Customer Churn Prediction

Hüseyin Durmaz

```
datachurn <- read.csv("Bank Customer Churn Prediction.csv")

# First of all, we need libraries for functions etc.


library(DALEX)
library(ROCR)

#for oversampling and undersampling
library(ROSE)
library(caret)
library(pROC)
str(datachurn)
```

```
'data.frame':   10000 obs. of  12 variables:
 $ customer_id     : int  15634602 15647311 15619304 15701354 15737888 15574012 15592531 1565
 $ 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 ...
```

Details about the task

Problem The problem is the customer churn of ABC Multinational Bank.

Features

customer_id : Account Number

credit_score : Credit Score

country : Country of Residence

gender : Sex

age : Age

tenure : From how many years he/she is having bank acc in ABC Bank

balance : Account Balance

products_number : Number of Product from bank

credit_card : Is this customer have credit card ? (1 = Yes, 0 = No)

active_member : Is he/she is active Member of bank ? (1 = Yes, 0 = No)

Target

The aim of the data will be predicting the Customer Churn.

In addition

''churn" variable is the target.

Describing the data set

Dimension

There are 10,000 observations and 12 features in the dataset.

Variable Type

The categorical variables are country and gender.

The numerical variables are credit_score, age, tenure, balance, products_number, credit_card, active_member, estimated_salary, and churn.

```
datachurn <- datachurn[, -c(1,3)]
```

Remove `country` and `customer_id` feature from the data set. It is a problematic categorical feature in model training, because it has many classes.

```
set.seed(4826) # for reproducibility
index <- sample(1 : nrow(datachurn), round(nrow(datachurn) * 0.80))
train <- datachurn[index, ]
test  <- datachurn[-index, ]
```

Training data here.

```r
lr_model <- glm(churn ~ ., data = train, family = "binomial")
```

We used the 'glm()' function to train a logistic regression model.

```r
lr_model
```

```
Call:  glm(formula = churn ~ ., family = "binomial", data = train)

Coefficients:
     (Intercept)      credit_score        genderMale               age
      -3.402e+00        -7.094e-04        -5.457e-01         7.226e-02
          tenure           balance   products_number       credit_card
      -6.002e-03         4.973e-06        -5.018e-02        -7.705e-02
   active_member  estimated_salary
      -1.026e+00         3.842e-07

Degrees of Freedom: 7999 Total (i.e. Null);  7990 Residual
Null Deviance:      8067
Residual Deviance: 6995     AIC: 7015
```

```r
summary(lr_model)
```

```
Call:
glm(formula = churn ~ ., family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1201  -0.6745  -0.4732  -0.2856   2.9031

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      -3.402e+00  2.708e-01 -12.564   <2e-16 ***
credit_score     -7.094e-04  3.110e-04  -2.281   0.0225 *
genderMale       -5.457e-01  6.028e-02  -9.053   <2e-16 ***
age               7.226e-02  2.859e-03  25.276   <2e-16 ***
tenure           -6.002e-03  1.033e-02  -0.581   0.5611
```

```
balance            4.973e-06  5.126e-07   9.701   <2e-16 ***
products_number   -5.018e-02  5.182e-02  -0.968   0.3328
credit_card       -7.705e-02  6.531e-02  -1.180   0.2381
active_member     -1.026e+00  6.355e-02 -16.144   <2e-16 ***
estimated_salary   3.842e-07  5.246e-07   0.732   0.4639
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8067.1  on 7999  degrees of freedom
Residual deviance: 6995.4  on 7990  degrees of freedom
AIC: 7015.4

Number of Fisher Scoring iterations: 5
```

```r
predicted_probs <- predict(lr_model, test, type = "response")
head(predicted_probs)
```

```
        11         12         15         17         18         19
0.15706247 0.06111598 0.07288306 0.57653213 0.03526677 0.24279732
```

```r
predicted_classes <- ifelse(predicted_probs > 0.5, 1, 0)
head(predicted_classes)
```

```
11 12 15 17 18 19
 0  0  0  1  0  0
```

We will calculate metrics based on the confusion matrix, assuming that 1 represents the positive class and 0 represents the negative class.

```r
TP <- sum(predicted_classes[which(test$churn == 1)] == 1)
FP <- sum(predicted_classes[which(test$churn == 1)] == 0)
TN <- sum(predicted_classes[which(test$churn == 0)] == 0)
FN <- sum(predicted_classes[which(test$churn == 0)] == 1)
recall      <- TP / (TP + FN)
specificity <- TN / (TN + FP)
precision   <- TP / (TP + FP)
accuracy    <- (TN + TP) / (TP + FP + TN + FN)
```

```r
recall
```

```
[1] 0.6447368
```

```r
specificity
```

```
[1] 0.8284632
```

```r
precision
```

```
[1] 0.2361446
```

```r
accuracy
```

```
[1] 0.8145
```

Recall (Sensitivity) is 0.6447368: This indicates that only about two-thirds of all true positive cases are correctly classified as positive according to the given model.

Specificity is 0.8284632: This indicates that approximately 82.8% of all true negative cases are correctly classified as negative according to the given model.

Precision is 0.2361446: This indicates that only about 23.6% of positively classified examples are actually positive according to the given model.

Accuracy is 0.8145: This indicates that approximately 81% of all examples are correctly classified according to the given model. However, in cases of class imbalance, accuracy can be misleading, and other performance metrics should also be considered.

```r
table(train$churn) / dim(train)[1]
```

```

      0       1 
0.79725 0.20275 
```

We observe that the data set is imbalanced in terms of the sample size.

```r
confusion_matrix <- table(test$churn, predicted_classes)
confusionMatrix(confusion_matrix, positive = "1")
```

```
Confusion Matrix and Statistics

   predicted_classes
       0    1
  0 1531   54
  1  317   98

               Accuracy : 0.8145
                 95% CI : (0.7968, 0.8313)
    No Information Rate : 0.924
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2638

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6447
            Specificity : 0.8285
         Pos Pred Value : 0.2361
         Neg Pred Value : 0.9659
             Prevalence : 0.0760
         Detection Rate : 0.0490
   Detection Prevalence : 0.2075
      Balanced Accuracy : 0.7366

       'Positive' Class : 1
```

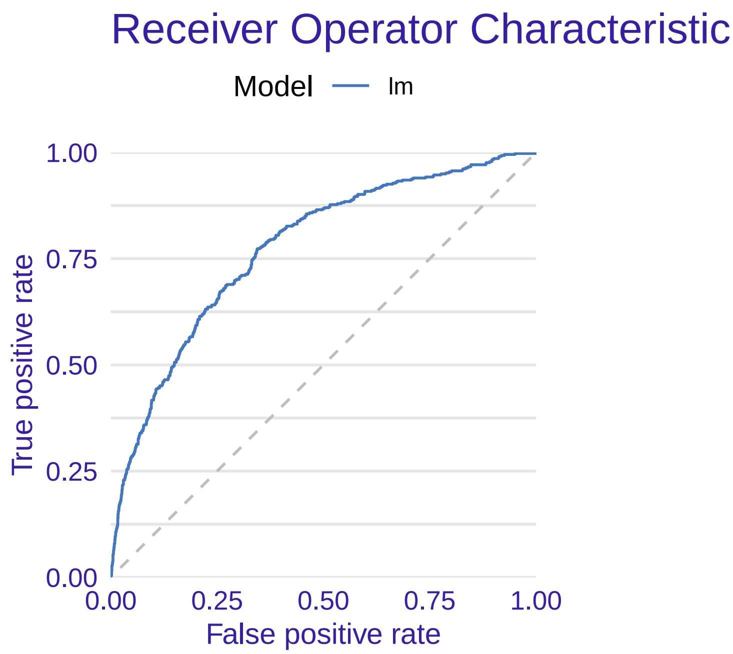
This code creates a confusion matrix by comparing the predicted classes (predicted_classes) with the true datas (test$churn) using the table() function. Then, the confusionMatrix() function is used to compute performance metrics, such as sensitivity and specificity, based on the generated confusion matrix. It is noted that the positive class is defined as "1".

```r
explain_lr <- explain(model    = lr_model,
                      data     = test[, -10],
                      y        = test$churn == 1,

                  type     = "classification", verbose = FALSE)
```

This code creates an explainer object using the explain() function from the DALEX package. The explainer object provides a detailed report or explanation for a logistic regression model (lr_model) using a test dataset (test[, -10]) and observed values of the target variable (test$churn == 1).

```
performance_lr <- model_performance(explain_lr)
plot(performance_lr, geom = "roc")
```




```
performance_lr
```

```
Measures for:  classification
recall     : 0.2361446
precision  : 0.6447368
f1         : 0.345679
accuracy   : 0.8145
auc        : 0.773944

Residuals:
         0%         10%         20%         30%         40%         50%
-0.79757816 -0.33868105 -0.22716241 -0.17400515 -0.13211265 -0.10195767
        60%         70%         80%         90%        100%
-0.07560689 -0.05292294  0.29840456  0.69878012  0.98724517
```

"ROC curve is a tool used to visually evaluate the performance of a classification model. The curve shows the sensitivity (true positive rate) and specificity (true negative rate) of the model. It is also possible to calculate the area under the curve (AUC).

The ROC curve of this model is a curve with an **AUC** value of **0.7739**. This indicates that the classification performance of the model is reasonably good. However, since the recall value is very low (about 23%), the model is weak in correctly classifying the positive class.

The precision value of the model is approximately 64%, which indicates that about 64% of the predictions in the positive class are correct.

In conclusion, the performance of this model can be improved. Especially, improvement can be made due to the low sensitivity value, which is weak in correctly classifying the positive class."

First, we check for the imbalance problem.

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

```
   0    1
6378 1622
```

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

```
      0       1
0.79725 0.20275
```

The proportion of observations with class label '0' is approximately 80%, while that of class label '1' is approximately 20%. Therefore, it may be difficult to learn the minority class (class label '1') accurately and the model's accuracy may be misleadingly high. Hence, when evaluating model performance, accuracy alone may not be sufficient and different metrics such as precision, recall or F1 score may need to be used

We apply the same encoding procedures used in the training dataset for oversampling and undersampling.

Firstly, oversampling..

```
oversampled <- ovun.sample(churn ~ ., data = train, method = "over", N = 12756)$data
table(oversampled$churn)
```

```
   0    1 
6378 6378 
```

We balance the data set by increasing the sample size through data augmentation.

The ovun.sample() function was used to increase the number of observations belonging to the minority class in the train dataset and bring them closer to each other.

Over-sampling is a technique used to increase the number of observations belonging to the minority class in a data set with class imbalance. With this technique, observations from the minority class are duplicated to reduce class imbalance in the data set. This process can help the observations from the minority class to gain more weight and be better learned. However, duplicating patterns and relationships in the data set after the over-sampling process can cause the model to overfit.

```
set.seed(4826) # for reproducibility
over_index <- sample(1 : nrow(oversampled), round(nrow(oversampled) * 0.80))
over_train <- oversampled[index, ]
over_test  <- oversampled[-index, ]
```

Training oversampled data here.

```
lr_model_over <- glm(churn ~ ., data = over_train, family = "binomial")
lr_model_over
```

```
Call:  glm(formula = churn ~ ., family = "binomial", data = over_train)

Coefficients:
     (Intercept)      credit_score        genderMale               age  
      -2.830e+00        -5.923e-04        -5.541e-01         7.404e-02  
          tenure           balance   products_number       credit_card  
       3.046e-03         5.150e-06        -8.570e-02        -2.643e-02  
   active_member  estimated_salary  
      -8.648e-01         3.220e-07  

Degrees of Freedom: 7999 Total (i.e. Null);  7990 Residual
Null Deviance:      10500 
Residual Deviance: 9066     AIC: 9086
```

We used the 'glm()' function to train a logistic regression model.

```r
summary(lr_model_over)
```

```

Call:
glm(formula = churn ~ ., family = "binomial", data = over_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max  
-2.4052  -0.8751  -0.5753   1.0419   2.5992  

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)    
(Intercept)      -2.830e+00  2.302e-01 -12.297   <2e-16 ***
credit_score     -5.923e-04  2.614e-04  -2.265   0.0235 *  
genderMale       -5.541e-01  5.126e-02 -10.809   <2e-16 ***
age               7.404e-02  2.643e-03  28.016   <2e-16 ***
tenure            3.046e-03  8.723e-03   0.349   0.7270    
balance           5.150e-06  4.266e-07  12.071   <2e-16 ***
products_number  -8.570e-02  4.071e-02  -2.105   0.0353 *  
credit_card      -2.643e-02  5.584e-02  -0.473   0.6360    
active_member    -8.648e-01  5.224e-02 -16.553   <2e-16 ***
estimated_salary  3.220e-07  4.426e-07   0.727   0.4669    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 10498.7  on 7999  degrees of freedom
Residual deviance:  9066.4  on 7990  degrees of freedom
AIC: 9086.4

Number of Fisher Scoring iterations: 4
```

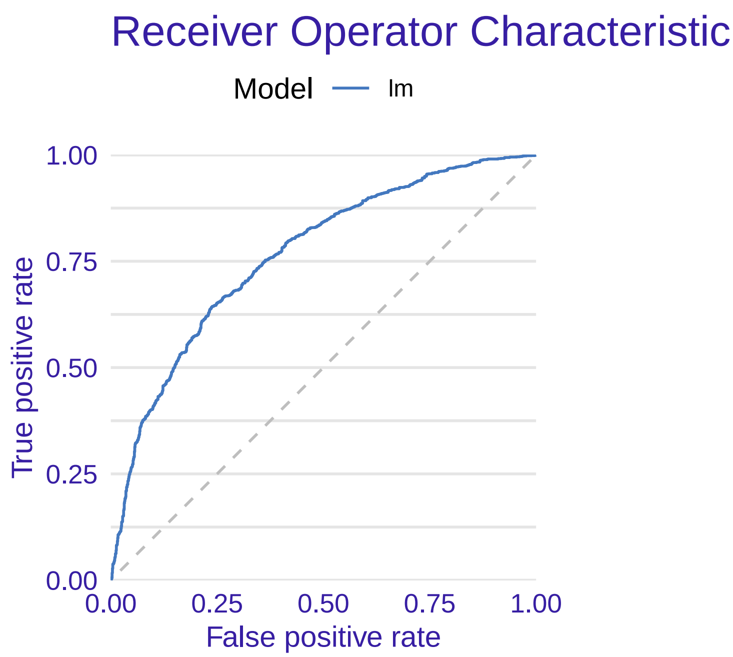
```r
predicted_probs_over <- predict(lr_model_over, test, type = "response")
head(predicted_probs_over)
```

```
        11         12         15         17         18         19 
0.26817542 0.11244764 0.16312487 0.74999386 0.08247472 0.39731983 
```

```r
explain_lr_over <- explain(model    = lr_model_over,
                           data     = over_test[, -10],
                           y        = over_test$churn == 1,

                       type     = "classification", verbose = FALSE)
```

```r
performance_lr2 <- model_performance(explain_lr_over)
plot(performance_lr2, geom = "roc")
```




```r
performance_lr2
```

```
Measures for:  classification
recall     : 0.4680544
precision  : 0.9039643
f1         : 0.6167619
accuracy   : 0.5769554
auc        : 0.7644088

Residuals:
        0%         10%         20%         30%         40%         50%         60%
```

```
-0.9537321 -0.3235322 -0.1741125  0.1704706  0.3256858  0.4047501  0.4909336
       70%        80%        90%       100%
 0.5701419  0.6571939  0.7466092  0.9658809
```

The ROC curve of this model is a curve with an **AUC** value of **0.7644**.

Then under sampling data,

We balance the imbalanced data by reducing the sample size to equalize the data set.

```
undersampled <- ovun.sample(churn ~ ., data = train, method = "under", N = 3250)$data
table(undersampled$churn)
```

```
   0    1
1628 1622
```

We balance the data set by reducing the sample size.

```
set.seed(4826) # for reproducibility
under_index <- sample(1 : nrow(undersampled), round(nrow(undersampled) * 0.80))
under_train <- undersampled[index, ]
under_test  <- undersampled[-index, ]
```

Training undersampled data here.

```
lr_model_under <- glm(churn ~ ., data = under_train, family = "binomial")
lr_model_under
```

```
Call:  glm(formula = churn ~ ., family = "binomial", data = under_train)

Coefficients:
     (Intercept)      credit_score        genderMale               age
      -3.223e+00        -3.909e-04        -5.486e-01         9.202e-02
          tenure           balance   products_number       credit_card
      -9.434e-03         5.575e-06        -3.320e-02        -1.308e-01
   active_member  estimated_salary
      -8.018e-01         4.666e-07

Degrees of Freedom: 2592 Total (i.e. Null);  2583 Residual
```

```
  (5407 observations deleted due to missingness)
Null Deviance:      3593
Residual Deviance: 2997     AIC: 3017
```

We used the 'glm()' function to train a logistic regression model.

```
predicted_probs_under <- predict(lr_model_under, test, type = "response")
head(predicted_probs_under)
```

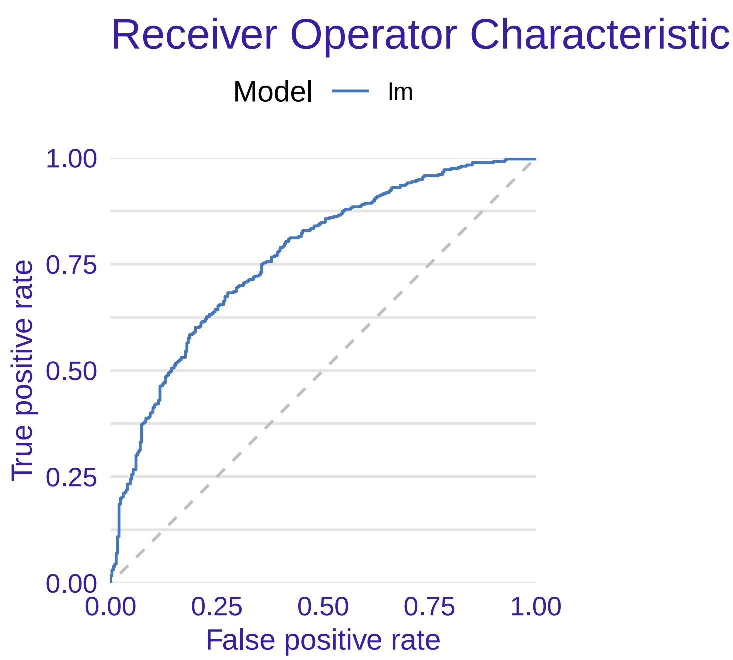
```
        11         12         15         17         18         19
0.34480563 0.12488565 0.21678611 0.86746470 0.09062997 0.53102121
```

We convert probabilities to classes to measure the performance of the model.

```
explain_lr_under <- explain(model   = lr_model_under,
                      data    = under_test[, -10],
                      y       = under_test$churn == 1,

                  type    = "classification", verbose = FALSE)
```

```
performance_lr3 <- model_performance(explain_lr_under)
plot(performance_lr3, geom = "roc")
```



```
performance_lr3
```

```
Measures for:  classification
recall      : 0.6348315
precision   : 0.7583893
f1          : 0.6911315
accuracy    : 0.6925419
auc         : 0.7681604

Residuals:
        0%        10%        20%        30%        40%        50%        60%
-0.9495543 -0.5126285 -0.3471393 -0.2336273 -0.1429836  0.1185262  0.2505390
       70%        80%        90%       100%
 0.3620777  0.4925105  0.6558755  0.9623069
```

The ROC curve of this model is a curve with an **AUC** value of **0.7681**.

To Conclude

When we look at the performance metrics of the data sets, precision, recall, F1 score, and AUC value differ in all three data sets.

In the first data set, precision, recall, and F1 score are low, but the accuracy and AUC value are at a reasonable level. In the second data set, precision, recall, and F1 score are highest value because of the over-sampling, but the accuracy is low. AUC value is at an reasonable level. In the third data set, precision, recall, and F1 score are the highest due to under-sampling, and the accuracy and AUC value are also at a reasonable level. Therefore, the best-performing data set is the third data set. Under-sampling is a correct method to balance the data set and helps the model to learn. Additionally, unlike over-sampling, under-sampling reduces the number of observations in the data set, so it requires less computational power and has a lower risk of overfitting that can occur due to over-sampling.