stat_model

Süleyman Erim, Giacomo Schiavo, Mattia Varagnolo

2023-05-18

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
# import libraries
library(MASS) #for stepAIC (stepwise)
library(tidyverse) # for data manipulation
library(corrplot) # for correlation plot
library(ggplot2) # for plotting
library(gridExtra) # for grid.arrange
library(correlation) # for partial correlation
library(stats) # for anova
library(car) # for vif package
library(ROCR) # for roc curve
library(ROSE) # for oversampling (just in case)
library(leaps) # for stepwise selection
library(regclass) # for VIF package
library(MLmetrics) # to create confusion matrix
library(pROC) # for ROC Curve
library(e1071) # for Naive Bayes Classifier
library(rpart) # for Decision Tree
library(randomForest) # for Random Forest
```

```
#read data
data = read.csv("data_encoded.csv")
#drop X column
data = data %>% select(-X)
attach(data)
```

```
#train test split
set.seed(123)
train_index = sample(1:nrow(data), 0.8*nrow(data))
# 80% of data is used for training
train = data[train_index,]
# 20% of data is used for testing
test = data[-train_index,]

# Set X and Y, where Y is the target variable "satisfaction"
```

```r
X_train = train %>% select(-satisfaction)
y_train = train$satisfaction

X_test = test %>% select(-satisfaction)
y_test = test$satisfaction
```

```r
# print proportion of satisfied and dissatisfied customers in train and test data
prop.table(table(y_train))
```

```
## y_train
##        0        1
## 0.566035 0.433965
```

```r
prop.table(table(y_test))
```

```
## y_test
##        0        1
## 0.563364 0.436636
```

LOGISTIC REGRESSION

```r
# Model definition with all features:
glm_full<- glm(data = train,
               satisfaction ~ .,
               family = "binomial")
# summary of full model
s<- summary(glm_full)
s
```

```
##
## Call:
## glm(formula = satisfaction ~ ., family = "binomial", data = train)
##
## Coefficients:
##                                    Estimate Std. Error  z value Pr(>|z|)
## (Intercept)                      -8.694e+00  8.492e-02 -102.378  < 2e-16 ***
## Gender                            5.759e-02  1.946e-02    2.960 0.003079 **
## Customer_Type                    -2.015e+00  2.960e-02  -68.065  < 2e-16 ***
## Age                              -8.084e-03  7.099e-04  -11.386  < 2e-16 ***
## Type_of_Travel                    2.769e+00  3.139e-02   88.204  < 2e-16 ***
## Class                            -3.433e-01  1.278e-02  -26.858  < 2e-16 ***
## Flight_Distance                   7.426e-06  1.118e-05    0.664 0.506399
## Inflight_wifi_service             3.832e-01  1.142e-02   33.544  < 2e-16 ***
## Departure_Arrival_time_convenient -1.362e-01  8.180e-03  -16.645  < 2e-16 ***
## Ease_of_Online_booking            -1.480e-01  1.126e-02  -13.149  < 2e-16 ***
## Gate_location                     3.416e-02  9.145e-03    3.735 0.000188 ***
## Food_and_drink                   -2.674e-02  1.067e-02   -2.505 0.012243 *
## Online_boarding                   6.119e-01  1.023e-02   59.816  < 2e-16 ***
## Seat_comfort                      6.394e-02  1.117e-02    5.723 1.05e-08 ***
## Inflight_entertainment            6.058e-02  1.423e-02    4.257 2.08e-05 ***
```

```
## On_board_service                   3.085e-01  1.020e-02   30.247  < 2e-16 ***
## Leg_room_service                   2.461e-01  8.511e-03   28.920  < 2e-16 ***
## Baggage_handling                   1.393e-01  1.144e-02   12.177  < 2e-16 ***
## Checkin_service                    3.288e-01  8.552e-03   38.445  < 2e-16 ***
## Inflight_service                   1.305e-01  1.203e-02   10.845  < 2e-16 ***
## Cleanliness                        2.284e-01  1.205e-02   18.951  < 2e-16 ***
## Departure_Delay_in_Minutes         4.330e-03  9.901e-04    4.373 1.23e-05 ***
## Arrival_Delay_in_Minutes          -9.159e-03  9.765e-04   -9.379  < 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: 141793  on 103588  degrees of freedom
## Residual deviance:  69386  on 103566  degrees of freedom
## AIC: 69432
##
## Number of Fisher Scoring iterations: 6
```

```
#  Here, the code calculates the coefficient of determination, often referred to as R-squared (R²). R²
r2<- 1 - (s$deviance/s$null.deviance)

# In this line, the code calculates the variance inflation factor (VIF) by taking the reciprocal of 1 m
1/(1-r2)
```

```
## [1] 2.04353
```

#COLLINEARITY #It refers to the situation in which 2 or more predictor variables are closely #related to one another. The presence of collinearity can destabilize the model #and make it harder to estimate the effect of each predictor variable.

#LOW collinearity is desirable.

#VIF (Variance Inflation Factor) is a measure of collinearity among predictor variables #within a multiple regression. It is calculated by taking the the ratio of the variance #of all a given model's betas divide by the variance of a single beta if it were fit alone.

#VIF = 1: no collinearity #1 < VIF < 5: moderate collinearity #VIF > 5: high collinearity (this is the case we want to avoid)

#Depending on what value of VIF you deem to be too high to include in the model, #you may choose to remove certain predictor variables and see if the corresponding #R-squared value or standard error of the model is affected.

```
# VIF Iteration 0
# Using the VIF function and comparing the obtained values with the
# computed quantity:
# (The process is done iteratively where we delete one variable at time)
vif_values <- VIF(glm_full)



# Create a data frame with variable names and their corresponding VIF values
vif_df <- data.frame(Variable = names(vif_values), VIF = vif_values,row.names = NULL)
```

```r
# Sort the data frame in decreasing order of VIF values
sorted_df <- vif_df[order(-vif_df$VIF), ]

# Print the sorted data frame
print(sorted_df)
```

```
##                              Variable       VIF
## 22          Arrival_Delay_in_Minutes 14.008175
## 21        Departure_Delay_in_Minutes 13.979107
## 14            Inflight_entertainment  3.245449
## 9             Ease_of_Online_booking  2.576084
## 20                       Cleanliness  2.451463
## 7             Inflight_wifi_service  2.215165
## 13                      Seat_comfort  2.037219
## 11                   Food_and_drink  2.009480
## 19                 Inflight_service  2.007913
## 4                    Type_of_Travel  1.844961
## 17                 Baggage_handling  1.822313
## 8  Departure_Arrival_time_convenient  1.715030
## 15                 On_board_service  1.641210
## 2                     Customer_Type  1.591834
## 5                             Class  1.578498
## 10                   Gate_location  1.516946
## 12                  Online_boarding  1.495898
## 6                   Flight_Distance  1.323544
## 16                 Leg_room_service  1.217731
## 18                  Checkin_service  1.207495
## 3                               Age  1.180871
## 1                            Gender  1.006894
```

```r
# VIF Iteration 1
# Using the VIF function and comparing the obtained values with the
# computed quantity:
# (The process is done iteratively where we delete one variable at time)
# Model definition:
glm_vif1<- glm(data = train,
               satisfaction ~ .-Arrival_Delay_in_Minutes,
               family = "binomial")
vif_values <- VIF(glm_vif1)

# Create a data frame with variable names and their corresponding VIF values
vif_df <- data.frame(Variable = names(vif_values), VIF = vif_values,row.names = NULL)

# Sort the data frame in decreasing order of VIF values
sorted_df <- vif_df[order(-vif_df$VIF), ]

# Print the sorted data frame
print(sorted_df)
```

```
##                              Variable      VIF
## 14            Inflight_entertainment 3.241132
## 9             Ease_of_Online_booking 2.575518
## 20                       Cleanliness 2.448739
```

```
## 7              Inflight_wifi_service 2.214585
## 13                      Seat_comfort 2.037669
## 11                    Food_and_drink 2.006126
## 19                   Inflight_service 2.005777
## 4                      Type_of_Travel 1.843421
## 17                   Baggage_handling 1.822409
## 8  Departure_Arrival_time_convenient 1.716160
## 15                   On_board_service 1.641836
## 2                       Customer_Type 1.590028
## 5                               Class 1.578602
## 10                     Gate_location 1.517501
## 12                   Online_boarding 1.494308
## 6                    Flight_Distance 1.323924
## 16                   Leg_room_service 1.217844
## 18                    Checkin_service 1.207435
## 3                                 Age 1.180391
## 21        Departure_Delay_in_Minutes 1.017993
## 1                              Gender 1.006829
```

```
# VIF Iteration 2
# Using the VIF function and comparing the obtained values with the
# computed quantity:
# (The process is done iteratively where we delete one variable at time)
# Model definition:
glm_vif2<- glm(data = train,
               satisfaction ~ .
              -Arrival_Delay_in_Minutes
              -Inflight_entertainment,
               family = "binomial")
vif_values <- VIF(glm_vif2)

# Create a data frame with variable names and their corresponding VIF values
vif_df <- data.frame(Variable = names(vif_values), VIF = vif_values,row.names = NULL)

# Sort the data frame in decreasing order of VIF values
sorted_df <- vif_df[order(-vif_df$VIF), ]

# Print the sorted data frame
print(sorted_df)
```

```
##                              Variable      VIF
## 9             Ease_of_Online_booking 2.568115
## 7              Inflight_wifi_service 2.168272
## 19                       Cleanliness 2.054477
## 13                      Seat_comfort 1.912449
## 18                   Inflight_service 1.869601
## 4                      Type_of_Travel 1.807889
## 16                   Baggage_handling 1.779945
## 11                    Food_and_drink 1.741701
## 8  Departure_Arrival_time_convenient 1.713822
## 5                               Class 1.566037
## 2                       Customer_Type 1.541106
## 14                   On_board_service 1.532743
## 10                     Gate_location 1.517825
```

```
## 12                   Online_boarding 1.478537
## 6                    Flight_Distance 1.323590
## 15                 Leg_room_service 1.203038
## 3                               Age 1.175829
## 17                   Checkin_service 1.165839
## 20        Departure_Delay_in_Minutes 1.016321
## 1                            Gender 1.005421
```

```
# VIF Iteration 3
# Using the VIF function and comparing the obtained values with the
# computed quantity:
# (The process is done iteratively where we delete one variable at time)
# Model definition:
glm_vif3<- glm(data = train,
               satisfaction ~ .
               -Arrival_Delay_in_Minutes
               -Inflight_entertainment
               -Ease_of_Online_booking ,
               family = "binomial")
vif_values <- VIF(glm_vif3)

# Create a data frame with variable names and their corresponding VIF values
vif_df <- data.frame(Variable = names(vif_values), VIF = vif_values,row.names = NULL)

# Sort the data frame in decreasing order of VIF values
sorted_df <- vif_df[order(-vif_df$VIF), ]

# Print the sorted data frame
print(sorted_df)
```

```
##                            Variable      VIF
## 18                       Cleanliness 2.043409
## 12                      Seat_comfort 1.899349
## 17                  Inflight_service 1.869288
## 4                     Type_of_Travel 1.785648
## 15                  Baggage_handling 1.780261
## 10                    Food_and_drink 1.739464
## 8  Departure_Arrival_time_convenient 1.583158
## 5                               Class 1.561466
## 7              Inflight_wifi_service 1.551222
## 13                  On_board_service 1.532021
## 2                      Customer_Type 1.519759
## 11                   Online_boarding 1.417851
## 9                      Gate_location 1.389852
## 6                    Flight_Distance 1.323397
## 14                 Leg_room_service 1.198476
## 3                               Age 1.168966
## 16                   Checkin_service 1.164720
## 19        Departure_Delay_in_Minutes 1.015387
## 1                            Gender 1.004730
```

```
# VIF Iteration 4
# Using the VIF function and comparing the obtained values with the
```

```
# computed quantity:
# (The process is done iteratively where we delete one variable at time)
# Model definition:
glm_vif4<- glm(data = train,
               satisfaction ~ .
              -Arrival_Delay_in_Minutes
              -Inflight_entertainment
              -Ease_of_Online_booking
              -Cleanliness ,
               family = "binomial")
vif_values <- VIF(glm_vif4)

# Create a data frame with variable names and their corresponding VIF values
vif_df <- data.frame(Variable = names(vif_values), VIF = vif_values,row.names = NULL)

# Sort the data frame in decreasing order of VIF values
sorted_df <- vif_df[order(-vif_df$VIF), ]

# Print the sorted data frame
print(sorted_df)
```

```
##                            Variable      VIF
## 17                 Inflight_service 1.877111
## 15                 Baggage_handling 1.789002
## 4                    Type_of_Travel 1.776978
## 8  Departure_Arrival_time_convenient 1.587584
## 12                     Seat_comfort 1.582754
## 5                             Class 1.560780
## 7              Inflight_wifi_service 1.553323
## 13                  On_board_service 1.537258
## 2                      Customer_Type 1.513158
## 10                   Food_and_drink 1.431039
## 11                  Online_boarding 1.412727
## 9                     Gate_location 1.397708
## 6                   Flight_Distance 1.324727
## 14                 Leg_room_service 1.202714
## 3                               Age 1.165981
## 16                  Checkin_service 1.162132
## 18       Departure_Delay_in_Minutes 1.012514
## 1                           Gender 1.004091
```

```
glm_reduced<- glm(data = train,
                  satisfaction ~ .
                  -Arrival_Delay_in_Minutes
                  -Inflight_entertainment
                  -Ease_of_Online_booking
                  -Cleanliness    ,
                  family = "binomial")
# Observation of the model summary:
summary(glm_reduced)
```

```
##
```

```
## Call:
## glm(formula = satisfaction ~ . - Arrival_Delay_in_Minutes - Inflight_entertainment -
##     Ease_of_Online_booking - Cleanliness, family = "binomial",
##     data = train)
##
## Coefficients:
##                                    Estimate Std. Error  z value Pr(>|z|)
## (Intercept)                      -8.785e+00  8.216e-02 -106.922  < 2e-16 ***
## Gender                            6.208e-02  1.927e-02    3.221  0.00128 **
## Customer_Type                    -2.050e+00  2.898e-02  -70.759  < 2e-16 ***
## Age                              -8.272e-03  7.011e-04  -11.799  < 2e-16 ***
## Type_of_Travel                    2.777e+00  3.083e-02   90.077  < 2e-16 ***
## Class                            -3.179e-01  1.262e-02  -25.201  < 2e-16 ***
## Flight_Distance                   6.086e-06  1.105e-05    0.551  0.58178
## Inflight_wifi_service             3.091e-01  9.468e-03   32.647  < 2e-16 ***
## Departure_Arrival_time_convenient -1.700e-01  7.822e-03  -21.733  < 2e-16 ***
## Gate_location                    -1.173e-03  8.689e-03   -0.135  0.89258
## Food_and_drink                    9.003e-02  8.910e-03   10.104  < 2e-16 ***
## Online_boarding                   5.996e-01  9.936e-03   60.348  < 2e-16 ***
## Seat_comfort                      1.927e-01  9.772e-03   19.719  < 2e-16 ***
## On_board_service                  3.250e-01  9.763e-03   33.284  < 2e-16 ***
## Leg_room_service                  2.465e-01  8.387e-03   29.388  < 2e-16 ***
## Baggage_handling                  1.582e-01  1.119e-02   14.136  < 2e-16 ***
## Checkin_service                   3.331e-01  8.308e-03   40.098  < 2e-16 ***
## Inflight_service                  1.523e-01  1.148e-02   13.264  < 2e-16 ***
## Departure_Delay_in_Minutes       -4.383e-03  2.648e-04  -16.553  < 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: 141793  on 103588  degrees of freedom
## Residual deviance:  70230  on 103570  degrees of freedom
## AIC: 70268
##
## Number of Fisher Scoring iterations: 5
```

```r
# While checking the p value in summary, as we see Flight_Distance and Gate_location does not have sign

glm_last<- glm(data = train,
               satisfaction ~ .
               -Arrival_Delay_in_Minutes
               -Inflight_entertainment
               -Ease_of_Online_booking
               -Cleanliness-Gate_location
               -Flight_Distance
               -Gate_location,
               family = "binomial")


# Computing the predictions with the model on the test set:
pred_glm_last<- predict(glm_last, test, type = "response")
```

Classification Outputs : Accuracy, F1, Precision, Recall

```r
#The function will return the calculated evaluation metrics in the results data frame
calculate_evaluation_metrics <- function(thresholds, output_list, y_test) {
  # Create an empty data frame to store the results
  results_df <- data.frame(
    Threshold = numeric(length(thresholds)),
    Accuracy = numeric(length(thresholds)),
    F1_Score = numeric(length(thresholds)),
    Precision = numeric(length(thresholds)),
    Recall = numeric(length(thresholds))
  )

  # Calculate evaluation metrics for each threshold and store the results in the data frame
  for (i in 1:length(thresholds)) {
    threshold <- thresholds[i]

    pred_output <- output_list[[as.character(threshold)]]

    results_df[i, "Threshold"] <- threshold
    results_df[i, "Accuracy"] <- Accuracy(y_pred = pred_output, y_true = y_test)
    results_df[i, "F1_Score"] <- F1_Score(y_pred = pred_output, y_true = y_test)
    results_df[i, "Precision"] <- Precision(y_pred = pred_output, y_true = y_test)
    results_df[i, "Recall"] <- Recall(y_pred = pred_output, y_true = y_test)
  }

  # Format the floating-point numbers with two decimal places
  results_df$Accuracy <- round(results_df$Accuracy, 4)
  results_df$F1_Score <- round(results_df$F1_Score, 4)
  results_df$Precision <- round(results_df$Precision, 4)
  results_df$Recall <- round(results_df$Recall, 4)

  return(results_df)
}
```

```r
# Converting the prediction in {0,1} according to the chosen threshold:
thresholds <- c(0.4, 0.5, 0.6, 0.7)
output_list <- list()

for (threshold in thresholds) {
  output <- ifelse(pred_glm_last > threshold, 1, 0)
  output_list[[as.character(threshold)]] <- output
}

# Access the outputs using the threshold values as keys
#output_list$`0.4`
#output_list$`0.5`
#output_list$`0.6`
#output_list$`0.7`
```

```r
# Calculate evaluation metrics
results <- calculate_evaluation_metrics(thresholds, output_list, y_test)
print(results)
```

```
##   Threshold Accuracy F1_Score Precision Recall
## 1       0.4   0.8607   0.8739    0.8922 0.8563
## 2       0.5   0.8725   0.8887    0.8745 0.9034
## 3       0.6   0.8722   0.8921    0.8508 0.9375
## 4       0.7   0.8618   0.8870    0.8222 0.9630
```

```
# Print the results as a table in R Markdown
knitr::kable(results, align = "c")
```

Threshold	Accuracy	F1_Score	Precision	Recall
0.4	0.8607	0.8739	0.8922	0.8563
0.5	0.8725	0.8887	0.8745	0.9034
0.6	0.8722	0.8921	0.8508	0.9375
0.7	0.8618	0.8870	0.8222	0.9630

```
#Model definition:
# Here we don't re-apply the VIF method because we start from the
# previous result.
glm_full<- glm(data = train,
               satisfaction ~ .,
               family = "binomial")

# Application of the Stepwise method, specifying that we consider
# both the forward and the backward directions. We consider as
# reference metric the Akaike Information Criterion:
glm_step <- stepAIC(glm_full, direction = "both",trace = FALSE)
# Observation of the model summary:
summary(glm_step)
```

```
##
## Call:
## glm(formula = satisfaction ~ Gender + Customer_Type + Age + Type_of_Travel +
##     Class + Inflight_wifi_service + Departure_Arrival_time_convenient +
##     Ease_of_Online_booking + Gate_location + Food_and_drink +
##     Online_boarding + Seat_comfort + Inflight_entertainment +
##     On_board_service + Leg_room_service + Baggage_handling +
##     Checkin_service + Inflight_service + Cleanliness + Departure_Delay_in_Minutes +
##     Arrival_Delay_in_Minutes, family = "binomial", data = train)
##
## Coefficients:
##                                    Estimate Std. Error  z value Pr(>|z|)
## (Intercept)                      -8.6834688  0.0833554 -104.174  < 2e-16 ***
## Gender                            0.0576110  0.0194566    2.961 0.003066 **
## Customer_Type                    -2.0196473  0.0286649  -70.457  < 2e-16 ***
## Age                              -0.0081124  0.0007086  -11.448  < 2e-16 ***
## Type_of_Travel                    2.7714471  0.0311571   88.951  < 2e-16 ***
## Class                            -0.3459367  0.0121620  -28.444  < 2e-16 ***
## Inflight_wifi_service             0.3827055  0.0113993   33.573  < 2e-16 ***
## Departure_Arrival_time_convenient -0.1360790  0.0081796  -16.636  < 2e-16 ***
```

```
## Ease_of_Online_booking            -0.1478984  0.0112555  -13.140  < 2e-16 ***
## Gate_location                      0.0341116  0.0091455    3.730 0.000192 ***
## Food_and_drink                    -0.0268544  0.0106711   -2.517 0.011851 *
## Online_boarding                    0.6121134  0.0102223   59.880  < 2e-16 ***
## Seat_comfort                       0.0641066  0.0111716    5.738 9.56e-09 ***
## Inflight_entertainment             0.0605781  0.0142310    4.257 2.07e-05 ***
## On_board_service                   0.3086085  0.0101990   30.259  < 2e-16 ***
## Leg_room_service                   0.2463465  0.0085058   28.962  < 2e-16 ***
## Baggage_handling                   0.1391618  0.0114369   12.168  < 2e-16 ***
## Checkin_service                    0.3288115  0.0085515   38.451  < 2e-16 ***
## Inflight_service                   0.1303791  0.0120341   10.834  < 2e-16 ***
## Cleanliness                        0.2284507  0.0120531   18.954  < 2e-16 ***
## Departure_Delay_in_Minutes         0.0043350  0.0009901    4.378 1.20e-05 ***
## Arrival_Delay_in_Minutes          -0.0091635  0.0009766   -9.383  < 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: 141793  on 103588  degrees of freedom
## Residual deviance:  69387  on 103567  degrees of freedom
## AIC: 69431
##
## Number of Fisher Scoring iterations: 6
```

```r
# Computing the predictions with the model on the test set:
pred_glm_step<- predict(glm_step, test, type = "response")
```

```r
# Converting the prediction in {0,1} according to the chosen threshold:
thresholds <- c(0.4, 0.5, 0.6, 0.7)
output_list <- list()

for (threshold in thresholds) {
  output <- ifelse(pred_glm_step > threshold, 1, 0)
  output_list[[as.character(threshold)]] <- output
}

# Access the outputs using the threshold values as keys
#output_list$`0.4`
#output_list$`0.5`
#output_list$`0.6`
#output_list$`0.7`
```

```r
# Calculate evaluation metrics
results <- calculate_evaluation_metrics(thresholds, output_list, y_test)
print(results)
```

```
##   Threshold Accuracy F1_Score Precision Recall
## 1       0.4   0.8618   0.8750    0.8918 0.8589
## 2       0.5   0.8726   0.8889    0.8738 0.9044
## 3       0.6   0.8738   0.8934    0.8523 0.9385
## 4       0.7   0.8643   0.8888    0.8256 0.9624
```

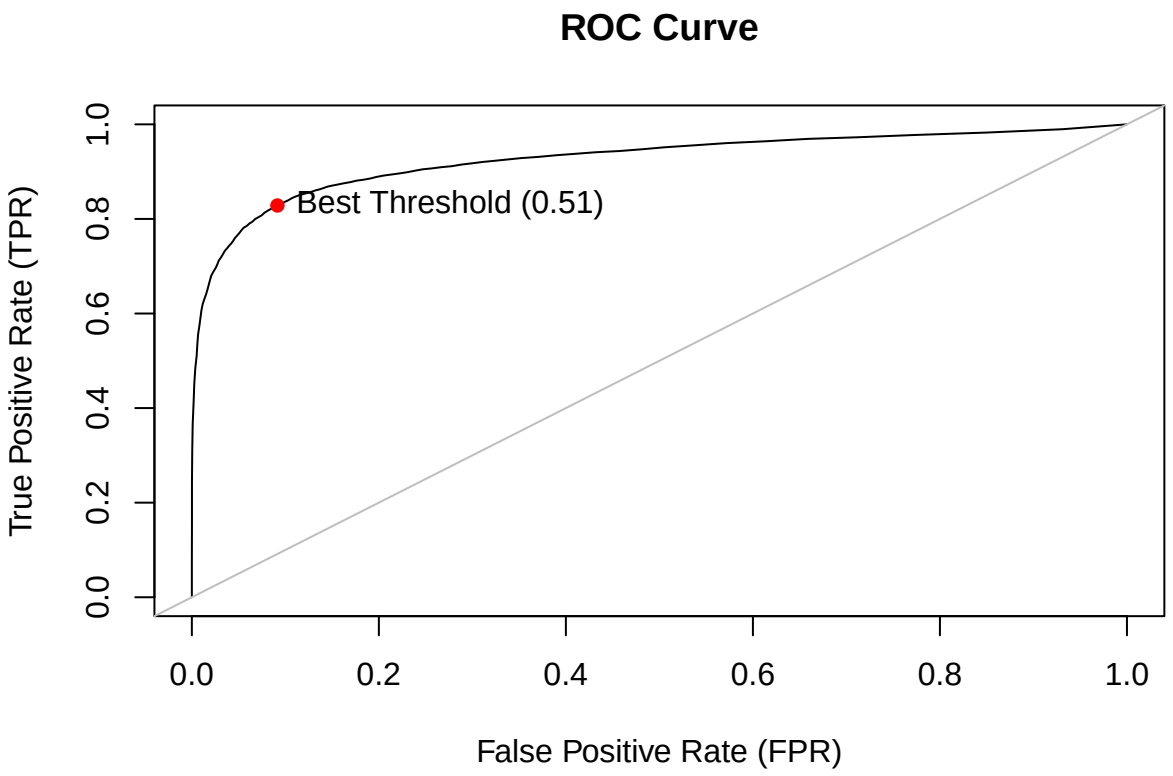
```r
# Print the results as a table in R Markdown
knitr::kable(results, align = "c")
```

Threshold	Accuracy	F1_Score	Precision	Recall
0.4	0.8618	0.8750	0.8918	0.8589
0.5	0.8726	0.8889	0.8738	0.9044
0.6	0.8738	0.8934	0.8523	0.9385
0.7	0.8643	0.8888	0.8256	0.9624

ROC CURVE

```r
plot_roc_curve <- function(predicted_probabilities, true_labels) {
  # Calculate TPR and FPR for different threshold values
  thresholds <- seq(0, 1, by = 0.01)
  tpr <- numeric(length = length(thresholds))
  fpr <- numeric(length = length(thresholds))

  for (i in 1:length(thresholds)) {
    threshold <- thresholds[i]
    predicted_labels <- ifelse(predicted_probabilities >= threshold, 1, 0)

    tp <- sum(predicted_labels == 1 & true_labels == 1)
    tn <- sum(predicted_labels == 0 & true_labels == 0)
    fp <- sum(predicted_labels == 1 & true_labels == 0)
    fn <- sum(predicted_labels == 0 & true_labels == 1)

    tpr[i] <- tp / (tp + fn)
    fpr[i] <- fp / (fp + tn)
  }

  # Plot ROC curve
  plot(fpr, tpr, type = "l", main = "ROC Curve", xlab = "False Positive Rate (FPR)", ylab = "True Positi
  abline(0, 1, col = "gray")

  # Find the best threshold point
  best_threshold <- thresholds[which.max(tpr - fpr)]
  best_tpr <- tpr[which.max(tpr - fpr)]
  best_fpr <- fpr[which.max(tpr - fpr)]

  # Add the best threshold point to the graph
  points(best_fpr, best_tpr, col = "red", pch = 16)
  text(best_fpr, best_tpr, sprintf("Best Threshold (%.2f)", best_threshold), pos = 4)

  # Return the best threshold and its TPR value
  return(list(best_threshold = best_threshold, best_tpr = best_tpr))
}
```

```r
result<-plot_roc_curve(pred_glm_step, y_test)
```




```
print(result$best_threshold)
```

```
## [1] 0.51
```

```
print(result$best_tpr)
```

```
## [1] 0.82844
```

Naive Bayes

```
#1. Estimating the p-dimensional joint distribution of predictors may be
#challenging;
#2. the conditional independence assumption allows us to estimate only
#the marginal distribution of predictors;
#3. in most cases the conditional independence assumption is unrealistic
#and mainly made for convenience...
#4. ... however despite the strong assumption Naive Bayes often
#produces good classification results, especially in settings where n is
#not large enough relative to p to effectively estimate the joint
#distribution of predictors within each class.
```

```r
#The Naive Bayes Classifier is a probabilistic algorithm used for binary classification. It assumes tha

nb.fit <- naiveBayes(data = X_train,
                     y_train ~ .)

# Make predictions on the test data
predictions <- predict(nb.fit, newdata = X_test)

# Evaluate the performance of the classifier
table(predictions, y_test)
```

```
##            y_test
## predictions     0     1
##           0 13087  2061
##           1  1503  9247
```

```r
mean(predictions == y_test)
```

```
## [1] 0.8623832
```

Decision Trees

```r
# Build the decision tree model

#choose some variables
# Specify the names of variables to drop
#variables_to_drop <- c("Arrival_Delay_in_Minutes", "Inflight_entertainment", "Ease_of_Online_booking",

# Drop the variables from the training dataset
#tree_data <- X_train[, !(names(X_train) %in% variables_to_drop)]


tree_model <- rpart(data = X_train,
                    y_train ~ .,
                   method = "class")

# Display the tree plot
#plot(tree_model)
#text(tree_model)

# Make predictions on the test data
predictions <- predict(tree_model, newdata = X_test, type = "class")

# Evaluate the performance of the classifier
table(predictions, y_test)
```

```
##            y_test
## predictions     0     1
##           0 12627  1062
##           1  1963 10246
```

```
mean(predictions == y_test)
```

```
## [1] 0.8831956
```

Random Forest

```
# Build random forest model

# Build the Random Forest model
# number of trees : 100
rf_model <- randomForest(data = X_train,
                         factor(y_train) ~ .,
                         ntree = 100)

# Make predictions on the test data
predictions <- predict(rf_model, newdata = X_test,)

# Evaluate the performance of the classifier
table(predictions, y_test)
```

```
##            y_test
## predictions     0     1
##           0 14300   668
##           1   290 10640
```

```
mean(predictions == y_test)
```

```
## [1] 0.9630087
```

Training error vs Test error

We are interested in assessing the accuracy of predictions when the statistical learning method is applied to previously unseen test observations not used to train the method. Recall the distinction between the test error and the training error: - The test error is the expected error that results from using a statistical learning method to predict the response on a new observation, one that was not used in training the method. It can be estimated if a designated test set is available. -In contrast, the training error can be easily computed by applying the statistical learning method to the observations used in its training. -But the training error rate often is quite different from the test error rate, and in particular the former can dramatically underestimate the latter.

```
# Then we try to reproduce the same plot as above, considering the
# classifications obtained with the models
a <- ggplot(X_test,
            aes(
              x = Flight_Distance ,
              y = Departure_Delay_in_Minutes ,
              color = factor(output_list$`0.6`) # 0 or 1
            )) +
```

```
  geom_point(size = 2) +
  labs(
    x = "Flight Distance",
    y = "Departure Delay (minutes)",
    color = "Satisfaction",
    title = "Simple GLM : 0.6"
  ) +
  scale_color_manual(values = c("0" = "red", "1" = "darkgreen")) +
  theme(legend.position = c(0.8, 0.8))

b <- ggplot(X_test,
            aes(
              x = Flight_Distance ,
              y = Departure_Delay_in_Minutes ,
              color = factor(output_list$`0.5`) # 0 or 1
            )) +
  geom_point(size = 2) +
  labs(
    x = "Flight Distance",
    y = "Departure Delay (minutes)",
    color = "Satisfaction",
    title = "Simple GLM  : 0.5"
  ) +
  scale_color_manual(values = c("0" = "red", "1" = "darkgreen")) +
theme(legend.position = c(0.8, 0.8))

grid.arrange(a, b,ncol = 2)
```


```
# We compare the results obtained with the two different models, plotting
# now an estimation of the logistic curve using the predictions given by
# the models:
predicted_data <-
  data.frame(prob.of.Satisfaction = pred_glm_step, Satisfaction = y_test)
predicted_data <-
  predicted_data[order(predicted_data$prob.of.Satisfaction, decreasing = FALSE), ]
predicted_data$rank <- 1:nrow(predicted_data)
a <- ggplot(data = predicted_data, aes(x = rank, y = prob.of.Satisfaction)) +
  geom_point(
    aes(color = as.factor(Satisfaction)),
    alpha = 1,
    shape = 1,
    stroke = 1
  ) +
  xlab("Index") +
  ylab("Predicted probability") +
  ggtitle("Estimated Logistic Curve - Simple GLM")


predicted_data <-
  data.frame(prob.of.Satisfaction = pred_glm_last, Satisfaction = y_test)
predicted_data <-
  predicted_data[order(predicted_data$prob.of.Satisfaction, decreasing = FALSE), ]
predicted_data$rank <- 1:nrow(predicted_data)
b <- ggplot(data = predicted_data, aes(x = rank, y = prob.of.Satisfaction)) +
```

```
  geom_point(
    aes(color = as.factor(Satisfaction)),
    alpha = 1,
    shape = 1,
    stroke = 1
  ) +
  xlab("Index") +
  ylab("Predicted probability") +
  ggtitle("Estimated Logistic Curve - GLM with Stepwise")

grid.arrange(a, b, nrow = 2)
```