Cancer Detection

Gizem Altun

Supervised Learning: Logistic Regression Models

In this task, the following steps were followed and the logistic regression was dealt with.

1. Definition of the Problem, Target and Features
2. Describing the Data Set
3. Splitting the Data Set(Train and Test Set)
4. Training a Logistic Regression Model
5. Performance of the Trained model
6. Checking the Imbalance Problem
7. Under-sampling Methods to Balance the Data Set
8. Compare the Performance of the Models which are Trained on Imbalanced, Undersampled Train Data.

1. Definition of the Problem, Target and Features

In this task, to predict the alive probability of the which have patients breast cancer.

Target variable and features were examined in this dataset. A dependent variable(target) is a variable that is observed to change in response to independent variables(features). Then the target variable in this data set is the "Status" variable.(Y: Status) Since we have a target variable in this dataset, this process as "supervised learning" process. The target variable is categorical variable. It means that the target variable or any categorical variable can only text the values of some classes. In this task, only two possible outcome of this variable so then we an call the status variable as categorical variable which takes just two values. The values are alive or dead. This is called the "Binary Classification Task" because the target variable(Status) takes only two classes.(alive or dead)

Independent variables(features) are variables that cause a change in dependent variable(target). The features in this dataset are; X1:Age, X2:Race, X3:Marital Status, X4:T Stage, X5:N Stage, X6:Sixth Stage, X7:differentiate, X8:Grade, X9:A Stage, X10:Tumor Size, X11:Estrogen Status, X12:Progesterone Status, X13:Regional Node Examined X14:Reginol Node Positive, X15:Survival Months. There are a total of 15 features and 1 target variable(Status) in this task.

Packages

Before proceeding to the steps written above about logistic regression, the packages required for these steps have been installed.As a first step, the packages were installed. As a second step, ıt was called from the library.

```r
#install.packages("DALEX")
#Comparing performance across multiple models convenient.
#install.packages("readxl")
#This package allows us to read our dataset.
#install.packages("caret")
#set of functions that attempt to streamline the process for creating predictive models.
#install.packages("ROCR")
#ROCR is a package for evaluating and visualizing the performance of scoring classifiers.
library(DALEX)
library(readxl)
library(caret)
library(ROCR)
```

The first line is hashtaged to faster this process.

2. Describing the Data Set

This dataset of breast cancer patients was obtained from the 2017 November update of the SEER Program of the NCI, which provides information on population-based cancer statistics. The dataset involved female patients with infiltrating duct and lobular carcinoma breast cancer (SEER primary cites recode NOS histology codes 8522/3) diagnosed in 2006-2010. Patients with unknown tumour size, examined regional LNs, positive regional LNs, and patients whose survival months were less than 1 month were excluded; thus, 4024 patients were ultimately included. This dataset is taken from

Kaggle.(https://www.kaggle.com/datasets/reihanenamdari/breast-cancer) The task is to predict the alive probability of the which have patients breast cancer. The dataset contains several informations about the patients and their alive status. The readxl package was used because the data set's Excel file is of the xlsx file type. At the same time, import dataset from environment window to add the dataset.

```
Breast_Cancer <- read_excel("Breast_Cancer.xlsx")
```

The data set was looked at using the str() function. This function gives us the following information; number of observation, number of features, name of features, type of features, a few observations of features.

```
str(Breast_Cancer)
```

```
tibble [4,024 x 16] (S3: tbl_df/tbl/data.frame)
 $ Age                   : num [1:4024] 68 50 58 58 47 51 51 40 40 69 ...
 $ Race                  : chr [1:4024] "White" "White" "White" "White" ...
 $ Marital_Status        : chr [1:4024] "Married" "Married" "Divorced" "Married" ...
 $ T_Stage               : chr [1:4024] "T1" "T2" "T3" "T1" ...
 $ N_Stage               : chr [1:4024] "N1" "N2" "N3" "N1" ...
 $ Sixth_Stage           : chr [1:4024] "IIA" "IIIA" "IIIC" "IIA" ...
 $ differentiate         : chr [1:4024] "Poorly differentiated" "Moderately differentiated" "
 $ Grade                 : chr [1:4024] "3" "2" "2" "3" ...
 $ A_Stage               : chr [1:4024] "Regional" "Regional" "Regional" "Regional" ...
 $ Tumor_Size            : num [1:4024] 4 35 63 18 41 20 8 30 103 32 ...
 $ Estrogen_Status       : chr [1:4024] "Positive" "Positive" "Positive" "Positive" ...
 $ Progesterone_Status   : chr [1:4024] "Positive" "Positive" "Positive" "Positive" ...
 $ Regional_Node_Examined: num [1:4024] 24 14 14 2 3 18 11 9 20 21 ...
 $ Reginol_Node_Positive : num [1:4024] 1 5 7 1 1 2 1 1 18 12 ...
 $ Survival_Months       : num [1:4024] 60 62 75 84 50 89 54 14 70 92 ...
 $ Status                : chr [1:4024] "Alive" "Alive" "Alive" "Alive" ...
```

Describing the Dataset:

This dataset contains 4024 observation of 16 variables. This also equates to 4024 rows and 16 columns. Variables:

1.Age: Numeric and continuous variable. Continuous because, it takes the values between zero and positive infinity it takes any values.

2.Race: Categoric and unordered. These data are data that cannot be measured numerically. The variables that we cannot superiority with, that we cannot rate, are unordered.

3.Marital Status: Categoric and unordered variable.

4.T Stage: Categoric and ordered variable. The relationships that we can rate with a logical sequence are ordered.

5.N Stage: Categoric and ordered variable.

6.Sixth Stage: Categoric and ordered variable.

7.differentiate: Categoric and unordered variable.

8.Grade: Categoric and ordered variable.

9.A Stage: Categoric and ordered variable.

10.Tumor Size: Numeric and continuous variable.

11.Estrogen Status: Categoric and unordered variable.

12.Progesterone Status: Categoric and unordered variable.

13.Regional Node Examined: Numeric and continuous variable.

14.Reginol Node Positive: Numeric and continuous variable.

15.Survival Months: Numeric and continuous variable.

16.Status: Categoric and unordered variable.

From the first variable to the fifteenth variable, they are all features. The sixteenth variable(Status) is the target variable. As can be seen, the dataset contains the target variable. This process as "Supervised Learning" process. According to type of the target variable we can conclude that type of task. The type of target variable is categoric this is called classification task and only two possible outcome of this variable so then we an call the status variable as categorical variable which takes just two values. The values are alive or dead. This is called the "Binary Classification Task" because the target variable(Status) takes only two classes.(alive or dead) Since the type of target variable is categorical, logistic regression is used.

```
dim(Breast_Cancer)
```

```
[1] 4024   16
```

The dim() function shows us to dimension of dataset. When there are how many observations in the first column, we see how many variables there are in the second column. This dataset contains 4024 observation of 16 varibles.

Before training the model, let's look at the target variable type and missing value states.

Since our target variable remains a character type, we need to make it categorical. While doing this, we will use the as.factor function. The syntax is as.factor(input), where the input is a vector, column, or data frame and returns the requested column specified as a factor rather than a character one. The reason we do this is that we should attribute it to taking values as a probability in the following way; y is a binary target variable takes the values (0,1). p is the probability of y=1 (alive) , p =P(Y=1), p is the probability of y=0(dead) , p =P(Y=0). So to put it briefly, a value of 1 was assigned for alive and a value of 0 for dead.

```r
Breast_Cancer["Status"][Breast_Cancer["Status"] == "Alive"] <- "1"
Breast_Cancer["Status"][Breast_Cancer["Status"] == "Dead"] <- "0"
```

Later, used to the as.factor function, it was turned into a categorical data type. In order to check, the class was checked used to the class function.

```r
Breast_Cancer$Status <- as.factor(Breast_Cancer$Status)
class(Breast_Cancer$Status)
```

```
[1] "factor"
```

We checked that the data type of the target variable is correct.

In addition, we convert all features read as characters to the categorical variable type. Then we examine the entire data set with the str function.

```r
Breast_Cancer$Race <- as.factor(Breast_Cancer$Race)
Breast_Cancer$Marital_Status <- as.factor(Breast_Cancer$Marital_Status)
Breast_Cancer$T_Stage <- as.factor(Breast_Cancer$T_Stage)
Breast_Cancer$N_Stage <- as.factor(Breast_Cancer$N_Stage)
Breast_Cancer$Sixth_Stage <- as.factor(Breast_Cancer$Sixth_Stage)
Breast_Cancer$differentiate <- as.factor(Breast_Cancer$differentiate)
Breast_Cancer$Grade <- as.factor(Breast_Cancer$Grade)
Breast_Cancer$A_Stage <- as.factor(Breast_Cancer$A_Stage)
Breast_Cancer$Estrogen_Status <- as.factor(Breast_Cancer$Estrogen_Status)
Breast_Cancer$Progesterone_Status <- as.factor(Breast_Cancer$Progesterone_Status)
str(Breast_Cancer)
```

```
tibble [4,024 x 16] (S3: tbl_df/tbl/data.frame)
 $ Age                     : num [1:4024] 68 50 58 58 47 51 51 40 40 69 ...
 $ Race                    : Factor w/ 3 levels "Black","Other",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ Marital_Status          : Factor w/ 5 levels "Divorced","Married",..: 2 2 1 2 2 4 2 2 1 2 .
 $ T_Stage                 : Factor w/ 4 levels "T1","T2","T3",..: 1 2 3 1 2 1 1 2 4 4 ...
$ N_Stage               : Factor w/ 3 levels "N1","N2","N3": 1 2 3 1 1 1 1 1 3 3 ...
$ Sixth_Stage           : Factor w/ 5 levels "IIA","IIB","IIIA",..: 1 3 5 1 2 1 1 2 5 5 ...
$ differentiate         : Factor w/ 4 levels "Moderately differentiated",..: 2 1 1 2 2 1 4
$ Grade                 : Factor w/ 4 levels "1","2","3","anaplastic; Grade IV": 3 2 2 3 3
$ A_Stage               : Factor w/ 2 levels "Distant","Regional": 2 2 2 2 2 2 2 2 2 1 ...
$ Tumor_Size            : num [1:4024] 4 35 63 18 41 20 8 30 103 32 ...
$ Estrogen_Status       : Factor w/ 2 levels "Negative","Positive": 2 2 2 2 2 2 2 2 2 2 ...
$ Progesterone_Status   : Factor w/ 2 levels "Negative","Positive": 2 2 2 2 2 2 2 2 2 2 ...
$ Regional_Node_Examined: num [1:4024] 24 14 14 2 3 18 11 9 20 21 ...
$ Reginol_Node_Positive : num [1:4024] 1 5 7 1 1 2 1 1 18 12 ...
$ Survival_Months       : num [1:4024] 60 62 75 84 50 89 54 14 70 92 ...
$ Status                : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 2 ...
```

Before training the model, excluded the all NA variables in the dataset.

```
Breast_Cancer <- na.exclude(Breast_Cancer)
```

It is necessary because when we split dataset, some of the classes may not be seen in the train set then the model can not learn anything about these classes. Thus, it is not possible to predict the value of target feature with the predictors(features) that have these unseen classes.

3. Splitting the Data Set(Train and Test Set)

Train/ Test split is used rather than just validating the model on train set, it gives also an estimate how well the model performances on new data.That we should train model a dataset then we check performance of model in a different dataset so far doing this. We can seperate our original dataset to train and test set data. We can do this steps by selecting randomly observation so in practice mostly the number of observation in train dataset much more than test data. The ratio like 80% train data, 20% test data. This dataset was also splitted at this ratio. The split the dataset is important because it allows us to “generalizability” the model.

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

We should split the data set to two different set named train and test set. We will use sample function to create a new object which is index. In this sample we need to use a sequence.

When we create an sequence starting with 1 up to the number of rows of the Breast_Canser data set. then we use the same dataset to split 80% of the dataset.

4. Training a Linear Regression Model

The process to estimate the model coefficients is called model training. This task predict the alive probability of the which have patients breast cancer. So, the glm() function was used to the lr_model object to train the model. Model formula tips: (y ~ .) Here "y" refers to the target variable, dot refers to the features. Since the dot mark is set, this includes all the features. Finally, the family distribution of the target feature. It must be setted as "binomial" in binary logistic regression models.

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

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

Coefficients:
                    (Intercept)                              Age
                      -1.294415                        -0.031739
                      RaceOther                        RaceWhite
                       0.867929                         0.515519
          Marital_StatusMarried          Marital_StatusSeparated
                       0.272381                        -0.620519
           Marital_StatusSingle            Marital_StatusWidowed
                       0.218506                         0.109418
                      T_StageT2                        T_StageT3
                      -0.409576                        -0.967772
                      T_StageT4                        N_StageN2
                      -1.651890                        -0.813027
                      N_StageN3                   Sixth_StageIIB
                      -0.240321                        -0.212427
                Sixth_StageIIIA                  Sixth_StageIIIB
                       0.544258                         0.274990
                Sixth_StageIIIC  differentiatePoorly differentiated
                             NA                        -0.381933
  differentiateUndifferentiated    differentiateWell differentiated
                      -1.758900                         0.726406
                         Grade2                           Grade3
```

```
                               NA                                    NA
         Gradeanaplastic; Grade IV                     A_StageRegional
                               NA                             -0.227727
                       Tumor_Size             Estrogen_StatusPositive
                         0.004066                              0.416672
       Progesterone_StatusPositive             Regional_Node_Examined
                         0.534972                              0.028956
             Reginol_Node_Positive                     Survival_Months
                        -0.086619                              0.063085

Degrees of Freedom: 3218 Total (i.e. Null);  3193 Residual
Null Deviance:      2780
Residual Deviance: 1786     AIC: 1838
```

This code output gives information about the model. The estimated values of the model parameters which are Betas and the trained data. From this code output, it shows which variables are numerical and which variables are categorical.(name + name of the category) summary() function was used to see more details about the model.

```r
summary(lrm_model)
```

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.3413   0.1348   0.2623   0.4676   2.3556

Coefficients: (4 not defined because of singularities)
                                   Estimate Std. Error z value Pr(>|z|)
(Intercept)                       -1.294415   0.659200  -1.964  0.04957 *
Age                               -0.031739   0.007393  -4.293 1.76e-05 ***
RaceOther                          0.867929   0.318963   2.721  0.00651 **
RaceWhite                          0.515519   0.207703   2.482  0.01306 *
Marital_StatusMarried              0.272381   0.185532   1.468  0.14208
Marital_StatusSeparated           -0.620519   0.529481  -1.172  0.24122
Marital_StatusSingle               0.218506   0.229882   0.951  0.34185
Marital_StatusWidowed              0.109418   0.287810   0.380  0.70382
T_StageT2                         -0.409576   0.267279  -1.532  0.12543
T_StageT3                         -0.967772   0.421580  -2.296  0.02170 *
```

```
T_StageT4                          -1.651890   0.654122  -2.525  0.01156 *
N_StageN2                          -0.813027   0.322273  -2.523  0.01164 *
N_StageN3                          -0.240321   0.410256  -0.586  0.55802
Sixth_StageIIB                     -0.212427   0.307610  -0.691  0.48983
Sixth_StageIIIA                     0.544258   0.396693   1.372  0.17007
Sixth_StageIIIB                     0.274990   0.759259   0.362  0.71722
Sixth_StageIIIC                           NA         NA      NA       NA
differentiatePoorly differentiated -0.381933   0.138259  -2.762  0.00574 **
differentiateUndifferentiated      -1.758900   0.845417  -2.081  0.03748 *
differentiateWell differentiated    0.726406   0.238369   3.047  0.00231 **
Grade2                                    NA         NA      NA       NA
Grade3                                    NA         NA      NA       NA
Gradeanaplastic; Grade IV                 NA         NA      NA       NA
A_StageRegional                    -0.227727   0.363445  -0.627  0.53094
Tumor_Size                          0.004066   0.005289   0.769  0.44207
Estrogen_StatusPositive             0.416672   0.258087   1.614  0.10643
Progesterone_StatusPositive         0.534972   0.171746   3.115  0.00184 **
Regional_Node_Examined              0.028956   0.008816   3.284  0.00102 **
Reginol_Node_Positive              -0.086619   0.020392  -4.248 2.16e-05 ***
Survival_Months                     0.063085   0.003146  20.051  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2780.2  on 3218  degrees of freedom
Residual deviance: 1786.4  on 3193  degrees of freedom
AIC: 1838.4

Number of Fisher Scoring iterations: 6
```

We can see some significant statistics for the features. If any class of a categorical variable is significant which means the p-value is lower than the significance level we can evaluate the whole level of categorical variables is significant. For example in here, two different classes of race categorical variable two of them significant but even if only one of them was significant again, in this case, too it means that race feature significant. It has an important contribution to the target variable. It is same for any other categorical variable in here. We can use the AIC value to compare the models.

Coefficients shows us the logs odds ratios of the coefficient. We can take the odds ratios of features for doing this. We can take the exponential of them by using the exp() function.

```
exp(lrm_model$coefficients)
```

```
                     (Intercept)                                 Age 
                       0.2740580                           0.9687596 
                       RaceOther                           RaceWhite 
                       2.3819719                           1.6745066 
           Marital_StatusMarried             Marital_StatusSeparated 
                       1.3130875                           0.5376655 
            Marital_StatusSingle               Marital_StatusWidowed 
                       1.2442167                           1.1156289 
                       T_StageT2                           T_StageT3 
                       0.6639317                           0.3799284 
                       T_StageT4                           N_StageN2 
                       0.1916873                           0.4435134 
                       N_StageN3                     Sixth_StageIIB 
                       0.7863758                           0.8086190 
                 Sixth_StageIIIA                     Sixth_StageIIIB 
                       1.7233285                           1.3165177 
                 Sixth_StageIIIC differentiatePoorly differentiated 
                              NA                           0.6825409 
    differentiateUndifferentiated   differentiateWell differentiated 
                       0.1722342                           2.0676356 
                          Grade2                              Grade3 
                              NA                                  NA 
        Gradeanaplastic; Grade IV                     A_StageRegional 
                              NA                           0.7963416 
                      Tumor_Size             Estrogen_StatusPositive 
                       1.0040741                           1.5169043 
    Progesterone_StatusPositive              Regional_Node_Examined 
                       1.7074013                           1.0293791 
           Reginol_Node_Positive                     Survival_Months 
                       0.9170264                           1.0651176 
```

We can see the odds ratios of the features of glm model. If we are going to look at how to interpret it, let's interpret one of them as an example. Focus on the Age, ıt means that you are familiar from linear regression models. If we increase the value of age by one units it is contribution is point 0.96 to the target feature but in here for odds ratios we can interpret them then we increase the age one unit it contributes the odds ratio of the target variable 0.96 times so it means that we the log multiply the odds ratio of the target variable by 0.96. If we multiply any value by another value which is lower than one, it causes the lower in the calculation. In here we can say that, if the odds ratio value any feature is lower than one, it is contribution is negative. Another way of say, if age of patients increases their odds ratio of alive status decreasing. It means that, if the age of patients increases their alive probability decreases. Likewise, if the odds ratio value any feature is higher than one, it is contribution is positive. We can interpret this kind coefficients in logistic regression model.

5. Performance of the Trained model

It is necessary to check the model performance of the model on test set. Because we are interested in train such a model has a good generalizability. To do this, we first calculated the predicted values of the target variable on test set. There are some additional steps different from the regression task. In here, to predict the value of target variable, we use additional argument. It is type when we set the type as response we can calculate the probability of the target variable or predicted probabilities of target variable. We assign predicted_probs to predict underscore props object. Exclude the true values of target variable from test set.

```
predicted_probs <- predict(lrm_model, test[ , -16], type = "response")
head(predicted_probs)
```

```
        1         2         3         4         5         6 
0.8432563 0.9899364 0.6367093 0.7404053 0.9491666 0.6107392 
```

We can see that this is are whole probability values because it differs from 0 and 1.

Then we should convert this probability values to the values of classes. There are two classes in our target variable these are “alive” or “dead”. Convert them one and zero. One is alive, zero is dead. We can use ifelse() function for doing this. So, first argument is a condition. It means that if the values of predicted props object is higher than 0.5 let’s assign it as one. If otherwise assign in as zero.

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

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

Measuring the Model Performance

There is a binary classification task and the aim of the model is to predict the value of or classes of target variable correctly possible. Confusion Matrix, two different part; observed class and predicted class. Classes of the model of target variable are positive and negative class. Model assign any positive values as positive this is a kind of correct decision and we call the number of observation this cell “True Positive” (TP), if the observed class of the target variable is negative then model may label this observation like positive so we call this cell “False Positive” (FP), ıf the observed class of the target variable is positive then the model may label them as negative so this is wrong decision so we called this cell as “False Negative” (FN) and last possibility is observed class of target variable may be negative and model may assign these observation as negative so this correct decision we call this cell is “True Negative” . The following assignments were made in this way. The metrics is calculated based on these four cell.

```
TP <- sum(predicted_classes[which(test$Status == "1")] == 1)
FP <- sum(predicted_classes[which(test$Status == "1")] == 0)
TN <- sum(predicted_classes[which(test$Status == "0")] == 0)
FN <- sum(predicted_classes[which(test$Status == "0")] == 1)

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

recall
```

```
[1] 0.9102041
```

```
specificity
```

```
[1] 0.7142857
```

```
precision
```

```
[1] 0.9709724
```

```
accuracy
```

```
[1] 0.8931677
```

Some information about these metrics. Recall addresses the question: “Among all the true positives, how many of them are indeed correctly captured by the model?” In here, 91 percent. Specificity, It tells us the proportion of correctly identified negative labels (TN) among all the negative labels (TN + FP). In here, 71 percent. Precision addresses the question: How many percent of the positive class is classified correctly? In here, we can say that 97 percent of the right patients in the test set is classified correctly. Accuracy is important metric. Because it means that 89 percent of the observation is classified correctly by the model. Looking at just one metric can sometimes be misleading, so it’s also useful to examine most of them. Although these metric values look good, it will be checked whether there is an imbalance problem in the future.

The following steps to calculate the metrics are not user-friendly. So, we use confusionMatrix() function in the {caret} package. This function requires an obligatory an an optional argument that we should check its value carefully. First one is a table that contains the observed and predicted class of target variable, and the second is positive class label.

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

```
Confusion Matrix and Statistics

   predicted_classes
      0   1
  0  50  66
  1  20 669

               Accuracy : 0.8932
                 95% CI : (0.8697, 0.9137)
    No Information Rate : 0.913
    P-Value [Acc > NIR] : 0.978

                  Kappa : 0.4814

 Mcnemar's Test P-Value : 1.219e-06

            Sensitivity : 0.9102
            Specificity : 0.7143
         Pos Pred Value : 0.9710
         Neg Pred Value : 0.4310
             Prevalence : 0.9130
         Detection Rate : 0.8311
   Detection Prevalence : 0.8559
      Balanced Accuracy : 0.8122
```

```
        'Positive' Class : 1
```

In the output of the function, we can see some new statistics. The most commonly used one is Balanced Accuracy. Balanced Accuracy (average of accuracy), ıt's the arithmetic mean of sensitivity and specificity, its use case is when dealing with imbalanced data, when one of the target classes appears a lot more than the other. The reason why it is important is that if the Balanced Accuracy value is lower than the accuracy value, it allows us to make the comment that there is an imbalanced problem here. Then we say that there is an imbalanced problem here for this task. This situation will be clearly checked in the future.

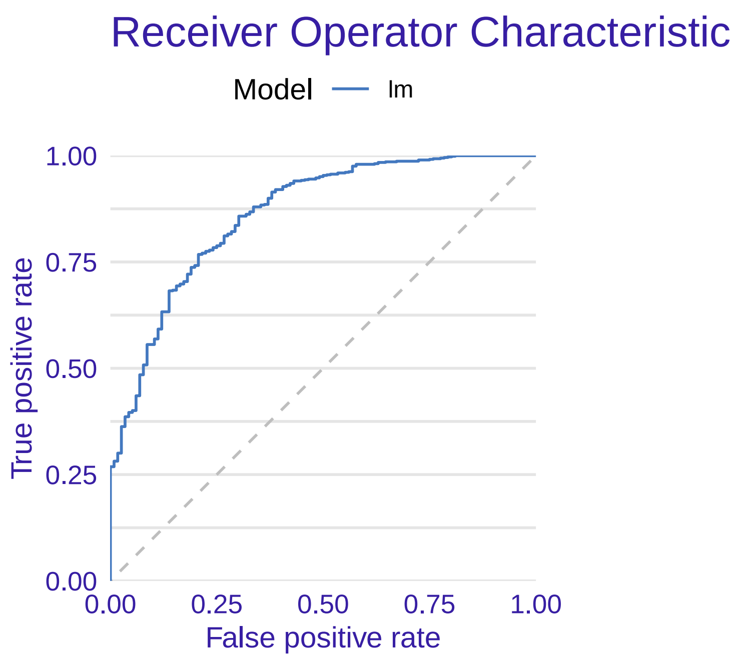
Any model performance metrics in classification is depend on the cutoff value (c) to assign a class to observation. In logistic regression models we predict the probability of intended class then we convert it to class label. For example, when we use logistic regression model we predict the probability of an intended class. So, then we should convert this probability to class label. So, in this measuring step we use 0.5 because we must define a cutoff value in practice. So, the result of this metrics is heavily depend on the value of cutoff value. The model performance can be manipulated by fixing the values of c, if we use only this 4 metrics. So, we use some alternative metrics. ROC curve gives us more detailed information about the model performances which is "independently" from the value of c. Area Under the Curve metrics gives us the area under the ROC curve.

We use explain an object in DALEX package.

```r
explain_lr <- explain(model = lrm_model, #trained model
                      data = test[ , -16], #test set without target
                      y = test$Status == "1", #observed values of target
                      type = "classification", #type of task
                      verbose = FALSE) #remove some messages
```

Then, we can use model_performance() function from {DALEX} package with explainer object that we created above to draw ROC curves and some others.

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



ROC curve getting closer to the left top, we can say that model performance is going to perfect. At the moment, the model looks good, but it is not perfect.

To calculate area under the curve(AUC) value, just print the performance_lr object.

```
performance_lr
```

```
Measures for:  classification
recall      : 0.9709724
precision   : 0.9102041
f1          : 0.9396067
accuracy    : 0.8931677
auc         : 0.8623943

Residuals:
          0%          10%          20%          30%          40%          50% 
-0.982202733 -0.302637308  0.006436929  0.011499261  0.020552443  0.035012123 
         60%          70%          80%          90%         100% 
 0.056762992  0.088278876  0.139532283  0.242106512  0.866188429 
```

It is about 0.86 means that the AUC equals to 0.86 because the axes range between 0 and 1. Thus maximum value of AUC can be 1. In here, the model performance is looks good.

6. Checking the Imbalance Problem

We must check the imbalancedness of the classes in the target feature.

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

```
        0         1 
0.1553277 0.8446723 
```

There is a imbalanced in these two classes are not equal, classes of the target feature is not balanced. Thus, it may cause that the model can learn less from minority class, so the model could not achieve a satisfying performance on classifying of the minority class. In that case, some solution ways can be performed based on the balancing of imbalanced classes of the target feature.

7. Under-sampling Methods to Balance the Data Set

Two solutions way imbalanced problem; - Decrease the number of observation in majority class. - Increase the number of observation in minority class.

We use the first solution in here. We use downSample() function in {caret} package.

```r
down_train <- downSample(x = train[, -ncol(train)],
                         y = train$Status)
table(down_train$Class)
```

```
  0   1 
500 500 
```

In here, in these two classes are equal. (500-500)

We use the second solution in here. We use upSample() function in {caret} package. It is possible to obtain the ROC graph using the same codes. But here we will make a comparison using it, since the first method gives a better result.

```r
up_train <- upSample(x = train[, -ncol(train)],
                     y = train$Status)
```

```r
table(up_train$Class)
```

```
   0    1 
2719 2719 
```

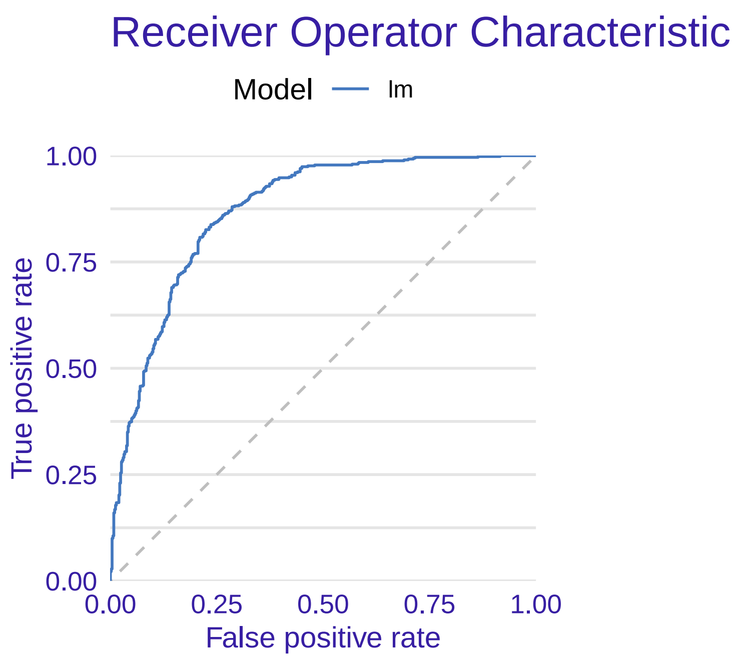
In here, in these two classes are equal.(2719-2719)

8. Compare the Performance of the Models

If we have or we train two different kind of model or we train two logistic regression model with different features we can compare their performances by using "AUC". The higher AUC indicates higher performance.

```r
explain_lrd <- explain(model = lrm_model,data = down_train[, -16],
                       y = down_train$Class == "1",
                       type = "classification",
                       verbose = FALSE)
```

```r
performance_d <- model_performance(explain_lrd)
plot(performance_d, geom = "roc")
```



We see that the curve of the ROC graph of the balanced data set is getting closer to the top left. The imbalanced one is a little further away in the data set. This means that the ROC graph of the balanced data set is better, so we can say that the model performance is approaching perfection.

```
performance_d
```

```
Measures for:  classification
recall     : 0.978
precision  : 0.6400524
f1         : 0.7737342
accuracy   : 0.714
auc        : 0.870092

Residuals:
          0%          10%          20%          30%          40%          50% 
-0.996235152 -0.863983086 -0.678415561 -0.411516090 -0.155747212 -0.001130646 
         60%          70%          80%          90%         100% 
 0.014720768  0.034983570  0.073933681  0.151601406  0.937620151 
```

In the imbalanced data set, the accuracy value of the model was 0.89. In the balanced data set, this value was 0.71. This is proof that the value of accuracy can manipulate us. In the imbalanced data set, the AUC value of the model was 0.86. In the balanced data set, this value was 0.87. This indicates that the model is approaching perfection and showing improvement.