Prediction of Customer Churn by Using Steps of Classification Models

Sahranur İnce

4/3/23

Calling The Data Set

```
library(readr)
Bank_Customer_Churn_Prediction <- read_csv("Bank Customer Churn Prediction.csv")
```

1.Detail your task with the problem, features, and target.

Problem:

We are dealing with the problem of `Customer churn prediction`. Our aim is to Predict the Customer Churn for ABC Bank. We will look for an answer and prediction to our problem with the features in the dataset.

Features:

There are 10 features in this dataset. These are: `credit_score`, `country`, `gender`, `age`, `tenure`, `balance`, `products_number`, `credit_card`, `active_member` and `estimated_salary`.

Target:

Churn: Used as the target. 1 if the client has left the bank during some period or 0 if he/she has not.

2.Describe the dataset in your task in terms of the dimension, variable type, and some other that you want to add.

```
# install.packages("DALEX")
# install.packages("caret")
# install.packages("ROCR")
library(DALEX)
library(caret)
library(ROCR)
```

Here, `customer_id` is omitted from the dataset because it is an unused feature. Therefore, our new dataset contains 11 features.

```
Bank_Customer_Churn_Prediction <-Bank_Customer_Churn_Prediction[,-1]
```

Here, the `str()` function is used to determine dimensions and what the types of features are.

```
str(Bank_Customer_Churn_Prediction)
```

```
tibble [10,000 x 11] (S3: tbl_df/tbl/data.frame)
 $ credit_score    : num [1:10000] 619 608 502 699 850 645 822 376 501 684 ...
 $ country         : chr [1:10000] "France" "Spain" "France" "France" ...
 $ gender          : chr [1:10000] "Female" "Female" "Female" "Female" ...
 $ age             : num [1:10000] 42 41 42 39 43 44 50 29 44 27 ...
 $ tenure          : num [1:10000] 2 1 8 1 2 8 7 4 4 2 ...
 $ balance         : num [1:10000] 0 83808 159661 0 125511 ...
 $ products_number : num [1:10000] 1 1 3 2 1 2 2 4 2 1 ...
 $ credit_card     : num [1:10000] 1 0 1 0 1 1 1 1 0 1 ...
 $ active_member   : num [1:10000] 1 1 0 0 1 0 1 0 1 1 ...
 $ estimated_salary: num [1:10000] 101349 112543 113932 93827 79084 ...
 $ churn           : num [1:10000] 1 0 1 0 0 1 0 1 0 0 ...
```

According to the results obtained from the output;

The dimension of this dataset is [10,000 $\times$ 11] matrix. The features credit_score, age, tenure, balance, products_number, credit_card, active_member, estimated_salary and the target churn are numeric. The features country and gender are categorical.

Here, the `sum(is.na())` function is used to check if there are missing observations in the dataset.

```
sum(is.na(Bank_Customer_Churn_Prediction))
```

```
[1] 0
```

According to the results obtained, no missing observations were found in the dataset.

3.Train a logistic regression model.

Splitting The Data Set

Here, the `sample()` function used to split the data set as test and train set. The `set.seed()` function is used for reproducibility.

```
set.seed(123)
index <- sample(1 : nrow(Bank_Customer_Churn_Prediction),
                round(nrow(Bank_Customer_Churn_Prediction) * 0.80))
train <- Bank_Customer_Churn_Prediction[index, ]
test  <- Bank_Customer_Churn_Prediction[-index, ]
```

As a result, there are 8000 observation and 11 features in the train set. There are 2000 observation and 11 features in the test set.

Here, the `dim()` function is used to show the row and column counts of the test and train sets.

```
dim(train)
```

```
[1] 8000   11
```

```
dim(test)
```

```
[1] 2000   11
```

Train a Logistic Regression Model

Here, the `glm()` function is used to train a logistic regression model. We used the train data and model formula that we split in the previous step. We used `y ~.` instead of defining all features you want to input to the model.

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

```r
lr_model
```

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

Coefficients:
     (Intercept)      credit_score    countryGermany      countrySpain
      -3.406e+00        -6.923e-04         7.884e-01         3.842e-02
      genderMale               age            tenure           balance
      -5.475e-01         7.399e-02        -1.979e-02         2.592e-06
 products_number       credit_card     active_member  estimated_salary
      -1.036e-01        -3.221e-02        -1.090e+00         5.595e-07

Degrees of Freedom: 7999 Total (i.e. Null);  7988 Residual
Null Deviance:      8100
Residual Deviance: 6834     AIC: 6858
```

```r
summary(lr_model)
```

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

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
```

In the output above; was obtained the minimum, maximum, median, 1st quartile and 3rd quartile values of the features. We see some significant statistics for the features. For example, we can look at the p-values for the significance of the variables. If only one class of categorical variables is significant, the others are also significant.

4.Report the performance of the trained model.

Measuring Model Performance

Here, `predicted()` function is used to predict values of the target variable on test set. We used test set for check the model performance because we are interested to train such a model has a good generalizability performance. We should exclude the true values of the target variable from the test set. Therefore, the target variable in the 11th column of the dataset has been exclude.

```r
predicted_probs <- predict(lr_model, test[,-11], type = "response")
head(predicted_probs)
```

```
        1         2         3         4         5         6
0.24512258 0.24506937 0.25785568 0.11421165 0.04111743 0.04277063
```

Here, to measure the model performance, we transform the probabilities to classes. `1` indicates the client has left the bank during some period, while `0` indicates the client has not. In here we used `ifelse()` function. The first argument condition means that if the values of the `predicted_probs` vector is higher than the 0.5 assings it as 1, if it is not assigns it as 0.

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

```
1 2 3 4 5 6
0 0 0 0 0 0
```

Since all the values of the `predicted_probs` vector above are less than 0.5, we got the labels of 0 here.

Here, we calculated the metrics based on the confusion matrix. We assumed that 1 is the positive and 0 is 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)
```

```r
recall      <- TP / (TP + FN)
specificity <- TN / (TN + FP)
precision   <- TP / (TP + FP)
accuracy    <- (TN + TP) / (TP + FP + TN + FN)
```

```r
recall
```

```
[1] 0.5620915
```

```r
specificity
```

```
[1] 0.8283703
```

```r
precision
```

```
[1] 0.2133995
```

```r
accuracy
```

```
[1] 0.808
```

According to the values of performance metrics, the model classifies the observations with 0.808 accuracy. Its precision is 0.213 means that the model classify only 21% of the positive class correctly.

The fact that the precision value is not good and not close to the accuracy value signals us that there is an imbalance problem here. So, we should check imbalancedness of the classes in the target feature.

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

```
      0       1
0.79575 0.20425
```

According to the results 79% of the variables in the train set belong to the 0 class (clients who do not left the bank). 20% of the variables belong to class 1 (clients who left the bank). So we can see that, the classes of the target feature is not balanced. In other words, the number of observations in these two classes are not equal or approximate.There is a signal for imbalancedness problem here. This situation, it may cause that the model can learn less from the minority class, so the model could not achieve a satisfying performance on classifying of the minority class.

We can easily calculate `confusion matrix` and `performance metrics` thanks to the `confusionMatrix()` function in the `{caret}` package.

```
confusionMatrix(table(ifelse(test$churn == "1", "1", "0"),
                      predicted_classes),
                positive = "1")
```

```
Confusion Matrix and Statistics

   predicted_classes
       0    1
  0 1530   67
  1  317   86

               Accuracy : 0.808
                 95% CI : (0.79, 0.8251)
    No Information Rate : 0.9235
```

```
   P-Value [Acc > NIR] : 1

                 Kappa : 0.2232

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

            Sensitivity : 0.5621
            Specificity : 0.8284
         Pos Pred Value : 0.2134
         Neg Pred Value : 0.9580
             Prevalence : 0.0765
         Detection Rate : 0.0430
   Detection Prevalence : 0.2015
      Balanced Accuracy : 0.6952

       'Positive' Class : 1
```

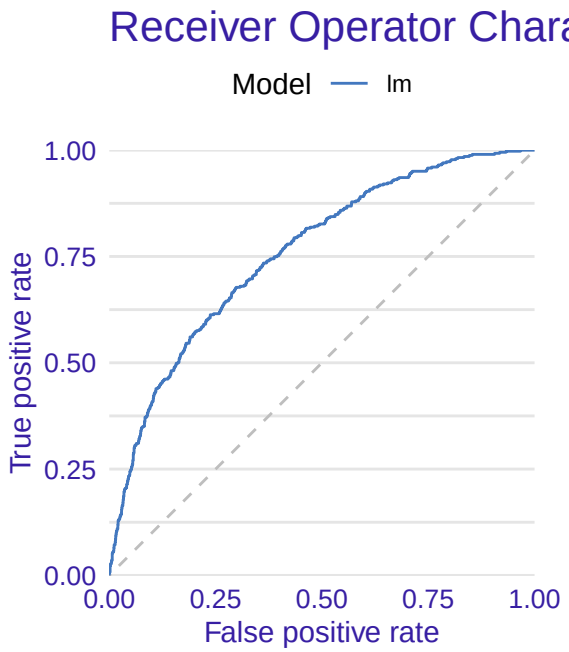
Here, we see that the results obtained are the same as those we calculated manually above. In here, we can look at the values of `balanced accuracy` and `accuracy` to determine if there is a problem with imbalancedness. Since the value of balanced accuracy is smaller than the value of accuracy, it may signal that there is a problem with imbalancedness here. Since there is an imbalancedness problem here, the balanced accuracy value is taken into account instead of accuracy as a general performance metrics.

Here, the `ROC curve` is calculated. The ROC curve is a graph showing the performance of a classification model.

```
explain_lr <- explain(model   = lr_model,
                      data    = test[, -11],
                       y       = test$churn == "1",
                      type    = "classification",
                      verbose = FALSE)
```

In this graph, there is the FP(False Positive) rate on the x-axis and the TP(True Positive) rate on the y-axis.

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




```
performance_lr
```

```
Measures for:  classification
recall     : 0.2133995
precision  : 0.5620915
f1         : 0.3093525
accuracy   : 0.808
auc        : 0.7577918

Residuals:
         0%         10%         20%         30%         40%         50%
-0.87267532 -0.33010251 -0.23453307 -0.17147023 -0.13220738 -0.09748351
        60%         70%         80%         90%        100%
-0.06758197 -0.04495876  0.12343130  0.70324206  0.97167062
```

AUC stands for area under the ROC Curve. The higher AUC indicates higher performance. AUC takes values between the 0 and 1. According to the results obtained above, we see that the AUC value is 0.75. So, we can say that the performance of the model is going to perfect. In other words, the model works perfectly.