Customer Churn Prediction

Customer churn prediction

Firstly, we need to dowload packages

```r
# to make data manipulation we use tidyverse
library(tidyverse)
# to make machine learning we use caret
library(caret)
# To measure performence, ROC Curve, and AUC are going to be used with
library(ROCR)
library(pROC)
# over-under sampling and checking the imbalance problem
library(ROSE)
library(DMwR2)
library(rpart)
```

Problem:

- The Bank Customer Churn dataset can be used to understand the factors that may affect a bank's customers' exit (also known as churn) from their banking services. This dataset can help the bank understand the reasons for customer churn and develop strategies to reduce it.
- Also, to reach data set click here.

```r
bankcustomer <- read.csv("~/bank_customer.csv")

str(bankcustomer)
```

```
'data.frame':	10000 obs. of  12 variables:
 $ customer_id     : int  15634602 15647311 15619304 15701354 15737888 15574012 15592531 156
 $ credit_score    : int  619 608 502 699 850 645 822 376 501 684 ...
 $ country         : chr  "France" "Spain" "France" "France" ...
 $ gender          : chr  "Female" "Female" "Female" "Female" ...
 $ age             : int  42 41 42 39 43 44 50 29 44 27 ...
 $ tenure          : int  2 1 8 1 2 8 7 4 4 2 ...
 $ balance         : num  0 83808 159661 0 125511 ...
 $ products_number : int  1 1 3 2 1 2 2 4 2 1 ...
 $ credit_card     : int  1 0 1 0 1 1 1 1 0 1 ...
 $ active_member   : int  1 1 0 0 1 0 1 0 1 1 ...
 $ estimated_salary: num  101349 112543 113932 93827 79084 ...
 $ churn           : int  1 0 1 0 0 1 0 1 0 0 ...
```

Features

- CustomerId: Customer identification number
- CreditScore: Customer's credit score
- Country: Customer's geographic location (France, Spain, or Germany)
- Gender: Customer's gender (Male or Female)
- Age: Customer's age
- Tenure: Number of years the customer has been with the bank
- Balance: Customer's account balance
- NumOfProducts: Number of bank products used by the customer
- HasCrCard: Does the customer have a credit card? (0 = No, 1 = Yes)
- IsActiveMember: Is the customer an active member of the bank? (0 = No, 1 = Yes)
- EstimatedSalary: Customer's estimated salary
- Churn: Did the customer leave the bank? (0 = No, 1 = Yes)

About Data set

- The data set contains 12 columns (variables), and 10000 rows (observations)
- The dataset contains a mix of categorical and numerical variables.
- The categorical variables are "Country" and "Gender".
- The numerical variables are "CreditScore", "Age", "Tenure", "Balance", "NumberProducts", "HasCard", "ActiveMember", "EstimatedSalary", and "Churn".

Add-ons on Data Set

- There are no missing values in the dataset.
- No need any imputation in this dataset.
- The "Churn" variable is the target variable.
- The variable "CustomerId" is also a unique identifier for each customer and is not useful for our analysis.

```
# Customer ID is a useless variable to our analysis, therefore, i decided to remove c
bankcustomer2 <- subset(bankcustomer, select = -c(customer_id))
```

Logistic Regression Model

```
# Splitting data set to train(%80), and test(%20) to make LGM.

set.seed(123)
trainIndex <- createDataPartition(bankcustomer2$churn, p = 0.8, list = FALSE)
train <- bankcustomer2[trainIndex,]
test <- bankcustomer2[-trainIndex,]


# we need to Cross-validation to get more logical results.
ctrl <- trainControl(method = "cv", number = 10)
# To train logistic regression model

model <- train(churn ~ ., data = train, method = "glm", trControl = ctrl, family = "binomi

#Let's look at the how's look like

summary(model)
```

```
Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3358  -0.6581  -0.4535  -0.2649   3.0118
```

```
Coefficients:
                   Estimate Std. Error z value Pr(>|z|)    
(Intercept)      -3.406e+00  2.750e-01 -12.385  < 2e-16 ***
credit_score     -6.923e-04  3.138e-04  -2.206   0.0274 *  
countryGermany    7.884e-01  7.585e-02  10.395  < 2e-16 ***
countrySpain      3.842e-02  7.883e-02   0.487   0.6260    
genderMale       -5.475e-01  6.098e-02  -8.978  < 2e-16 ***
age               7.399e-02  2.905e-03  25.468  < 2e-16 ***
tenure           -1.979e-02  1.049e-02  -1.887   0.0592 .  
balance           2.592e-06  5.738e-07   4.517 6.27e-06 ***
products_number  -1.036e-01  5.289e-02  -1.959   0.0501 .  
credit_card      -3.221e-02  6.648e-02  -0.485   0.6280    
active_member    -1.090e+00  6.471e-02 -16.843  < 2e-16 ***
estimated_salary  5.595e-07  5.322e-07   1.051   0.2931    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8099.8  on 7999  degrees of freedom
Residual deviance: 6834.2  on 7988  degrees of freedom
AIC: 6858.2

Number of Fisher Scoring iterations: 5
```

- As we can see The AIC value is 6858, if we tranied more than one models we can choose lower AIC value which means better model but CV has made, so we dont need the AIC value because of it.

What are those?

- "CreditScore": Its coefficient is negative (-0.0069), so the probability of a customer leaving decreases as their credit score increases.
- "Age": Its coefficient is positive (0.074), so the probability of a customer leaving increases as their age increases.
- "Tenure": Its coefficient is negative (-0.019), so the probability of a customer leaving decreases as their tenure with the bank increases.
- "Balance": Its coefficient is positive (0.00025), so the probability of a customer leaving increases as their account balance increases.

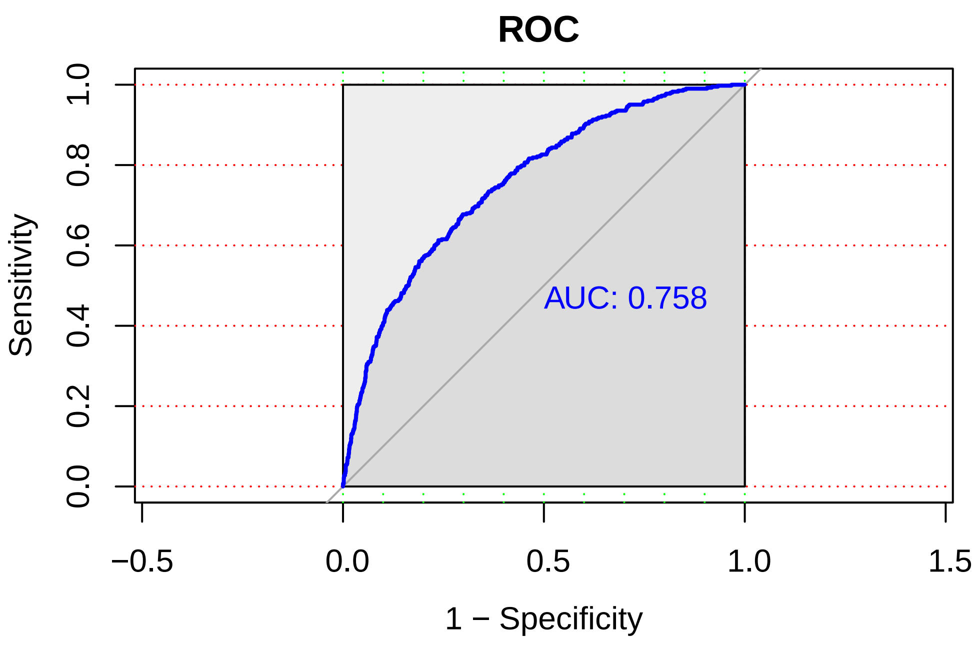
- "ProductsNumber": Its coefficient is negative (-0.0103), so the probability of a customer leaving decreases as the number of products they have purchased increases.
- "HasCreditCard": Its coefficient is negative (-0.032), but since its p-value is high, we can say that this coefficient is not statistically significant.
- "IsActiveMember": Its coefficient is negative (-1.0109), so customers who are not active members have a higher probability of leaving.

performance of the trained model.

WHY Roc Curve and AUC Value?

- The ROC curve uses the AUC (Area Under the Curve) metric, which is one of the most important metrics used to evaluate the accuracy of a classification model. AUC is a value between 0 and 1 and represents the area under the curve. The higher the AUC value, the better the classification performance of the model.
- The ROC curve also shows the imbalance of the model's error types (false positive and false negative rates). So the ROC curve helps us to evaluate the performance of the model more accurately by showing this imbalance of error types.

```r
#firstly, we need "raw" type to get probabilities of results.
pred <- predict(model, newdata = test, type = "raw")

# calculate ROC curve and AUC score
roc <- roc(test$churn, pred)
auc <- auc(roc)

# plot ROC curve
plot(roc, main = "ROC ", col = "blue", lwd = 2,
     print.auc = TRUE, auc.polygon = TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon = TRUE, legacy.axes = TRUE)
```



- The AUC value was calculated as 0.758. This indicates that the model has an average performance. Being close to the upper left corner of the curve indicates that the model has a high accuracy rate. However, the curve only shows a significant increase after the probability threshold of 0.5, indicating that the model's classification performance is not particularly high.

Checking The Imbalance Problem

- firstly, we need to check imbalancy

```
table(train$churn)
```

```

   0    1 
6366 1634 
```

```
prop.table(table(train$churn))
```

```

      0       1 
```

```
0.79575 0.20425
```

- As we can see, there is a class imbalance problem in the dataset. The “Left Customer(0)” class contains much more observations than the “still client(1)” class. This may cause the model to learn more about the “left Customer” class and misclassify the “still client” class.

Over-sampled

```
oversampled_data <- ovun.sample(churn ~ ., data = train, method = "over",N = 13000)$data
table(oversampled_data$churn)
```

```

   0    1
6366 6634
```

- Over-sampling is a technique used to increase the number of samples in a class that is underrepresented compared to the other class in an imbalanced dataset. This technique helps to reduce class imbalance and obtain a more balanced dataset.
- This code performs the sampling and increases the size of the original dataset to 13,000. This way, the number of classes will be balanced.

Under-sampled

```
undersampled_data = ovun.sample(churn ~ ., data = train, method = "under", N = 3000, seed
table(undersampled_data$churn)
```

```

   0    1
1366 1634
```

- Under-sampling is a technique used to reduce the number of samples in a class that is overrepresented compared to the other class in an imbalanced dataset. This technique helps to reduce class imbalance and obtain a more balanced dataset.
- This code performs the sampling and decreases the size of the original dataset to 3,000. This way, the number of classes will be balanced.

Model is balanced, but...?

- we've lost significant information from the sample. Let's do both undersampling and oversampling on this imbalanced data. This can be achieved using method = "both". In this case, the minority class is oversampled with replacement and majority class is undersampled without replacement.

```
#p refers to the probability of positive class in newly generated sample.
data_balanced_both <- ovun.sample(churn ~ ., data = train, method = "both", p=0.5, N=2100,
table(data_balanced_both$churn)
```

```

   0    1 
1090 1010 
```

- Now, we almost same amont value (0,1)

#Training the over-undersampling Models

```
# Splitting the data
set.seed(123)
train_idx <- createDataPartition(oversampled_data$churn, p = 0.8, list = FALSE)
train_balanced <- oversampled_data[train_idx, ]
test_balanced <- oversampled_data[-train_idx, ]

# Training LGM
model_os <- glm(churn ~ ., data = train_balanced, family = "binomial")

# Prediction
pred_os <- predict(model_os, newdata = test_balanced)
```

- Training Model of over-sampled data.

```
set.seed(123)
train_idx2 <- createDataPartition(undersampled_data$churn, p = 0.8, list = FALSE)
train_balanced2 <- undersampled_data[train_idx, ]
test_balanced2 <- undersampled_data[-train_idx, ]


model_us <- glm(churn ~ ., data = train_balanced2, family = "binomial")
```

```r
pred_us <- predict(model_us, newdata = test_balanced2)
```

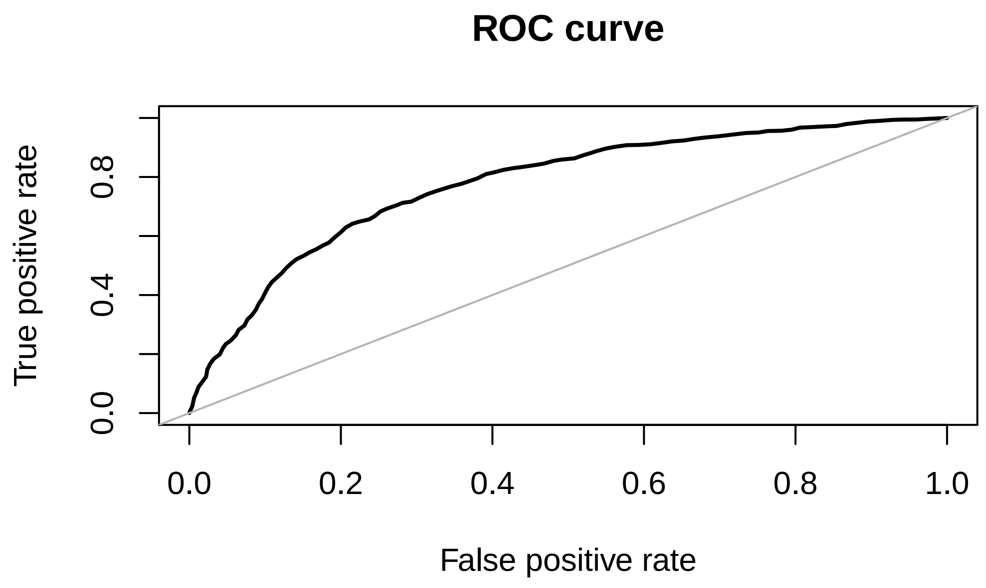
- Training Model of under-sampled data.

```r
set.seed(123)
train_idx3 <- createDataPartition(data_balanced_both$churn, p = 0.8, list = FALSE)
train_balanced3 <- data_balanced_both[train_idx, ]
test_balanced3 <- data_balanced_both[-train_idx, ]


model_both <- glm(churn ~ ., data = train_balanced3, family = "binomial")


pred_both <- predict(model_both, newdata = test_balanced3)
```

- Training Model of Both sample data.

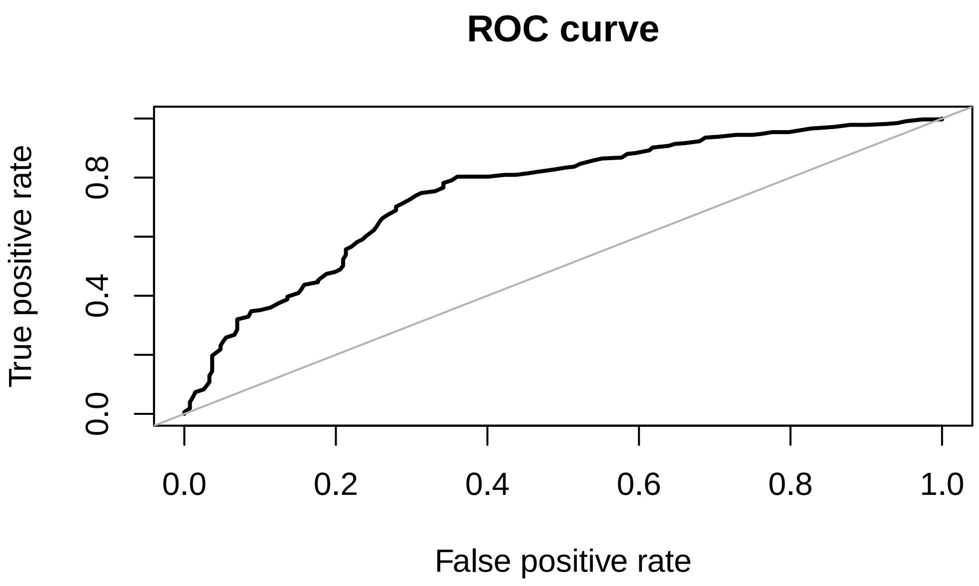
ROC Curves of Under-Oversamplings

```r
#AUC Oversampling
roc.curve(test_balanced$churn, pred_os)
```



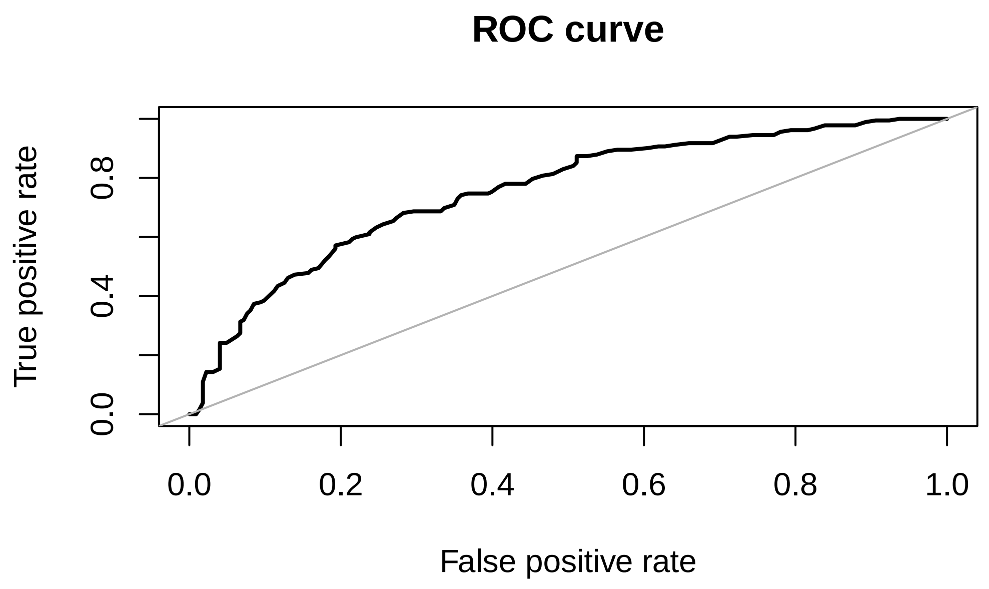
```
Area under the curve (AUC): 0.777
```

```
#AUC Undersampling
roc.curve(test_balanced2$churn, pred_us)
```



```
Area under the curve (AUC): 0.752
```

```
#AUC BOTH Sampling
roc.curve(test_balanced3$churn, pred_both)
```



```
Area under the curve (AUC): 0.756
```

- About oversampling we can see that Area under the curve (AUC): 0.777
- About undersampling we can see that Area under the curve (AUC): 0.752
- Aboout Both sampling we can see that Area under the curve (AUC): 0.756
- Both Sampling has not the highest AUC Value, So there is no reason to choose.

Conclusion:

- Oversampling's AUC value is higher than Undersampling's one, so I have to choose Oversampling. Because AUC values gives us the under the curve that means a higher AUC value indicates that the model has a better classification performance. The closer the AUC value is to 1, the better the classification performance of the model. If the AUC value is close to 0.5, the classification performance of the model is similar to a random classifier. Therefore, a higher AUC value indicates that the model can classify more accurately and makes fewer errors.