Breast Cancer Diagnosis Prediction using Logistic Regression

Sercan Öncü

8/4/23

```
options(repos = list(CRAN="http://cran.rstudio.com/"))
install.packages("tidyverse")
install.packages("caret")
install.packages("dplyr")
install.packages("DALEX")
install.packages("ROSE")
install.packages("rpart")
library(rpart)
library(ROSE)
library(DALEX)
library(dplyr)
library(tidyverse)
library(caret)
```

```
library(readr)
data <- read_csv("C:/Users/serca/Downloads/archive (3)/Breast_Cancer.csv")
```

Problem, Features, and Target

The dataset is related to breast cancer diagnosis, and it includes features such as Age, Race, Marital Status, Tumor Size, T Stage, N Stage, 6th Stage, Estrogen Status, Progesterone Status, and Survival Months. The target variable is Status, which indicates whether the patient is alive or dead after the treatment. The aim of the analysis is to build a model to predict the survival status of patients based on the given features.

Terms of the dimension, variable type

The dataset includes 4024 observations and 16 variables in total, including 10 character variables and 6 numeric(double) variables.

```
glimpse(data)
```

```
Rows: 4,024
Columns: 16
$ Age                      <dbl> 68, 50, 58, 58, 47, 51, 51, 40, 40, 69, 68, 4~
$ Race                     <chr> "White", "White", "White", "White", "White", ~
$ `Marital Status`         <chr> "Married", "Married", "Divorced", "Married", ~
$ `T Stage`                <chr> "T1", "T2", "T3", "T1", "T2", "T1", "T1", "T2~
$ `N Stage`                <chr> "N1", "N2", "N3", "N1", "N1", "N1", "N1", "N1~
$ `6th Stage`              <chr> "IIA", "IIIA", "IIIC", "IIA", "IIB", "IIA", "~
$ differentiate            <chr> "Poorly differentiated", "Moderately differen~
$ Grade                    <chr> "3", "2", "2", "3", "3", "2", "1", "2", "3", ~
$ `A Stage`                <chr> "Regional", "Regional", "Regional", "Regional~
$ `Tumor Size`             <dbl> 4, 35, 63, 18, 41, 20, 8, 30, 103, 32, 13, 59~
$ `Estrogen Status`        <chr> "Positive", "Positive", "Positive", "Positive~
$ `Progesterone Status`    <chr> "Positive", "Positive", "Positive", "Positive~
$ `Regional Node Examined` <dbl> 24, 14, 14, 2, 3, 18, 11, 9, 20, 21, 9, 11, 1~
$ `Reginol Node Positive`  <dbl> 1, 5, 7, 1, 1, 2, 1, 1, 18, 12, 1, 3, 3, 7, 1~
$ `Survival Months`        <dbl> 60, 62, 75, 84, 50, 89, 54, 14, 70, 92, 64, 9~
$ Status                   <chr> "Alive", "Alive", "Alive", "Alive", "Alive", ~
```

In this code, the value of "Alive" in the "Status" variable was replaced with 0, and the value of "Dead" was replaced with 1. This operation was performed to convert the "Status" variable in the dataset into a binary format.

```
data$Status <- ifelse(data$Status == "Alive", 0, 1)
```

Train a logistic regression model

This code sets a random seed, then samples 80% of the data to be used for training by selecting a random set of row indices. The remaining 20% of the data is assigned to the test dataset.

```
set.seed(123)
index <- sample(1 : nrow(data), round(nrow(data) * 0.80))
```

```r
train <- data[index, ]
test  <- data[-index, ]
```

The following code creates a logistic regression model using the 'glm' function. The target variable is 'Status', and all other variables in the 'train' dataset are used as predictors. The model uses the binomial family since the target variable is binary.

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

```r
summary(model)
```

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

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

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 Status`Married          -0.272381   0.185532  -1.468  0.14208
`Marital Status`Separated         0.620519   0.529481   1.172  0.24122
`Marital Status`Single           -0.218506   0.229882  -0.951  0.34185
`Marital Status`Widowed          -0.109418   0.287810  -0.380  0.70382
`T Stage`T2                       0.409576   0.267279   1.532  0.12543
`T Stage`T3                       0.967772   0.421580   2.296  0.02170 *
`T Stage`T4                       1.651890   0.654122   2.525  0.01156 *
`N Stage`N2                       0.813027   0.322273   2.523  0.01164 *
`N Stage`N3                       0.240321   0.410256   0.586  0.55802
`6th Stage`IIB                    0.212427   0.307610   0.691  0.48983
`6th Stage`IIIA                  -0.544258   0.396693  -1.372  0.17007
`6th Stage`IIIB                  -0.274990   0.759259  -0.362  0.71722
`6th Stage`IIIC                         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 Stage`Regional                   0.227727   0.363445   0.627  0.53094
`Tumor Size`                       -0.004066   0.005289  -0.769  0.44207
`Estrogen Status`Positive          -0.416672   0.258087  -1.614  0.10643
`Progesterone Status`Positive      -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
```

The model's fit was evaluated using the null deviance (2780.2 with 3218 degrees of freedom) and the residual deviance (1786.4 with 3193 degrees of freedom). A lower residual deviance indicates a better fit. The AIC value (1838.4) can be used to compare the quality of different models, with lower AIC values indicating better models.

Report the performance of the trained model

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

```
        1         2         3         4         5         6
0.15674374 0.01006363 0.36329066 0.25959466 0.05083337 0.38926082
```

```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
```

The code above generates predicted probabilities of the test set using the logistic regression model that was previously built. The 'predict' function is used with the argument 'type = "response"'. The resulting output is a vector containing the predicted probabilities for each observation in the test set. The second part of the code creates a vector of predicted classes by applying a threshold of 0.5 to the predicted probabilities. If the predicted probability is greater than 0.5, the class is set to 1, otherwise it is set to 0. This threshold can be adjusted depending on the specific needs of the analysis.

```
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] 0.7142857
```

```
specificity
```

```
[1] 0.9102041
```

```
precision
```

```
[1] 0.4310345
```

```
accuracy
```

```
[1] 0.8931677
```

The model's performance can be summarized as follows: The recall (sensitivity) is 0.7143, indicating that 71.43% of the actual positive cases are correctly identified by the model. The specificity is 0.9102, meaning that 91.02% of the actual negative cases are correctly classified. The precision is 0.4310, suggesting that 43.10% of the predicted positive cases are truly positive. Finally, the overall accuracy is 0.8932, which shows that 89.32% of all cases are correctly classified by the model

Check the imbalance problem

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

```
        0         1
0.8446723 0.1553277
```

The class distribution in the training dataset is imbalanced, with approximately 84.47% of examples belonging to the negative class and 15.53% belonging to the positive class. This can cause issues with model evaluation metrics, as the model may be biased towards the larger class. Resampling techniques such as oversampling or undersampling can be used to address this problem.

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

```
Confusion Matrix and Statistics

   predicted_classes
      0   1
  0 669  20
  1  66  50

               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 : 0.71429
            Specificity : 0.91020
         Pos Pred Value : 0.43103
         Neg Pred Value : 0.97097
             Prevalence : 0.08696
         Detection Rate : 0.06211
```

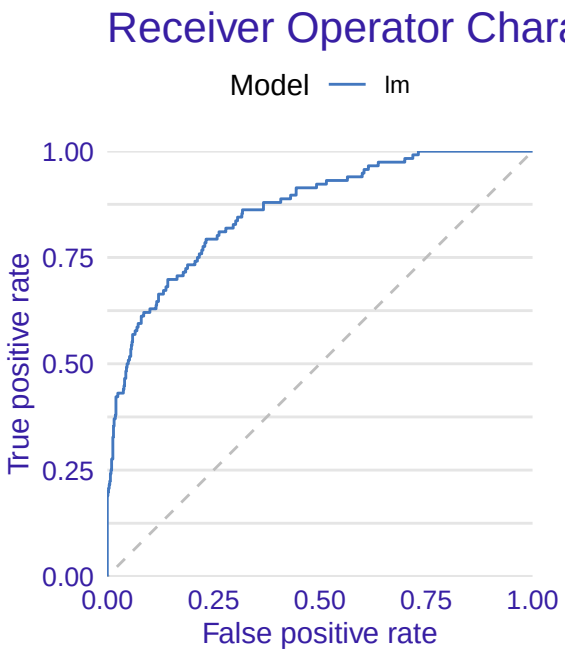
```
Detection Prevalence : 0.14410
   Balanced Accuracy : 0.81224

     'Positive' Class : 1
```

Model evaluation metrics for the test dataset are as follows: Accuracy = 0.8932 (proportion of correct predictions), Kappa = 0.4814 (an index measuring classification accuracy ranging between 0 and 1, with higher values indicating better performance), Sensitivity/Recall = 0.71429 (proportion of true positives to the sum of true positives and false negatives), Specificity = 0.91020 (proportion of true negatives to the sum of true n egatives and false positives), Precision/Pos Pred Value = 0.43103 (proportion of true positives to the sum of true positives and false positives), Neg Pred Value = 0.97097 (proportion of true negatives to the sum of true negatives and false negatives), Balanced Accuracy = 0.81224 (average of sensitivity and specificity). Overall, the model's performance on the test dataset appears to be good, with high accuracy, specificity, and negative predictive value. However, the sensitivity and precision values are lower, which is often seen in imbalanced datasets with a low number of positive examples. The accuracy of the model can be improved, but it is important to consider sensitivity and precision values as well.

```r
explain_lr <- DALEX::explain(model   = model,
                      data    = test[, -16],
                      y       = test$Status == "1",
                      type    = "classification",
                      verbose = FALSE)
```

The above code provides an explanation of the logistic regression model's performance on the test dataset, excluding the 16th column. The explanation is for a classification problem

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




```
performance_lr
```

```
Measures for:  classification
recall     : 0.4310345
precision  : 0.7142857
f1         : 0.5376344
accuracy   : 0.8931677
auc        : 0.8623943

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

Recall: 0.4310345 - The proportion of true positive predictions made by the model. Precision: 0.7142857 - The proportion of positive predictions that are actually true positives. F1 Score: 0.5376344 - The harmonic mean of precision and recall, used to measure the model's balanced performance. Accuracy: 0.8931677 - The proportion of correct predictions made by the model. AUC (Area Under Curve): 0.8623943 - The area under the Receiver Operating Characteristic (ROC) curve, used to measure the model's classification performance (values closer to 1 indicate better performance).

OVER AND UNDER SAMPLING

```
table(train$Status)
```

```
   0    1
2719  500
```

Over Sampling

```
data_balanced_over <- ovun.sample(Status ~ ., data = train, method = "over",N = 5438)$data
table(data_balanced_over$Status)
```

```
   0    1
2719 2719
```

Under Sampling

```
data_balanced_under <- ovun.sample(Status ~ ., data = train, method = "under", N = 1000, s
table(data_balanced_under$Status)
```

```
  0   1
500 500
```

Both undersampling and oversampling on this imbalanced data

```
data_balanced_both <- ovun.sample(Status ~ ., data = train, method = "both", p=0.5, N=1000
table(data_balanced_both $Status)
```

```
  0   1
520 480
```

```
data.rose <- ROSE(Status ~ Age  + Tumor_size + Regional_node_examined +
                    Regional_node_pos + survival_months,
                  data = train, seed = 1)$data

table(data.rose$Status)
```

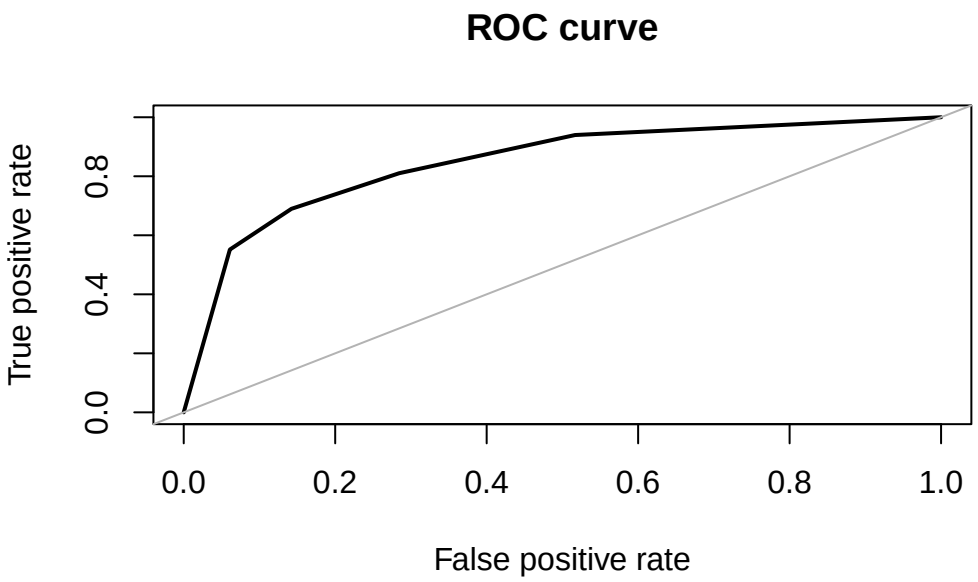
```
   0    1
1668 1551
```

```
tree.rose <- rpart(Status ~ ., data = data.rose)
tree.over <- rpart(Status ~ ., data = data_balanced_over)
tree.under <- rpart(Status ~ ., data = data_balanced_under)
tree.both <- rpart(Status ~ ., data = data_balanced_both)

pred.tree.rose <- predict(tree.rose, newdata = test)
pred.tree.over <- predict(tree.over, newdata = test)
pred.tree.under <- predict(tree.under, newdata = test)
pred.tree.both <- predict(tree.both, newdata = test)
```

Evaluate the accuracy of respective predictions

Over Sampling ROC CURVE AND AUC

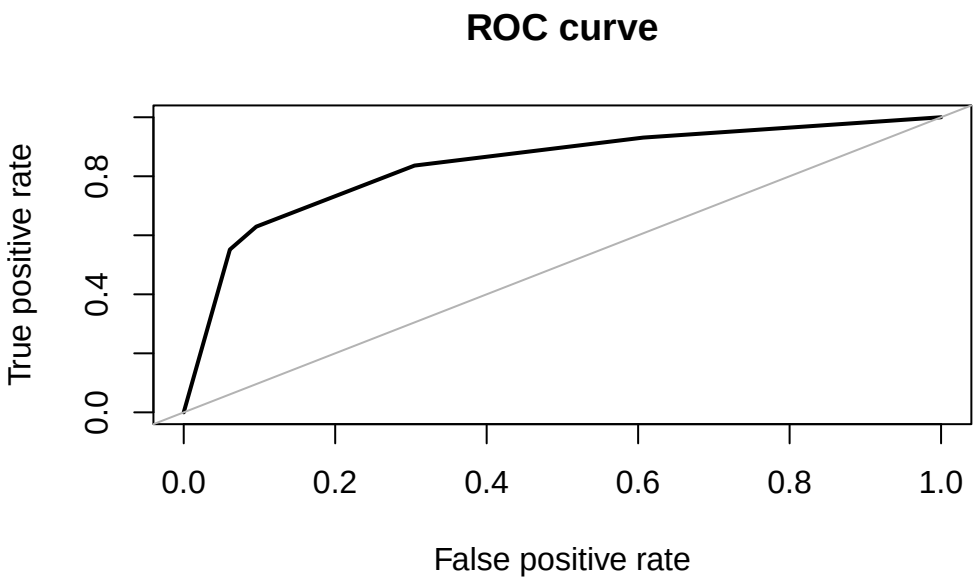
```
roc.curve(test$Status, pred.tree.over)
```




```
Area under the curve (AUC): 0.846
```

Under Sampling ROC CURVE AND AUC

```
roc.curve(test$Status, pred.tree.under)
```




```
Area under the curve (AUC): 0.837
```

Following the operations, the AUC value did not show much difference.