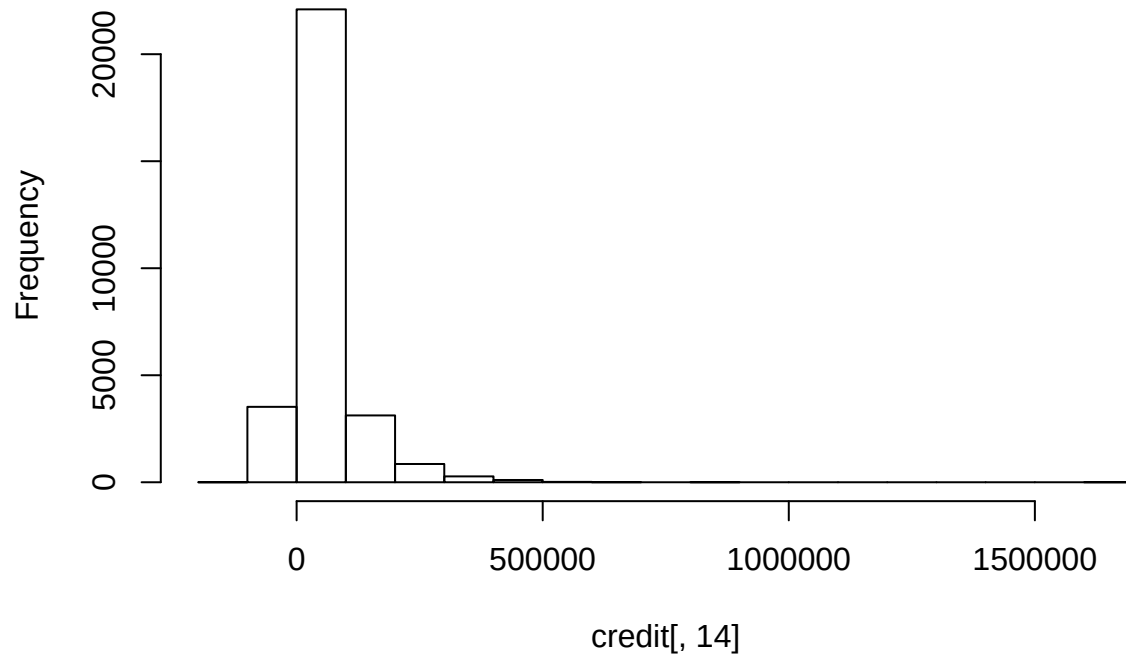
```
hist(credit[,13], main = "Histogram for BILL_AMT2")
```

Histogram for BILL_AMT2



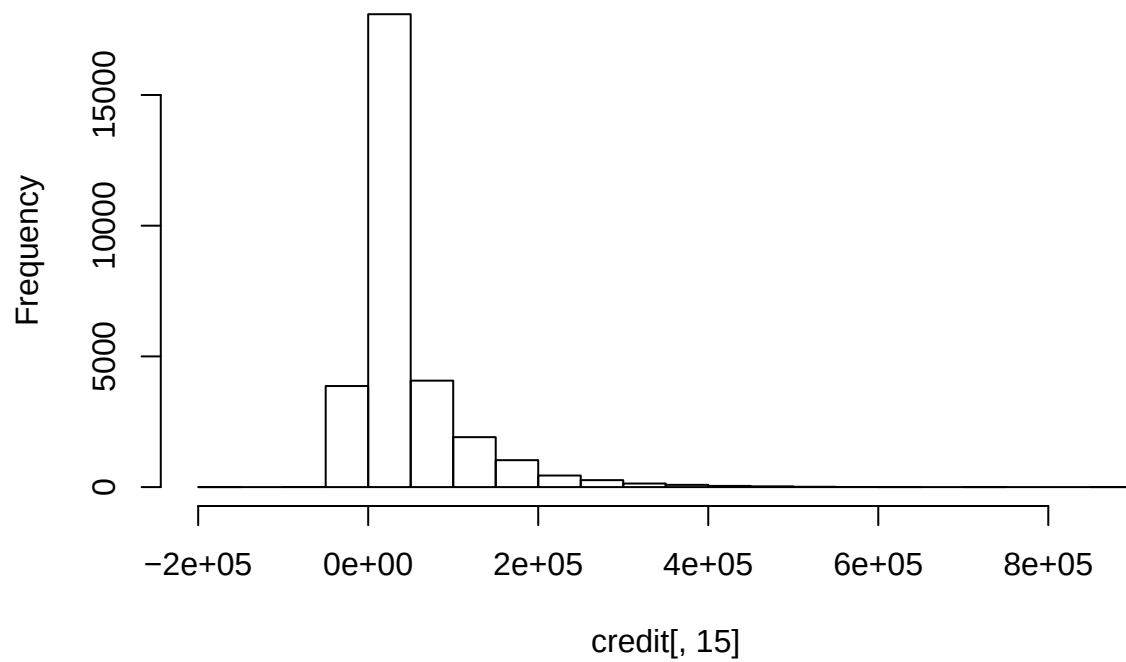
```
hist(credit[,14], main = "Histogram for BILL_AMT3")
```

Histogram for BILL_AMT3



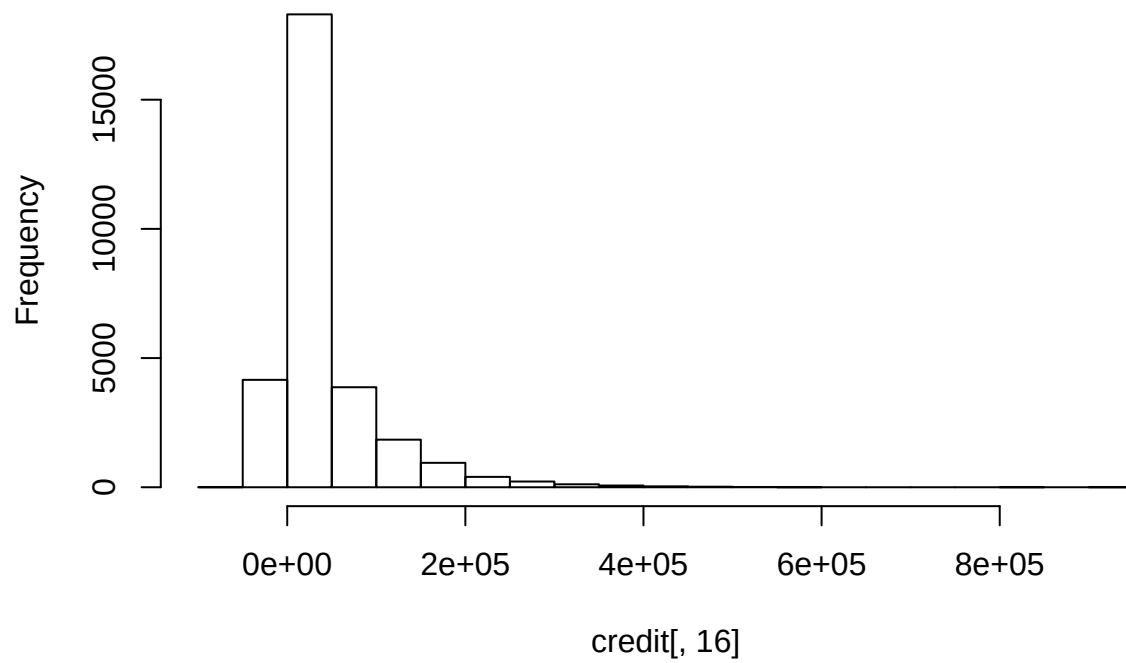
```
hist(credit[,15], main = "Histogram for BILL_AMT4")
```

Histogram for BILL_AMT4



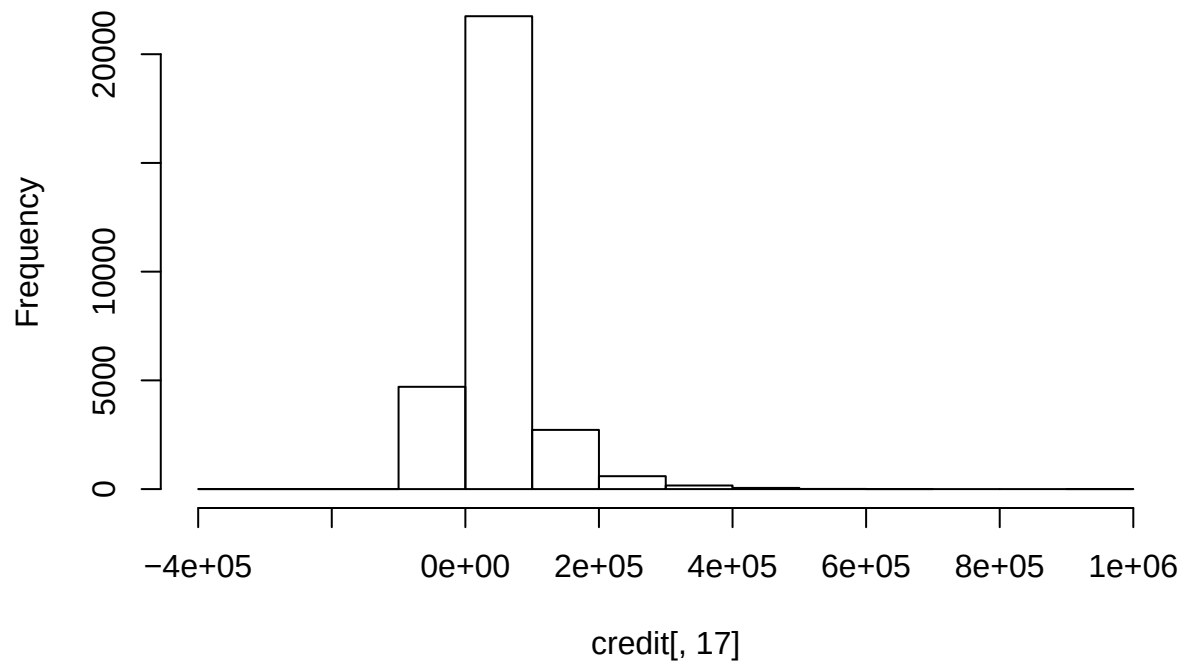
```
hist(credit[,16], main = "Histogram for BILL_AMT5")
```

Histogram for BILL_AMT5



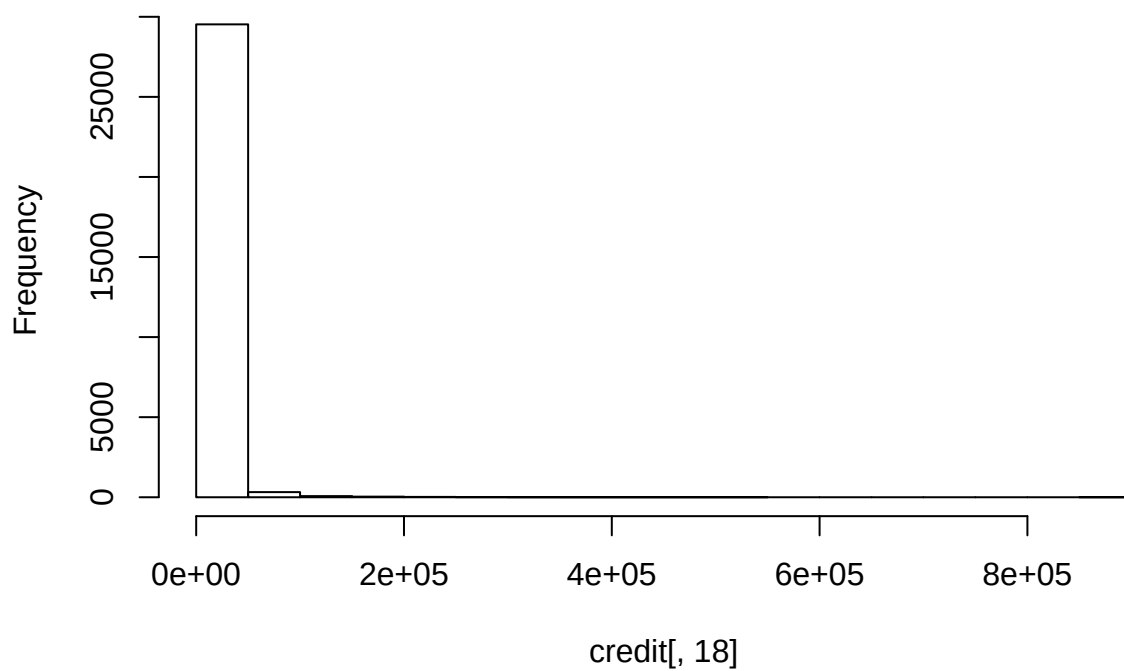
```
hist(credit[,17], main = "Histogram for BILL_AMT6")
```

Histogram for BILL_AMT6



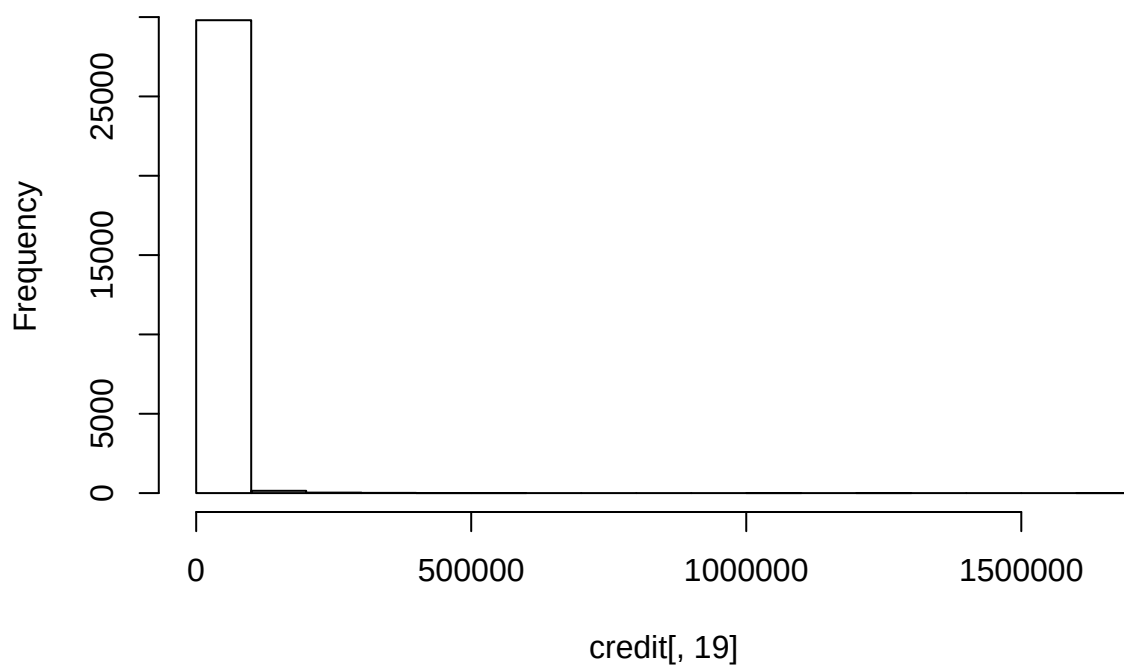
```
hist(credit[,18], main = "Histogram for PAY_AMT1")
```

Histogram for PAY_AMT1



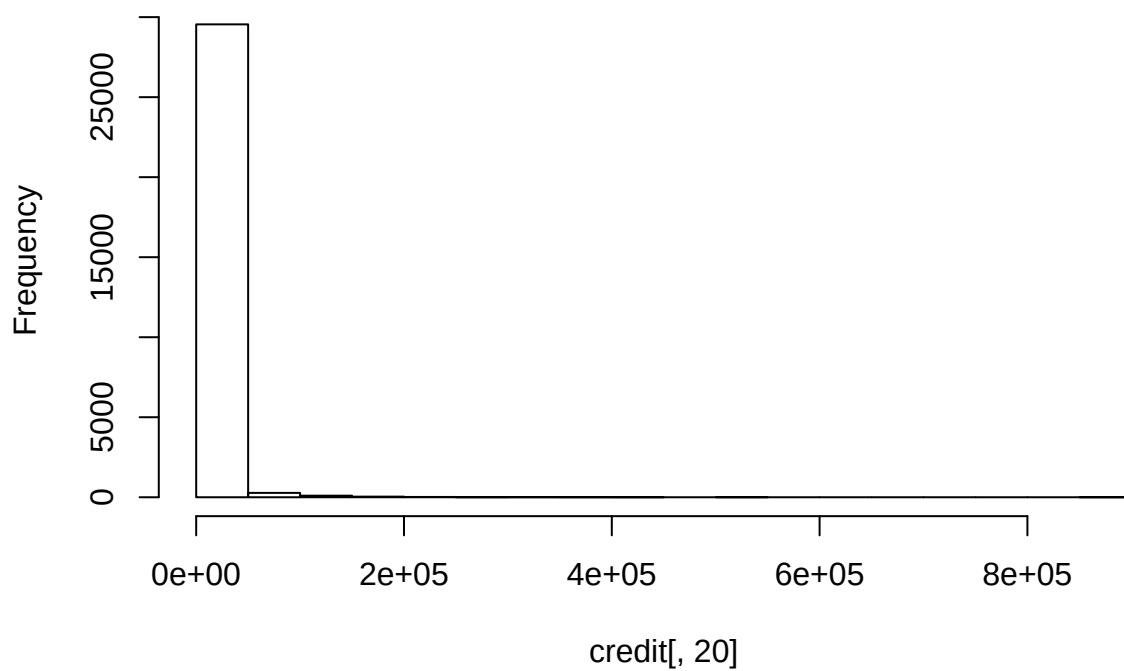
```
hist(credit[,19], main = "Histogram for PAY_AMT2")
```

Histogram for PAY_AMT2



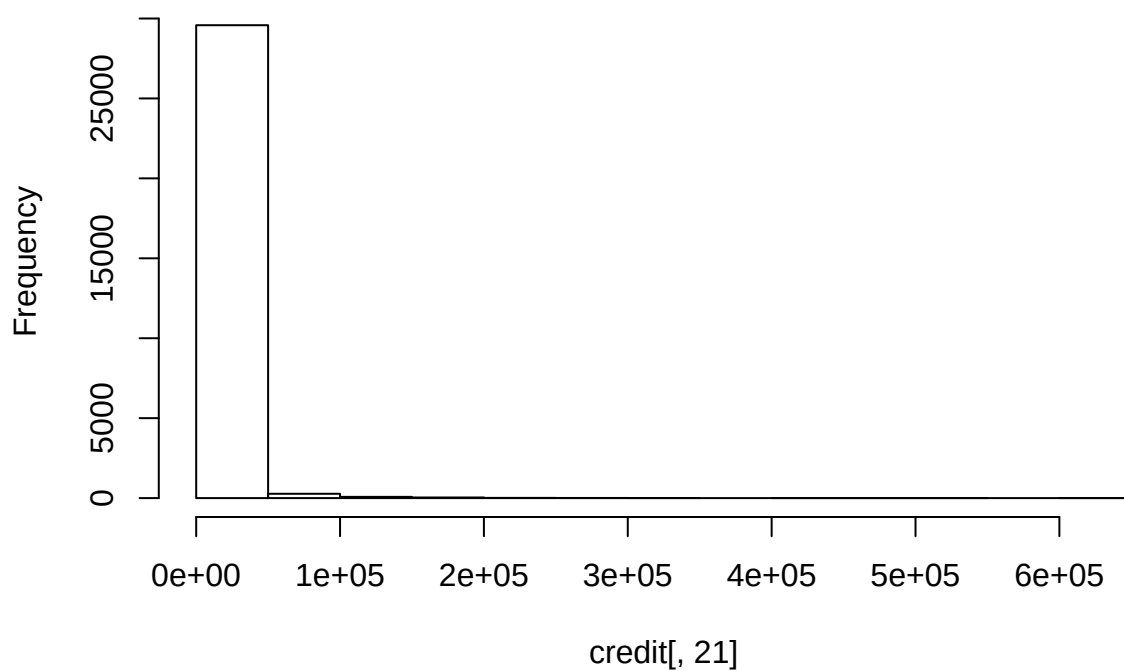
```
hist(credit[,20], main = "Histogram for PAY_AMT3")
```

Histogram for PAY_AMT3



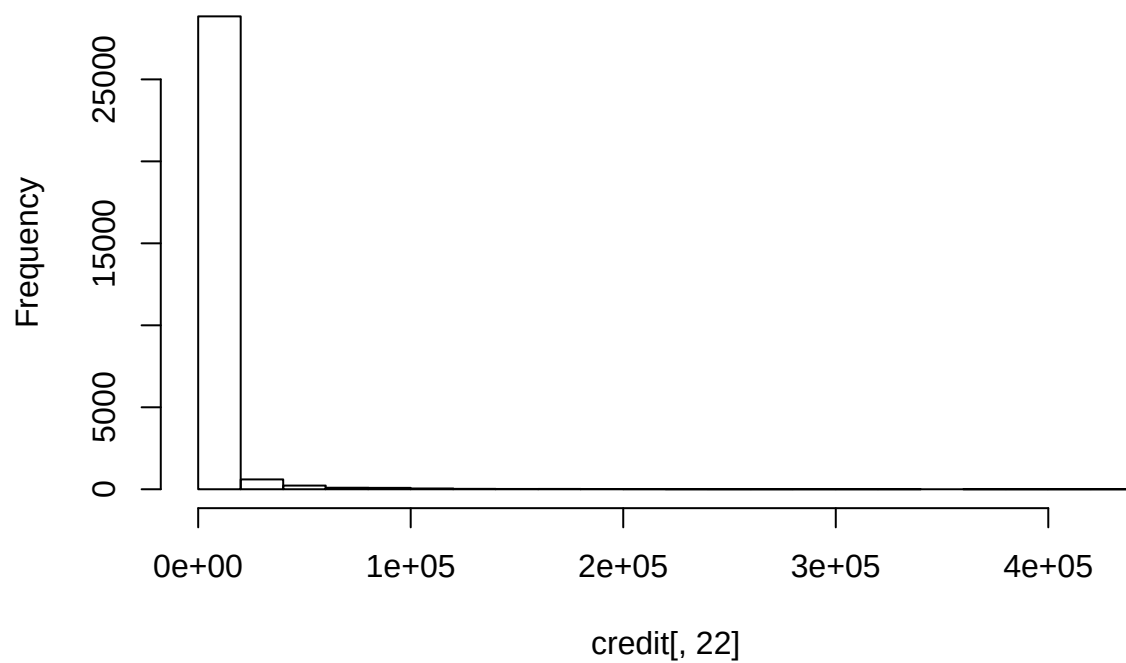
```
hist(credit[,21], main = "Histogram for PAY_AMT4")
```

Histogram for PAY_AMT4



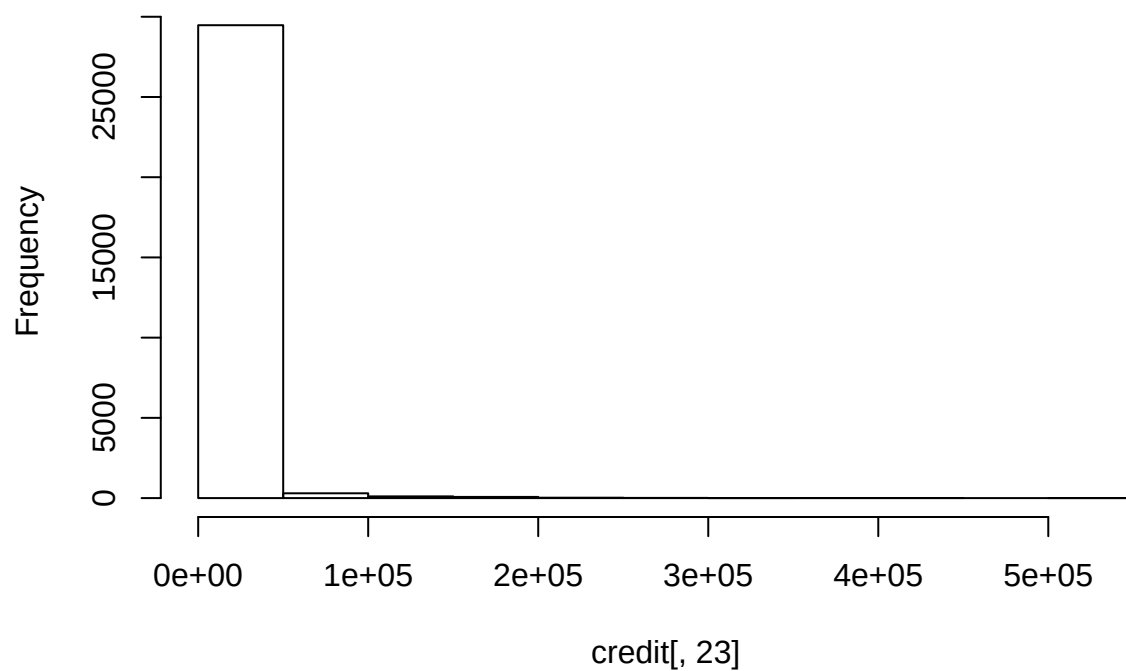
```
hist(credit[,22], main = "Histogram for PAY_AMT5")
```

Histogram for PAY_AMT5



```
hist(credit[,23], main = "Histogram for PAY_AMT6")
```

Histogram for PAY_AMT6



```
#Restrict bill and payment amounts to positive values  
for (i in 12:23){  
  credit <- credit[credit[,i] > 0, ]  
}
```

```

}

#Log transformations:
for (i in 12:23){
  credit[,i] <- log(credit[,i])
}

#Scale transformations:
credit$LIMIT_BAL <- scale(credit$LIMIT_BAL)
credit$AGE <- scale(credit$AGE)
for (i in 12:23){
  credit[,i] <- scale(credit[,i])
}

```

Visualization

- Bivariate plots
- Correlation map
- Principal Component Analysis

For these techniques, I saw a remarkable difference in the correlation plots and segregation of observations belonging to defaulters and non-defaulters after removing the missing data (repayment status = -2 and 0) which I had referred to above.

Hence, only for the purposes of these visualization techniques, in addition to the transformations used hereinbefore, I also removed the observations with missing data from the original UCI dataset.

For all the classification methods, I have used the transformed dataset credit and not this dataset cred.

```

cred <- read.csv("Credit_Card_default.csv")
cred <- cred[,-1]

#Restrict repayment status to -1 and values strictly greater than 0
for (i in 6:11){
  cred <- cred[cred[,i] == -1 | cred[,i] > 0, ]
}

##Calculate Total Number of defaults for every observation
cred <- cred %>% mutate(PAY_0 = ifelse(PAY_0 < 3, 0, 1))
cred <- cred %>% mutate(PAY_2 = ifelse(PAY_2 < 3, 0, 1))
cred <- cred %>% mutate(PAY_3 = ifelse(PAY_3 < 3, 0, 1))
cred <- cred %>% mutate(PAY_4 = ifelse(PAY_4 < 3, 0, 1))
cred <- cred %>% mutate(PAY_5 = ifelse(PAY_5 < 3, 0, 1))
cred <- cred %>% mutate(PAY_6 = ifelse(PAY_6 < 3, 0, 1))
for (i in 1:nrow(cred)){
  cred$Number_of_defaults[i] <- (cred$PAY_0[i] + cred$PAY_2[i] + cred$PAY_3[i] + cred$PAY_4[i] + cred$PAY_5[i] + cred$PAY_6[i])
}

#Restrict bill and payment amounts to positive values
for (i in 12:23){
  cred <- cred[cred[,i] > 0, ]
}

#Log transformations:
for (i in 12:23){
  cred[,i] <- log(cred[,i])
}

```



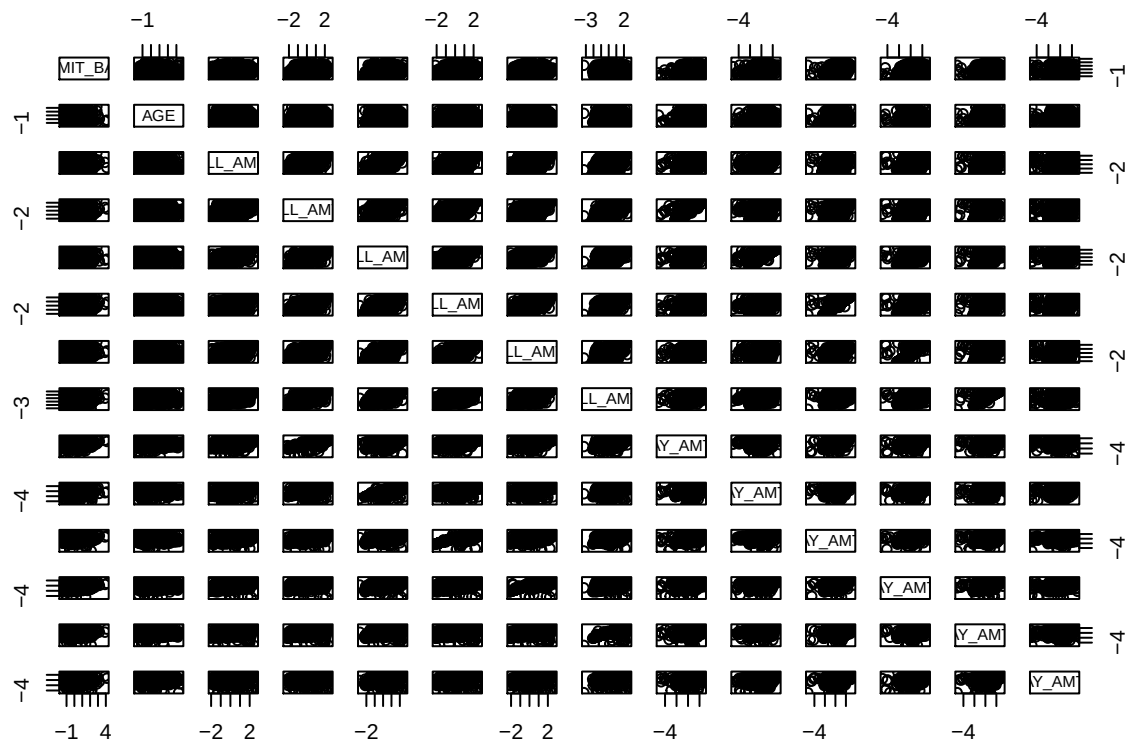
```

}

#Scale tranformations:
cred$LIMIT_BAL <- scale(cred$LIMIT_BAL)
cred$AGE <- scale(cred$AGE)
for (i in 12:23){
  cred[,i] <- scale(cred[,i])
}

#Bivariate plots for continuous variables
pairs(cred[,c(1,5,12:23)])

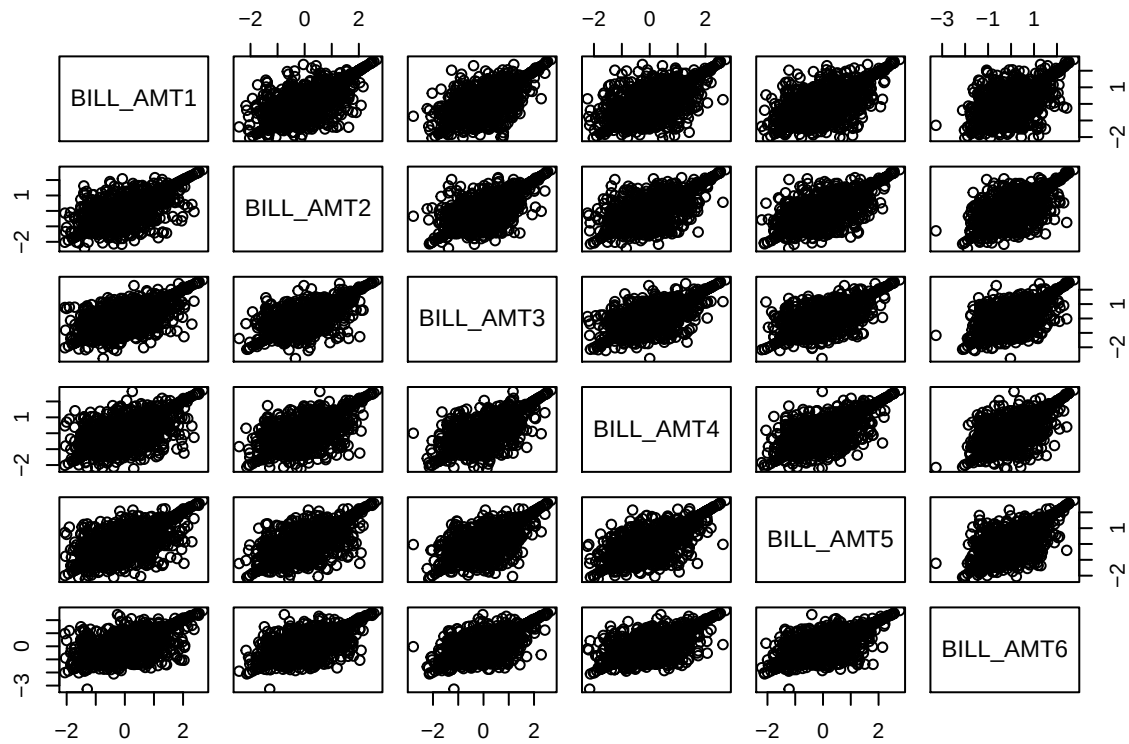
```



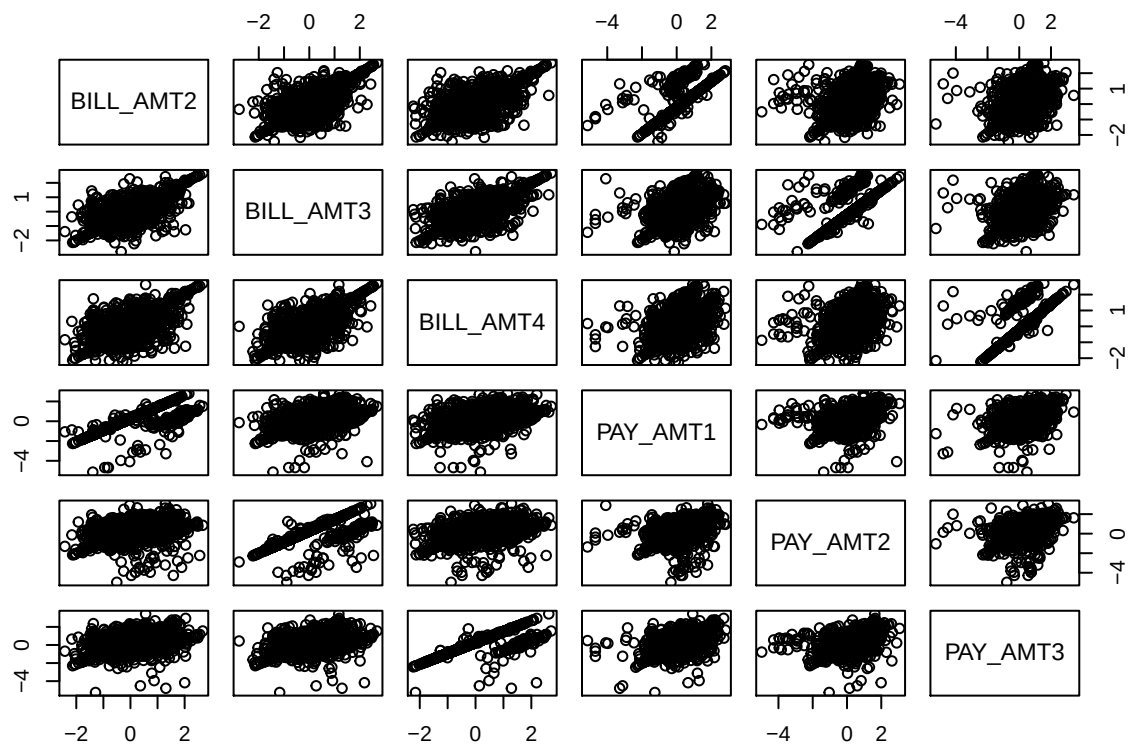
```

pairs(cred[,c(12:17)])

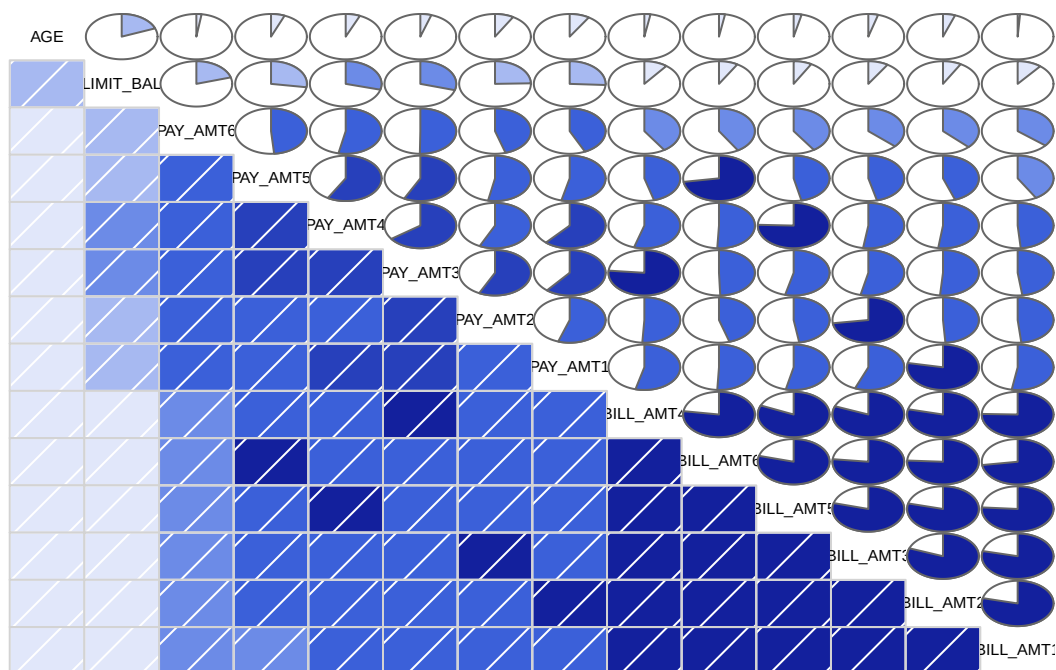
```



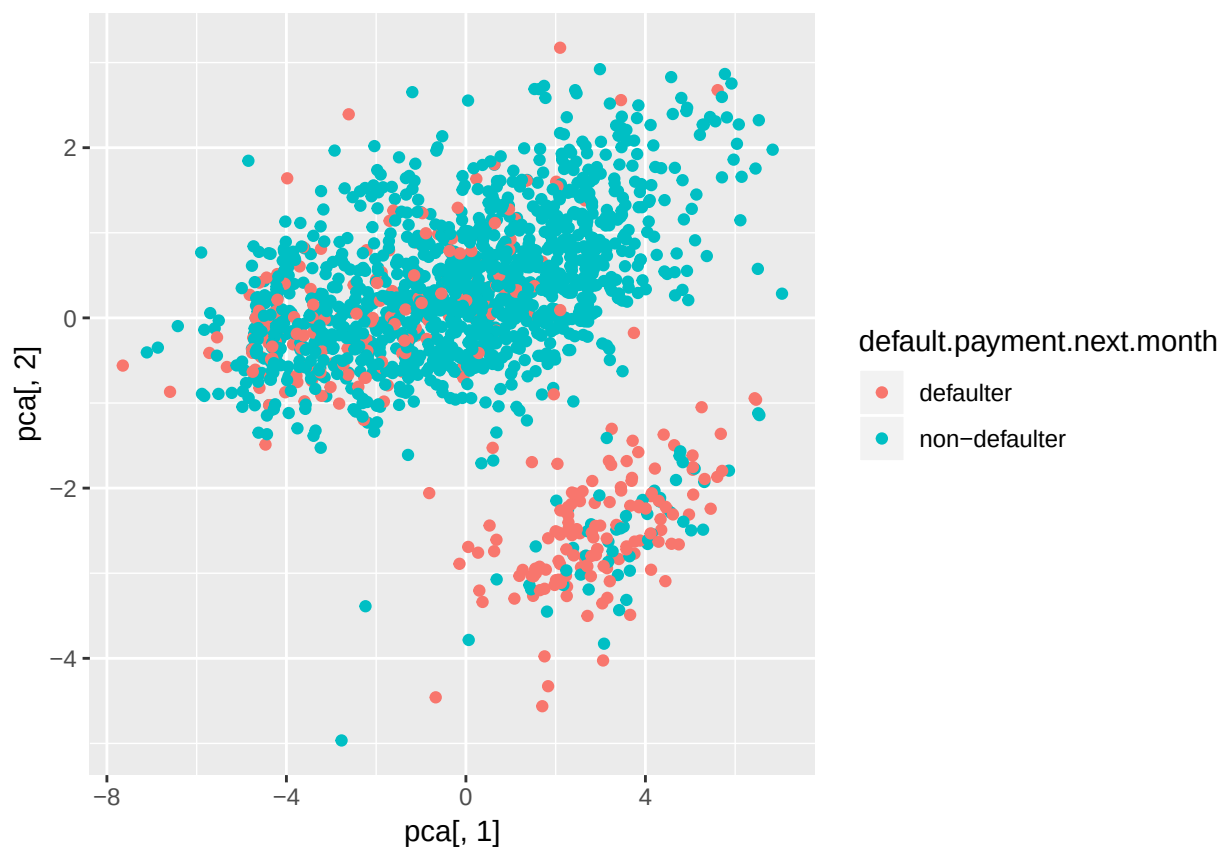
```
pairs(cred[,c(13:15,18:20)])
```



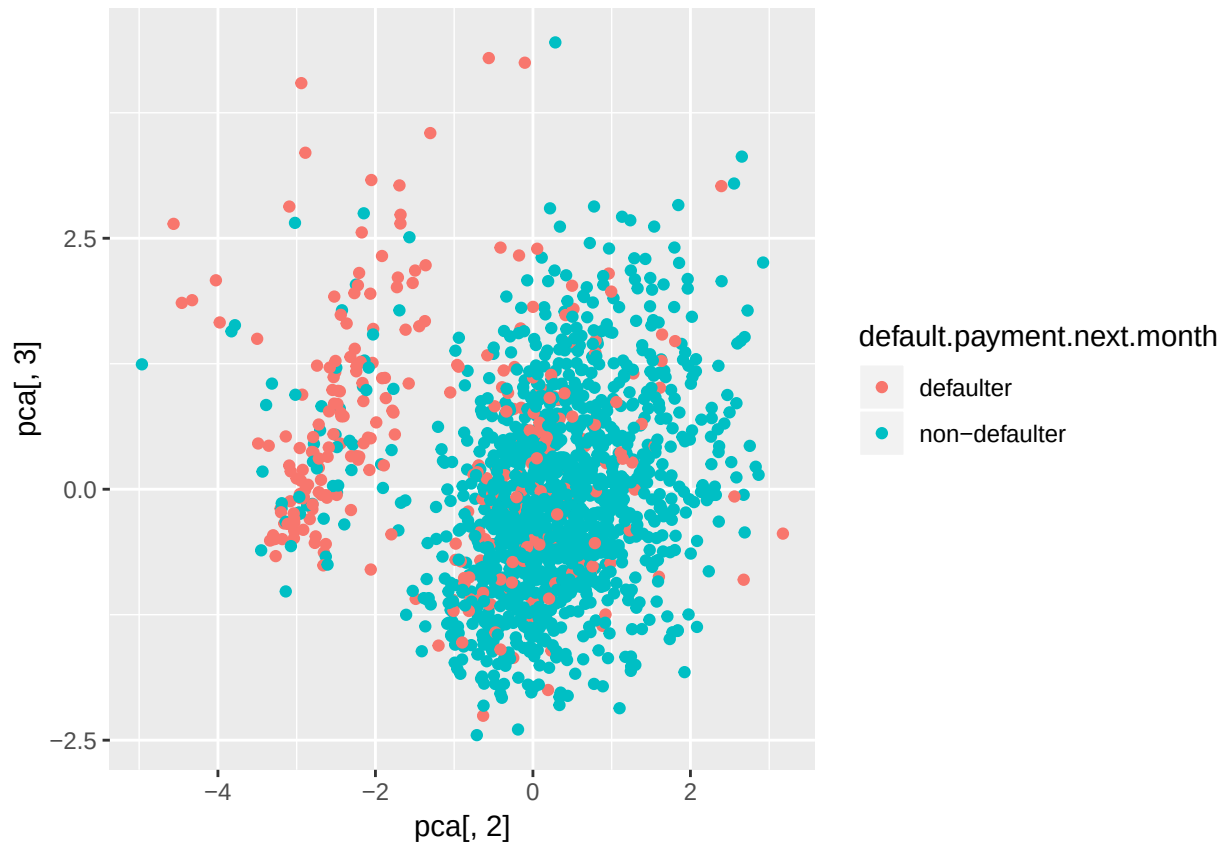
```
##Correlation map
corrgram(cred[,c(1,5,12:23)], order = TRUE, lower.panel = panel.shade,
         upper.panel = panel.pie, text.panel = panel.txt)
```



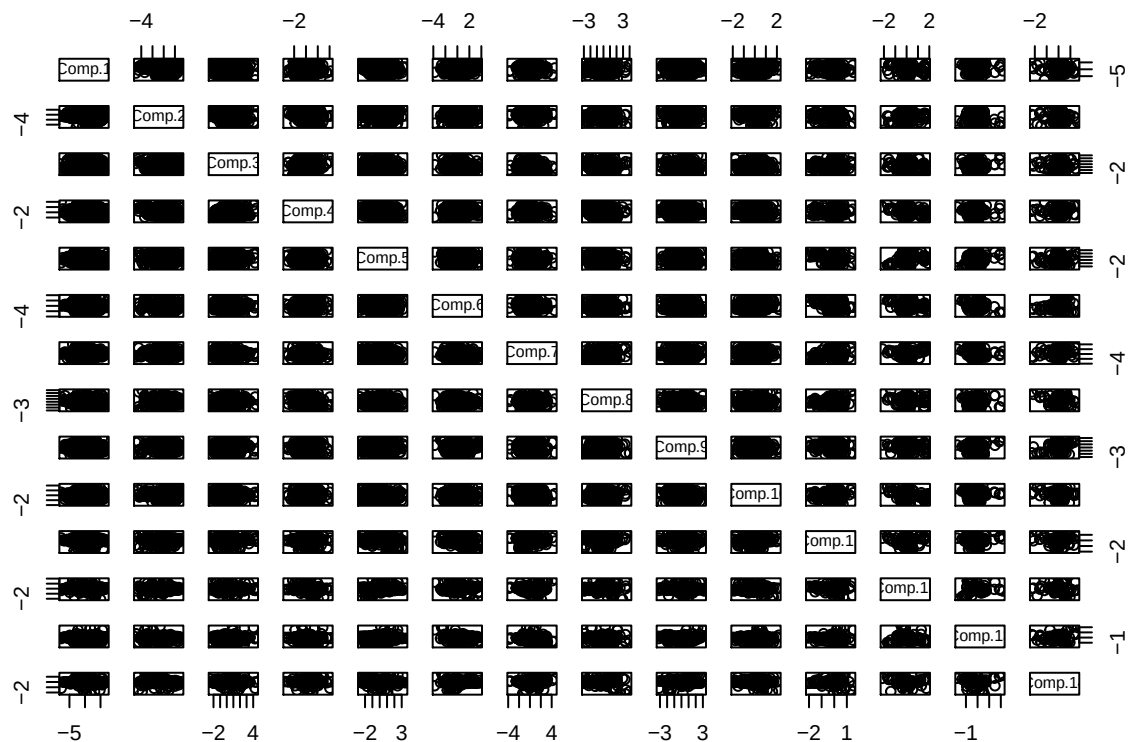
```
##PCA analysis
cred.new <- cred
cred.new$default.payment.next.month <- ifelse(cred.new$default.payment.next.month == 1, "defaulter", "non-defaulter")
pca <- princomp(cred.new[,c(1,5,12:23)], cor = T)$scores
ggplot(data = cred.new, aes(pca[,1], pca[,2], colour = default.payment.next.month))+geom_jitter()
```



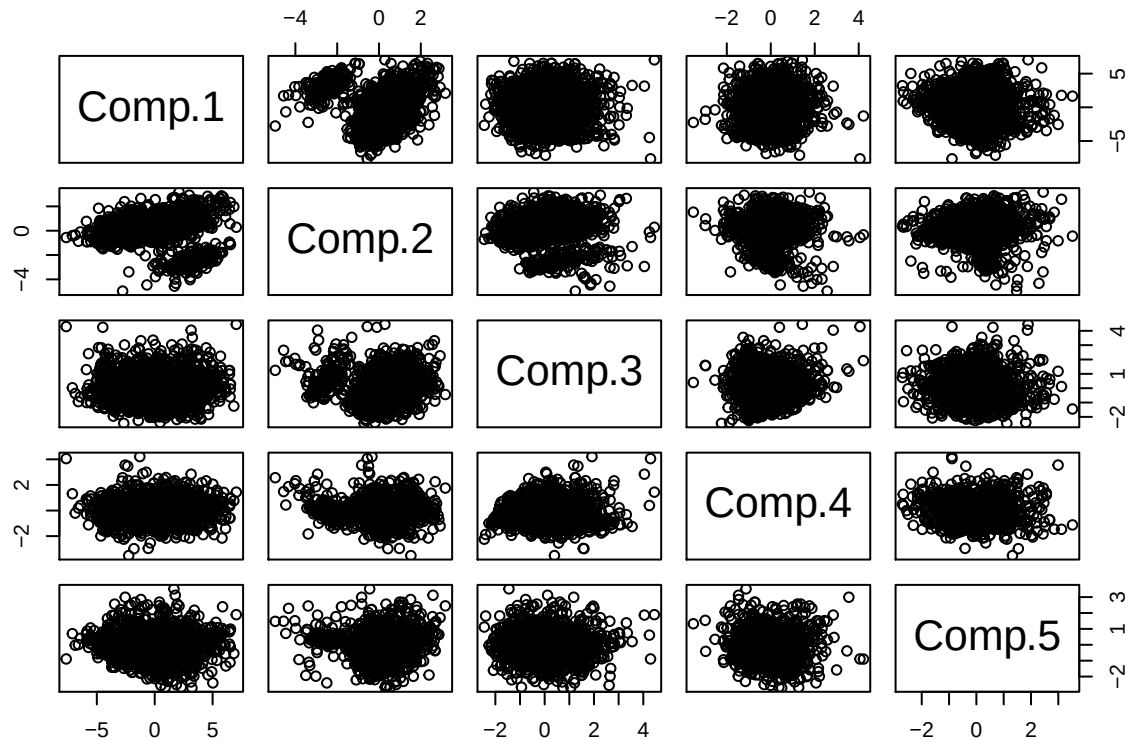
```
ggplot(data = cred.new, aes(pca[,2], pca[,3], colour = default.payment.next.month))+geom_jitter()
```



```
pairs(pca)
```



```
pairs(pca[,1:5])
```



Bivariate plots:

From the bivariate plots, we can see strong linear correlation between the different bill amounts as well as between the bill amounts and amounts paid. For example: there is a strong linear correlation between BILL_AMT2 and PAY_AMT1

Correlation Map:

The correlation map, like the bivariate plots, shows a strong correlation between the different bill amounts as well as between the bill amounts and amounts paid. For example: there is a strong linear correlation between BILL_AMT2 and PAY_AMT1

Principal Component Analysis:

Using the continuous variables in the given dataset, I calculated the principal components and plotted the pairs of the first five components. The plots of first versus second component and second versus third component show two distinct clusters.

However, when I colored them using the default payment values, the larger cluster seems to have some defaulter observations.

Hence, though the observations are not completely separated according to their default payment status, with such an imbalanced dataset, I think PCA does well in forming the two clusters, one with majority observations belonging to default payment = 1 status and other with majority observations belonging to default payment = 0 status .

Balancing the dataset and split into train and test datasets:

The transformed dataset credit has only 14% observations for defaulters and is quite imbalanced. Hence, I used the random undersampling technique to balance the dataset for the defaulter and non-defaulter observations.

The dataset thus obtained is also called credit with 4137 observations, out of which 51% belong to the defaulters category.

Henceforth, every reference to dataset or original/transformed dataset would imply a reference to this new balanced dataset - credit.

Thereafter, I divided this dataset into two parts in the ratio of 80:20

The new data with 80% of the observations is termed as train dataset whereas the remaining part of the dataset "credit" is termed as test dataset.

The train dataset has 3309 observations, whereas the test dataset has 828 observations. Approximately 52% of the observations in the train dataset belong to the default payment = 1 category.

I deleted the columns corresponding to repayment status from April 2005 to September 2005 and used my newly created variable -Number_of_defaults- for the purposes of classification.

```
set.seed(2011)
##Undersampling to create a balanced dataset
credit <- credit[,-(6:11)]
credit_balanced <- ovun.sample(as.factor(default.payment.next.month) ~.,data = credit, method = "under")
credit <- credit_balanced

percentDefault <- sum(credit$default.payment.next.month)/nrow(credit)
percentDefault

## [1] 0.5090645

##Create train and test datasets
n.total.new <- nrow(credit)
n.train.new <- floor(0.80 * nrow(credit))
training.idx.new <- sample.int(n.total.new, n.train.new, replace=FALSE)
train <- credit[training.idx.new, ]
test <- credit[-training.idx.new, ]

percentDefaultTrain <- sum(train$default.payment.next.month)/nrow(train)
percentDefaultTrain

## [1] 0.5176791

percentDefaultTest <- sum(test$default.payment.next.month)/nrow(test)
percentDefaultTest

## [1] 0.4746377
```

Logistic Regression:

```
##Logistic regression
fit.logit <- glm(default.payment.next.month ~. , data = train, family = "binomial")
prediction <- predict(fit.logit, newdata = test, type = "response")

test$Prediction.log <- ifelse(prediction > 0.5, 1, 0)

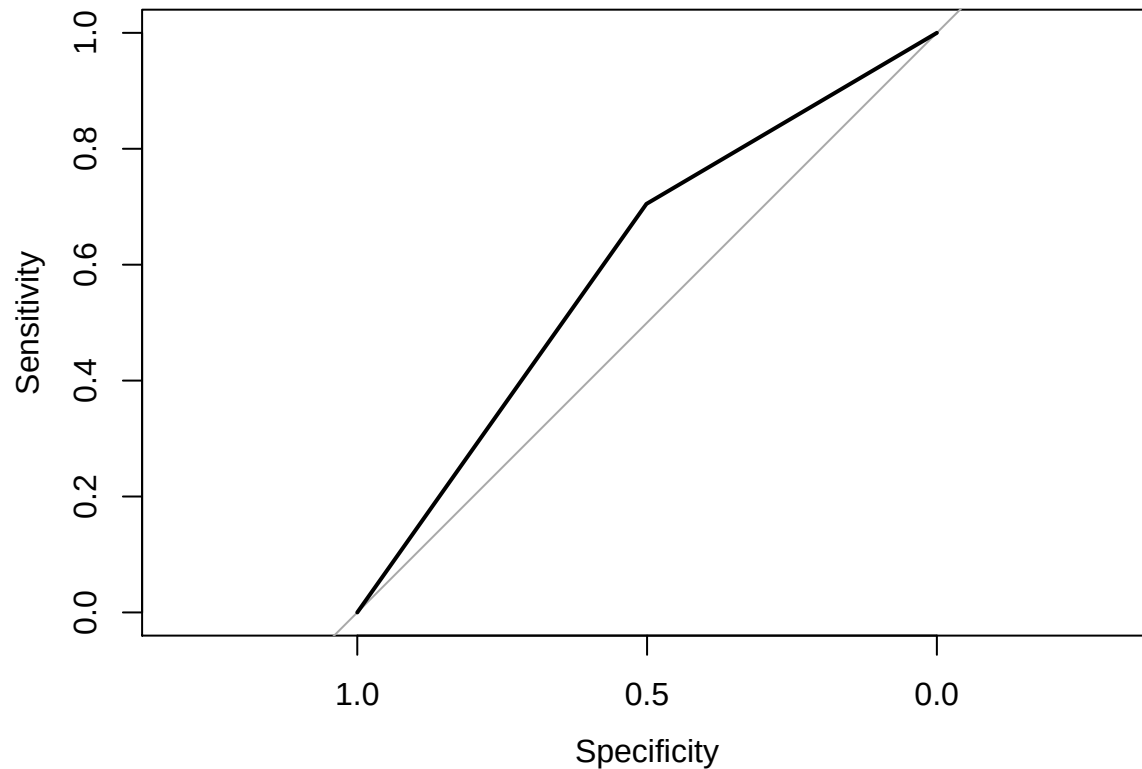
accuracy.log <- sum(test$default.payment.next.month == test$Prediction.log)/nrow(test)

confusionMatrix(as.factor(test$Prediction.log),as.factor(test$default.payment.next.month))

## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction  0   1
##           0 218 116
##           1 217 277
##
##           Accuracy : 0.5978
##           95% CI : (0.5635, 0.6314)
##           No Information Rate : 0.5254
##           P-Value [Acc > NIR] : 1.610e-05
##
##           Kappa : 0.2035
##           McNemar's Test P-Value : 4.254e-08
##
##           Sensitivity : 0.5011
##           Specificity : 0.7048
##           Pos Pred Value : 0.6527
##           Neg Pred Value : 0.5607
##           Prevalence : 0.5254
##           Detection Rate : 0.2633
##           Detection Prevalence : 0.4034
##           Balanced Accuracy : 0.6030
##
##           'Positive' Class : 0
##
```

```
rocCurve <- roc(test$default.payment.next.month ~ test$Prediction.log)
plot(rocCurve)
```



```
coords(rocCurve, "best", ret = "threshold")
```

```
## [1] 0.5
```

The accuracy in prediction achieved with this method was

```
sum(test$default.payment.next.month == test$Prediction.log)/nrow(test)
```

```
## [1] 0.5978261
```

Support Vector Machines:

Method: Build the model on train dataset (SVM - Radial and SVM- Linear) and make predictions on the test dataset

On the train dataset created previously, I used cross validation function to find the best tuning parameter (cost and gamma) values and fit the data.

With this model, I made predictions about the response variable using the observations in the test dataset.

```
test <- test[,-20]
```

```
##SVM - kernel method :Radial
```

```
svm.Radial <- tune.svm(x = train[,-19], y = train$default.payment.next.month, data= train, kernel = "radial",
```

```
svm.Radial$best.parameters
```

```
##      gamma cost
```

```
## 11    0.5    1
```

```
credit.svmRad <- e1071::svm(as.factor(default.payment.next.month) ~ ., data = train, kernel = "radial",
```

```
#Get predictions on the test dataset
```

```
predictions.svmRad <- predict(credit.svmRad, test)
```

```
#Attach predictions to the dataset - test
```

```
test$Predictions.svmRad <- predictions.svmRad
```

```
#Compute the accuracy
```

```
accuracy.svmRad <- sum(test$default.payment.next.month == test$Predictions.svmRad)/nrow(test)
```

```
accuracy.svmRad
```

```
## [1] 0.5869565
```

```
#Confusion Matrix
```

```
confusionMatrix(as.factor(test$Predictions.svmRad),as.factor(test$default.payment.next.month))
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction  0    1
```

```
##           0 213 120
```

```
##           1 222 273
```

```
##
```

```
##           Accuracy : 0.587
```

```
##           95% CI : (0.5526, 0.6207)
```

```
##           No Information Rate : 0.5254
```

```
##           P-Value [Acc > NIR] : 0.0002115
```

```
##
```

```
##           Kappa : 0.182
```

```
##           McNemar's Test P-Value : 4.722e-08
```

```
##
```



```
##          Sensitivity : 0.4897
##          Specificity : 0.6947
##          Pos Pred Value : 0.6396
##          Neg Pred Value : 0.5515
##          Prevalence : 0.5254
##          Detection Rate : 0.2572
##          Detection Prevalence : 0.4022
##          Balanced Accuracy : 0.5922
##
##          'Positive' Class : 0
##
```

```
test <- test[,-20]
##Linear SVM
svm.linear <- tune.svm(x = train[,-19], y = train$default.payment.next.month, data= train, kernel = "lin
svm.linear$best.parameters
```

```
##      gamma cost
## 11    0.5    1
```

```
#Train the model
```

```
model.svmLin <- svm(as.factor(default.payment.next.month) ~ ., data = train, method="svmLinear", cost =
```

```
#Get the predictions on the test dataset
```

```
predictions.svmLin <- predict(model.svmLin, test)
predictions.svmLin
```

```
##      2      6      12      15      17      24      28      32      34      35      48      54      57      64      75
##      1      0      1      0      1      0      1      1      1      1      0      0      1      0      0
##      84      87      89      101      103      109      111      120      124      127      135      141      142      146      152
##      0      0      1      0      1      1      0      1      1      1      0      1      1      1      0
##      154      163      166      171      174      177      178      179      184      185      188      189      200      211      214
##      1      1      0      0      0      1      0      1      0      0      1      0      0      0      0
##      216      224      226      232      234      237      239      244      249      250      260      265      268      284      287
##      1      1      1      1      0      0      1      1      0      1      0      0      1      0      1
##      290      292      297      299      303      311      312      313      318      323      326      329      331      342      346
##      0      0      1      0      1      0      0      1      0      1      0      1      1      1      0
##      357      358      360      363      364      369      372      386      394      405      409      410      412      413      419
##      1      1      1      0      1      0      0      0      1      0      1      1      0      0      1
##      420      422      436      437      440      441      447      455      457      458      477      478      486      489      491
##      1      0      1      1      0      1      1      1      0      0      1      0      1      1      0
##      501      507      513      514      517      519      522      525      526      530      531      538      541      543      544
##      1      1      1      0      1      0      1      1      1      1      0      0      0      0      1
##      546      547      557      567      569      570      578      580      584      592      594      600      606      610      611
##      1      1      0      1      1      1      0      0      0      1      1      1      0      0      1
##      615      638      639      643      652      654      656      658      668      671      673      674      676      680      681
##      0      0      1      0      1      0      1      0      1      1      1      1      1      0      1
##      682      686      691      695      707      710      729      755      763      767      782      788      794      798      801
##      1      1      0      0      1      1      1      0      0      0      0      0      0      0      0
##      803      810      817      821      832      834      837      838      839      865      879      880      883      885      890
##      0      0      1      0      1      0      1      1      1      0      1      1      1      1      0
##      891      899      901      908      913      916      918      919      922      923      928      936      937      939      940
##      0      1      0      1      1      0      0      0      0      1      1      1      1      1      0
##      941      942      946      951      955      965      969      970      981      982      984      994      997      1005      1016
##      1      1      1      0      0      0      0      0      1      1      1      0      1      1      1
```

##	1019	1022	1027	1034	1038	1039	1040	1053	1054	1055	1064	1069	1080	1084	1091
##	0	1	0	1	0	1	1	0	1	0	1	1	0	0	0
##	1092	1097	1100	1101	1107	1111	1112	1114	1120	1127	1128	1131	1134	1142	1146
##	1	0	0	0	1	0	0	0	1	0	0	1	0	0	1
##	1152	1155	1156	1159	1162	1166	1167	1176	1181	1184	1188	1196	1208	1209	1212
##	1	0	1	0	1	0	0	1	0	0	0	0	0	0	1
##	1219	1221	1229	1235	1236	1237	1239	1241	1245	1246	1248	1249	1254	1259	1271
##	0	1	1	0	0	1	1	1	1	1	0	1	0	0	1
##	1272	1282	1286	1288	1295	1300	1301	1304	1308	1320	1322	1335	1340	1342	1343
##	0	0	1	1	1	1	1	0	1	0	1	0	1	1	1
##	1344	1345	1346	1356	1359	1363	1365	1378	1380	1386	1389	1390	1405	1408	1417
##	0	0	1	1	0	1	0	1	1	0	0	0	1	1	1
##	1418	1420	1422	1429	1430	1434	1438	1442	1453	1455	1463	1464	1465	1473	1475
##	0	1	0	0	0	0	0	0	0	1	1	1	0	0	0
##	1480	1483	1490	1497	1499	1500	1511	1516	1519	1525	1527	1539	1544	1545	1546
##	1	0	0	1	0	0	0	1	0	0	0	0	1	1	0
##	1548	1554	1558	1561	1564	1574	1577	1587	1588	1592	1594	1596	1605	1606	1607
##	1	1	0	0	0	0	1	1	1	0	0	1	1	0	1
##	1615	1617	1618	1620	1621	1627	1630	1635	1650	1651	1662	1667	1669	1675	1677
##	1	1	0	1	0	0	1	0	1	1	0	0	0	1	0
##	1686	1687	1688	1706	1707	1711	1714	1715	1717	1721	1724	1726	1734	1735	1737
##	0	1	0	1	0	1	1	0	0	0	0	0	1	1	1
##	1742	1751	1755	1756	1784	1785	1787	1790	1793	1795	1801	1805	1806	1816	1818
##	1	0	0	1	1	0	0	0	1	0	1	1	1	0	1
##	1824	1827	1833	1834	1835	1837	1841	1842	1843	1847	1852	1855	1858	1859	1875
##	0	1	0	1	1	0	1	1	1	1	1	0	1	1	1
##	1877	1887	1893	1900	1901	1903	1904	1913	1917	1919	1930	1945	1946	1947	1962
##	0	1	0	0	1	0	1	1	1	0	0	1	1	1	0
##	1968	1976	1980	1983	1985	1986	1987	1991	1997	2003	2012	2013	2014	2023	2024
##	1	0	0	1	0	1	0	1	1	1	1	0	0	0	0
##	2033	2036	2037	2039	2040	2042	2060	2065	2069	2072	2077	2082	2085	2090	2091
##	0	1	1	1	1	1	1	1	0	1	1	0	0	1	1
##	2094	2095	2096	2108	2119	2124	2126	2127	2130	2135	2141	2144	2155	2156	2161
##	1	1	1	0	1	0	1	1	1	1	1	0	1	1	0
##	2171	2182	2190	2191	2193	2196	2204	2207	2209	2210	2213	2220	2222	2231	2232
##	1	0	1	1	1	1	0	0	1	1	1	0	1	1	0
##	2238	2244	2246	2257	2261	2262	2265	2285	2305	2316	2324	2326	2330	2333	2335
##	1	1	1	0	1	1	1	1	1	0	1	0	1	1	0
##	2340	2341	2343	2345	2353	2360	2370	2372	2377	2378	2379	2395	2399	2416	2419
##	0	1	1	1	1	1	1	1	0	1	1	1	0	1	1
##	2438	2444	2448	2455	2458	2460	2461	2474	2481	2488	2489	2493	2503	2505	2507
##	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1
##	2509	2510	2511	2529	2532	2535	2536	2539	2541	2556	2561	2562	2583	2586	2597
##	0	0	0	1	0	0	0	1	0	1	1	1	1	1	1
##	2598	2600	2601	2604	2610	2617	2618	2623	2627	2642	2668	2670	2680	2681	2689
##	1	0	1	1	0	1	1	0	0	1	1	0	1	0	1
##	2697	2699	2716	2719	2722	2731	2741	2756	2758	2760	2766	2773	2776	2777	2779
##	1	0	1	0	1	1	0	1	1	1	0	1	0	0	1
##	2789	2792	2794	2795	2810	2815	2818	2820	2830	2848	2849	2850	2852	2855	2859
##	1	1	1	1	1	0	1	1	0	1	1	1	0	0	1
##	2863	2884	2893	2895	2898	2900	2902	2912	2915	2921	2924	2927	2930	2940	2946
##	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1
##	2949	2953	2955	2958	2966	2971	2977	2979	3004	3005	3007	3008	3018	3020	3024
##	0	1	1	1	1	0	1	1	0	1	1	0	1	1	1

```
## 3029 3033 3038 3045 3056 3059 3065 3071 3076 3083 3084 3094 3096 3098 3102
##      1      1      1      1      0      1      1      1      0      1      1      1      1      0      1
## 3103 3104 3105 3109 3118 3122 3129 3148 3155 3158 3159 3164 3169 3174 3184
##      1      1      0      0      0      1      1      1      0      1      1      1      1      1      1
## 3185 3187 3188 3190 3207 3208 3219 3223 3224 3227 3232 3246 3250 3257 3259
##      1      1      1      1      1      1      1      1      1      0      0      1      0      1      1
## 3270 3271 3274 3276 3278 3279 3282 3284 3288 3294 3305 3308 3311 3319 3324
##      1      1      0      0      1      0      0      1      1      1      1      1      1      1      1
## 3327 3333 3341 3346 3347 3365 3369 3378 3380 3384 3386 3387 3403 3407 3409
##      1      1      0      1      1      0      1      0      1      1      1      1      0      1      1
## 3410 3415 3429 3432 3434 3435 3437 3442 3447 3452 3453 3461 3472 3474 3481
##      1      1      1      1      0      1      0      0      0      0      1      1      1      0      1
## 3488 3494 3502 3504 3506 3510 3511 3512 3519 3522 3526 3529 3536 3538 3548
##      1      1      1      1      0      1      0      0      1      1      1      1      1      1      1
## 3554 3565 3570 3571 3575 3579 3581 3584 3587 3593 3601 3608 3617 3625 3641
##      0      1      0      1      1      1      1      0      1      1      0      0      0      0      1
## 3646 3653 3654 3665 3672 3674 3688 3705 3708 3715 3718 3722 3727 3731 3737
##      0      1      0      1      1      1      1      1      0      1      0      1      1      0      1
## 3739 3742 3761 3764 3776 3781 3788 3789 3796 3799 3815 3827 3830 3831 3854
##      1      1      1      0      1      0      1      0      1      1      1      0      1      0      1
## 3858 3872 3873 3875 3878 3879 3880 3889 3892 3895 3901 3905 3907 3913 3915
##      1      0      0      0      1      0      1      1      1      1      1      0      0      1      0
## 3917 3918 3919 3923 3924 3929 3932 3943 3945 3949 3967 3972 3979 3986 3988
##      0      1      0      0      0      1      0      0      1      1      0      1      1      0      1
## 3991 3995 3996 3998 4001 4028 4029 4036 4042 4043 4045 4048 4051 4052 4060
##      1      1      1      1      1      1      1      0      1      0      1      0      0      0      1
## 4063 4065 4069 4071 4072 4073 4074 4076 4081 4089 4098 4112 4114 4115 4118
##      1      1      0      1      1      1      1      0      1      0      1      1      1      1      1
## 4120 4123 4137
##      0      0      1
## Levels: 0 1
```

```
#Attach predictions to the dataset - test
```

```
test$Predictions.svmLin <- predictions.svmLin
```

```
#Accuracy
```

```
accuracy.svmLin <- sum(test$default.payment.next.month == test$Predictions.svmLin)/nrow(test)
accuracy.svmLin
```

```
## [1] 0.5869565
```

```
#Confusion Matrix
```

```
confusionMatrix(as.factor(test$Predictions.svmLin), as.factor(test$default.payment.next.month))
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction  0    1
```

```
##           0 213 120
```

```
##           1 222 273
```

```
##
```

```
##           Accuracy : 0.587
```

```
##           95% CI : (0.5526, 0.6207)
```

```
## No Information Rate : 0.5254
```

```
## P-Value [Acc > NIR] : 0.0002115
```

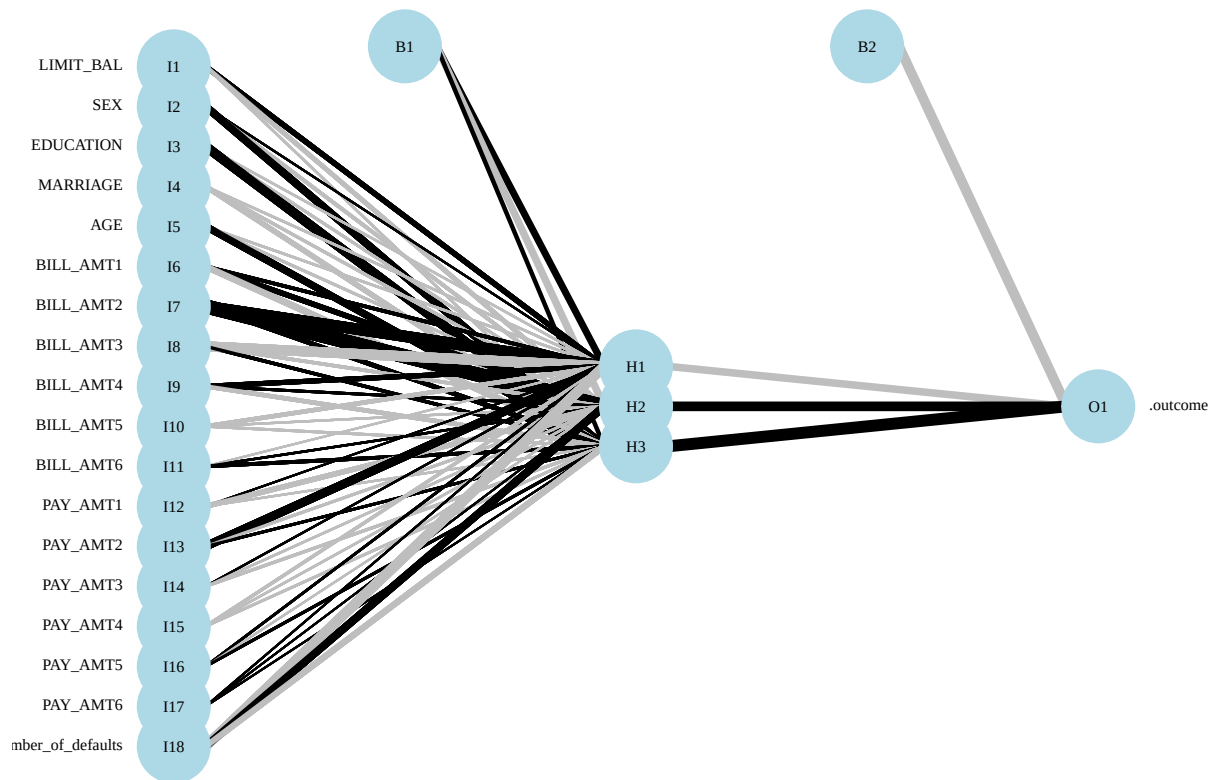
```
##
```

```
##           Kappa : 0.182
## McNemar's Test P-Value : 4.722e-08
##
##           Sensitivity : 0.4897
##           Specificity : 0.6947
##           Pos Pred Value : 0.6396
##           Neg Pred Value : 0.5515
##           Prevalence : 0.5254
##           Detection Rate : 0.2572
##           Detection Prevalence : 0.4022
##           Balanced Accuracy : 0.5922
##
##           'Positive' Class : 0
##
```

Artificial Neural Networks: (use CV for choosing tuning parameters)

```
test <- test[,-20]
trControl <- trainControl(method = "cv", number = 10, search = "grid")
test.nn <- capture.output(nn.credit.1 <- caret::train(as.factor(default.payment.next.month) ~. , data =
#Get predictions
pred.nn.credit <- predict(nn.credit.1, test)
print(nn.credit.1)
```

```
## Neural Network
##
## 3309 samples
## 18 predictor
## 2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2979, 2978, 2979, 2979, 2977, 2978, ...
## Resampling results across tuning parameters:
##
##  size  decay  Accuracy  Kappa
##  1      0e+00  0.6022957  0.2031436
##  1      1e-04  0.6031984  0.2050318
##  1      1e-01  0.6032057  0.2045030
##  3      0e+00  0.5950659  0.1825646
##  3      1e-04  0.5983874  0.1908209
##  3      1e-01  0.6119826  0.2194416
##  5      0e+00  0.5977822  0.1911978
##  5      1e-04  0.5896059  0.1735074
##  5      1e-01  0.6059220  0.2073512
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 3 and decay = 0.1.
##
##Plot the neural network
par(mar = numeric(4), family = "serif")
plotnet(nn.credit.1$finalModel, cex_val = 0.5)
```



```
#Print confusion matrix
print(cm.credit.1 <- confusionMatrix(pred.nn.credit, as.factor(test$default.payment.next.month)))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0    1
##           0 218 109
##           1 217 284
##
##           Accuracy : 0.6063
##           95% CI : (0.5721, 0.6397)
##           No Information Rate : 0.5254
##           P-Value [Acc > NIR] : 1.661e-06
##
##           Kappa : 0.2209
##           McNemar's Test P-Value : 3.101e-09
##
##           Sensitivity : 0.5011
##           Specificity : 0.7226
##           Pos Pred Value : 0.6667
##           Neg Pred Value : 0.5669
##           Prevalence : 0.5254
##           Detection Rate : 0.2633
##           Detection Prevalence : 0.3949
##           Balanced Accuracy : 0.6119
##
##           'Positive' Class : 0
##
```

```

#Accuracy for the neural network model
acc <- nn.credit.1$results$Accuracy[best.ch <- as.numeric(dimnames(nn.credit.1$bestTune)[[1]])]
acc

## [1] 0.6119826

kable(data.frame(Logit = accuracy.log, SvmLin= accuracy.svmLin, SvmRad= accuracy.svmRad, NNet= acc))

```

Logit	SvmLin	SvmRad	NNet
0.5978261	0.5869565	0.5869565	0.6119826

Choice of classification methods and rationale:

Logistic regression is used as a classification method to model binary outcome variables. Since, the outcome variable in this case - default payment- is also a binary variable, I decided to use logistic regression for the purpose of classification.

Support Vector Machine: (SVM)

Support Vector Machine algorithm is used to find the hyperplane with the maximum margin that distinctly classifies the observations in the data. This method has some advantages over logistic regression i.e it is less sensitive to outliers than logistic regression and it calculates the absolute predictions instead of probabilistic predictions given by logistic regression. Thus, in SVM, we are not required to decide a threshold to obtain the absolute probabilities.

Artificial Neural Network: (NNet)

According to L.M. Salchenberger, E.M. Cinar, N.A. Lash¹, neural network is better when multicollinearity exists and a non-linear relationship is found between the explanatory and response variables. Since, from the visualization maps, I found strong correlations between some of the explanatory variables, I decided to try using one of the most widely used machine learning method - Artificial Neural Network.

Substantive conclusion:

In financial fraud/default cases, ideally we would prefer maximizing the true positive and true negative outcomes. In addition to this, to avoid substantial monetary losses, we need to be more skeptical of the false negative (predicting non-default for a defaulter) than a false positive outcomes (predicting default for a non defaulter).

False negatives imply loss of money due to fraudulent transactions and false positives imply loss of revenue due to turning away legitimate customers. Accuracy implies the correct predictions of defaulters and non-defaulters.

In my analysis, the Artificial Neural network model had the highest accuracy with approximately 61%

Logistic regression performs well too with an accuracy of approximately 60% followed by SVM (linear and radial) with an accuracy of approximately 59%

Further research:

I would like to explore the methods of features selection and check if that improves the accuracy of these models.

¹(Neural networks: a new tool for predicting thrift failures, Decision Sciences 23 (1992) 899–915)