Bank Loan modelling

Germán Camargo Ortega

2022-10-21

Contents

Introduction/overview/executive:

Description

Background and Context

AllLife Bank is a US bank that has a growing customer base. The majority of these customers are liability customers (depositors) with varying sizes of deposits. The number of customers who are also borrowers (asset customers) is quite small, and the bank is interested in expanding this base rapidly to bring in more loan business and in the process, earn more through the interest on loans. In particular, the management wants to explore ways of converting its liability customers to personal loan customers (while retaining them as depositors).

A campaign that the bank ran last year for liability customers showed a healthy conversion rate of over 9% success. This has encouraged the retail marketing department to devise campaigns with better target marketing to increase the success ratio.

Main goal: I have to build a model that could help the marketing department to identify the potential customers who have a higher probability of purchasing the loan. I will need to predict whether a liability customer will buy a personal loan or not. Which variables are most significant? Which segment of customers should be targeted more?

Data Dictionary

- ID: Customer ID
- Age: Customer's age in completed years
- Experience: #years of professional experience
- Income: Annual income of the customer (in thousand dollars)
- ZIP Code: Home Address ZIP code.
- Family: the Family size of the customer
- CCAvg: Average spending on credit cards per month (in thousand dollars)
- Education: Education Level. 1: Undergrad; 2: Graduate;3: Advanced/Professional
- Mortgage: Value of house mortgage if any. (in thousand dollars)
- Personal_Loan: Did this customer accept the personal loan offered in the last campaign?
- Securities_Account: Does the customer have securities account with the bank?

- CD_Account: Does the customer have a certificate of deposit (CD) account with the bank?
- Online: Do customers use internet banking facilities?
- CreditCard: Does the customer use a credit card issued by any other Bank (excluding All life Bank)?

1. Load libraries and data set, and check the structure

1.1 Libraries for data analysis and visualization

```r
# install packeges if needed
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.0      v stringr 1.4.0
## v readr   2.0.1      v forcats 0.5.1

## Warning: package 'tibble' was built under R version 4.1.2

## Warning: package 'tidyr' was built under R version 4.1.2

## Warning: package 'dplyr' was built under R version 4.1.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

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

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

## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
if(!require(janitor)) install.packages("janitor", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: janitor
```

```
##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```r
if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: knitr

## Warning: package 'knitr' was built under R version 4.1.2
```

```r
if(!require(DataExplorer)) install.packages("DataExplorer", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: DataExplorer
```

```r
if(!require(ggpubr)) install.packages("ggpubr", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: ggpubr
```

```r
if(!require(GGally)) install.packages("GGally", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: GGally

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
if(!require(performanceEstimation)) install.packages("performanceEstimation", repos = "http://cran.us.r
```

```
## Loading required package: performanceEstimation
```

```r
if(!require(ROSE)) install.packages("ROSE", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: ROSE

## Loaded ROSE 0.0-4
```

```r
if(!require(smotefamily)) install.packages("smotefamily", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: smotefamily
```

```r
if(!require(readxl)) install.packages("readxl", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: readxl
```

```r
if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: rpart.plot

## Warning: package 'rpart.plot' was built under R version 4.1.2

## Loading required package: rpart
```

```r
# load libraries
library(tidyverse)
library(caret)
library(data.table)
library(janitor)
library(knitr)
library(DataExplorer)
library(ggpubr)
library(GGally)
library(performanceEstimation)
library(ROSE)
library(smotefamily)
library(readxl)
library(rpart.plot)
```

1.2 Load the data set

The original data set is found under: https://www.kaggle.com/datasets/itsmesunil/bank-loan-modelling/download?datasetVersionNumber=1 It is not super heavy and you can download it into your personal computer. I will load it here from my github.

```
# load the data
Loan_Modelling  <- read.csv(
  "https://raw.githubusercontent.com/GermanCamargoOrtega/edxProject/main/BankLoan.csv"
  )




# create a copy to leave original unchanged
df <- Loan_Modelling

# clean column names
df <- clean_names(df)

set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

1.3 Check the structure of the data.

In the next step I will have a check at the shape of the data set. In addition to tidyverse-related functions, I will use other functions from the package DataExplorer, which is designed specifically for data structure analysis and EDA!

```
# check data characteristics
head(df, 10)
```

```
##    id age experience income zip_code family cc_avg education mortgage
## 1   1  25          1     49    91107      4    1.6         1        0
## 2   2  45         19     34    90089      3    1.5         1        0
## 3   3  39         15     11    94720      1    1.0         1        0
## 4   4  35          9    100    94112      1    2.7         2        0
## 5   5  35          8     45    91330      4    1.0         2        0
## 6   6  37         13     29    92121      4    0.4         2      155
## 7   7  53         27     72    91711      2    1.5         2        0
## 8   8  50         24     22    93943      1    0.3         3        0
## 9   9  35         10     81    90089      3    0.6         2      104
## 10 10  34          9    180    93023      1    8.9         3        0
##    personal_loan securities_account cd_account online credit_card
## 1              0                  1          0      0           0
## 2              0                  1          0      0           0
## 3              0                  0          0      0           0
## 4              0                  0          0      0           0
## 5              0                  0          0      0           1
## 6              0                  0          0      1           0
## 7              0                  0          0      1           0
## 8              0                  0          0      0           1
## 9              0                  0          0      1           0
## 10             1                  0          0      0           0
```

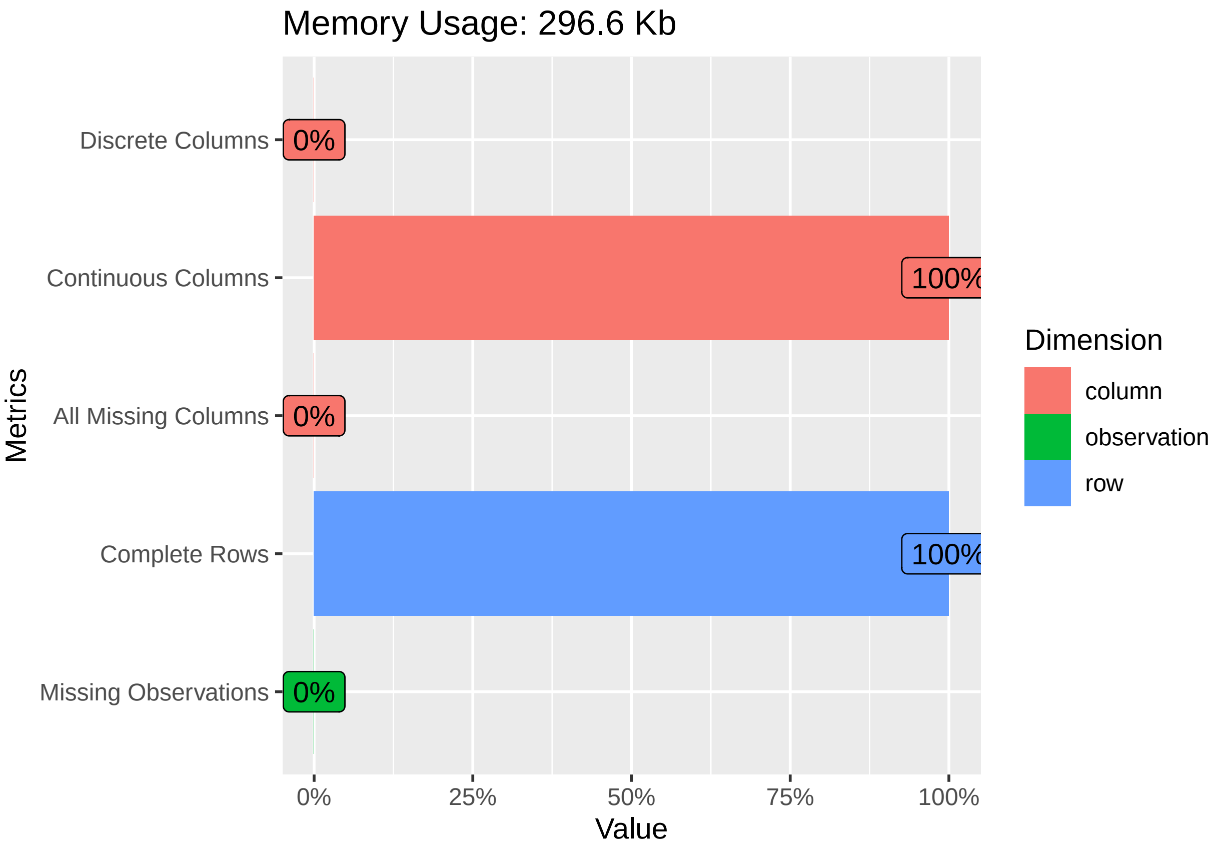
```
# further check data characteristics
glimpse(df)
```

```
## Rows: 5,000
## Columns: 14
## $ id                 <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ~
## $ age                <int> 25, 45, 39, 35, 35, 37, 53, 50, 35, 34, 65, 29, 48,~
## $ experience         <int> 1, 19, 15, 9, 8, 13, 27, 24, 10, 9, 39, 5, 23, 32, ~
## $ income             <int> 49, 34, 11, 100, 45, 29, 72, 22, 81, 180, 105, 45, ~
## $ zip_code           <int> 91107, 90089, 94720, 94112, 91330, 92121, 91711, 93~
## $ family             <int> 4, 3, 1, 1, 4, 4, 2, 1, 3, 1, 4, 3, 2, 4, 1, 1, 4, ~
## $ cc_avg             <dbl> 1.6, 1.5, 1.0, 2.7, 1.0, 0.4, 1.5, 0.3, 0.6, 8.9, 2~
## $ education          <int> 1, 1, 1, 2, 2, 2, 2, 3, 2, 3, 3, 2, 3, 2, 1, 3, 3, ~
## $ mortgage           <int> 0, 0, 0, 0, 0, 155, 0, 0, 104, 0, 0, 0, 0, 0, 0, 0,~
## $ personal_loan      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, ~
## $ securities_account <int> 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, ~
## $ cd_account         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ online             <int> 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, ~
## $ credit_card        <int> 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, ~
```

```
t(introduce(df))
```

```
##                        [,1]
## rows                   5000
## columns                  14
## discrete_columns          0
## continuous_columns       14
## all_missing_columns       0
## total_missing_values      0
## complete_rows          5000
## total_observations    70000
## memory_usage         303752
```

```
plot_intro(df)
```



```
# check the number of unique values fr each variable
# note, map gives a list, so we use here map_df; pivot longer is used here to transpose the df
map_df(df, n_distinct) %>% pivot_longer(everything(),
                                        names_to = "variables",
                                        values_to = "unique values") %>% kable()
```

variables	unique values
id	5000
age	45
experience	47
income	162
zip_code	467
family	4
cc_avg	108
education	3
mortgage	347
personal_loan	2
securities_account	2
cd_account	2
online	2
credit_card	2

```
# calculate the number of missing values (another way of chicking this)
map_df(df, function(x) sum(length(which(is.na(x))))) %>%
  pivot_longer(everything(),
               names_to = "variables",
               values_to = "missing values") %>% kable()
```

variables	missing values
id	0
age	0
experience	0
income	0
zip_code	0
family	0
cc_avg	0
education	0
mortgage	0
personal_loan	0
securities_account	0
cd_account	0
online	0
credit_card	0

```
# statistical summary
t(as.data.frame(apply(df, 2, summary)))
```

```
##                      Min.  1st Qu.  Median         Mean  3rd Qu.  Max.
## id                      1  1250.75  2500.5  2500.500000  3750.25  5000
## age                    23    35.00    45.0    45.338400    55.00    67
## experience             -3    10.00    20.0    20.104600    30.00    43
## income                  8    39.00    64.0    73.774200    98.00   224
## zip_code             9307 91911.00 93437.0 93152.503000 94608.00 96651
## family                  1     1.00     2.0     2.396400     3.00     4
## cc_avg                  0     0.70     1.5     1.937938     2.50    10
## education               1     1.00     2.0     1.881000     3.00     3
## mortgage                0     0.00     0.0    56.498800   101.00   635
## personal_loan           0     0.00     0.0     0.096000     0.00     1
## securities_account      0     0.00     0.0     0.104400     0.00     1
## cd_account              0     0.00     0.0     0.060400     0.00     1
## online                  0     0.00     1.0     0.596800     1.00     1
## credit_card             0     0.00     0.0     0.294000     1.00     1
```

```
# check which is the smallest ZIP code and identify the row number
df$zip_code %>% min()
```

```
## [1] 9307
```

```
df$zip_code[385]
```

```
## [1] 9307
```

```
slice_min(df, zip_code, n=2) %>% select("zip_code")
```

```
##   zip_code
## 1     9307
## 2    90005
## 3    90005
## 4    90005
## 5    90005
## 6    90005
```

Observations

- File has 5000 rows and 14 columns.
- All values are indeed numeric, more precisely double-precision floating point number.
- There are 0 missing values.
- Some columns have repeated values.
- Column ID have only unique values and does not contribute. **I will drop it.**
- Otherwise the data seems rather clean.

From the statistical summary

- balanced columns: Age, Experienced, Family, Education.
- possibly right skewed columns: CCAvg, Income, Mortgage.
- ZIPCodes are from California, except the one in row 385 (i.e. 9307). I will assume this code is in reality 93007, which corresponds to Ventura, CA. **I will adjust this value to 93007.**

```
# drop column ID
df <- select(df, -id)

# adjust the zip code in row 385
df$zip_code[385] <- 93007
```

I will transform columns "personal_loan", "securities_account", "cd_account", "online", "credit_card" (i.e. columns 10 to 14) into factors (categorical) as they are only 0 and 1, and doing so may help with plotting and model generation.

```
df[,9:13] <- map(df[,9:13], as_factor)
```

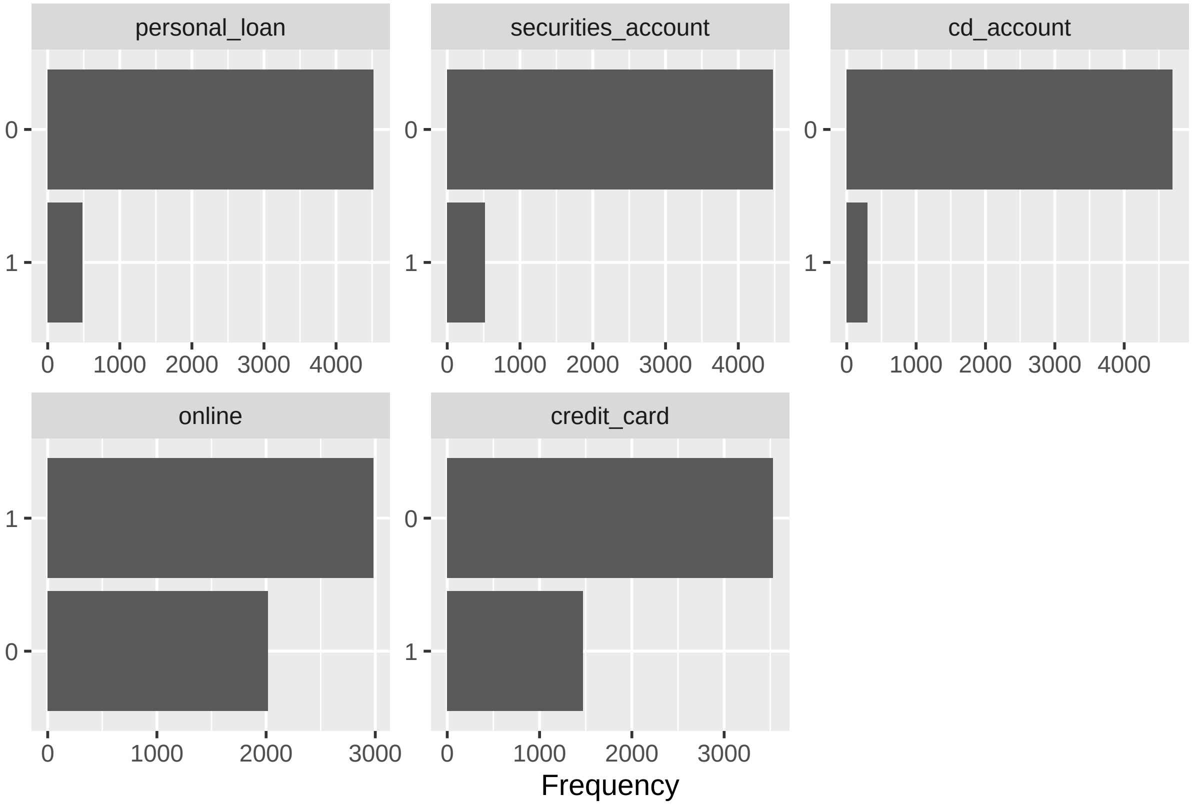
2. EDA of the data set

Notes:

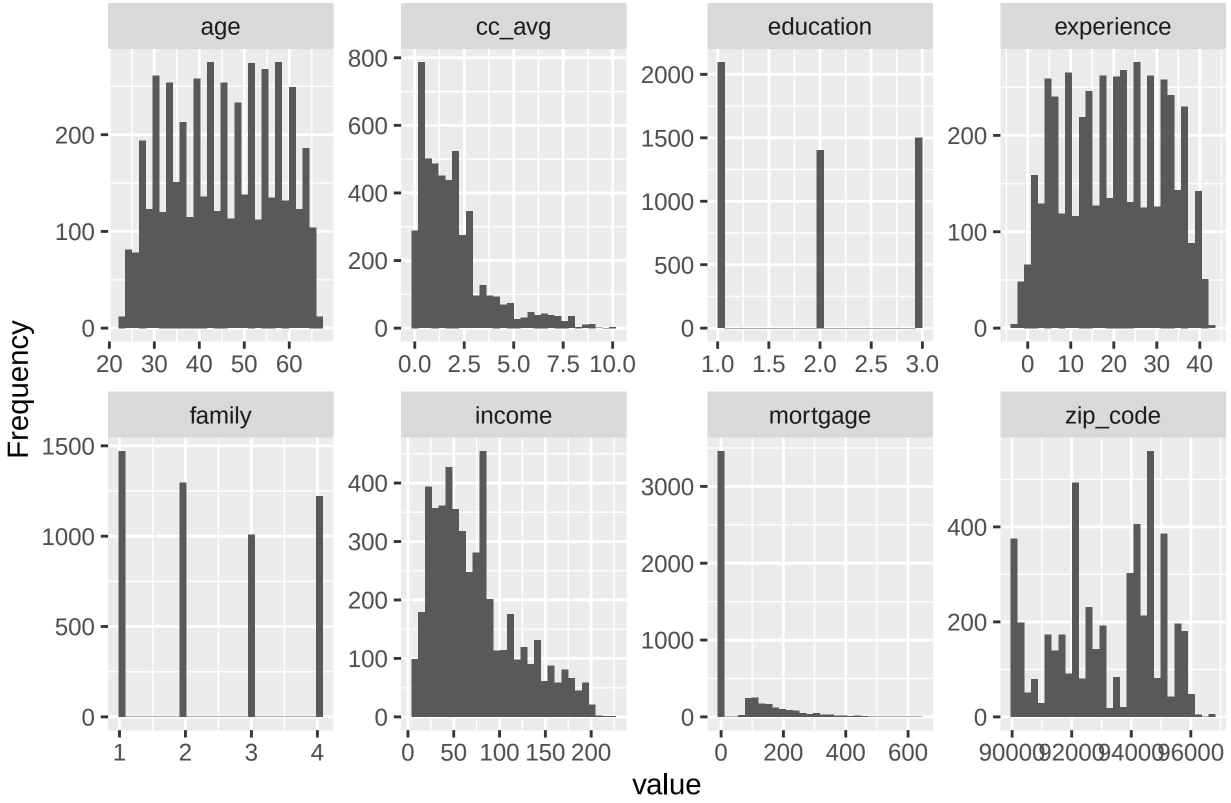
- Univariate analysis
- Bivariate analysis
- Key meaningful observations on the relationship between variables

I will start with a unimodal screen of the distribution of each variable. No need for nice plots at this place, barplots and histograms are enough to make the point!

```
# barplots for bimodal variables
plot_bar(df)
```



```r
# histograms for multimodal variables
plot_histogram(df)
```

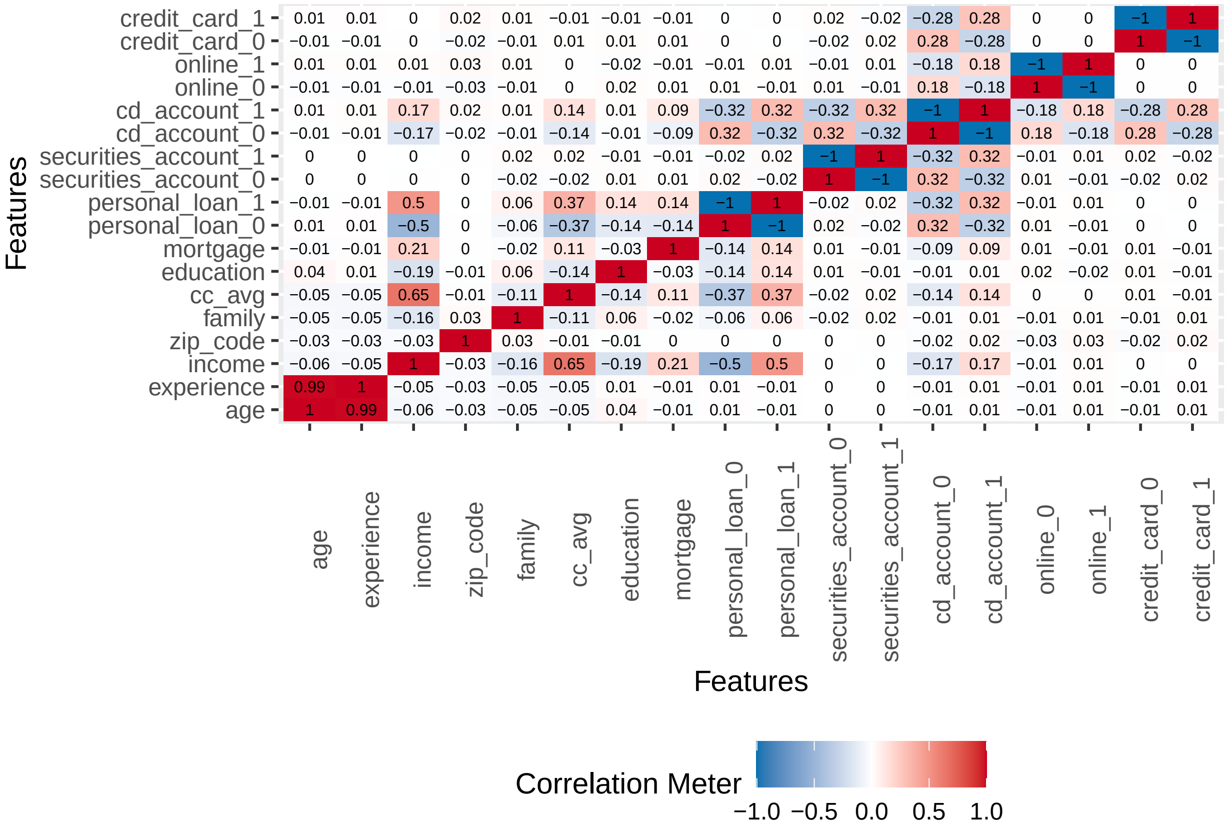


Observations:

- Age, Experience, Family, Education are homogeneously distributed as previously deducted.
- CCAvg, Income are indeed right-skewed as previously deducted. **Test log transformation.**
- Mortgage is strongly right-skewed. **Test log transformation.**
- Education: around 42% (2096/5000) of customers are undergrad, 28% Graduate and 30% Advanced/Professional
- Personal_Loan: 10% of customers accepted the personal loan offered in the last campaign.
- Securities_Account: around 10% of customers have securities account with the bank
- CD_Account: around 5-6% of customers have a certificate of deposit (CD) account with the bank.
- Online: 40% of customers use internet banking facilities.
- CreditCard: around 30% of customers use credit card from another bank.
- Importantly, notice the percentage of classes in training set; it shows that there is a class imbalance. This will be addressed later during he preparation of the model.

Now I will do bivariate analyses and correlation matrix to try to spot relationships between variables.

```r
# plot a correlation matrix (Pearson)
plot_correlation(na.omit(df), geom_text_arg = c(size = 2))
```

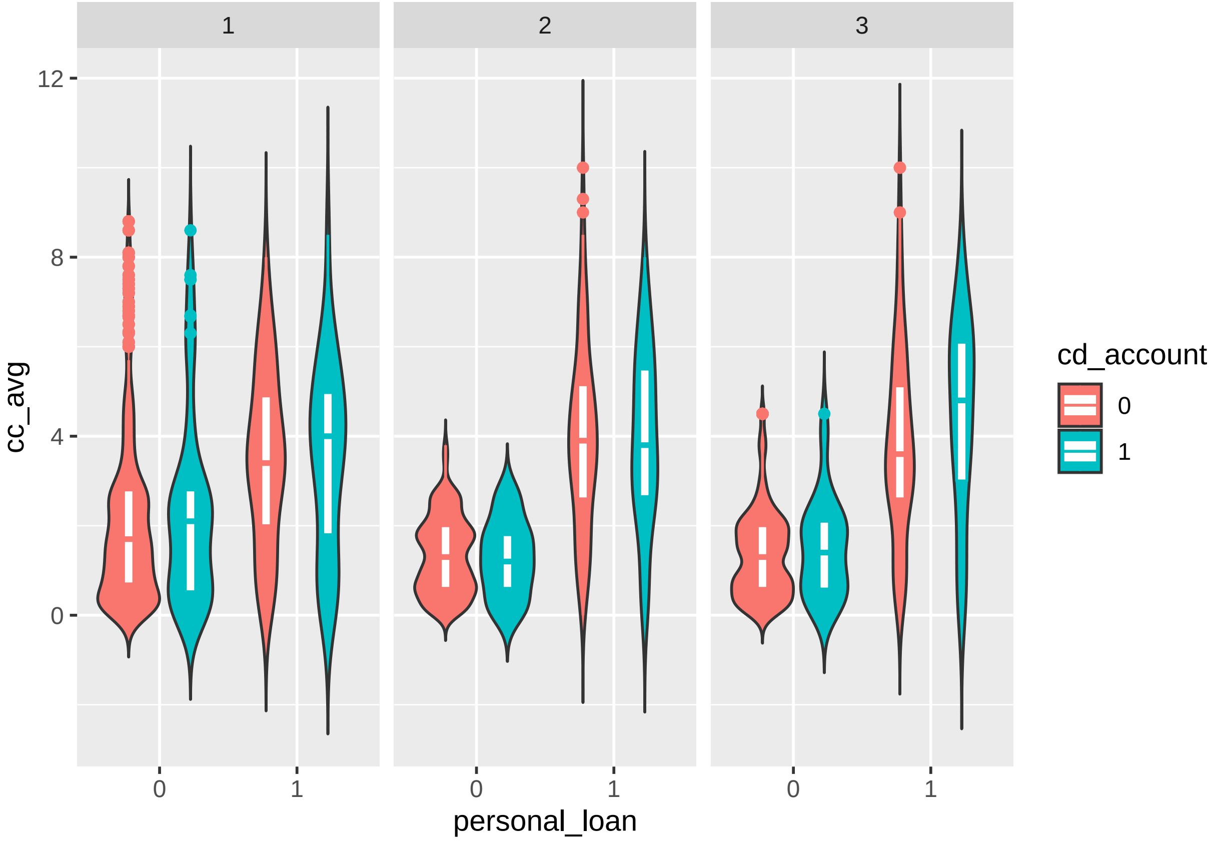


Observations:

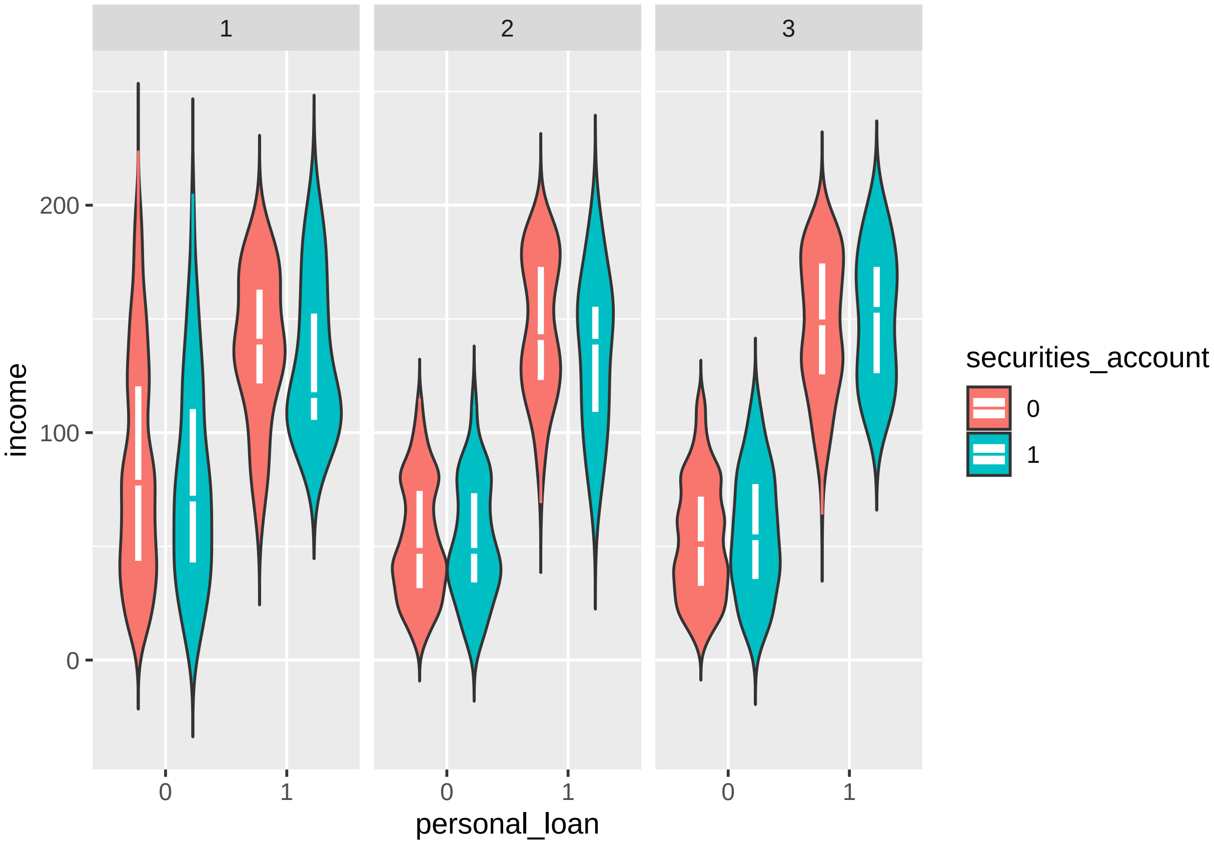
- With a threshold for the R value of 0.3, "Personal loan" (dependent variable) seems correlated (Pearson!) with Income, CCAvg, CD_Account, so these may be the most relevant independent variables.
- Income is correlated with CCAvg (dependency!).
- CD_Account is slightly correlated to Securities_Account and CreditCard.
- No correlation of ZIPCode, Family, Education, Mortage, and Online with any other variable.

- High correlation between Age and Experience.

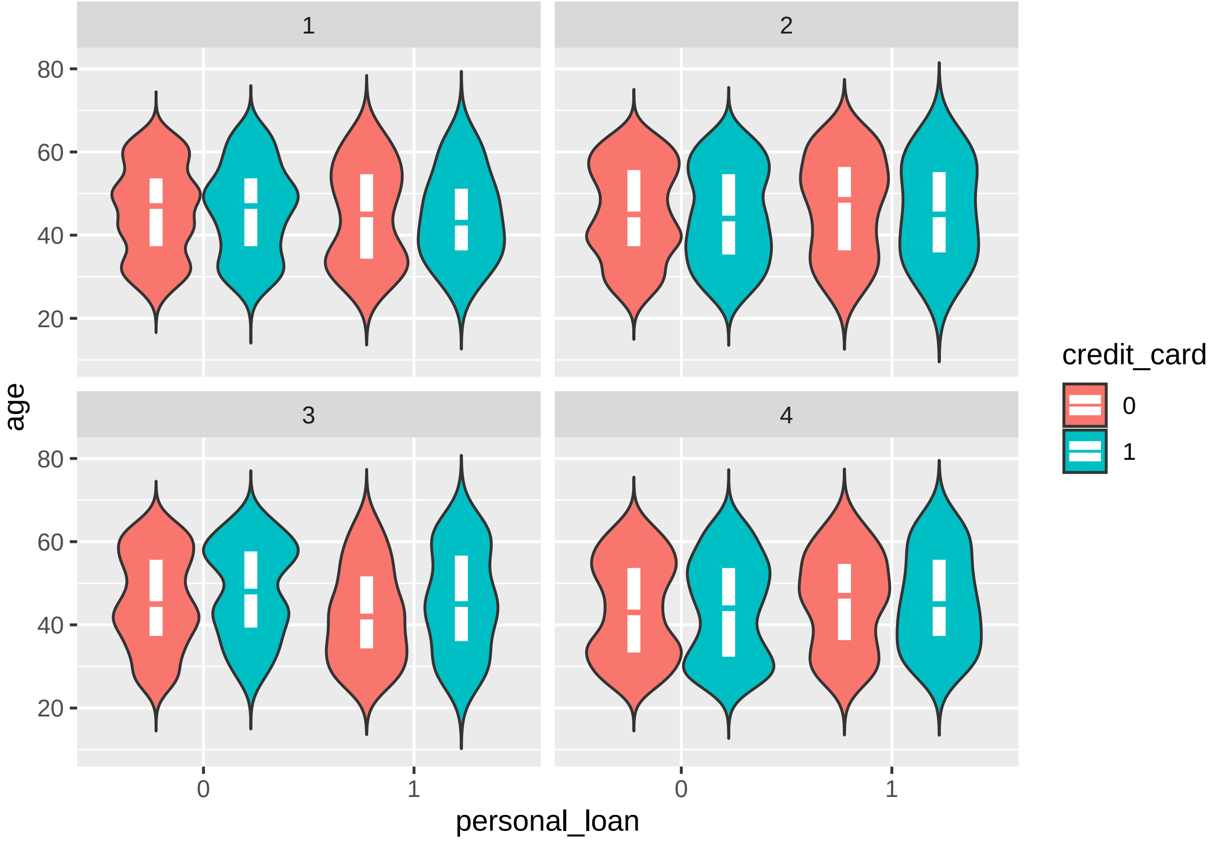
```
# a) Personal_Loan vs. CCAvg vs. CD_Account vs. Education
ggplot(df, aes(x = personal_loan, y = cc_avg)) +
    geom_violin(aes(fill = cd_account), trim = FALSE, position = position_dodge(0.9)) +
    geom_boxplot(aes(color = cd_account), width = 0.15,position = position_dodge(0.9)) +
    facet_wrap(~education)
```



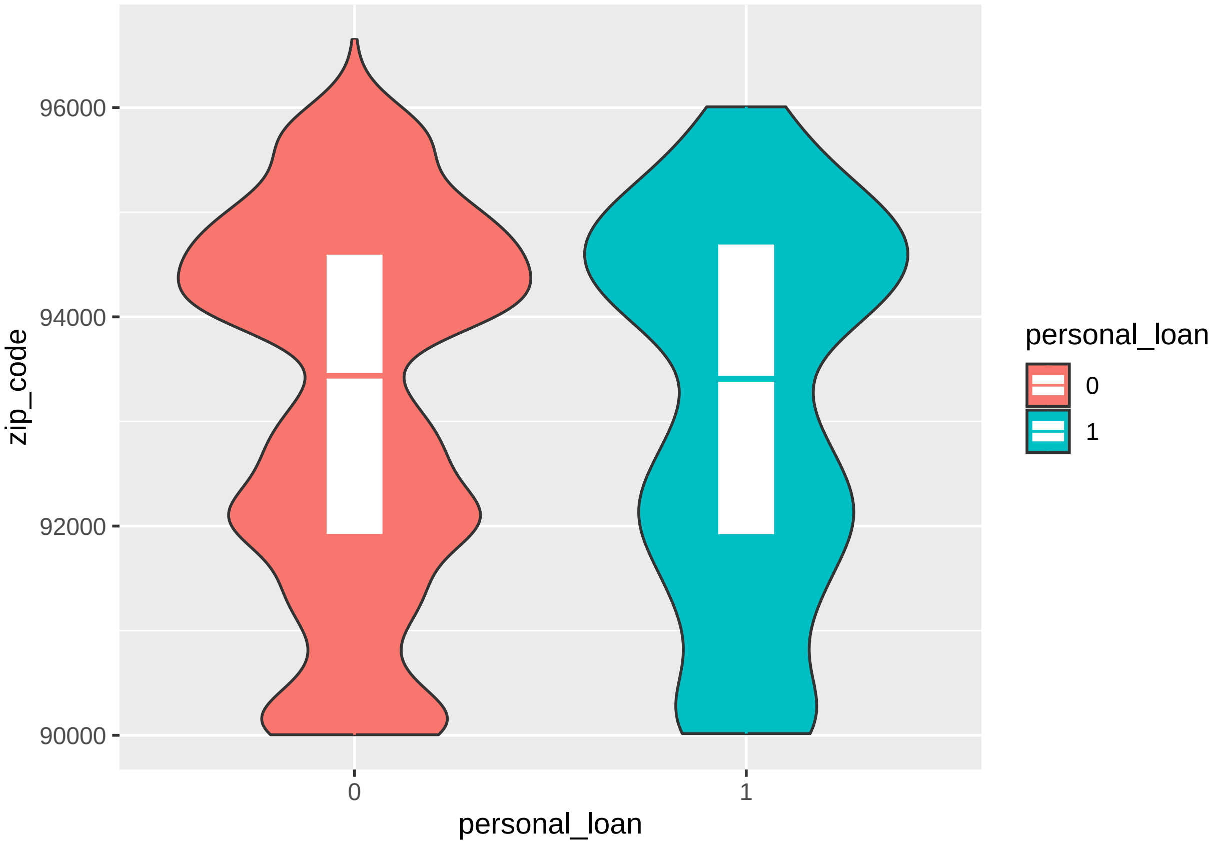
```
# b) Personal_Loan vs. Income vs. Securities_Account vs. Education
ggplot(df, aes(x = personal_loan, y = income)) +
    geom_violin(aes(fill = securities_account), trim = FALSE, position = position_dodge(0.9)) +
    geom_boxplot(aes(color = securities_account), width = 0.15,position = position_dodge(0.9)) +
    facet_wrap(~education)
```



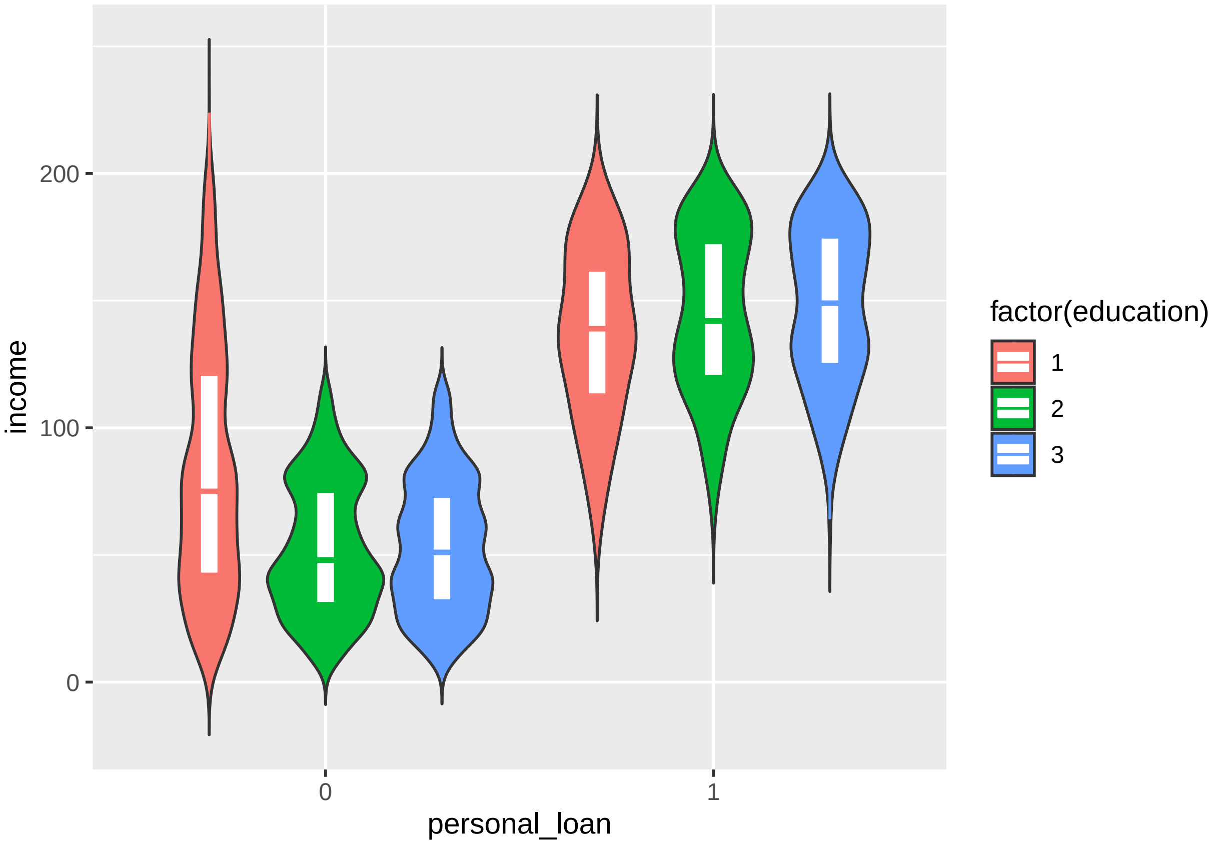
```
# c) Personal_Loan vs. Age vs. CreditCard vs. Family
ggplot(df, aes(x = personal_loan, y = age)) +
    geom_violin(aes(fill = credit_card), trim = FALSE, position = position_dodge(0.9)) +
    geom_boxplot(aes(color = credit_card), width = 0.15,position = position_dodge(0.9)) +
    facet_wrap(~family)
```



```
# d) Personal_Loan vs. ZIPCode
ggplot(df, aes(x = personal_loan, y = zip_code)) +
    geom_violin(aes(fill = personal_loan), trim = T, position = position_dodge(0.9)) +
    geom_boxplot(aes(color = personal_loan), width = 0.15,position = position_dodge(0.9))
```



```
# e) Personal_Loan vs. Experience
ggplot(df, aes(x = personal_loan, y = income)) +
    geom_violin(aes(fill = factor(education)), trim = FALSE, position = position_dodge(0.9)) +
    geom_boxplot(aes(color = factor(education)), width = 0.15,position = position_dodge(0.9))
```



Observations:

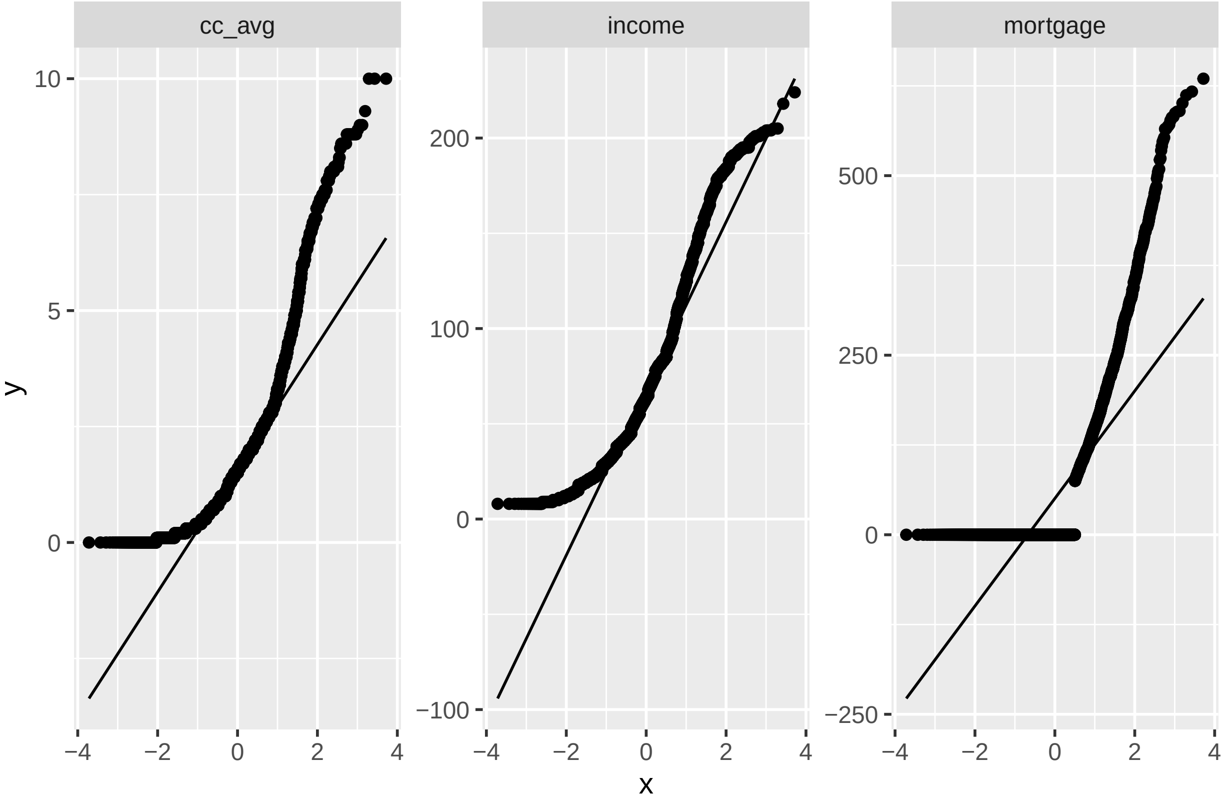
- People with the higher average credit card usage have accepted the personal loan offered in the last campaign; and this seems independent of education and certificate of deposit account.
- Clearly people with the highest incomes have accepted the personal loan offered in the last campaign; and this seems independent of education and Securities_Account.

3. Data pre-processing

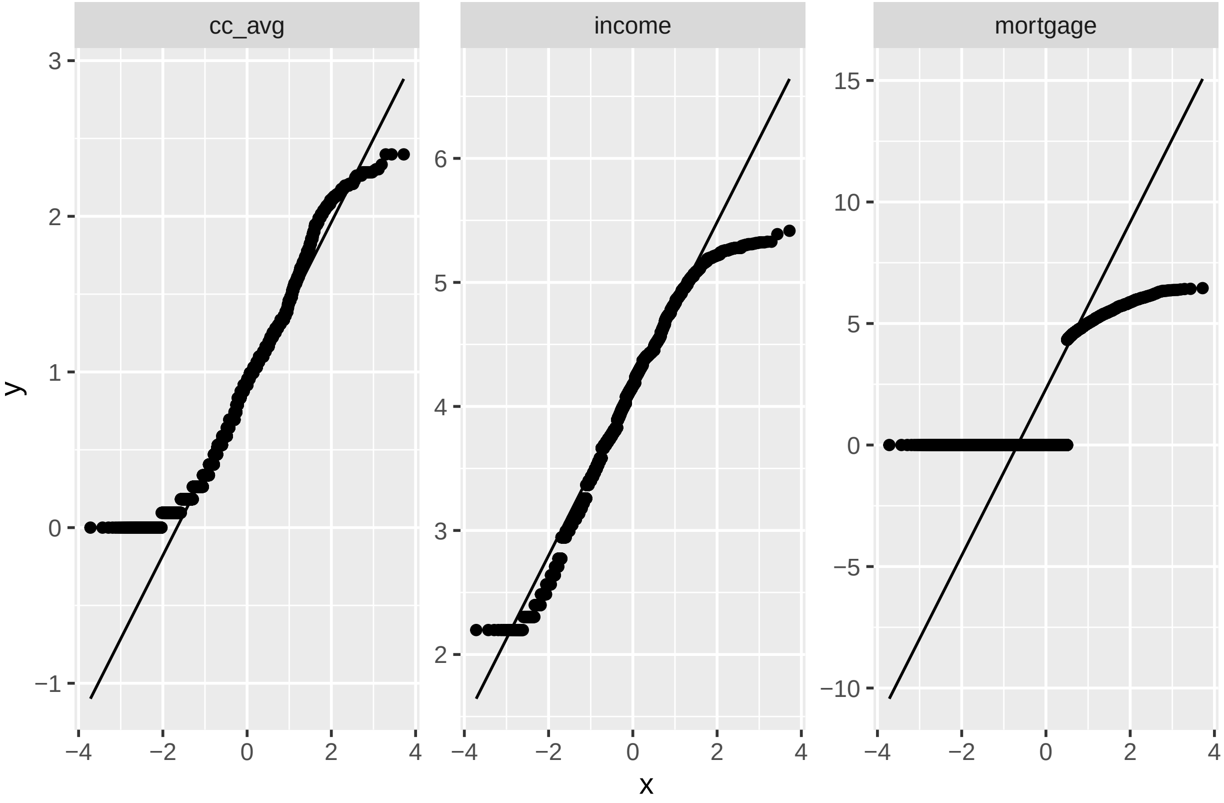
We saw from the histograms that the distribution of some variables (CCAvg, Income, Mortgage) are skewed, and thus they may need treatment. I will double check this with QQ-plots and then perform logarithmic transformation. QQ-plots are a way to visualize the deviation from a specific probability distribution (in this case, normal distribution). Then, I will treat the skwed columns in the dataframe and will drop Age, Experience, ZIPCode as they seem not to be contributing to accepting a personal loan. Family could be also eliminated, but may be relevant perhaps, so I keep it.

Now I will check and treat the skewedness of "CCAvg", "Income", "Mortgage"

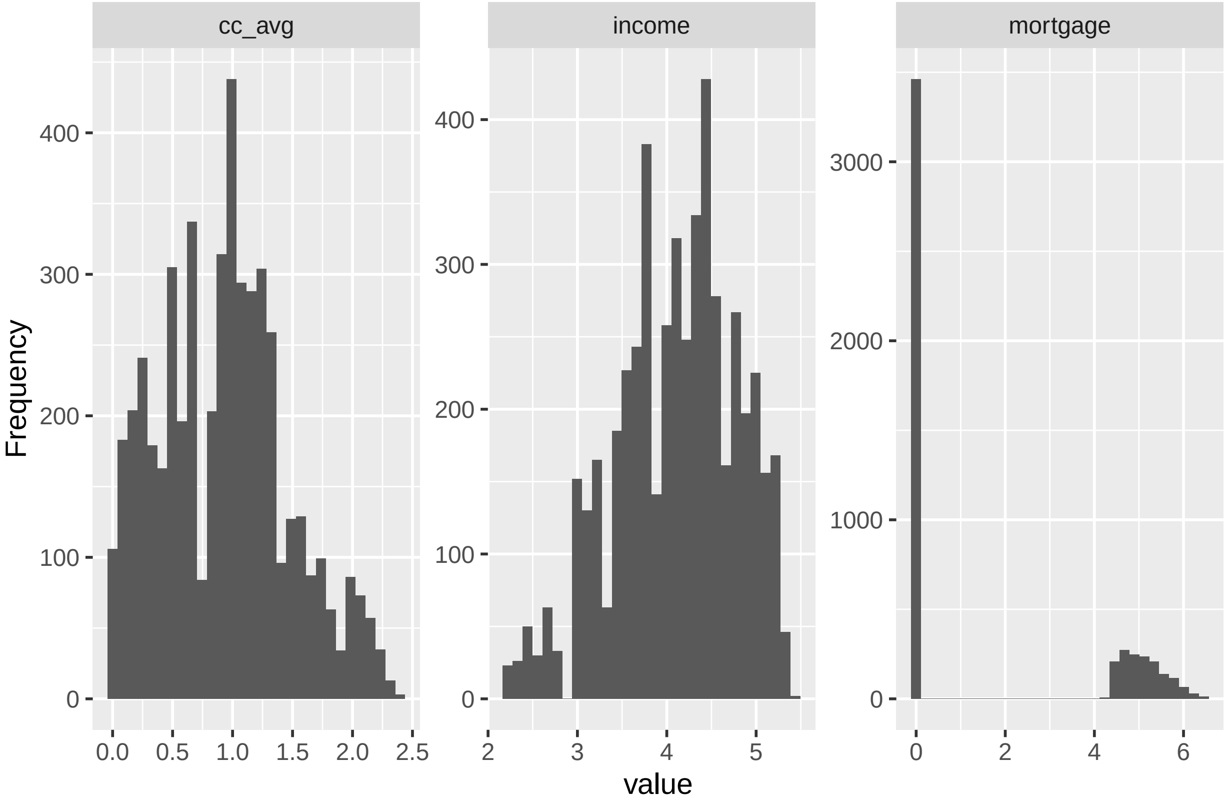
```r
# QQ plot for skewed data
qq_data <- df[, c("cc_avg", "income", "mortgage")]
plot_qq(qq_data)
```



```r
# QQ plot for log-transformed  data
log_qq_data <- update_columns(qq_data, 1:3, function(x) log(x + 1))
plot_qq(log_qq_data)
```



```r
# Histograms of log-transformed  data
plot_histogram(log_qq_data)
```



Observations:

- The QQ-plots shows indeed the three variables are skewed on both ends as they deviate form the normal distribution (straight line).
- After log transformation, CCAvg" and "Income" look more normally distribution as shown by QQ-plots and histograms. The distribution of "Mortgage" does not change much though, but I will still use log-transformation to lower the scale of the values.
- Now I will pre-process the data accordingly prior splintting into training and test sets. I will (1) drop column ID, Age, Experience, ZIPCode, and (2) log transform "CCAvg", "Income", "Mortgage".
- For this project, I will not do z-score normalization since the data is more or less already in the same scale after log-transformation.

```r
# Drop columns ID, Age, Experience, ZIPCode
df <- select(df, - c(age, experience, zip_code))

# Log transform CCAvg, Income, Mortgage
df <- update_columns(df, c(1, 3, 5), function(x) log(x + 1))
head(df)
```

```
##     income family    cc_avg education mortgage personal_loan securities_account
## 1 3.912023      4 0.9555114         1 0.000000             0                  1
## 2 3.555348      3 0.9162907         1 0.000000             0                  1
## 3 2.484907      1 0.6931472         1 0.000000             0                  0
## 4 4.615121      1 1.3083328         2 0.000000             0                  0
## 5 3.828641      4 0.6931472         2 0.000000             0                  0
## 6 3.401197      4 0.3364722         2 5.049856             0                  0
##   cd_account online credit_card
## 1          0      0           0
## 2          0      0           0
## 3          0      0           0
## 4          0      0           0
## 5          0      0           1
## 6          0      1           0
```

4. Model development

I will test two model types, namely *logistic regression* and *decision trees.* Before that, I will split the data into training and test sets. For this project, I will not split again the training data into training and validation, but notice that this could/should be done, to ensure that data leaking happens.

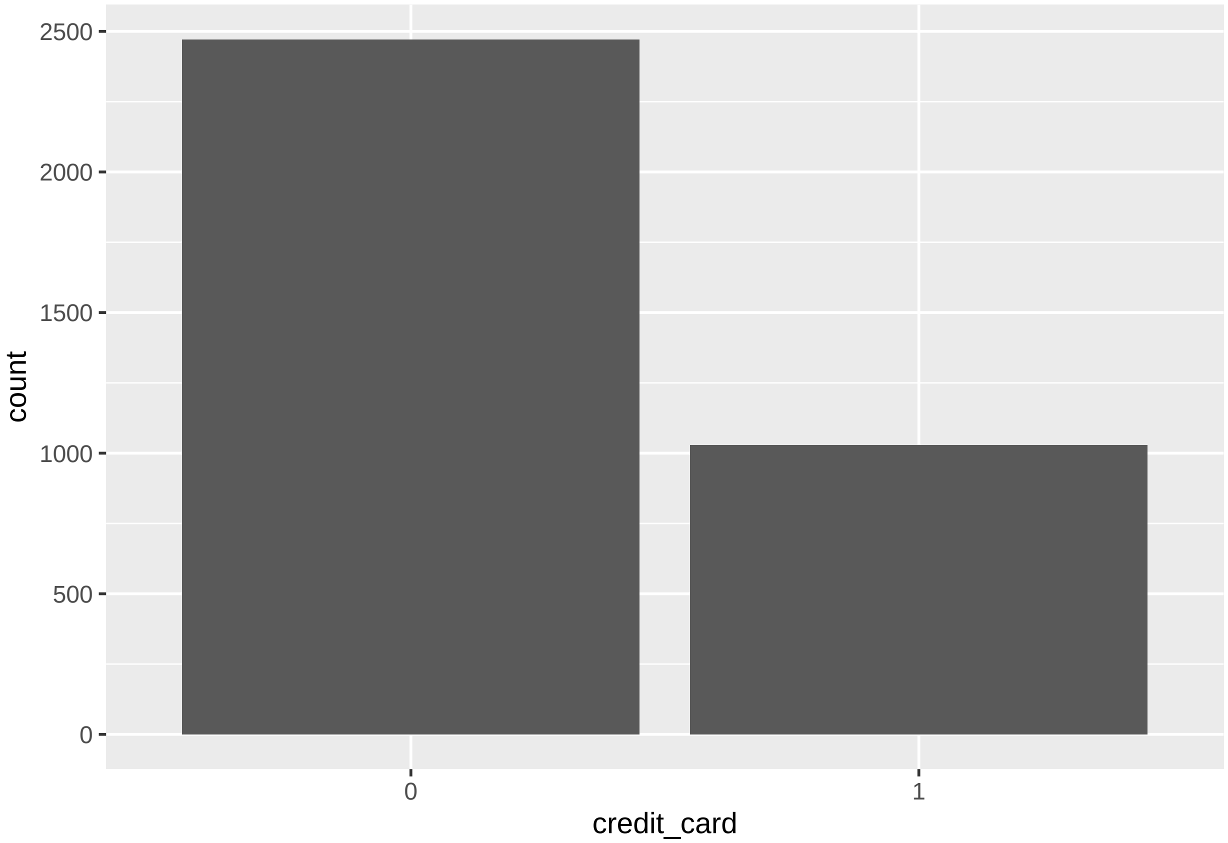
Which metric is the most appropriate metric to evaluate the model according to the problem statement? I would say **recall** because we want minimize false negative, i.e. reduce the chances of missing the oportunity to identify a liability customer that will buy a personal loan. Remember, recall = sensitivity, true pos rate or TP/(TP+FN).

4.1 Split data into training and test sets

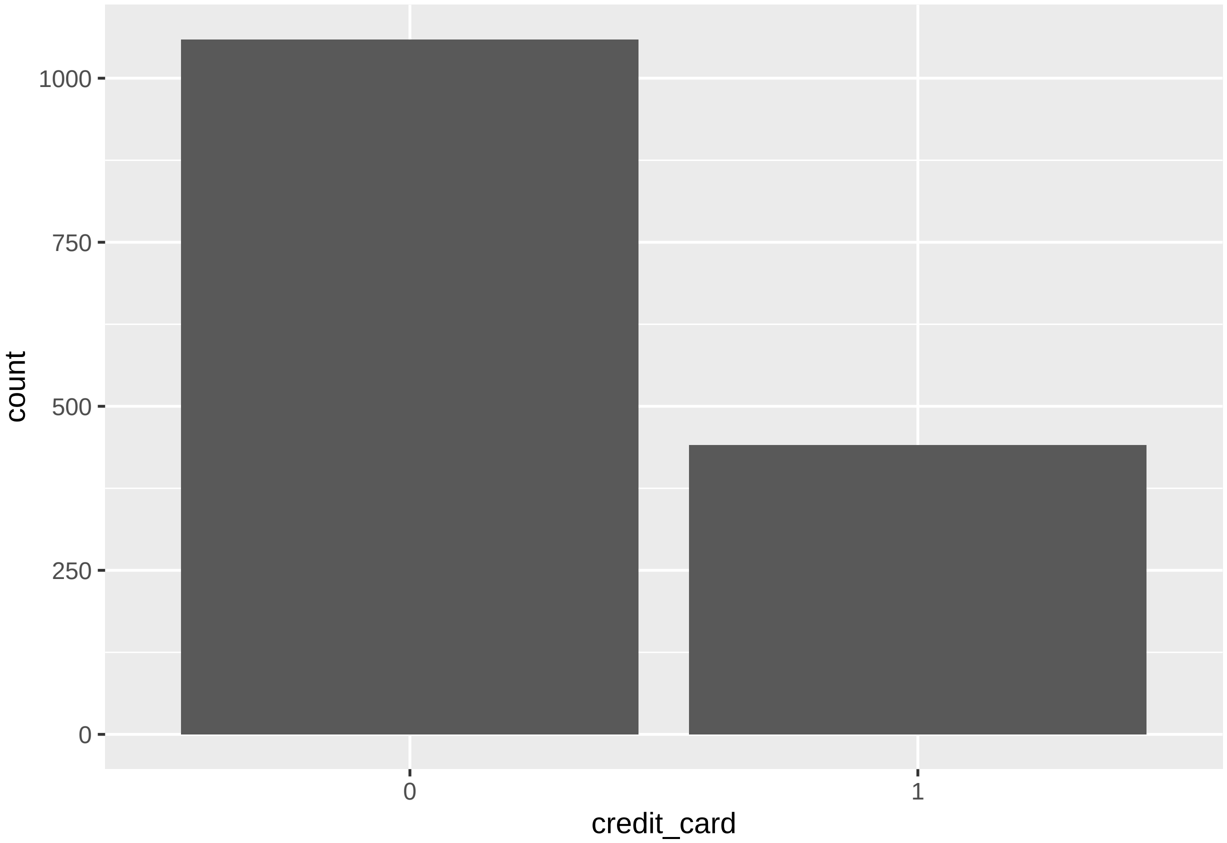
```r
# Split data into 30% test/ 70% training
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
test_index <- createDataPartition(y = df$credit_card, times = 1, p = 0.3, list = FALSE)

# Subset test data by test_index (all raws in test_index, and all columns)
test_data <- df[test_index, ]

# Subset training data by test_index (rest of raws, and all columns)
train_data <- df[-test_index, ]

# Remove test_index
rm(test_index)

# check the data distribution in training and test set
ggplot(train_data, aes(x=credit_card)) + geom_bar(aes(y = ..count.., group = 1))
```



```
ggplot(test_data, aes(x=credit_card)) + geom_bar(aes(y = ..count.., group = 1))
```



Observations:

- Clearly, the distribution of the dependent variable CreditCard is imbalanced, as there is about half 1s than 0s. This will be considere for model generation and thus treated next.

4.2 Treat data imbalance

I will test 4 different approaches using the help of the Caret, performanceEstimation and ROSE packages: down-sampling, up-sampling, SMOTE (hybrid method) and ROSE (hybrid method). You can find more information about this on https://topepo.github.io/caret/subsampling-for-class-imbalances.html . I will treat the training dataset and use it for fitting the algorithm. The test will done, however, with untreated data.

```r
# compare to the original data
table(train_data$credit_card)
```

```
##
##    0    1
## 2471 1029
```

```r
# down-sampling
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
down_train <- downSample(x = train_data[, -ncol(train_data)], y = factor(train_data$credit_card))
down_train <- rename(down_train, credit_card = Class)
table(down_train$credit_card)
```

```
##
##    0    1
## 1029 1029
```

```
# up-sampling
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
up_train <- upSample(x = train_data[, -ncol(train_data)], y = factor(train_data$credit_card))
up_train <- rename(up_train, credit_card = Class)
table(up_train$credit_card)
```

```
##
##    0    1
## 2471 2471
```

```
# SMOTE
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
smote_train <- performanceEstimation::smote(credit_card ~ ., data  = train_data,
                                            perc.over = 0.5, perc.under = 2)
#smote_train$credit_card <- as.numeric(smote_train$credit_card)
table(smote_train$credit_card)
```

```
##
##    0    1
## 1028 1543
```

```
# ROSE
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
rose_train <- ROSE(credit_card ~ ., data  = train_data)$data
table(rose_train$credit_card)
```

```
##
##    0    1
## 1801 1699
```

Observations:

- Except for up-sampling, all balancing approaches subset all the classes in the training set so that their class frequencies match (or approximates) the least prevalent class (the target value CreditCard)

4.3 Build the logistic regression models

Compare unbalanced vs. balanced data (training data!!), check the performance (test data!!) and calculate the summaries of the models.

Note: 1. predict() generates the logits (probabilities) for each sample in the test set. 2. A decision boundary of 0.5 will decide the binary classification of the sample. I use here ifelse() to get the output vector. 3. I will compile the residuals of each model for a comparison.

```r
# Model with unbalanced data
lr_model <- glm(credit_card ~ ., data = train_data, family = "binomial")
residuals_lr_model <- summary(residuals(lr_model))

# Model with down-sampling balancing
lr_model_downSampling <- glm(credit_card ~ ., data = down_train, family = "binomial")
residuals_lr_model_downSampling <- summary(residuals(lr_model_downSampling))

# Model with up-sampling balancing
lr_model_upSampling <- glm(credit_card ~ ., data = up_train, family = "binomial")
residuals_lr_model_upSampling <- summary(residuals(lr_model_upSampling))

# Model with SMOTE balancing
lr_model_SMOTE <- glm(credit_card ~ ., data = smote_train, family = "binomial")
residuals_lr_model_SMOTE <- summary(residuals(lr_model_SMOTE))

# Model with ROSE balancing
lr_model_ROSE <- glm(credit_card ~ ., data = rose_train, family = "binomial")
residuals_lr_model_ROSE <- summary(residuals(lr_model_ROSE))

# Compile and compare the residuals of each model
residual_models <- rbind(residuals_lr_model, residuals_lr_model_downSampling,
                         residuals_lr_model_upSampling, residuals_lr_model_SMOTE,
                         residuals_lr_model_ROSE)
knitr::kable(residual_models)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
residuals_lr_model	-2.009337	-0.8310498	-0.7527627	-0.1225308	0.7659143	2.138705
residuals_lr_model_downSampling	-2.263796	-1.1269051	-0.1194492	-0.0052443	1.1759890	1.740583
residuals_lr_model_upSampling	-2.346624	-1.1302209	-0.1137739	-0.0060114	1.1786993	1.743963
residuals_lr_model_SMOTE	-2.328237	-1.2717668	0.8573288	0.0554429	1.0364436	1.556474
residuals_lr_model_ROSE	-2.253125	-1.1280839	-0.6154311	-0.0140236	1.1976381	1.852180

Interpretation of the Deviance Residuals:

- The deviance residuals help identifying the overall discrepancies between models and data outliers.
- As they are normal standardized values, they are good when close to being centered on 0 and and the values (1Q and 3Q i particular!) are roughly similarly distant from the median.
- Here we observe that in the unbalanced data indeed the media is rather away from 0 and Q1/Q3 are not similarly distant from the media. The median is more near to Q1!
- We also observed that all down-sampling and up-sampling shift indeed the median towards 0, and the Q1/Q3 values are more similarly distant from the median. SMOTE and ROSE do not shift significantly the median towards 0.
- It seems that up-sampling makes the best balancing, followed by down-sampling.

Now let's check the performance as measured by recall in test data.

```r
# Model with unbalanced data
fitted.results.lr_model <- ifelse(predict(lr_model,
                                          newdata = test_data,
                                          type = "response") > 0.5, 1, 0) %>% as_factor()

lr_model_recall <- recall(data = fitted.results.lr_model,
                          reference = test_data$credit_card)
```

```r
# Model with down-sampling balancing
fitted.results.lr_model_downSampling <- ifelse(predict(lr_model_downSampling,
                                                       newdata = test_data,
                                                       type = "response") > 0.5, 1, 0) %>% as_factor()

lr_model_downSampling_recall <- recall(data = fitted.results.lr_model_downSampling,
                                       reference = test_data$credit_card)

# Model with up-sampling balancing
fitted.results.lr_model_upSampling <- ifelse(predict(lr_model_upSampling,
                                                     newdata = test_data,
                                                     type = "response") > 0.5, 1, 0) %>% as_factor()

lr_model_upSampling_recall <- recall(data = fitted.results.lr_model_upSampling,
                                     reference = test_data$credit_card)

# Model with SMOTE balancing
fitted.results.lr_model_SMOTE <- ifelse(predict(lr_model_SMOTE,
                                                newdata = test_data,
                                                type = "response") > 0.5, 1, 0) %>% as_factor()

lr_model_SMOTE_recall <- recall(data = fitted.results.lr_model_SMOTE,
                                reference = test_data$credit_card)

# Model with ROSE balancing
fitted.results.lr_model_ROSE <- ifelse(predict(lr_model_ROSE,
                                               newdata = test_data,
                                               type = "response") > 0.5, 1, 0) %>% as_factor()

lr_model_ROSE_recall <- recall(data = fitted.results.lr_model_ROSE,
                               reference = test_data$credit_card)


recall_models <- rbind(lr_model_recall, lr_model_downSampling_recall,
                       lr_model_upSampling_recall, lr_model_SMOTE_recall,
                       lr_model_ROSE_recall)

knitr::kable(recall_models)
```

lr_model_recall	0.9801700
lr_model_downSampling_recall	0.6430595
lr_model_upSampling_recall	0.6676110
lr_model_SMOTE_recall	0.1718602
lr_model_ROSE_recall	0.7780925

Observations: performance of the model on test data - Here the best recall value was given by the model using a up-sampling balancing approach. - The recall values are in general relative low however. - Interestingly, the recall value of the unbalanced data is highest. See discussion in chapter 5. - Let's check the coefficients of the model using imbalanced and balanced data using a **up-sampling** approach and the confusion matrix (cm).

First, let's check the coefficients using imbalanced data.

```
# display the summary with coefficients
summary(lr_model)
```

```
##
## Call:
## glm(formula = credit_card ~ ., family = "binomial", data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0093  -0.8310  -0.7528   0.7659   2.1387
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.694587   0.328100  -2.117 0.034260 *
## income              -0.015318   0.075511  -0.203 0.839250
## family               0.066770   0.034509   1.935 0.053006 .
## cc_avg              -0.119058   0.094528  -1.259 0.207851
## education           -0.007748   0.048255  -0.161 0.872433
## mortgage            -0.015587   0.016909  -0.922 0.356630
## personal_loan1      -0.984020   0.191559  -5.137 2.79e-07 ***
## securities_account1 -1.286538   0.181944  -7.071 1.54e-12 ***
## cd_account1          3.727961   0.242487  15.374  < 2e-16 ***
## online1             -0.294634   0.080383  -3.665 0.000247 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4239.9  on 3499  degrees of freedom
## Residual deviance: 3876.8  on 3490  degrees of freedom
## AIC: 3896.8
##
## Number of Fisher Scoring iterations: 4
```

Interpretation of the coefficients:

- Coefficients = this follows the equation Y = intercept + b1X1 + b2X2 + … bnXn, for the **log(odds)** graph; where b ar the Estimates and the intercept is the value at which the line cross the Y axis when all other values of Xs are 0. This means, since the intercept is negative (and significant!), the odds are against getting a Credit card if all other variables are 0.
- We see CD Account has a positive significant Estimate Values, so that an increase in X will lead to a corresponding fold increase in chances of the dependent variable (e.g. 3.6 fold)
- In contrast, Personal Loan, Securities Account, and Online have a negative significant effect on the dependent variable, meaning if they increase the dependent variable will decrease by the corresponding fold.
- As these five Estimates are statistically significant, it indicate that they could be useful to predict the outcome.

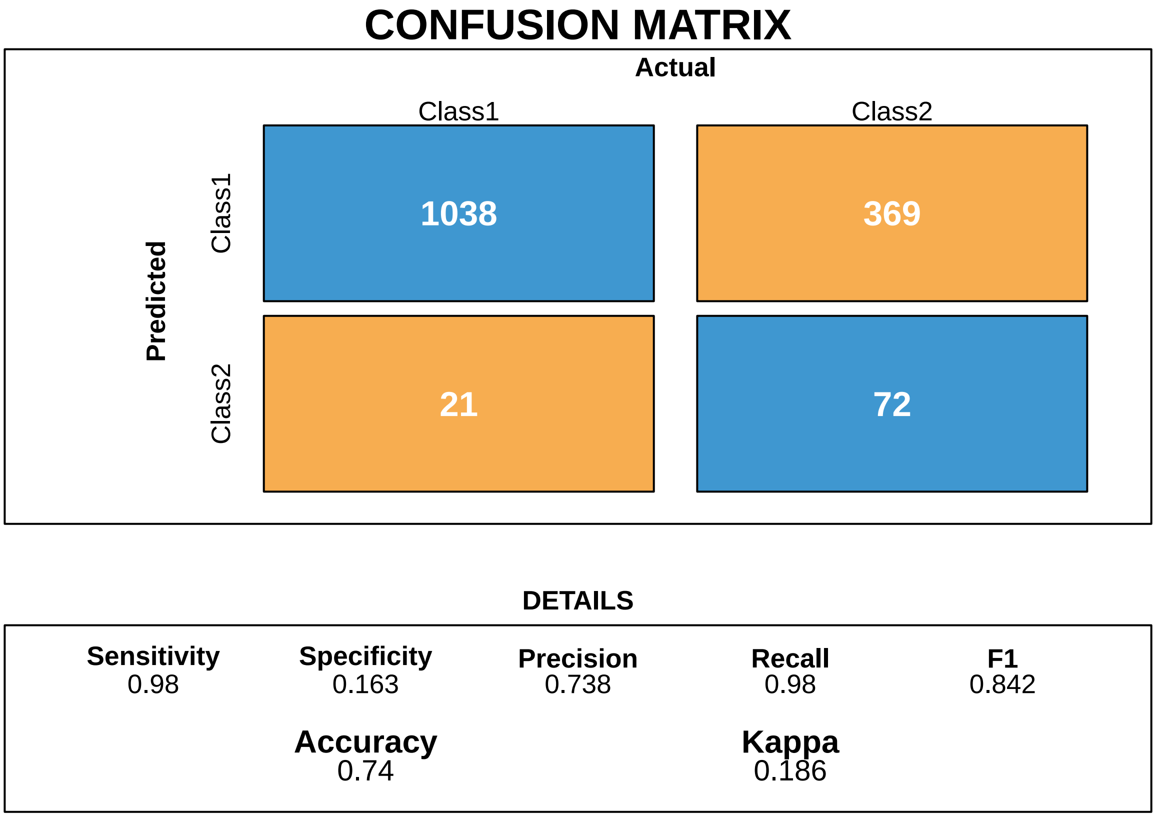
Next, let's generate the function to nicely draw the matrix. I will then fit the function with the data from the confusion matrix generated with Caret's confusionMatrix(). The code for drawing the the matrix is borrowed from https://stackoverflow.com/questions/23891140/r-how-to-visualize-confusion-matrix-using-the-caret-package .

```
draw_confusion_matrix <- function(cm) {

  layout(matrix(c(1,1,2)))
  par(mar=c(2,2,2,2))
  plot(c(100, 345), c(300, 450), type = "n", xlab="", ylab="", xaxt='n', yaxt='n')
  title('CONFUSION MATRIX', cex.main=2)

  # create the matrix
  rect(150, 430, 240, 370, col='#3F97D0')
  text(195, 435, 'Class1', cex=1.2)
  rect(250, 430, 340, 370, col='#F7AD50')
  text(295, 435, 'Class2', cex=1.2)
  text(125, 370, 'Predicted', cex=1.3, srt=90, font=2)
  text(245, 450, 'Actual', cex=1.3, font=2)
  rect(150, 305, 240, 365, col='#F7AD50')
  rect(250, 305, 340, 365, col='#3F97D0')
  text(140, 400, 'Class1', cex=1.2, srt=90)
  text(140, 335, 'Class2', cex=1.2, srt=90)

  # add in the cm results
  res <- as.numeric(cm$table)
  text(195, 400, res[1], cex=1.6, font=2, col='white')
  text(195, 335, res[2], cex=1.6, font=2, col='white')
  text(295, 400, res[3], cex=1.6, font=2, col='white')
  text(295, 335, res[4], cex=1.6, font=2, col='white')

  # add in the specifics
  plot(c(100, 0), c(100, 0), type = "n", xlab="", ylab="", main = "DETAILS", xaxt='n', yaxt='n')
  text(10, 85, names(cm$byClass[1]), cex=1.2, font=2)
  text(10, 70, round(as.numeric(cm$byClass[1]), 3), cex=1.2)
  text(30, 85, names(cm$byClass[2]), cex=1.2, font=2)
  text(30, 70, round(as.numeric(cm$byClass[2]), 3), cex=1.2)
  text(50, 85, names(cm$byClass[5]), cex=1.2, font=2)
  text(50, 70, round(as.numeric(cm$byClass[5]), 3), cex=1.2)
  text(70, 85, names(cm$byClass[6]), cex=1.2, font=2)
  text(70, 70, round(as.numeric(cm$byClass[6]), 3), cex=1.2)
  text(90, 85, names(cm$byClass[7]), cex=1.2, font=2)
  text(90, 70, round(as.numeric(cm$byClass[7]), 3), cex=1.2)

  # add in the accuracy information
  text(30, 35, names(cm$overall[1]), cex=1.5, font=2)
  text(30, 20, round(as.numeric(cm$overall[1]), 3), cex=1.4)
  text(70, 35, names(cm$overall[2]), cex=1.5, font=2)
  text(70, 20, round(as.numeric(cm$overall[2]), 3), cex=1.4)
}
```

Now let's generate and plot the confusion matrix using imbalanced data.

```
# generate the CM (lr_cm)
lr_cm <- confusionMatrix(
  fitted.results.lr_model,
  test_data$credit_card,
  positive = NULL,
  dnn = c("Prediction", "Reference"),
  prevalence = NULL,
  mode = "everything")
```

```
draw_confusion_matrix(lr_cm)
```



Now, let's check the coefficients using balanced data via up-sampling.

```
# display the summary with coefficients
summary(lr_model_upSampling)
```

```
##
## Call:
## glm(formula = credit_card ~ ., family = "binomial", data = up_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3466  -1.1302  -0.1138   1.1787   1.7440
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)            0.395538   0.251219   1.574  0.11538
## income                -0.080146   0.057971  -1.383  0.16682
## family                 0.094470   0.026313   3.590  0.00033 ***
## cc_avg                -0.098567   0.072357  -1.362  0.17313
## education             -0.031090   0.036801  -0.845  0.39822
## mortgage              -0.006984   0.012845  -0.544  0.58665
## personal_loan1        -0.846883   0.138095  -6.133 8.65e-10 ***
## securities_account1 -1.215914   0.128815  -9.439  < 2e-16 ***
## cd_account1            3.610353   0.200368  18.019  < 2e-16 ***
## online1               -0.328559   0.061853  -5.312 1.08e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

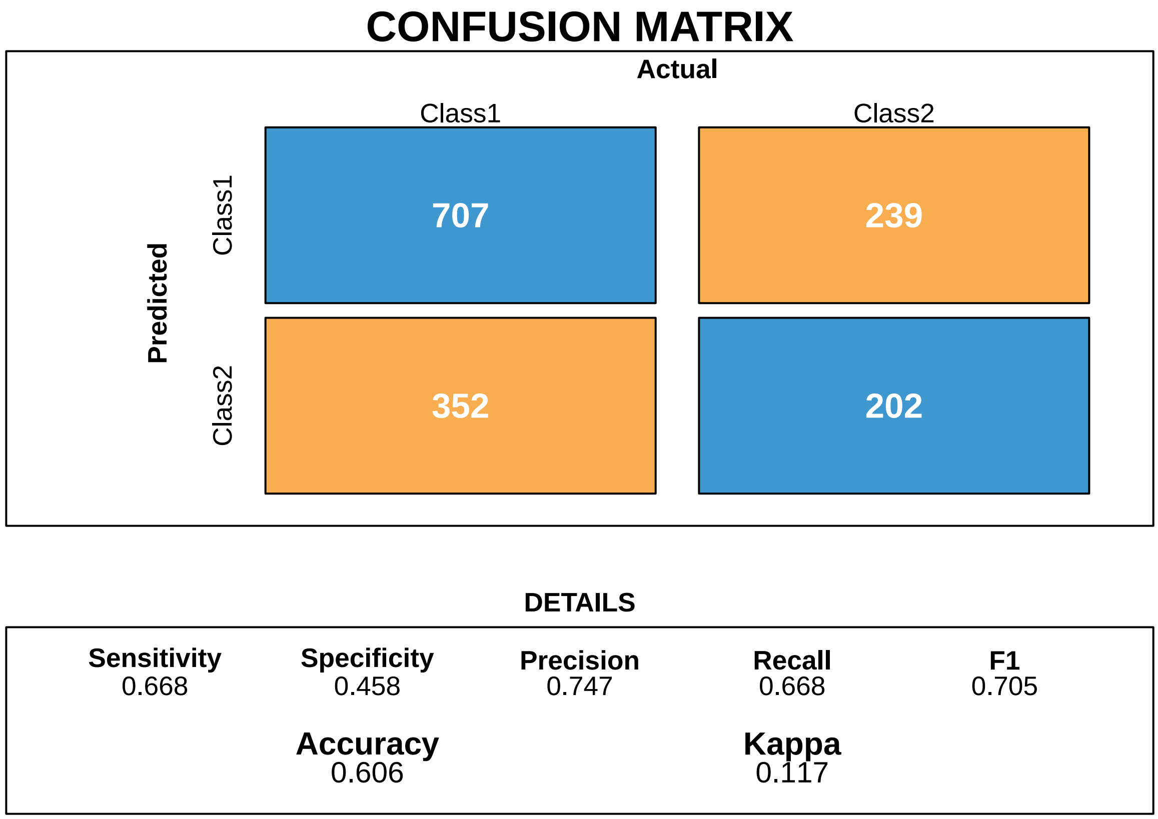
```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6851.1  on 4941  degrees of freedom
## Residual deviance: 6272.1  on 4932  degrees of freedom
## AIC: 6292.1
##
## Number of Fisher Scoring iterations: 5
```

Interpretation of the coefficients:

- Here the intercept is positive (and significant!), meaning that the odds are for getting a Credit card if all other variables are 0.
- We see CD Account **and Family** have a positive significant Estimate Values, so that an increase in X will lead to a corresponding fold increase in chances of the dependent variable.
- In contrast, as with imbalanced data, Personal Loan, Securities Account, and Online have a negative significant effect on the dependent variable, meaning if they increase the dependent variable will decrease by the corresponding fold.
- As these six Estimates are statistically significant, it indicate that they could be useful to predict the outcome.

Now let's check the confusion matrix using balanced data via up-sampling.

```r
# generate the CM (lr_cm)
lr_cm_upSampling <- confusionMatrix(
  fitted.results.lr_model_upSampling,
  test_data$credit_card,
  positive = NULL,
  dnn = c("Prediction", "Reference"),
  prevalence = NULL,
  mode = "everything")

draw_confusion_matrix(lr_cm_upSampling)
```



CONFUSION MATRIX

Predicted	Actual: Class1	Actual: Class2
Class1	707	239
Class2	352	202

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.668	0.458	0.747	0.668	0.705

Accuracy	Kappa
0.606	0.117

Observations:

- Now we see how many values were predicted correctly or incorrectly as well as several performance metric besides recall, for both models.
- We see again that most metrics are higher using imbalanced data compared to balanced data via up-scaling.
- We also observe that the F1 score, is not high, but also not so bad.
- Notably, specificity is higher with balanced data (0.458) compared to imbalanced data (0.163), indicating that the former is better at predicting true negatives.

Next I will compare to a decision tree classifier.

4.4 Build the decision tree classifier

I will check first a classifier using default parameters with imbalanced data.

```
# 1. Set the seed
set.seed(1, sample.kind="Rounding")
```

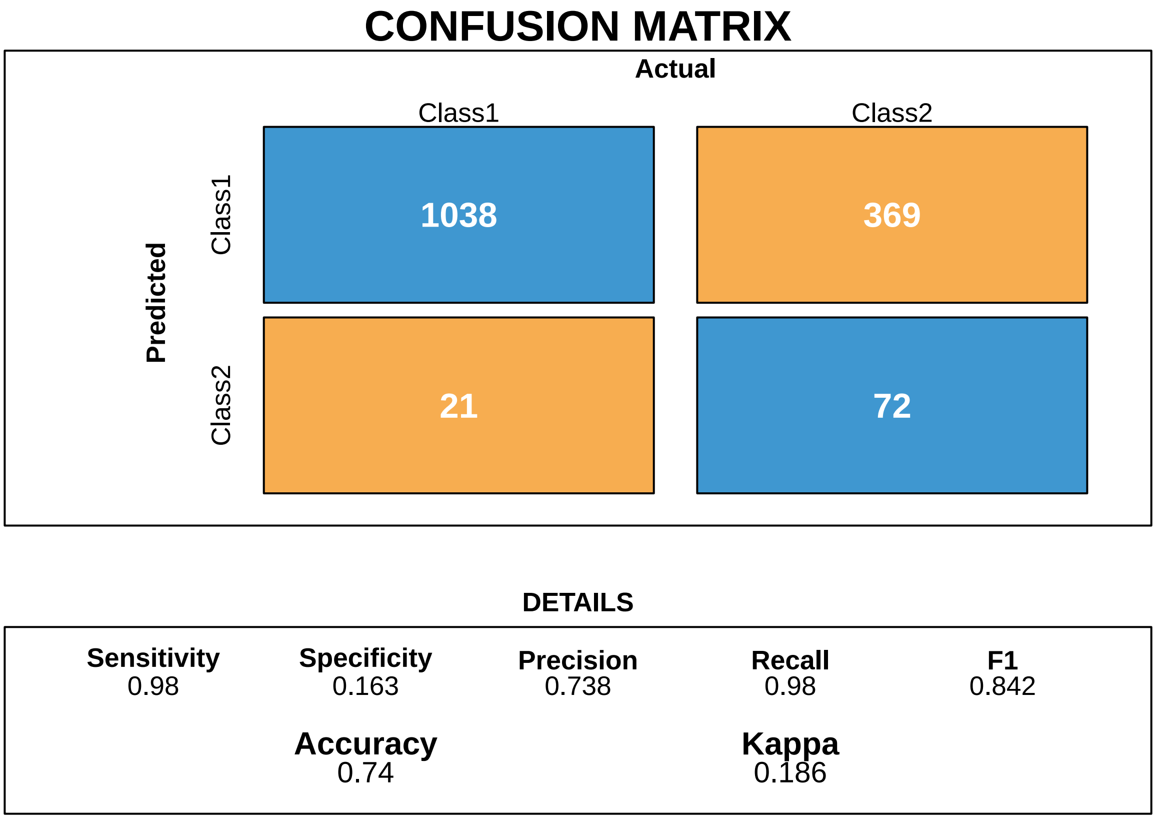
```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
# 2. Generate and fit the model with imbalanced data and default parameters
rt_model <- train(credit_card ~ ., data = train_data, method = "rpart")
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```

```
fitted.results.rt_model <- predict(rt_model,test_data)
```

```
# 3. Generate and plot the confusion matrix
rt_model_cm <- confusionMatrix(
  fitted.results.rt_model,
  test_data$credit_card,
  positive = NULL,
  dnn = c("Prediction", "Reference"),
  prevalence = NULL,
  mode = "everything")

draw_confusion_matrix(rt_model_cm)
```



CONFUSION MATRIX

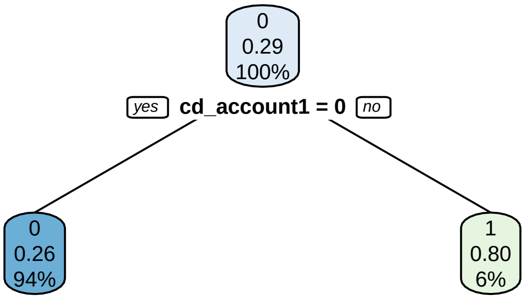
Predicted	Actual: Class1	Actual: Class2
Class1	1038	369
Class2	21	72

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.98	0.163	0.738	0.98	0.842

Accuracy	Kappa
0.74	0.186

```
# 4. Plot the decision tree
rpart.plot(rt_model$finalModel, fallen.leaves = FALSE)
```



Next, I will check a classifier with up-sampling-based balanced data, using default parameters.

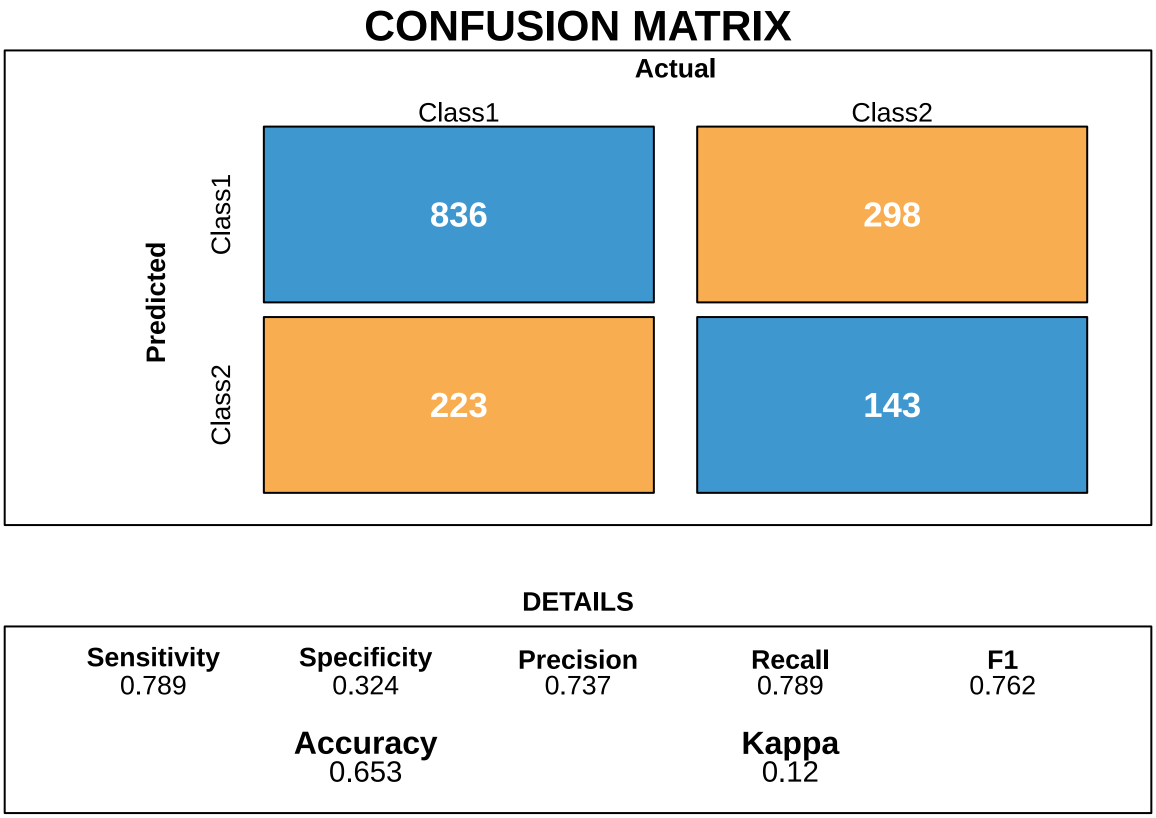
```
# 1. Set the seed
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

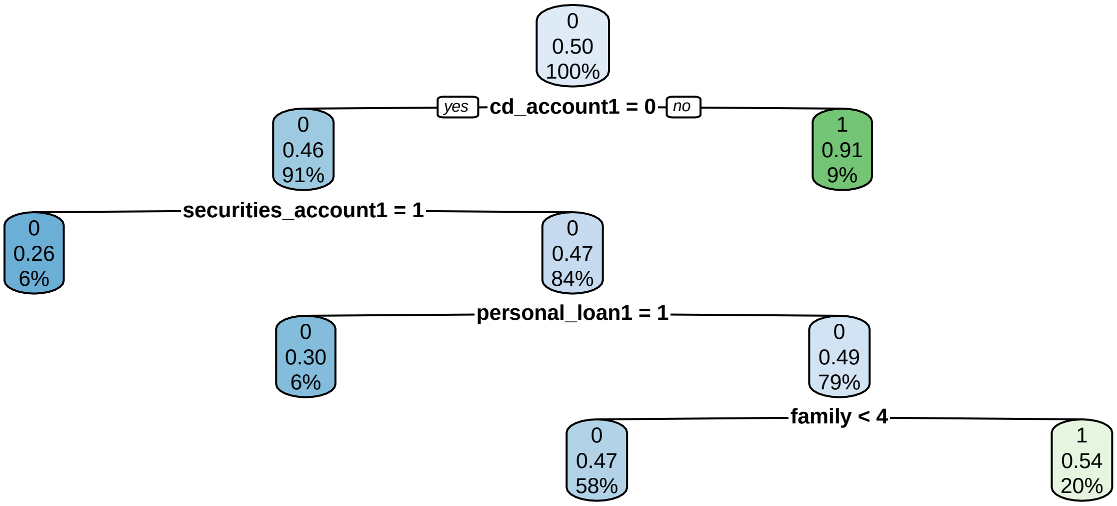
```
# 2. Generate and fit the model with imbalanced data and default parameters
rt_model_upSampling <- train(credit_card ~ ., data = up_train, method = "rpart")
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```

```
fitted.results.rt_model_upSampling <- predict(rt_model_upSampling,test_data)

# 3. Generate and plot the confusion matrix
rt_model_cm_upSampling <- confusionMatrix(
  fitted.results.rt_model_upSampling,
  test_data$credit_card,
  positive = NULL,
  dnn = c("Prediction", "Reference"),
  prevalence = NULL,
  mode = "everything")

draw_confusion_matrix(rt_model_cm_upSampling)
```



```
# 4. Plot the decision tree
rpart.plot(rt_model_upSampling$finalModel, fallen.leaves = FALSE)
```



Finally, I will check a classifier with up-sampling-based balanced data, using tunned parameters.

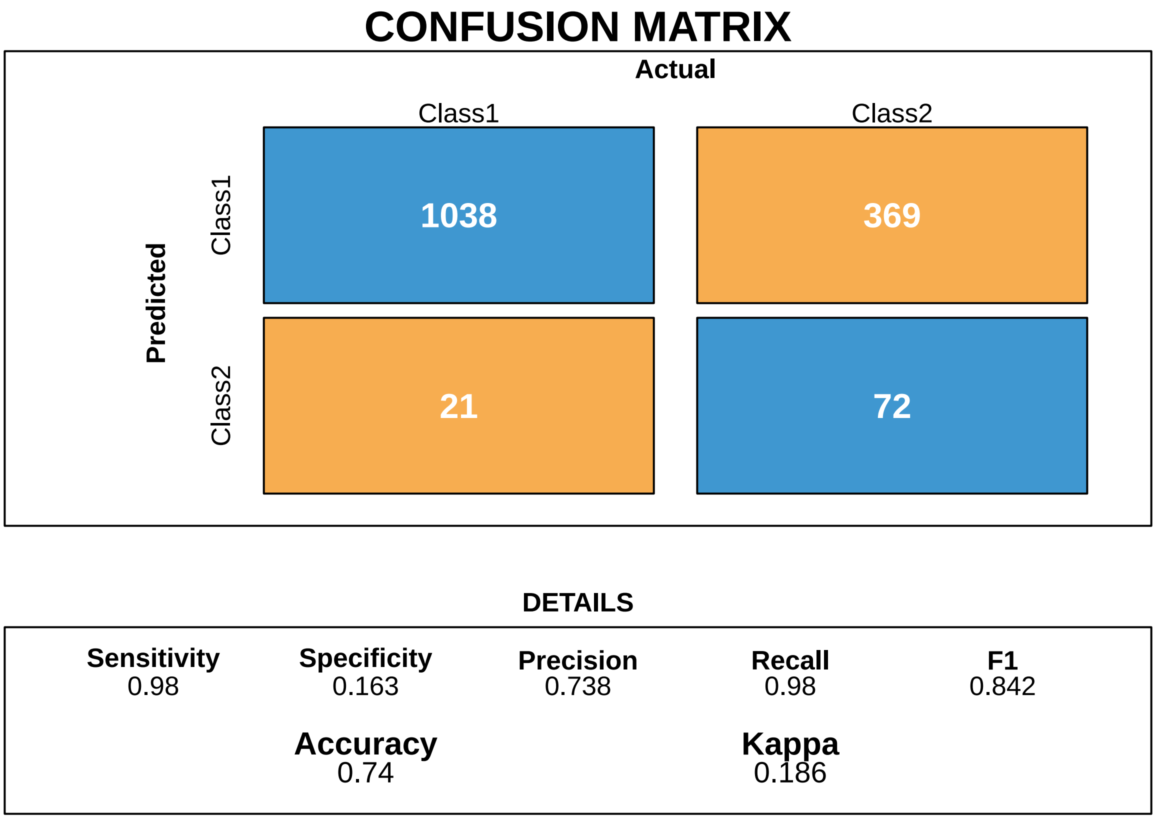
```
# 1. Set the seed
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

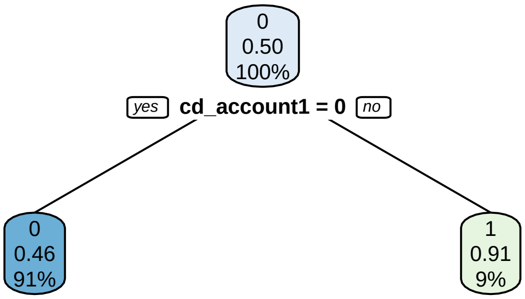
```
# 2. Generate and fit the model with imbalanced data and default parameters
rt_model_upSampling_tunned <- train(credit_card ~ .,
                                    data = up_train,
                                    method = "rpart",
                                    trControl = trainControl(method  = "cv", number  = 5),
                                    metric="Accuracy",
                                    maximize = T,
                                    tuneGrid = data.frame(cp = 0.05),
                                    tuneLength = 30)
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```

```
fitted.results.rt_model_upSampling_tunned <- predict(rt_model_upSampling_tunned,test_data)

# 3. Generate and plot the confusion matrix
rt_model_cm_upSampling_tunned <- confusionMatrix(
  fitted.results.rt_model_upSampling_tunned,
  test_data$credit_card,
  positive = NULL,
  dnn = c("Prediction", "Reference"),
  prevalence = NULL,
  mode = "everything")

draw_confusion_matrix(rt_model_cm_upSampling_tunned)
```



```
# 4. Plot the decision tree
rpart.plot(rt_model_upSampling_tunned$finalModel, fallen.leaves = FALSE)
```



Observations

- Here, as with the logistic regression model, we see that the performance of the imbalanced data set is higher that the performance using balanced data via up-sampling. For discussion see chapter 5.
- Hyper-tuning the tree classifier can bring the performance using balanced data to the same level as the performance using imbalanced data.
- Using balanced data, the decision tree classifier performs better than the logistic regression model.
- Here again, we see that cd_account, securities_account, personal_loans, family) are relevant for the classifier (using balanced data!)
- It seems cd_account is particularly important!

5. Conclusion and recommendations

Conclusions:

- Here, both models performed equally well on imbalanced data.

- Performance was better using imbalanced data compared to up-sampling treated balanced data.
- Using balanced data, the decision tree performed much better than the logistic regression, and it required 4 parameters (cd_account, securities_account, personal_loans, family) for giving a better recall value.
- Interestingly, Family was an important parameter when using balanced data, which was not obvious from the EDA as there was no clear correlation.
- Why is the results better using imbalanced data? In brief, training data needs to reflect the real world, which in this case the imbalanced data do best that the balanced one. Thus, actually the classifiers trained on balanced data may need to be re-trained on proper test data, for which predictions and patterns may change significantly. Furthermore, there may not be a real need in this case for balancing the data as the minority class count may already suffice. Notice that conditional probabilities such as precision and recall should be evaluated and optimized within the natural distribution of your classes. For more detailed discussion see: https://towardsdatascience.com/why-balancing-classes-is-over-hyped-e382a8a410f7

Recommendations:

- The marketing team should aim at targeting individuals with securities_account and personal_loans, large families and with a certificate of deposit (CD) account with the bank.
- It make sense of course that individuals with higher income and bigger family will more likely to borrow more from the bank (to sustain the family) as they have higher chances of paying back.