Bank\_Loan\_Analysis.R

2023-12-05

# Bank\_Loan\_Analysis.R

#### 2023-12-05

# Load necessary packages  
install.packages("httpuv")

install.packages("naniar")

install.packages("forcats")

install.packages("ggpubr")

install.packages("ggthemes")

install.packages("gridExtra")

install.packages("corrplot")

install.packages("ROCR")

install.packages("pROC")

install.packages("ROSE")

install.packages("caTools")

install.packages("glmnet")

install.packages("Metrics")

install.packages("rpart")

install.packages("rpart.plot")

install.packages("kableExtra")

install.packages("naivebayes")

install.packages("e1071")

library(httpuv)

library(e1071)

library(naivebayes)

library(kableExtra)

library(rpart.plot)

library(rpart)  
library(Metrics)

library(glmnet)

library(naniar)

library(caTools)

library(pROC)

library(ROCR)

library(ROSE)

library(corrplot)

library(gridExtra)

library(naniar)  
library(forcats)

library(dplyr)

library(ggplot2)  
library(ggpubr)

library(ggthemes)

library(gridExtra)  
  
# Load the dataset  
BankDF <- read.csv("C:/Bank Loan Data/application\_data.csv")  
  
dim(BankDF)

## [1] 307511 122

# Find and drop columns with too many missing values  
  
miss <- c() # Initialize vector to store columns with too many missing values  
for (i in 1:ncol(BankDF)) {  
 if (sum(is.na(BankDF[, i])) > 140000) {  
 miss <- append(miss, i)   
 }  
}  
  
BankDF<- BankDF[,-miss]  
miss2 <- c() # Initialize vector to store rows with too many missing values  
for (i in 1:nrow(BankDF)) {  
 if (sum(is.na(BankDF[i, ])) > 0.5 \* ncol(BankDF)) {  
 miss2 <- append(miss2, i)   
 }  
}  
  
BankDF<- BankDF[-miss,]  
  
BankDF<- BankDF[,-c(52:71)]  
  
#check how many columns left  
dim(BankDF)

## [1] 307466 57

# Data cleaning: Replace missing values in specific columns  
# Replace missing values in specific columns with a specific value  
BankDF$EMERGENCYSTATE\_MODE[is.na(BankDF$EMERGENCYSTATE\_MODE)] <- ""  
BankDF$EMERGENCYSTATE\_MODE= fct\_explicit\_na(BankDF$EMERGENCYSTATE\_MODE, "Unknown")

## Warning: `fct\_explicit\_na()` was deprecated in forcats 1.0.0.  
## ℹ Please use `fct\_na\_value\_to\_level()` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was generated.

BankDF$OCCUPATION\_TYPE[is.na(BankDF$OCCUPATION\_TYPE)] <- "Unknown"  
BankDF$OCCUPATION\_TYPE= fct\_explicit\_na(BankDF$OCCUPATION\_TYPE, "Unknown")  
  
  
BankDF$FONDKAPREMONT\_MODE[is.na(BankDF$FONDKAPREMONT\_MODE)] <- "Unknown"  
BankDF <- replace\_with\_na(BankDF, replace = list(FONDKAPREMONT\_MODE = "Unknown", HOUSETYPE\_MODE = "Unknown", WALLSMATERIAL\_MODE = "Unknown"))  
  
BankDF$HOUSETYPE\_MODE[is.na(BankDF$HOUSETYPE\_MODE)] <- "Unknown"  
BankDF$HOUSETYPE\_MODE = fct\_explicit\_na(BankDF$HOUSETYPE\_MODE, "Unknown")  
  
BankDF$WALLSMATERIAL\_MODE[is.na(BankDF$WALLSMATERIAL\_MODE)] <- "Unknown"  
BankDF$WALLSMATERIAL\_MODE = fct\_explicit\_na(BankDF$WALLSMATERIAL\_MODE, "Unknown")  
  
# Extract numeric columns for replacement with mean  
numeric\_cols <- sapply(BankDF, is.numeric)  
  
# Replace missing values with mean in numeric columns  
for(i in 1:ncol(BankDF)) {   
 if(numeric\_cols[i]) {  
 BankDF[, i][is.na(BankDF[, i])] <- mean(BankDF[, i], na.rm = TRUE)  
 }  
}  
  
# Transform specific columns  
BankDF$DAYS\_BIRTH<-BankDF$DAYS\_BIRTH\*-1  
BankDF$DAYS\_EMPLOYED<-BankDF$DAYS\_EMPLOYED\*-1  
BankDF$DAYS\_REGISTRATION<-BankDF$DAYS\_REGISTRATION\*-1  
BankDF$DAYS\_ID\_PUBLISH<-BankDF$DAYS\_ID\_PUBLISH\*-1  
BankDF$DAYS\_LAST\_PHONE\_CHANGE<-BankDF$DAYS\_LAST\_PHONE\_CHANGE\*-1  
bins = c(0,350000,700000,1000000000)  
slots = c('Low','Medium','High')  
  
#Creating bins for Credit amount  
BankDF['AMT\_CREDIT\_RANGE']=cut(BankDF$AMT\_CREDIT,breaks=c(0,350000,700000,1000000000),labels=slots)  
bins = c(0,200000,400000,10000000000)  
slots = c('Low','Medium','High')  
  
# Creating bins for income amount  
BankDF['AMT\_INCOME\_RANGE']=cut(BankDF$AMT\_INCOME\_TOTAL,breaks=c(0,200000,400000,10000000000),labels=slots)  
bins = c(0,7300,10950,14600,18250,21900,25500)  
slots <- c('0-20','20-30','30-40','40-50','50-60','60-70')  
BankDF$AGE\_RANGE <- cut(BankDF$DAYS\_BIRTH, breaks = c(0,7300,10950,14600,18250,21900,25500), labels = slots)  
BankDF <- distinct(BankDF)  
  
# Finding and counting the duplicated rows in the dataset  
sum(duplicated(BankDF))

## [1] 0

# Counting the total number of missing values in the dataset  
sum(is.na(BankDF))

## [1] 0

# Displaying the dimensions of the dataset (rows and columns)  
dim(BankDF)

## [1] 307466 60

# Creating a table of proportions for the 'TARGET' variable  
prop.table(table(BankDF$TARGET))

##   
## 0 1   
## 0.91926262 0.08073738

# Removing unnecessary columns and storing the modified dataset in 'df'  
df= select(BankDF,-1,-31,)  
  
  
#Checking for missing values

# Total number of missing values in the dataset  
 cat("The total number of missing values in the dataset is" , sum(is.na(BankDF)))

## The total number of missing values in the dataset is 0

colSums(is.na(BankDF))

## SK\_ID\_CURR TARGET NAME\_CONTRACT\_TYPE   
## 0 0 0   
## CODE\_GENDER FLAG\_OWN\_CAR FLAG\_OWN\_REALTY   
## 0 0 0   
## CNT\_CHILDREN AMT\_INCOME\_TOTAL AMT\_CREDIT   
## 0 0 0   
## AMT\_ANNUITY AMT\_GOODS\_PRICE NAME\_TYPE\_SUITE   
## 0 0 0   
## NAME\_INCOME\_TYPE NAME\_EDUCATION\_TYPE NAME\_FAMILY\_STATUS   
## 0 0 0   
## NAME\_HOUSING\_TYPE REGION\_POPULATION\_RELATIVE DAYS\_BIRTH   
## 0 0 0   
## DAYS\_EMPLOYED DAYS\_REGISTRATION DAYS\_ID\_PUBLISH   
## 0 0 0   
## FLAG\_MOBIL FLAG\_EMP\_PHONE FLAG\_WORK\_PHONE   
## 0 0 0   
## FLAG\_CONT\_MOBILE FLAG\_PHONE FLAG\_EMAIL   
## 0 0 0   
## OCCUPATION\_TYPE CNT\_FAM\_MEMBERS REGION\_RATING\_CLIENT   
## 0 0 0   
## REGION\_RATING\_CLIENT\_W\_CITY WEEKDAY\_APPR\_PROCESS\_START HOUR\_APPR\_PROCESS\_START   
## 0 0 0   
## REG\_REGION\_NOT\_LIVE\_REGION REG\_REGION\_NOT\_WORK\_REGION LIVE\_REGION\_NOT\_WORK\_REGION   
## 0 0 0   
## REG\_CITY\_NOT\_LIVE\_CITY REG\_CITY\_NOT\_WORK\_CITY LIVE\_CITY\_NOT\_WORK\_CITY   
## 0 0 0   
## ORGANIZATION\_TYPE EXT\_SOURCE\_2 EXT\_SOURCE\_3   
## 0 0 0   
## FONDKAPREMONT\_MODE HOUSETYPE\_MODE WALLSMATERIAL\_MODE   
## 0 0 0   
## EMERGENCYSTATE\_MODE OBS\_30\_CNT\_SOCIAL\_CIRCLE DEF\_30\_CNT\_SOCIAL\_CIRCLE   
## 0 0 0   
## OBS\_60\_CNT\_SOCIAL\_CIRCLE DEF\_60\_CNT\_SOCIAL\_CIRCLE DAYS\_LAST\_PHONE\_CHANGE   
## 0 0 0   
## AMT\_REQ\_CREDIT\_BUREAU\_HOUR AMT\_REQ\_CREDIT\_BUREAU\_DAY AMT\_REQ\_CREDIT\_BUREAU\_WEEK   
## 0 0 0   
## AMT\_REQ\_CREDIT\_BUREAU\_MON AMT\_REQ\_CREDIT\_BUREAU\_QRT AMT\_REQ\_CREDIT\_BUREAU\_YEAR   
## 0 0 0   
## AMT\_CREDIT\_RANGE AMT\_INCOME\_RANGE AGE\_RANGE   
## 0 0 0

View(BankDF)  
  
#Outlier Analysis

# Extract the AGE variable  
 BankDF1<- BankDF  
  
  
# Assuming 'BankDF1'   
col\_list <- c('AMT\_INCOME\_TOTAL', 'AMT\_CREDIT', 'AMT\_ANNUITY', 'AMT\_GOODS\_PRICE')  
  
# Loop through the specified columns and divide each value by 100000  
for (col in col\_list) {  
 BankDF1[[col]] <- BankDF1[[col]] / 100000  
}  
  
  
  
# Extract the AGE variable  
age <- BankDF1$AGE  
  
# Plotting boxplot  
boxplot(age, main="Client's age", ylab="", col="lightblue", border="blue", horizontal=TRUE)

![](data:image/png;base64;base64,)

# Extract the Income Amount variable  
income <- BankDF$AMT\_INCOME\_TOTAL  
  
# Function for outlier plot  
  
  
income <- BankDF1$AMT\_INCOME\_TOTAL  
  
# Function for outlier plot  
outlier\_plot <- function(data, title, label) {  
 # Plotting boxplot  
 boxplot(data, main=title, ylab=label, col="lightblue", border="blue", horizontal=FALSE)  
}  
  
# Calling the function  
outlier\_plot(income, "Client's income", "Income in Lakhs")

![](data:image/png;base64;base64,)

credit <- BankDF1$AMT\_CREDIT  
  
# Function for outlier plot  
outlier\_plot <- function(data, title, label) {  
 # Plotting boxplot  
 boxplot(data, main=title, ylab=label, col="lightblue", border="blue", horizontal=FALSE)  
}  
  
# Calling the function  
outlier\_plot(credit, "Credit amount of the loan", "Amount in Lakhs")

![](data:image/png;base64;base64,)

#Univariate Analysis

# Creating plots for analyzing individual variables in the dataset  
   
 # Creating a bar plot to visualize the distribution of genders in the 'BankDF' dataset  
 genderplot<-ggplot(data=BankDF,aes(x=CODE\_GENDER,fill=CODE\_GENDER)) +  
 geom\_bar(stat="count") + ggtitle('Gender')+  
 theme(plot.title = element\_text (hjust = 0.5)) + theme(legend.position="none") + xlab('') + ylab ('')  
  
# Creating a bar plot to explore the distribution of credit amount ranges in 'BankDF'  
AMT\_CREDITRANGEplot<-ggplot(data=BankDF,aes(x=factor(AMT\_CREDIT\_RANGE, level=c('Low','Medium','High')),fill=AMT\_CREDIT\_RANGE))+  
 geom\_bar(stat="count") + ggtitle('Credit Amount (Range)') +   
 theme(plot.title = element\_text (hjust = 0.5)) + theme(legend.position="none") + xlab('') + ylab ('')  
  
# Creating a bar plot to analyze the distribution of income amount ranges in 'BankDF'  
AMT\_INCOME\_RANGEplot<-ggplot(data=BankDF,aes(x=factor(AMT\_INCOME\_RANGE, level=c('Low','Medium','High')),fill=AMT\_INCOME\_RANGE))+  
 geom\_bar(stat="count") + ggtitle('Income Amount (Range)') +   
 theme(plot.title = element\_text (hjust = 0.5)) + theme(legend.position="none") + xlab('') + ylab ('')  
  
# Creating a bar plot to visualize the distribution of age ranges in 'BankDF'  
AGE\_RANGEplot<-ggplot(data=BankDF,aes(x=AGE\_RANGE,fill=AGE\_RANGE))+   
 #scale\_fill\_manual(values = c("#F8766D","#A3A500","#00BF7D","#00B0F6","#E76BF3"))+  
 geom\_bar(stat="count") + ggtitle('Age Range') +  
 theme(plot.title = element\_text (hjust = 0.5)) + theme(legend.position="none") + xlab('') + ylab ('')  
  
# Arranging individual plots in a grid layout for comparative analysis  
p1 <- ggarrange(genderplot, AMT\_INCOME\_RANGEplot,  
 ncol = 1, nrow = 2)  
p2 <- ggarrange(AGE\_RANGEplot, AMT\_CREDITRANGEplot,   
 ncol = 1, nrow = 2)  
ggarrange(p1, p2, ncol = 2, nrow = 1)

![](data:image/png;base64;base64,)

#INCOME TYPE,EDUCATION TYPE, FAMILY STATUS, HOUSING TYPE, OCCUPATION TYPE

# Analysis of Categorical Variables  
 # Visualizing different categorical variables within the dataset  
   
 # Creating a bar plot to analyze the distribution of income types in the 'BankDF' dataset  
 incometypeplot <- ggplot(data = BankDF, aes(x = NAME\_INCOME\_TYPE, fill = NAME\_INCOME\_TYPE)) +  
 geom\_bar(stat = "count") + ggtitle('Types of Income') +  
 theme(plot.title = element\_text(hjust = 0.5)) +  
 theme(legend.position = "none") + xlab('') + ylab('')  
  
# Creating a bar plot to examine the distribution of education types in 'BankDF'  
edutypeplot <- ggplot(data = BankDF, aes(x = NAME\_EDUCATION\_TYPE, fill = NAME\_EDUCATION\_TYPE)) +  
 geom\_bar(stat = "count") + ggtitle('Types of Education') +  
 theme(plot.title = element\_text(hjust = 0.5)) +  
 theme(legend.position = "none") + xlab('') + ylab('')  
  
# Creating a bar plot to explore the distribution of marital statuses in the dataset  
marriagetypeplot <- ggplot(data = BankDF, aes(x = NAME\_FAMILY\_STATUS, fill = NAME\_FAMILY\_STATUS)) +  
 geom\_bar(stat = "count") + ggtitle('Marital Status') +  
 theme(plot.title = element\_text(hjust = 0.5)) +  
 theme(legend.position = "none") + xlab('') + ylab('')  
  
# Creating a bar plot to visualize the distribution of housing types  
housetypeplot <- ggplot(data = BankDF, aes(x = NAME\_HOUSING\_TYPE, fill = NAME\_HOUSING\_TYPE)) +  
 geom\_bar(stat = "count") + ggtitle('Types of Housing') +  
 theme(plot.title = element\_text(hjust = 0.5)) +  
 theme(legend.position = "none") + xlab('') + ylab('')  
  
# Creating a bar plot to analyze the distribution of occupation types  
worktypeplot <- ggplot(data = BankDF, aes(x = OCCUPATION\_TYPE, fill = OCCUPATION\_TYPE)) +  
 geom\_bar(stat = "count") + ggtitle('Types of Occupation') +  
 theme(plot.title = element\_text(hjust = 0.5)) +  
 theme(legend.position = "none") + xlab('') + ylab('')  
  
# Arranging the plots for income, education, marital status, housing, and occupation types  
p3 <- ggarrange(incometypeplot, edutypeplot,  
 ncol = 1, nrow = 2)  
p4 <- ggarrange(marriagetypeplot, housetypeplot,  
 ncol = 1, nrow = 2)  
ggarrange(p3, p4, ncol = 2, nrow = 1)

![](data:image/png;base64;base64,)

# Additional visualization: Plotting the occupation type with vertical categories for better display  
worktypeplot + coord\_flip()

![](data:image/png;base64;base64,)

# Bivariate analysis

# Exploratory Data Analysis - Visualizing Relationships between Different Variables and the Target  
   
 # AGE and Gender vs. TARGET  
 # Plotting the distribution of 'AGE\_RANGE' by 'TARGET' and 'CODE\_GENDER' by 'TARGET' in two separate bar plots  
 par(mfrow = c(2, 1))  
  
# Visualization of 'AGE\_RANGE' by 'TARGET'  
BankDF1 <- with(BankDF, table(TARGET, AGE\_RANGE))  
barplot(BankDF1, beside = TRUE, legend = TRUE, args.legend = list(x = "topright", inset = c(-0.05, 0)),  
 col = c("Red", "Green"), xlab = "AGE\_RANGE", ylab = "Values",  
 main = "Target=0: No Payment issue, Target=1: Payment issue")  
  
# Visualization of 'CODE\_GENDER' by 'TARGET'  
BankDF2 <- with(BankDF, table(TARGET, CODE\_GENDER))  
barplot(BankDF2, beside = TRUE, legend = TRUE, col = c("Red", "Green"), args.legend = list(x = "topright", inset = c(-0.05, 0)),  
 xlab = "CODE\_GENDER", ylab = "Values",  
 main = "Target=0: No Payment issue, Target=1: Payment issue")

![](data:image/png;base64;base64,)

# Amount of Credit (loan), Income type, and Amount of Income vs. TARGET  
par(mfrow = c(3, 1))  
  
# Visualizing 'AMT\_CREDIT\_RANGE' by 'TARGET'  
BankDF3 <- with(BankDF, table(TARGET, AMT\_CREDIT\_RANGE))  
barplot(BankDF3, beside = TRUE, legend = TRUE, args.legend = list(x = "topright", inset = c(0, 0)),  
 col = c("Red", "Green"), xlab = "Amount of Credit(Loan)", ylab = "Values",  
 main = "Target=0: No Payment issue, Target=1: Payment issue")  
  
# Visualizing 'NAME\_INCOME\_TYPE' by 'TARGET'  
BankDF4 <- with(BankDF, table(TARGET, NAME\_INCOME\_TYPE))  
barplot(BankDF4, beside = TRUE, legend = TRUE, args.legend = list(x = "topright", inset = c(0, 0)),  
 col = c("Red", "Green"), xlab = "NAME\_INCOME\_TYPE", ylab = "Values",  
 main = "Target=0: No Payment issue, Target=1: Payment issue")  
  
# Visualizing 'AMT\_INCOME\_RANGE' by 'TARGET'  
BankDF5 <- with(BankDF, table(TARGET, AMT\_INCOME\_RANGE))  
barplot(BankDF5, beside = TRUE, legend = TRUE, args.legend = list(x = "topright", inset = c(0, 0)),  
 col = c("Red", "Green"), xlab = "AMT\_INCOME\_RANGE", ylab = "Values",  
 main = "Target=0: No Payment issue, Target=1: Payment issue")

![](data:image/png;base64;base64,)

# Types of Education, Marital Status, and Types of Housing vs. TARGET  
par(mfrow = c(3, 1))  
  
# Visualizing 'NAME\_EDUCATION\_TYPE' by 'TARGET'  
BankDF6 <- with(BankDF, table(TARGET, NAME\_EDUCATION\_TYPE))  
barplot(BankDF6, beside = TRUE, legend = TRUE, args.legend = list(x = "topright", inset = c(0, 0)),  
 col = c("Red", "Green"), xlab = "Types of Education", ylab = "Values",  
 main = "Target=0: No Payment issue, Target=1: Payment issue")  
  
# Visualizing 'NAME\_FAMILY\_STATUS' by 'TARGET'  
BankDF7 <- with(BankDF, table(TARGET, NAME\_FAMILY\_STATUS))  
barplot(BankDF7, beside = TRUE, legend = TRUE, args.legend = list(x = "topright", inset = c(0, 0)),  
 col = c("Red", "Green"), xlab = "Marital Status", ylab = "Values",  
 main = "Target=0: No Payment issue, Target=1: Payment issue")  
  
# Visualizing 'NAME\_HOUSING\_TYPE' by 'TARGET'  
BankDF8 <- with(BankDF, table(TARGET, NAME\_HOUSING\_TYPE))  
barplot(BankDF8, beside = TRUE, legend = TRUE, args.legend = list(x = "topright", inset = c(0, 0)),  
 col = c("Red", "Green"), xlab = "Types of Housing", ylab = "Values",  
 main = "Target=0: No Payment issue, Target=1: Payment issue")

![](data:image/png;base64;base64,)

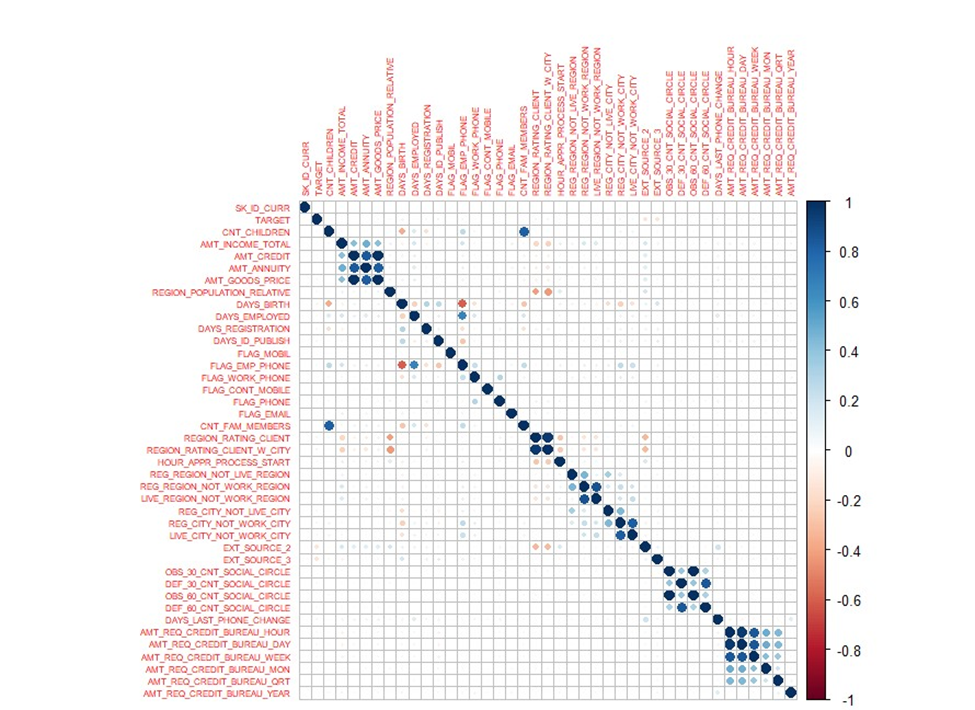
# AMT\_CREDIT, AMT\_ANNUITY, AMT\_GOODS\_PRICE vs. TARGET  
# Creating density plots for 'AMT\_CREDIT', 'AMT\_ANNUITY', 'AMT\_GOODS\_PRICE' based on the 'TARGET'  
p1 <- ggplot(data = BankDF, aes(x = AMT\_CREDIT, group = TARGET, fill = TARGET)) +  
 geom\_density(adjust = 1.5)  
  
p2 <- ggplot(data = BankDF, aes(x = AMT\_ANNUITY, group = TARGET, fill = TARGET)) +  
 geom\_density(adjust = 1.5)  
  
p3 <- ggplot(data = BankDF, aes(x = AMT\_GOODS\_PRICE, group = TARGET, fill = TARGET)) +  
 geom\_density(adjust = 1.5)  
  
# Arrange and display the density plots in a single column  
grid.arrange(p1, p2, p3, ncol = 1)

![](data:image/png;base64;base64,)

# Exploring Relationships and Data Insights  
  
# Correlation Matrix  
# Creating a correlation matrix of numeric variables in BankDF using Spearman method and visualizing the correlation matrix using corrplot.  
plot.new() # Creates a new plotting device  
dev.off() # Turns off the active plotting device

## null device   
## 1

data = select\_if(BankDF, is.numeric) # Selecting only numeric columns  
data.cor = cor(data, method = c("spearman")) # Calculating the Spearman correlation matrix  
corrplot(data.cor, tl.cex = 0.5) # Visualizing the correlation matrix with adjustable text size  
  
# Scatter Plots and Bar Plots  
  
# AMT\_INCOME\_TOTAL vs AMT\_CREDIT, AMT\_GOODS\_PRICE vs AMT\_CREDIT  
par(mfrow = c(1, 2)) # Divides the plotting area into a 1x2 grid  
  
# Scatter plot: Amount of Income vs Amount of Credit (Range)  
plot(data$AMT\_INCOME\_TOTAL, data$AMT\_CREDIT,   
 main = "Amount of Income vs Amount of Credit (Range)",  
 xlab = "AMT\_INCOME\_TOTAL", ylab = "AMT\_CREDIT", pch = 19) # Creates a



scatter plot  
  
# Scatter plot: Good Price vs Amount of Credit (Range)  
plot(data$AMT\_GOODS\_PRICE, data$AMT\_CREDIT,   
 main = "Good Price vs Amount of Credit (Range)",  
 xlab = "AMT\_GOODS\_PRICE", ylab = "AMT\_CREDIT", pch = 19) # Creates a scatter plot

A graph of a graph of a credit

Description automatically generated with medium confidence  
  
# Bar Plot: AMT\_INCOME\_RANGE vs CODE\_GENDER  
BankDF9 <- with(BankDF, table(AMT\_INCOME\_RANGE, CODE\_GENDER))  
barplot(BankDF9, beside = TRUE, legend = TRUE, args.legend = list(x = "topright", inset = c(0, 0)),  
 col = c("Red", "Green", "Blue"), xlab = "CODE\_GENDER", ylab = "values",  
 main = "The relationship between income range and gender")  
  
# Box Plot: AMT\_CREDIT vs NAME\_EDUCATION\_TYPE  
ggplot(BankDF, aes(x = NAME\_EDUCATION\_TYPE, y = AMT\_CREDIT, fill = NAME\_FAMILY\_STATUS)) +   
 geom\_boxplot() # Creates a box plot to visualize the relationship between education type and credit amount

# Data Manipulation  
  
# Dropping insignificant variables (REGION\_RATING\_CLIENT\_W\_CITY, SK\_ID\_CURR) and assigning the modified data to 'df'  
df = select(BankDF, -1, -31) # Drops columns based on their positions (-1, -31)

A screenshot of a graph

Description automatically generated

A graph of colorful bars

Description automatically generated with medium confidence

#Logistic regression

# Setting a seed for reproducibility in random processes  
 set.seed(123)  
  
# Splitting the dataset into training and testing sets  
split = sample.split(df$TARGET,SplitRatio = 0.70)  
training\_set = subset(df, split == TRUE)  
test\_set = subset(df, split == FALSE)  
  
# Standardizing continuous columns in the training and testing sets  
# List of continuous columns to standardize  
continuous\_column = c('AMT\_INCOME\_TOTAL', 'AMT\_CREDIT', 'AMT\_ANNUITY', 'AMT\_GOODS\_PRICE', 'REGION\_POPULATION\_RELATIVE', 'DAYS\_REGISTRATION', 'CNT\_FAM\_MEMBERS', 'EXT\_SOURCE\_2', 'EXT\_SOURCE\_3', 'OBS\_30\_CNT\_SOCIAL\_CIRCLE', 'DEF\_30\_CNT\_SOCIAL\_CIRCLE', 'OBS\_60\_CNT\_SOCIAL\_CIRCLE', 'DEF\_60\_CNT\_SOCIAL\_CIRCLE', 'DAYS\_LAST\_PHONE\_CHANGE','AMT\_REQ\_CREDIT\_BUREAU\_HOUR', 'AMT\_REQ\_CREDIT\_BUREAU\_DAY', 'AMT\_REQ\_CREDIT\_BUREAU\_WEEK', 'AMT\_REQ\_CREDIT\_BUREAU\_MON', 'AMT\_REQ\_CREDIT\_BUREAU\_QRT', 'AMT\_REQ\_CREDIT\_BUREAU\_YEAR')  
training\_set[continuous\_column] = scale(training\_set[continuous\_column])  
test\_set[continuous\_column] = scale(test\_set[continuous\_column])  
table(training\_set$TARGET)

##   
## 0 1   
## 197849 17377

# Balancing the training set using oversampling method to handle class imbalance  
bal\_training\_set=ovun.sample(TARGET~., data = training\_set, method = "over", N = 197849\*2)$data  
table(bal\_training\_set$TARGET)

##   
## 0 1   
## 197849 197849

# Defining a function to generate a confusion matrix  
get\_cm <- function(test\_set, y\_pred){  
 cm <- as.matrix(table(Actual = test\_set$TARGET, Predicted = y\_pred))  
}  
  
# Defining a function to evaluate the performance of the model  
simple\_eval <- function (test\_set, y\_pred) {  
 CM = get\_cm(test\_set, y\_pred)  
 n = sum(CM) # number of instances  
 nc = nrow(CM) # number of classes, should be 2 in our case  
 rowsums = apply(CM, 1, sum) # number of instances per class  
 colsums = apply(CM, 2, sum) # number of predictions per class  
 p = rowsums / n # distribution of instances over the actual classes  
 q = colsums / n # distribution of instances over the predicted classes  
 diag = diag(CM) # get TP and TN using identity matrix  
   
 accuracy = sum(diag) / n # overall classification accuracy  
 precision = diag / colsums # fraction of correct predictions for a certain class  
 recall = diag / rowsums # fraction of instances of a class that were correctly predicted  
 f1 = 2 \* precision \* recall / (precision + recall) # harmonic mean (or a weighted average) of precision and recall  
   
 # Ensuring all vectors have the same length  
 accuracy = rep(accuracy, length.out = length(precision))  
 f1 = rep(f1, length.out = length(precision))  
   
 perf\_df = data.frame(accuracy, precision, recall, f1)   
 AUC = auc(test\_set$TARGET, factor(y\_pred, ordered = TRUE)) # Area under ROC curve  
 expAccuracy = sum(p \* q)  
 test = (accuracy - expAccuracy) / (1 - expAccuracy)  
 macroPrecision = mean(precision)  
 macroRecall = mean(recall)  
 macroF1 = mean(f1)  
   
 list(Accuracy = accuracy, ROC\_area = AUC, test = test, Precision = macroPrecision,   
 Recall = macroRecall, Fscore = macroF1, performance\_df = perf\_df, confusion\_matrix = CM)  
}  
  
  
  
# Building a logistic regression model with balanced training set  
model\_glm <- glm(TARGET ~ ., data = bal\_training\_set, family = "binomial")  
  
# Predict test data based on model  
LR.y\_pred <- predict(model\_glm, newdata = test\_set, type = "response")

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == : prediction from  
## rank-deficient fit; attr(\*, "non-estim") has doubtful cases

LR.y\_pred <- ifelse(LR.y\_pred >0.5, 1, 0)  
  
# Confusion matrix  
LR.eval <- simple\_eval(test\_set, LR.y\_pred)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

LR.eval$confusion\_matrix

## Predicted  
## Actual 0 1  
## 0 58338 26455  
## 1 2434 5013

# Performance by output class  
print(round(LR.eval$performance\_df,2))

## accuracy precision recall f1  
## 0 0.69 0.96 0.69 0.80  
## 1 0.69 0.16 0.67 0.26

# Displaying confusion matrix and various performance metrics  
cat('Accuracy: ', round(LR.eval$Accuracy,2),  
 '\nMacro-precision: ', round(LR.eval$Precision,2),  
 '\nMacro-recall: ', round(LR.eval$Recall,2),  
 '\nMacro-F1: ', round(LR.eval$Fscore,2),  
 '\ntest: ', round(LR.eval$test,2),  
 '\nROC area: ', round(LR.eval$ROC\_area,2)  
)

## Accuracy: 0.69 0.69   
## Macro-precision: 0.56   
## Macro-recall: 0.68   
## Macro-F1: 0.53   
## test: 0.15 0.15   
## ROC area: 0.68

#Decision Tree (DT) Classification

# fit the decision tree classification  
 model = rpart(TARGET ~., data = bal\_training\_set, method = "class")  
# plot decision tree  
rpart.plot(model, extra = 106)

![](data:image/png;base64;base64,)

#Evaluate Decision Tree (DT) Model  
# make prediction  
DT.y\_pred = predict (model, test\_set, type = 'class')  
  
# Confusion Matrix  
DT.eval <- simple\_eval(test\_set, DT.y\_pred)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

DT.eval$confusion\_matrix

## Predicted  
## Actual 0 1  
## 0 67441 17352  
## 1 4225 3222

# Performance by output class  
print(round(DT.eval$performance\_df,2))

## accuracy precision recall f1  
## 0 0.77 0.94 0.80 0.86  
## 1 0.77 0.16 0.43 0.23

# Model Evaluation  
cat('Accuracy: ', round(DT.eval$Accuracy,2),  
 '\nMacro-precision: ', round(DT.eval$Precision,2),  
 '\nMacro-recall: ', round(DT.eval$Recall,2),  
 '\nMacro-F1: ', round(DT.eval$Fscore,2),  
 '\nROC area: ', round(DT.eval$ROC\_area,2)  
)

## Accuracy: 0.77 0.77   
## Macro-precision: 0.55   
## Macro-recall: 0.61   
## Macro-F1: 0.55   
## ROC area: 0.61

#Naive Bayes (NB)

# Fitting Naive Bayes Model to training dataset  
 model\_NB <- naiveBayes(TARGET ~ ., data = bal\_training\_set)  
  
# Predicting on test data'  
NB.y\_pred <- predict(model\_NB, newdata = test\_set)  
  
# Confusion Matrix  
NB.eval <- simple\_eval(test\_set, NB.y\_pred)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

NB.eval$confusion\_matrix

## Predicted  
## Actual 0 1  
## 0 54056 30737  
## 1 3113 4334

# Performance by output class  
print(round(NB.eval$performance\_df,2))

## accuracy precision recall f1  
## 0 0.63 0.95 0.64 0.76  
## 1 0.63 0.12 0.58 0.20

# Model Evaluation  
cat('Accuracy: ', round(NB.eval$Accuracy,2),  
 '\nMacro-precision: ', round(NB.eval$Precision,2),  
 '\nMacro-recall: ', round(NB.eval$Recall,2),  
 '\nMacro-F1: ', round(NB.eval$Fscore,2),  
 '\nROC area: ', round(NB.eval$ROC\_area,2)  
)

## Accuracy: 0.63 0.63   
## Macro-precision: 0.53   
## Macro-recall: 0.61   
## Macro-F1: 0.48   
## ROC area: 0.61

#Comparison between DT, NB and LR models

compare <- data.frame(Method=c('Decision Tree', 'Naive Bayes', 'Logistic Regression'), Accuracy = NA, Precision = NA, Recall = NA, FScore = NA, 'ROC' = NA)  
compare$Accuracy <- c(round(DT.eval$Accuracy[1],2),round(NB.eval$Accuracy[1],2),round(LR.eval$Accuracy[1],2))  
compare$Precision <- c(round(DT.eval$Precision,2),round(NB.eval$Precision,2),round(LR.eval$Precision,2))  
compare$Recall <- c(round(DT.eval$Recall,2),round(NB.eval$Recall,2),round(LR.eval$Recall,2))  
compare$FScore <- c(round(DT.eval$Fscore,2),round(NB.eval$Fscore,2),round(LR.eval$Fscore,2))  
compare$ROC <- c(round(DT.eval$ROC\_area,2),round(NB.eval$ROC\_area,2),round(LR.eval$ROC\_area,2))  
kable\_styling(kable(compare),c("striped","bordered"), full\_width = F)

| Method | Accuracy | Precision | Recall | FScore | ROC |
| --- | --- | --- | --- | --- | --- |
| Decision Tree | 0.77 | 0.55 | 0.61 | 0.55 | 0.61 |
| Naive Bayes | 0.63 | 0.53 | 0.61 | 0.48 | 0.61 |
| Logistic Regression | 0.69 | 0.56 | 0.68 | 0.53 | 0.68 |

#Performance Evaluation:

# select rows where target = 0 (able to replay the loan)  
df1 = subset(df, TARGET == 0)  
  
# Drop unnecessary or duplicated columns  
df1 = select (df1, -c('TARGET', 'AMT\_CREDIT\_RANGE'))  
  
set.seed(123)  
split = sample.split(df1$AMT\_CREDIT,SplitRatio = 0.70)  
training\_set = subset(df1, split == TRUE)  
test\_set = subset(df1, split == FALSE)  
  
continuous\_column = c('AMT\_INCOME\_TOTAL', 'AMT\_ANNUITY', 'AMT\_GOODS\_PRICE', 'REGION\_POPULATION\_RELATIVE', 'DAYS\_REGISTRATION', 'CNT\_FAM\_MEMBERS', 'EXT\_SOURCE\_2', 'EXT\_SOURCE\_3', 'OBS\_30\_CNT\_SOCIAL\_CIRCLE', 'DEF\_30\_CNT\_SOCIAL\_CIRCLE', 'OBS\_60\_CNT\_SOCIAL\_CIRCLE', 'DEF\_60\_CNT\_SOCIAL\_CIRCLE', 'DAYS\_LAST\_PHONE\_CHANGE','AMT\_REQ\_CREDIT\_BUREAU\_HOUR', 'AMT\_REQ\_CREDIT\_BUREAU\_DAY', 'AMT\_REQ\_CREDIT\_BUREAU\_WEEK', 'AMT\_REQ\_CREDIT\_BUREAU\_MON', 'AMT\_REQ\_CREDIT\_BUREAU\_QRT', 'AMT\_REQ\_CREDIT\_BUREAU\_YEAR')  
training\_set[continuous\_column] = scale(training\_set[continuous\_column])  
test\_set[continuous\_column] = scale(test\_set[continuous\_column])  
  
x\_train = data.matrix(select (training\_set, -c('AMT\_CREDIT'))) #select (training\_set, -c('AMT\_CREDIT'))  
y\_train = training\_set$AMT\_CREDIT  
  
x\_test = data.matrix(select (test\_set, -c('AMT\_CREDIT'))) #select (test\_set, -c('AMT\_CREDIT'))  
y\_test = test\_set$AMT\_CREDIT  
  
regr\_evaluation <- function(testDataset, predictionResult){  
 MSE <- mse(testDataset$AMT\_CREDIT, predictionResult)  
 RMSE <- rmse(testDataset$AMT\_CREDIT, predictionResult)  
 MAE <- mae(testDataset$AMT\_CREDIT, predictionResult)  
   
 error <- testDataset$AMT\_CREDIT - predictionResult  
 R2 <- 1-sum(error^2)/sum((testDataset$AMT\_CREDIT- mean(testDataset$AMT\_CREDIT))^2)  
 AD\_R2 <- 1-(MSE/var(testDataset$AMT\_CREDIT))  
 list(MSE = MSE, RMSE = RMSE, MAE = MAE, R\_square = R2, Adjusted\_R\_Square =AD\_R2)  
}  
  
  
show\_graph <- function(testDataset, predictionResult){  
 plot(x=predictionResult, y= testDataset$AMT\_CREDIT,  
 xlab='Predicted Values',  
 ylab='Actual Values',  
 main='Predicted vs. Actual Values')  
 abline(a=0, b=1)  
}

A graph of a graph with different colored lines

Description automatically generated with medium confidence  
  
  
# Visual Inspection and Data Details  
View(BankDF)   
dim(BankDF)

## [1] 307466 60