Predicting Credit Card Application Acceptance using Logistic Regression

Melih Gündüz / 41476216168

2024-03-24

```r
install.packages("AER")
install.packages("DALEX")
install.packages("car")
install.packages("caret")
install.packages("DescTools")
library(DescTools)
library(caret)
library(car)
library(DALEX)
library(AER)
```

1. Problem, Features, and Target

The dataset is related to credit card application acceptance. It includes features such as Card, Reports, Age, Income, Share, Expenditure, Owner, Selfemp, Dependents, Months, Majorcards, Active. The target variable is Card, which indicates whether the card applications are accepted after the application. The aim of the analysis is to build a model to predict whether applicants have cards based on the given features.

2. Dataset

The dataset has a total of 1319 observations and 12 variables. Among these, 9 are numeric variables and 3 are factor variables.

```r
data(CreditCard, data=AER)
str(CreditCard)
```

```
'data.frame':   1319 obs. of  12 variables:
 $ card       : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ reports    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ age        : num  37.7 33.2 33.7 30.5 32.2 ...
 $ income     : num  4.52 2.42 4.5 2.54 9.79 ...
 $ share      : num  0.03327 0.00522 0.00416 0.06521 0.06705 ...
 $ expenditure: num  124.98 9.85 15 137.87 546.5 ...
 $ owner      : Factor w/ 2 levels "no","yes": 2 1 2 1 2 1 1 2 2 1 ...
 $ selfemp    : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ dependents : num  3 3 4 0 2 0 2 0 0 0 ...
 $ months     : num  54 34 58 25 64 54 7 77 97 65 ...
 $ majorcards : num  1 1 1 1 1 1 1 1 1 1 ...
 $ active     : num  12 13 5 7 5 1 5 3 6 18 ...
```

```r
CreditCard$card <- ifelse(CreditCard$card == "yes", 1, 0)
CreditCard$owner <- ifelse(CreditCard$owner == "yes", 1, 0)
CreditCard$selfemp <- ifelse(CreditCard$selfemp == "yes", 1, 0)
```

The variables ‘card,’ ‘owner,’ and ‘selfemp’ were changed from factor to numeric variables. Their conditions were changed from ‘yes’ and ‘no’ to ‘1’ and ‘0,’ respectively.

3. Splitting the data set

It involves understanding the relationships of the parameters of the model in the dataset. It aims to predict the target variable and divides this dataset into two subsets, ‘train’ and ‘test’, according to 80% and 20%, respectively.

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

4. Train a logistic regression model

This code creates a logistic regression model using the ‘glm’ function. The target variable is used as predictors, and all other variables from the ‘train’ dataset are used as predictors. We use the binomial distribution as the model’s distribution.

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

```

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

Coefficients:
              Estimate Std. Error    z value Pr(>|z|)    
(Intercept)  4.787e+14  9.348e+06   51206972   <2e-16 ***
reports     -5.371e+14  1.606e+06 -334428307   <2e-16 ***
age          3.811e+12  2.446e+05   15578027   <2e-16 ***
income       4.829e+12  1.726e+06    2798188   <2e-16 ***
share        1.735e+16  4.738e+07  366212379   <2e-16 ***
expenditure -9.773e+11  1.694e+04  -57702295   <2e-16 ***
owner        3.688e+13  4.892e+06    7539631   <2e-16 ***
selfemp      1.781e+14  8.253e+06   21585985   <2e-16 ***
dependents  -2.570e+13  1.835e+06  -14005858   <2e-16 ***
months      -1.883e+11  3.557e+04   -5294713   <2e-16 ***
majorcards  -9.403e+12  5.447e+06   -1726345   <2e-16 ***
active      -1.254e+12  3.663e+05   -3423872   <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:  1121.6  on 1054  degrees of freedom
Residual deviance: 10885.2  on 1043  degrees of freedom
AIC: 10909

Number of Fisher Scoring iterations: 25
```

The model’s fit yielded a null deviance of 1121.6 with 1054 degrees of freedom, and a residual deviance of 10885.2 with 1043 degrees of freedom. The AIC value is 10909. AIC allows us to compare the quality of different models.

5. Measuring model performance

It predicts the probability of credit card application acceptance using a logistic regression model (lr_model) with features from the test dataset.

```r
predicted_probs <- predict(lr_model, test[,-1])
head(predicted_probs)
```

```
          14           15           21           22           33           43 
1.215334e+15 1.004421e+15 9.236412e+14 2.481678e+12 4.639263e+13 1.398705e+15 
```

It includes predicting the probability of credit card application acceptance for specific observations in the test dataset.

```r
error <- test$card - predicted_probs
mse <- mean(error^2)
rmse <- sqrt(mean(error^2))
mae <- median(abs(error))
head(error)
```

```
           14            15            21            22            33 
-1.215334e+15 -1.004421e+15 -9.236412e+14 -2.481678e+12 -4.639263e+13 
           43 
-1.398705e+15 
```

These metrics are used to measure how much the predictions of your model differ from the actual values. Metrics such as mean squared error (MSE), root mean squared error (RMSE), and median absolute error (MAE) can help evaluate the success of your model. While MSE and RMSE emphasize larger errors, MAE provides a more balanced measure and is more resistant to outliers. The combination of these metrics can help you understand your model's performance more comprehensively.

```r
mse
```

```
[1] 4.893511e+30
```

```r
rmse
```

```
[1] 2.212128e+15
```

```
mae
```

```
[1] 1.242154e+15
```

The MSE value (4.893511e+30) indicates the average of the squared errors, and when it is high, the model's predictions deviate significantly from the actual values. The RMSE value (2.212128e+15) represents the root mean square error, indicating the average error between predicted and actual values. A high RMSE value indicates poor model performance. The MAE value (1.242154e+15) shows the median of the absolute errors, measuring how much the model's predictions differ from the actual values. A large MAE value helps in improving the model.

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

```
14 15 21 22 33 43
 1  1  1  1  1  1
```

The 'ifelse' code assigns values to a new variable and sets the definition intervals. If the values are between 0 and 1, observations with values between '0-0.5' are rejected, and observations with values between '0.5-1' are not rejected.

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

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

```
recall
```

```
[1] 0.8451883
```

```
specificity
```

```
[1] 0.92
```

```
precision
```

```
[1] 0.9901961
```

```
accuracy
```

```
[1] 0.8522727
```

The performance of this model is as follows: The recall value (0.8451883) is observed, indicating a 84.5% rate of correctly predicting positives. The specificity value (0.92) is observed, indicating a 92% rate of correctly predicting negatives. The precision value (0.9901961) is observed, indicating a 99.02% rate of correctly predicting positives among predicted positives. The accuracy value (0.8522727) is observed, indicating the rate of correctly classifying all observations.

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

```

        0         1 
0.2236967 0.7763033 
```

The 'card' variable in the dataset represents class distributions as percentages. The "1" class represents 77.63%, while the "0" class represents 22.37%. This distribution indicates the imbalance in class distribution. Consequently, the model's performance in correctly predicting the minority class may be low due to this imbalance.

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

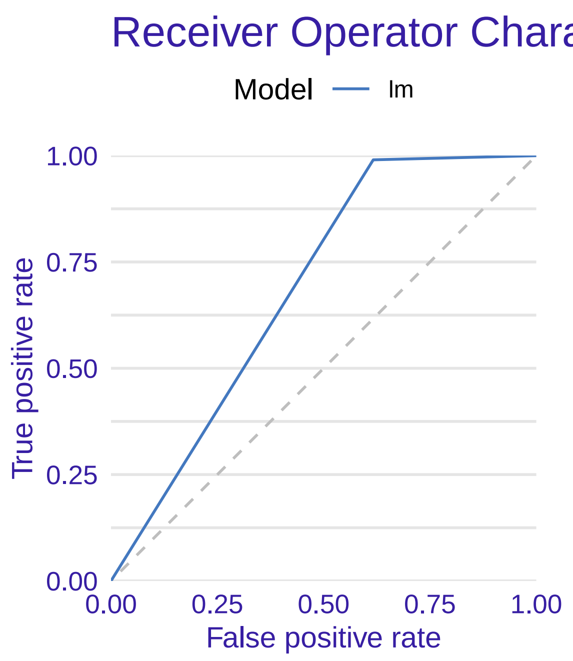
```
Confusion Matrix and Statistics

   predicted_classes
      0   1
  0  23  37
  1   2 202
                                          
               Accuracy : 0.8523          
                 95% CI : (0.8036, 0.8928)
```

```
    No Information Rate : 0.9053
    P-Value [Acc > NIR] : 0.9979

                  Kappa : 0.4704

 Mcnemar's Test P-Value : 5.199e-08

            Sensitivity : 0.8452
            Specificity : 0.9200
         Pos Pred Value : 0.9902
         Neg Pred Value : 0.3833
             Prevalence : 0.9053
         Detection Rate : 0.7652
   Detection Prevalence : 0.7727
      Balanced Accuracy : 0.8826

       'Positive' Class : 1
```

The results of model evaluation on the test dataset are as follows: Accuracy (0.8523), Kappa (0.4704), McNemar's Test P-Value (5.199e-08), Sensitivity (0.8452), Specificity (0.9200), Positive Predictive Value (0.9902), Negative Predictive Value (0.3833), Prevalence (0.9053), Detection Rate (0.7652), Detection Prevalence (0.7727), Balanced Accuracy (0.8826). Overall, it performs well in terms of performance.

6. ROC Curve

```r
explain_lr <- explain(model = lr_model,
                      data = test[, -1],
                      y=test$card==1,
                      type = "classification",
                      verbose = FALSE)
performance_lr <- model_performance(explain_lr)
plot(performance_lr, geom="roc")
```



```
performance_lr
```

```
Measures for:  classification
recall      : 0.9901961
precision   : 0.8451883
f1          : 0.9119639
accuracy    : 0.8522727
auc         : 0.6867647

Residuals:
           0%            10%            20%            30%            40% 
-1.000000e+00 -1.000000e+00 -2.220446e-16  2.220446e-16  2.220446e-16 
          50%            60%            70%            80%            90% 
 2.220446e-16  2.220446e-16  2.220446e-16  2.220446e-16  2.220446e-16 
         100% 
 1.000000e+00 
```

Recall (0.9901961) is the rate of correctly predicting positives. Precision (0.9901961) is the rate of true positives among predicted positives. F1 (0.9119639) is the harmonic mean of Recall and Precision, summarizing the model's classification performance in a single metric. Accuracy (0.8522727) is the rate of correctly classifying all observations. AUC (0.6867647) represents the area under the ROC curve and is used to measure the model's prediction performance. As the AUC value approaches 1, the model's performance improves. The Residuals section shows the residuals of the model's predictions, and generally, the residuals appear to have low values.

7. Brier Model

```
BrierScore(lr_model)
```

```
[1] 0.143128
```

The Brier Score is calculated as 0.143128. Therefore, a low Brier Score value indicates that the model's probability predictions are close to the actual values, demonstrating good performance.