The link to GitHub repository is

<https://github.com/dvolynskyy/CIND-820>

Initial Dataset Analysis

The very first and important step of the machine learning process while performing any data analysis it to prepare data for its proper future interpretation. Clean data can notably increase the accuracy of any model. Usually we start that in R with reading a dataset file using a command ‘read.csv’. We want to have the headers as in original file and do not have factor values. For this ‘header’ argument is set to ‘True’ and ‘stringAsFactors’ – to ‘False’. Now we can start investigating our data. We would need to learn the shape, size, type and general layout of the data that we have. A command ‘head’ shows us the first six rows and gives us understanding how the dataset looks like. As it was mentioned earlier in this project, there are twenty-five columns describing attributes of each credit card customer in terms of his/her sex, education, age, payment history, bill amount and status of payments in different months (see table 1). To avoid any complications further, our target variable ‘default.payment.next.month’ is going to be renamed to simpler name using the following code:

colnames(projectData)[colnames(projectData)=="default.payment.next.month"] <- "DEFAULT\_PAYMENT"

Then with ‘dim’ function we can easily check that the given data set has 30000 records in it. Using a standard command ‘str’ we may find out of what type is each variable in a data frame as it shown in the following screen capture:

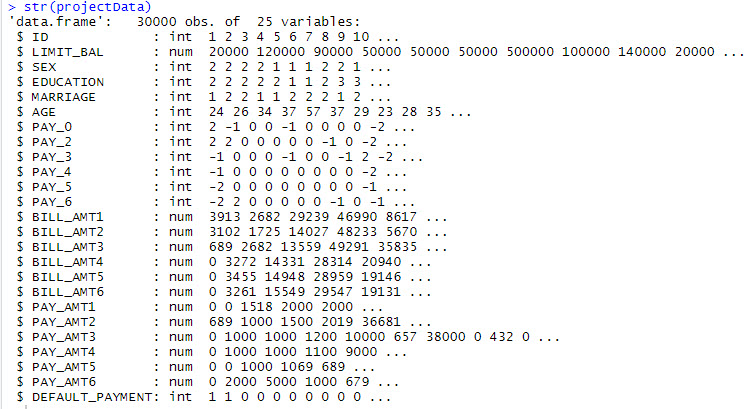


Figure 1 – results of str() function

As we can see, some variables are of integer type and others are numeric.

Since we have a classification problem in this project, there is a necessity to know the proportion of instances that belong to each class label. This is important because it may highlight an imbalance in the data, that if severe may need to be addressed with rebalancing techniques.

y <- projectData$DEFAULT\_PAYMENT

cbind(freq=table(y), percentage=prop.table(table(y))\*100)

The code above creates a useful table showing the number of instances that belong to each class as well as the percentage that this represents from the entire dataset:



Figure 2 – Frequency of class instances in a dataset

As it is seen, the given dataset has rather imbalanced data where class ‘0’ is presented triple as much as class ‘1’. Due to this we will need to address this issue with specific methods.

## Exploratory Data Analysis

Applying the summary() function to a data frame will return the summary showing main descriptive statistics (min, 25 percentile, median, mean, 75 percentile, max) for all numeric values. It also indicates, if applicable, the number of missing values for an attribute (marked NA). In this case there no missing values at all.

Comparing details of data description from table 1 and the results of summary() function, it is seen that two attributes, Education and Marriage, have categories either not included in the dataset description or are meaningless.

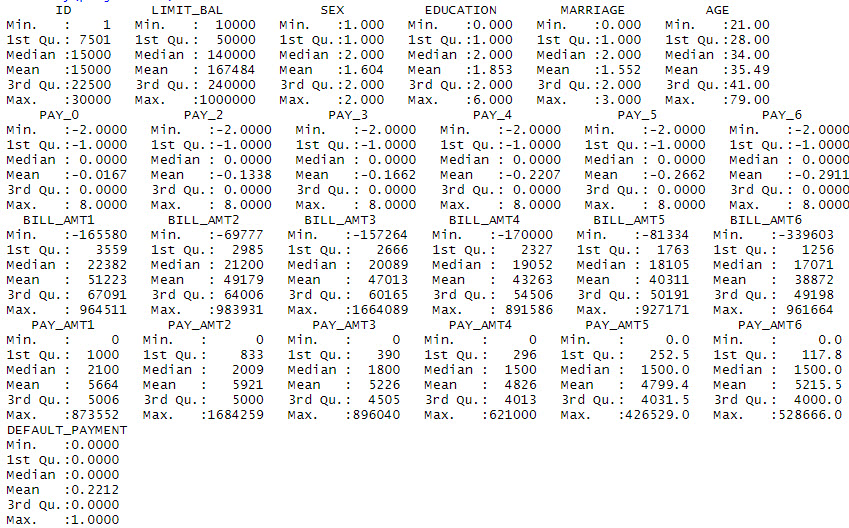


Figure 3 – Results of summary() function

From the data description received in table 1 we can see that the Education varible has the following categories: 1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown. In addition, we can observe that this attribute holds 0 number as its value, which is not described in the dataset. We may assume that these 0 values along with 5 and 6 can be categorized under value 4 (other).

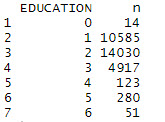


Figure 4 – Frequency of Education variable categories

A similar situation is for Marriage variable which holds values like 1=married, 2=single, 3 =others. Since the category 0 hasn't been defined anywhere, we will include it in the ‘others’ category marked as 3.

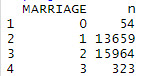


Figure 5 – Frequency of Marriage variable categories

The following code do the actions mentioned above:

projectData$EDUCATION[projectData$EDUCATION == 0] <- 4

projectData$EDUCATION[projectData$EDUCATION == 5] <- 4

projectData$EDUCATION[projectData$EDUCATION == 6] <- 4

projectData$MARRIAGE[projectData$MARRIAGE == 0] <- 3

projectData %>% count(EDUCATION, sort = FALSE)

projectData %>% count(MARRIAGE, sort = FALSE)

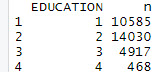
 

Figure 6 – Frequency of Education and Marriage variable categories after recategorizing

Another important aspect is to explore how the attributes related between each other. For this purpose, the correlation of each pair of numeric attributes will be considered.

correlations <- cor(projectData[,2:25])

print(correlations)

plot\_correlation(na.omit(projectData), maxcat = 5L)

The code above creates the correlation table between all pairs of attributes. Correlation values of above 0.75 or lower than minus 0.75 are of greater interest as they show stronger positive or negative correlations. As values approach one or minus one they tend to show either full positive or full negative correlation. This code creates a following heat map of variables correlations as well.

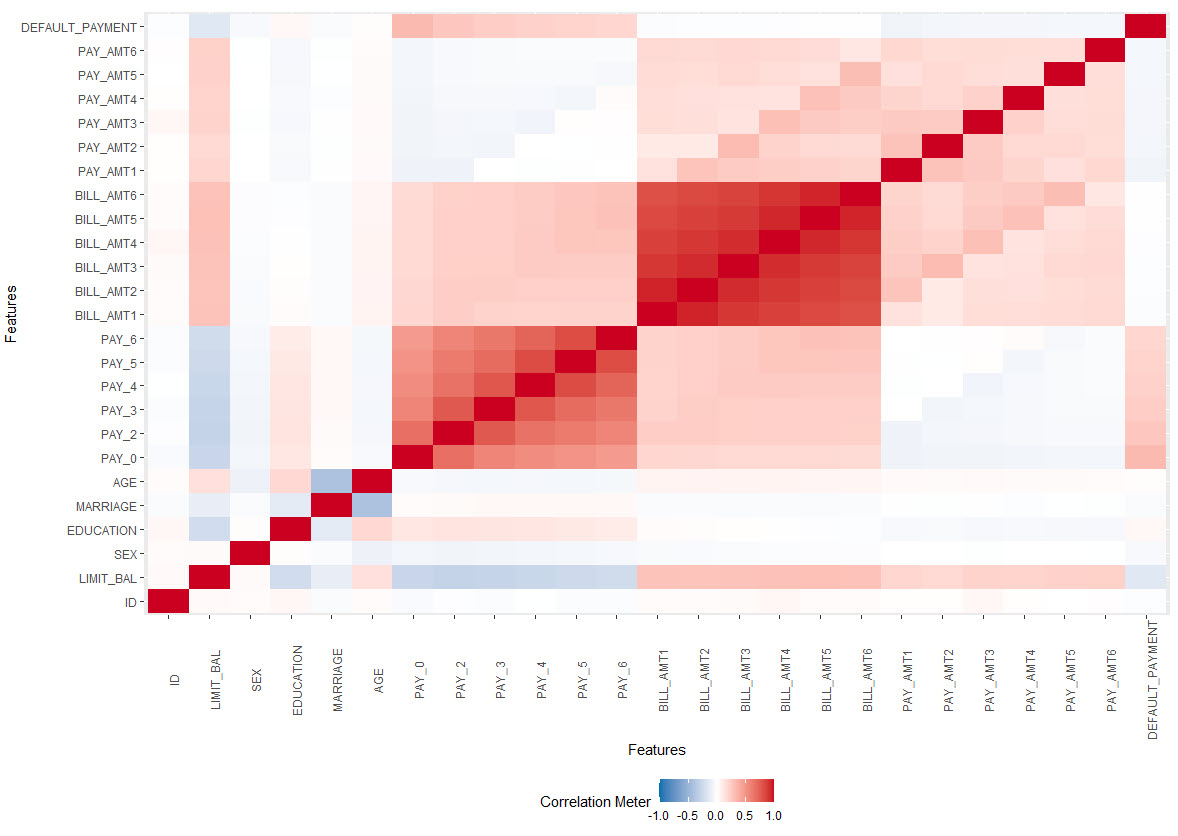
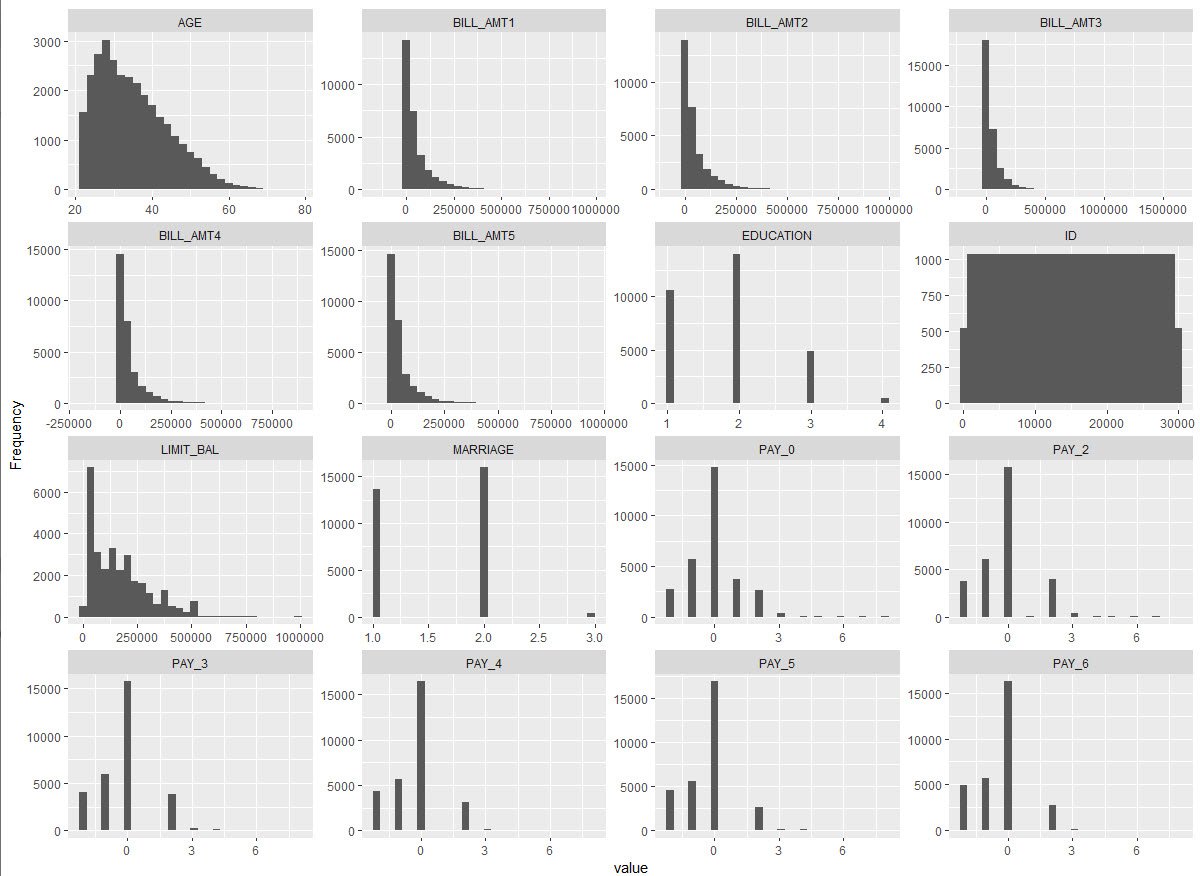


Figure 7 – Heat map of correlations between dataset attributes

From this heat map it is seen that particular variables like BILL\_AMT1, BILL\_AMT2, BILL\_AMT3, BILL\_AMT4, BILL\_AMT5, BILL\_AMT6 have a very weak correlation with a class attribute and thus have no impact on a final outcome. Due to this we will not include them in a machine learning model.

With a following piece of code, we create a histogram for each attribute and thus performing a univariate analysis of them:

plot\_histogram(projectData)



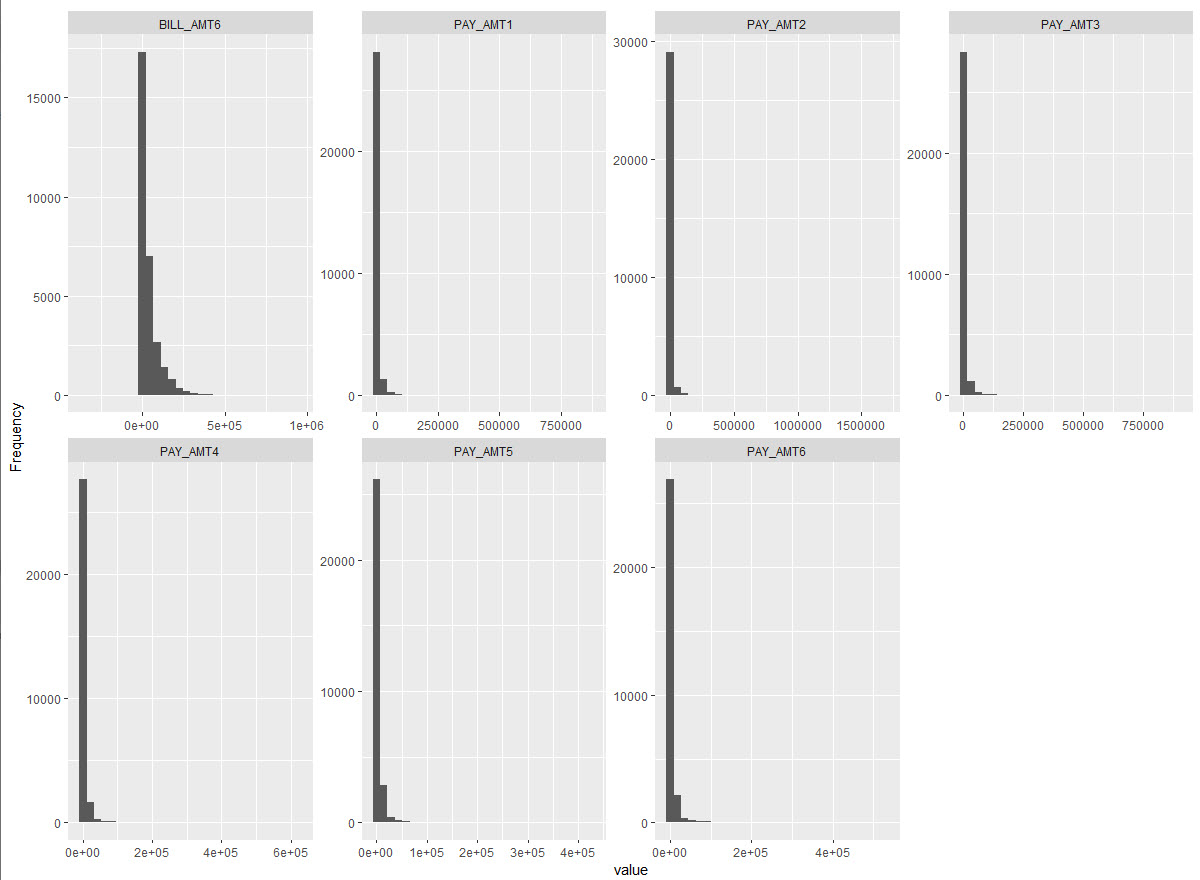


Figure 8 – Histograms for dataset attributes

The above graphs show that all PAY variables are skewed to the right. Otherwise there are no specific dependencies to highlight.

## Pre-Processing and Feature Selection

Next step is to prepare dataset for modelling. As it was mentioned above BILL\_AMT1, BILL\_AMT2, BILL\_AMT3, BILL\_AMT4, BILL\_AMT5, BILL\_AMT6 attributes have no impact on a class variable and should be deleted from it. That is done with the following code

modelData <- select(projectData, !c('ID', 'BILL\_AMT1','BILL\_AMT2', 'BILL\_AMT3','BILL\_AMT4','BILL\_AMT5','BILL\_AMT6'))

After executing the dataset has nineteen variables including dependent one and the same number of records as the initial dataset.

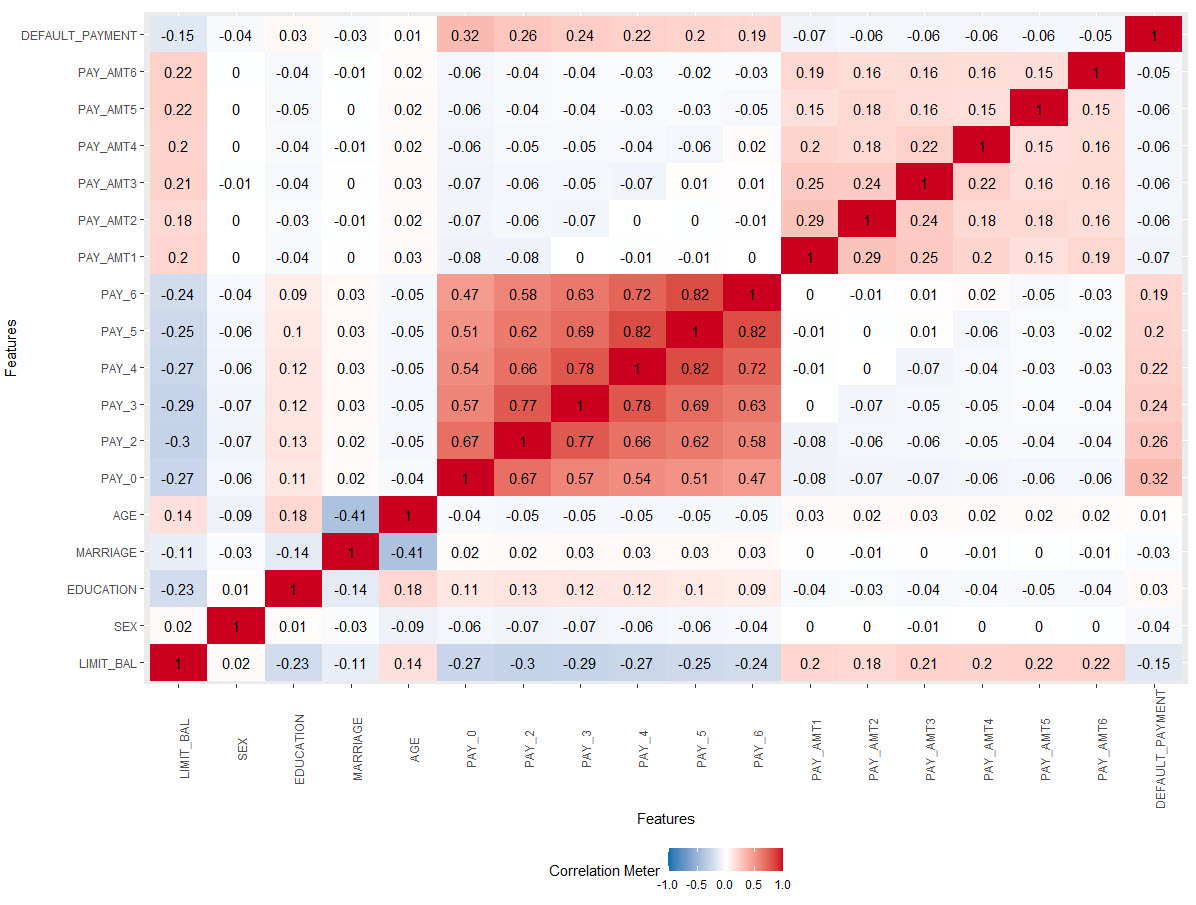


Figure 9 – Heat map of correlations between dataset attributes after removing features

Taking into consideration the heat map (Fig. 9) it is seen that even after removing specific attributes, all other ones have not so high correlation class variable. Due to that we will need to analyze and interpret very carefully the performance of machine learning modelling having almost independent variables.

Further we need to deal with class imbalance. For this reason, a CRAN package ‘Imbalance’ is installed. Specifically, a majority weighted minority oversampling technique (MWMOTE) for imbalance dataset will be used. It is a modification for SMOTE technique which overcomes some of the problems of the SMOTE technique when there are noisy instances, in which case SMOTE would generate more noisy instances out of them. After creating new artificial records as a separate dataframe we use function ‘rbind’ to merge two datasets and then we shuffle it in order not to have only the records of the same class outcome in each fold of the future *k*-fold cross validation modelling. Another correlation heat map (Fig. 10) is created to see how new instances in a dataset affected dependencies between each other. It is vividly seen that the values for PAY variables decreased and PAY\_AMT increased a little bit in comparison with previous heat map. Nevertheless, we assume that this change is not critical and will not biased the model itself very much.

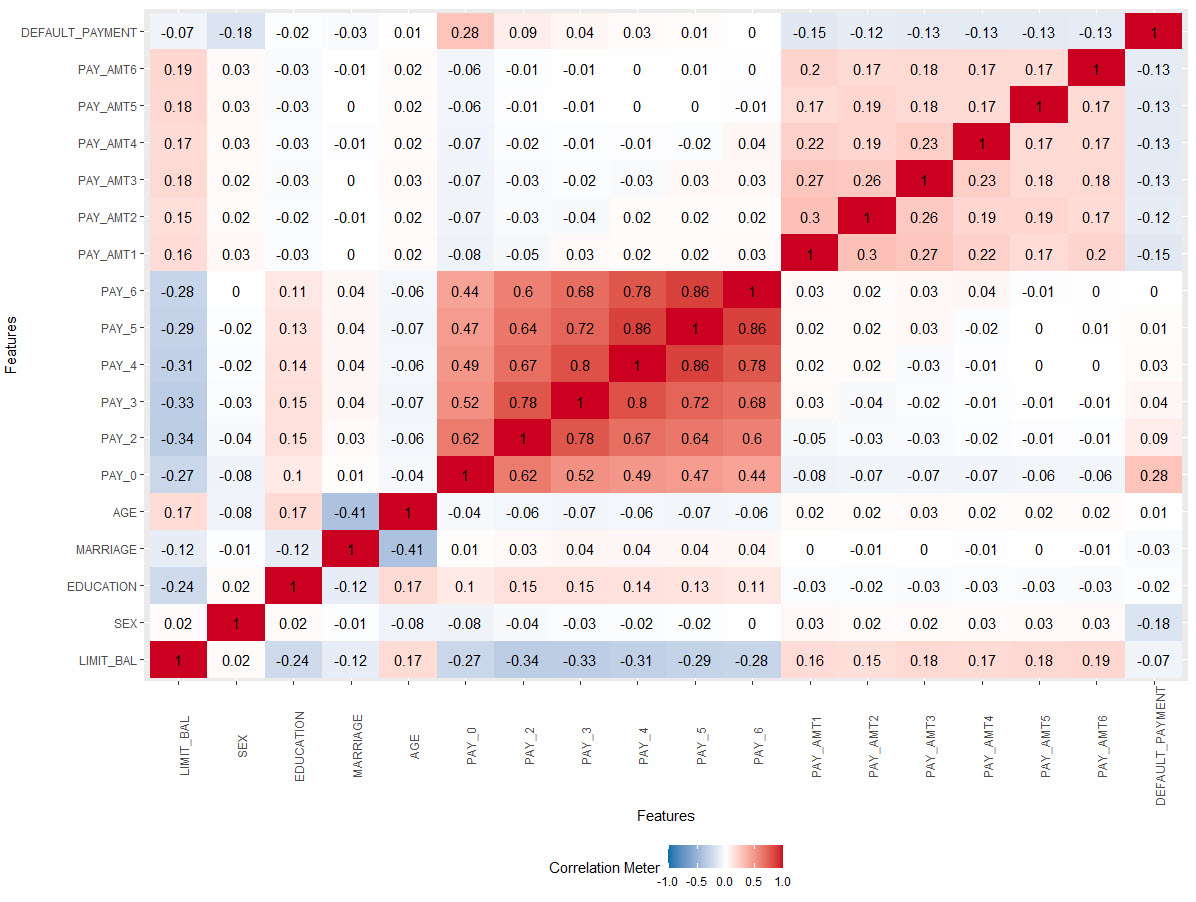


Figure 10 – Heat map of correlations between dataset attributes after data balanced with using MWMOTE() function

Then we check again in what proportion the records with different class value are presented. Execution of this code

z <- newData$DEFAULT\_PAYMENT

cbind(freq=table(z), percentage=prop.table(table(z))\*100)

shows that there is a ratio of 58.41% to 41.59% of instances that belong to class ‘0’ and class ‘1’ respectively. Due to this we assume that our dataset is balanced now and we can proceed to the machine learning modelling.