PREDICT OF HOTEL BOOKING

PACKAGES AND DATASET :

In R, the caret package is a package for training, testing, and comparing machine learning models, ROCR package is a package used to evaluate and visualize classifier performance , The tidymodels package is a collection of packages used for modeling and machine learning in accordance with tidyverse principles ,The rpart.plot package is a package used to visualize rpart models, mlbench package is a package containing artificial and real world machine learning problem sets , The ranger package is a package for generating fast and scalable random forests.

```r
#install.packages("caret")
#install.packages("ROCR")
#install.packages("tidymodels")
#install.packages("rpart.plot")
#install.packages("mlbench")
#install.packages("ranger")
library(ranger)
library(mlbench)
library(rpart.plot)
```

```
Zorunlu paket yükleniyor: rpart
```

```r
library(tidymodels)
```

```
-- Attaching packages -------------------------------------- tidymodels 1.1.0 --

v broom        1.0.4     v recipes      1.0.6
v dials        1.2.0     v rsample      1.1.1
v dplyr        1.1.2     v tibble       3.2.1
```

```
v ggplot2      3.4.2     v tidyr        1.3.0
v infer        1.0.4     v tune         1.1.1
v modeldata    1.1.0     v workflows    1.1.3
v parsnip      1.1.0     v workflowsets 1.0.1
v purrr        1.0.1     v yardstick    1.2.0

-- Conflicts ----------------------------------------- tidymodels_conflicts() --
x purrr::discard() masks scales::discard()
x dplyr::filter()  masks stats::filter()
x dplyr::lag()     masks stats::lag()
x dials::prune()   masks rpart::prune()
x recipes::step()  masks stats::step()
* Use suppressPackageStartupMessages() to eliminate package startup messages
```

```r
library(caret)
```

```
Zorunlu paket yükleniyor: lattice


Attaching package: 'caret'


The following objects are masked from 'package:yardstick':

    precision, recall, sensitivity, specificity


The following object is masked from 'package:purrr':

    lift
```

```r
library(ROCR)
hotel_bookings <- read.csv("hotel_bookings.csv")
str(hotel_bookings)
```

```
'data.frame':   119390 obs. of  32 variables:
 $ hotel                         : chr  "Resort Hotel" "Resort Hotel" "Resort Hotel" "Resort
 $ is_canceled                   : int  0 0 0 0 0 0 0 0 1 1 ...
 $ lead_time                     : int  342 737 7 13 14 14 0 9 85 75 ...
 $ arrival_date_year             : int  2015 2015 2015 2015 2015 2015 2015 2015 2015 2015 ..
```

```
$ arrival_date_month            : chr  "July" "July" "July" "July" ...
$ arrival_date_week_number      : int  27 27 27 27 27 27 27 27 27 27 ...
$ arrival_date_day_of_month     : int  1 1 1 1 1 1 1 1 1 1 ...
$ stays_in_weekend_nights       : int  0 0 0 0 0 0 0 0 0 0 ...
$ stays_in_week_nights          : int  0 0 1 1 2 2 2 2 3 3 ...
$ adults                        : int  2 2 1 1 2 2 2 2 2 2 ...
$ children                      : int  0 0 0 0 0 0 0 0 0 0 ...
$ babies                        : int  0 0 0 0 0 0 0 0 0 0 ...
$ meal                          : chr  "BB" "BB" "BB" "BB" ...
$ country                       : chr  "PRT" "PRT" "GBR" "GBR" ...
$ market_segment                : chr  "Direct" "Direct" "Direct" "Corporate" ...
$ distribution_channel          : chr  "Direct" "Direct" "Direct" "Corporate" ...
$ is_repeated_guest             : int  0 0 0 0 0 0 0 0 0 0 ...
$ previous_cancellations        : int  0 0 0 0 0 0 0 0 0 0 ...
$ previous_bookings_not_canceled: int  0 0 0 0 0 0 0 0 0 0 ...
$ reserved_room_type            : chr  "C" "C" "A" "A" ...
$ assigned_room_type            : chr  "C" "C" "C" "A" ...
$ booking_changes               : int  3 4 0 0 0 0 0 0 0 0 ...
$ deposit_type                  : chr  "No Deposit" "No Deposit" "No Deposit" "No Deposit"
$ agent                         : chr  "NULL" "NULL" "NULL" "304" ...
$ company                       : chr  "NULL" "NULL" "NULL" "NULL" ...
$ days_in_waiting_list          : int  0 0 0 0 0 0 0 0 0 0 ...
$ customer_type                 : chr  "Transient" "Transient" "Transient" "Transient" ...
$ adr                           : num  0 0 75 75 98 ...
$ required_car_parking_spaces   : int  0 0 0 0 0 0 0 0 0 0 ...
$ total_of_special_requests     : int  0 0 0 0 1 1 0 1 1 0 ...
$ reservation_status            : chr  "Check-Out" "Check-Out" "Check-Out" "Check-Out" ...
$ reservation_status_date       : chr  "2015-07-01" "2015-07-01" "2015-07-02" "2015-07-02"
```

This data set contains booking information for a city hotel and a resort hotel, and includes information such as when the booking was made, length of stay, the number of adults, children, and/or babies, and the number of available parking spaces.

Splitting the data set:

I use ‘sample()‘ function to split the data set as ‘test‘ and ‘train‘ set.

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

```
test  <- hotel_bookings[-index, ]
```

Train a logistic regression :

I use the ‘glm()‘ function to train a logistic regression model. To use this function, I need to edit some variables in the dataset.

```
hotel_bookings$is_canceled <- as.factor(hotel_bookings$is_canceled)
hotel_bookings$is_repeated_guest <- as.factor(hotel_bookings$is_repeated_guest)
hotel_bookings$lead_time <- as.factor(hotel_bookings$lead_time)
hotel_bookings$arrival_date_month <- as.factor(hotel_bookings$arrival_date_month)
hotel_bookings$reservation_status <- as.factor(hotel_bookings$reservation_status)


lr_model <- glm(is_canceled ~ is_repeated_guest + lead_time + arrival_date_month , data =
summary(lr_model)
```

```
Call:
glm(formula = is_canceled ~ is_repeated_guest + lead_time + arrival_date_month,
    family = "binomial", data = train)

Coefficients:
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                 -9.141e-01  2.336e-02 -39.126  < 2e-16 ***
is_repeated_guest           -8.793e-01  5.274e-02 -16.673  < 2e-16 ***
lead_time                    6.015e-03  7.308e-05  82.311  < 2e-16 ***
arrival_date_monthAugust    -3.285e-01  3.033e-02 -10.833  < 2e-16 ***
arrival_date_monthDecember  -1.516e-01  3.708e-02  -4.088 4.35e-05 ***
arrival_date_monthFebruary  -3.191e-02  3.514e-02  -0.908 0.363761
arrival_date_monthJanuary   -1.368e-01  3.958e-02  -3.457 0.000546 ***
arrival_date_monthJuly      -4.222e-01  3.120e-02 -13.534  < 2e-16 ***
arrival_date_monthJune      -2.227e-01  3.192e-02  -6.978 3.00e-12 ***
arrival_date_monthMarch     -2.582e-01  3.352e-02  -7.702 1.34e-14 ***
arrival_date_monthMay       -2.270e-01  3.140e-02  -7.230 4.84e-13 ***
arrival_date_monthNovember  -3.211e-01  3.790e-02  -8.471  < 2e-16 ***
arrival_date_monthOctober   -3.239e-01  3.228e-02 -10.033  < 2e-16 ***
arrival_date_monthSeptember -3.713e-01  3.279e-02 -11.324  < 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: 125994  on 95508  degrees of freedom
Residual deviance: 117081  on 95495  degrees of freedom
AIC: 117109

Number of Fisher Scoring iterations: 4
```

Performance :

```r
predicted_probs <- predict(lr_model, test[,-32], type = "response")
head(predicted_probs)
```

```
        2         3         4         6         9        11 
0.9567593 0.2151349 0.2212914 0.2223297 0.3046886 0.2318302 
```

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

```
 2  3  4  6  9 11 
 1  0  0  0  0  0 
```

DECISION TREE:

Splitting the data set :

```r
hotel_split <- initial_split(data = hotel_bookings, prop = 0.80)
hotel_train <- hotel_split |> training()
hotel_test  <- hotel_split |> testing()
```

Train a decision tree :

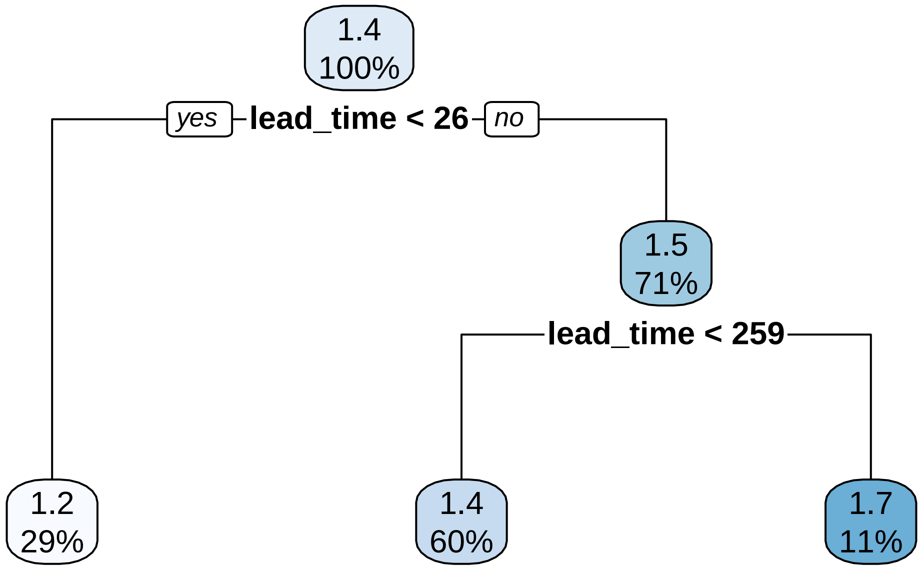
```r
hotel_train$is_canceled <- as.numeric(hotel_train$is_canceled)
hotel_train$is_repeated_guest <- as.numeric(hotel_train$is_repeated_guest)
hotel_train$lead_time <- as.numeric(hotel_train$lead_time)
hotel_train$arrival_date_month <- as.numeric(hotel_train$arrival_date_month)
```

```r
dt_model <- decision_tree() |> set_engine("rpart") |> set_mode("regression")

dt_hotel <- dt_model |>

  fit(is_canceled ~ is_repeated_guest + lead_time + arrival_date_month ,data = hotel_train

dt_hotel
```

```
parsnip model object

n= 95508

node), split, n, deviance, yval
      * denotes terminal node

1) root 95508 22268.640 1.370231
  2) lead_time< 25.5 27563  3775.413 1.163807 *
  3) lead_time>=25.5 67945 16842.290 1.453970
    6) lead_time< 258.5 57693 14021.810 1.416584 *
    7) lead_time>=258.5 10252  2286.056 1.664358 *
```

```r
rpart.plot(dt_hotel$fit)
```

```
Warning: Cannot retrieve the data used to build the model (so cannot determine roundint and i
To silence this warning:
    Call rpart.plot with roundint=FALSE,
    or rebuild the rpart model with model=TRUE.
```




```r
hotel_test$is_canceled <- as.numeric(hotel_test$is_canceled)
hotel_test$is_repeated_guest <- as.numeric(hotel_test$is_repeated_guest)
hotel_test$lead_time <- as.numeric(hotel_test$lead_time)
hotel_test$arrival_date_month <- as.numeric(hotel_test$arrival_date_month)
hotel_predictions <- dt_hotel |>
  predict(new_data = hotel_test)
hotel_predictions
```

```
# A tibble: 23,878 x 1
   .pred
   <dbl>
 1  1.66
 2  1.16
 3  1.16
 4  1.42
 5  1.42
 6  1.42
 7  1.42
 8  1.42
 9  1.42
10  1.42
# i 23,868 more rows
```

```r
hotel_results <- tibble(predicted = hotel_predictions$.pred,
                        actual    = hotel_test$is_canceled)
hotel_results
```

```
# A tibble: 23,878 x 2
   predicted actual
       <dbl>  <dbl>
 1      1.66      1
 2      1.16      1
 3      1.16      1
 4      1.42      1
 5      1.42      1
 6      1.42      1
 7      1.42      1
 8      1.42      2
 9      1.42      1
10      1.42      1
# i 23,868 more rows
```

```r
hotel_results |> rmse(truth = actual, estimate = predicted)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard       0.458
```

```r
hotel_results |> rsq(truth = actual, estimate = predicted)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rsq     standard      0.0998
```

RANDOM FOREST TREE :

```
hotel_split <- initial_split(data = hotel_bookings , prop = 0.80)
hotel_train_rf <- hotel_split |> training()
hotel_test_rf  <- hotel_split |> testing()
set.seed(123)
trained_rf <- ranger(is_canceled ~ is_repeated_guest + lead_time + arrival_date_month  ,da
trained_rf
```

```
Ranger result

Call:
 ranger(is_canceled ~ is_repeated_guest + lead_time + arrival_date_month,      data = hotel_

Type:                             Classification
Number of trees:                  500
Sample size:                      95508
Number of independent variables:  3
Mtry:                             1
Target node size:                 1
Variable importance mode:         none
Splitrule:                        gini
OOB prediction error:             35.06 %
```

```
preds_rf <- predict(trained_rf, hotel_test_rf)
confusionMatrix(preds_rf$predictions,
                hotel_test_rf$is_canceled,
                positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 14892  8139
         1   147   700

               Accuracy : 0.653
                 95% CI : (0.6469, 0.659)
    No Information Rate : 0.6298
    P-Value [Acc > NIR] : 5.114e-14
```

```
                 Kappa : 0.0853

Mcnemar's Test P-Value : < 2.2e-16

           Sensitivity : 0.07919
           Specificity : 0.99023
        Pos Pred Value : 0.82645
        Neg Pred Value : 0.64661
            Prevalence : 0.37017
        Detection Rate : 0.02932
  Detection Prevalence : 0.03547
     Balanced Accuracy : 0.53471

      'Positive' Class : 1
```