Machine Learning Project

Melih Gündüz/Fehmican Korkuter

2024-06-10

1. Problem, Features, and Target

The dataset is related to Hotel Reservations . It includes features such as number of adults, number of children, number of weekend,nights, booking status… The target variable is booking status.booking status will show us whether the reservation has been canceled or not. The aim of the analysis is to create models according to the given features and predicting whether reservations are canceled or not and evaluating the performance of the models

2. Dataset

```
library(readr)
hotel_new <- read_csv("Hotel Reservations.csv")
```

```
hotel_new <- hotel_new[,-1]
hotelnew <- na.omit(hotel_new)
hotelnew$booking_status <- ifelse(hotelnew$booking_status == "Canceled", 1, 0)
str(hotelnew)
```

```
tibble [36,275 x 18] (S3: tbl_df/tbl/data.frame)
 $ no_of_adults                        : num [1:36275] 2 2 1 2 2 2 2 2 3 2 ...
 $ no_of_children                      : num [1:36275] 0 0 0 0 0 0 0 0 0 0 ...
 $ no_of_weekend_nights                : num [1:36275] 1 2 2 0 1 0 1 1 0 0 ...
 $ no_of_week_nights                   : num [1:36275] 2 3 1 2 1 2 3 3 4 5 ...
 $ type_of_meal_plan                   : chr [1:36275] "Meal Plan 1" "Not Selected" "Meal Pla
 $ required_car_parking_space          : num [1:36275] 0 0 0 0 0 0 0 0 0 0 ...
 $ room_type_reserved                  : chr [1:36275] "Room_Type 1" "Room_Type 1" "Room_Type
 $ lead_time                           : num [1:36275] 224 5 1 211 48 346 34 83 121 44 ...
```

```
$ arrival_year                        : num [1:36275] 2017 2018 2018 2018 2018 ...
$ arrival_month                       : num [1:36275] 10 11 2 5 4 9 10 12 7 10 ...
$ arrival_date                        : num [1:36275] 2 6 28 20 11 13 15 26 6 18 ...
$ market_segment_type                 : chr [1:36275] "Offline" "Online" "Online" "Online"
$ repeated_guest                      : num [1:36275] 0 0 0 0 0 0 0 0 0 0 ...
$ no_of_previous_cancellations        : num [1:36275] 0 0 0 0 0 0 0 0 0 0 ...
$ no_of_previous_bookings_not_canceled: num [1:36275] 0 0 0 0 0 0 0 0 0 0 ...
$ avg_price_per_room                  : num [1:36275] 65 106.7 60 100 94.5 ...
$ no_of_special_requests              : num [1:36275] 0 1 0 0 0 1 1 1 1 3 ...
$ booking_status                      : num [1:36275] 0 0 1 1 1 1 0 0 0 0 ...
```

The dataset has a total of 36275 observations and 18 variables. Among these, 15 are numerical variables and 3 are categorical variables.

3. Check the imbalance problem

```
table(hotelnew$booking_status)/dim(hotelnew) [1]
```

```
        0         1
0.6723639 0.3276361
```

The class distribution in the training dataset is imbalanced. with approximately 67.23% of examples belonging to the negative class and 32.76% belonging to the positive class.Thus, It may cause the model to learn less from the minority class, thus not reaching a satisfactory result.

3.1

```
set.seed(123)
data_balanced_s <- ovun.sample(booking_status~., data = hotelnew,
                               method = "over", p=0.5)
data_balanced <- data_balanced_s$data
```

Oversampling is a technique used to balance unbalanced data sets in classification problems.

```
table(data_balanced$booking_status)/dim(data_balanced) [1]
```

```
        0         1
0.5028348 0.4971652
```

4. Splitting The Dataset

```
set.seed(123)
hotel_split <- initial_split(data = data_balanced, prop = 0.80)
hotel_train <- hotel_split |> training()
hotel_test <- hotel_split |> testing()
```

It involves understanding the relationships between the model's parameters within the dataset. The goal is to predict the target variable, and the dataset is divided into two subsets: 80% for 'train' and 20% for 'test'.

5.Train a Logistic Regression and A Decision Tree Model

5.1 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(hotel_train$booking_status~., data=hotel_train,
                family = "binomial")
summary(lr_model)
```

```
Call:
glm(formula = hotel_train$booking_status ~ ., family = "binomial",
    data = hotel_train)

Coefficients:
                                 Estimate Std. Error z value Pr(>|z|)
(Intercept)                    -6.824e+02  9.032e+01  -7.555 4.19e-14 ***
no_of_adults                    7.056e-02  2.904e-02   2.430  0.01510 *
no_of_children                  1.433e-01  4.671e-02   3.069  0.00215 **
no_of_weekend_nights            1.492e-01  1.526e-02   9.777  < 2e-16 ***
```

```
no_of_week_nights                      5.255e-02  9.305e-03   5.647 1.63e-08 ***
type_of_meal_planMeal Plan 2           2.130e-01  5.220e-02   4.081 4.49e-05 ***
type_of_meal_planMeal Plan 3           1.150e+01  9.325e+01   0.123  0.90184
type_of_meal_planNot Selected          2.155e-01  4.135e-02   5.212 1.87e-07 ***
required_car_parking_space            -1.712e+00  1.030e-01 -16.626  < 2e-16 ***
room_type_reservedRoom_Type 2         -4.605e-01  1.028e-01  -4.481 7.43e-06 ***
room_type_reservedRoom_Type 3         -2.965e-01  1.074e+00  -0.276  0.78254
room_type_reservedRoom_Type 4         -2.276e-01  4.129e-02  -5.513 3.53e-08 ***
room_type_reservedRoom_Type 5         -6.623e-01  1.656e-01  -3.998 6.38e-05 ***
room_type_reservedRoom_Type 6         -9.179e-01  1.175e-01  -7.814 5.55e-15 ***
room_type_reservedRoom_Type 7         -1.255e+00  2.291e-01  -5.478 4.30e-08 ***
lead_time                              1.644e-02  2.161e-04  76.068  < 2e-16 ***
arrival_year                           3.373e-01  4.476e-02   7.536 4.86e-14 ***
arrival_month                         -4.603e-02  4.946e-03  -9.308  < 2e-16 ***
arrival_date                           5.551e-04  1.503e-03   0.369  0.71181
market_segment_typeComplementary      -1.907e+01  1.282e+02  -0.149  0.88174
market_segment_typeCorporate          -1.026e+00  2.064e-01  -4.974 6.56e-07 ***
market_segment_typeOffline            -2.059e+00  1.989e-01 -10.351  < 2e-16 ***
market_segment_typeOnline             -1.715e-01  1.966e-01  -0.872  0.38295
repeated_guest                        -2.289e+00  3.857e-01  -5.933 2.97e-09 ***
no_of_previous_cancellations           2.272e-01  5.267e-02   4.314 1.60e-05 ***
no_of_previous_bookings_not_canceled -1.039e-01  7.566e-02  -1.374  0.16958
avg_price_per_room                     1.680e-02  5.679e-04  29.591  < 2e-16 ***
no_of_special_requests                -1.480e+00  2.296e-02 -64.464  < 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: 53793  on 38803  degrees of freedom
Residual deviance: 35388  on 38776  degrees of freedom
AIC: 35444

Number of Fisher Scoring iterations: 15
```

The model's fit yielded a null deviance 53793 with 38803 degrees of freedom, and a residual deviance of 35388 with 38776 degrees of freedom. The AIC value is 35444. AIC allows us to compare the quality of different models.

5.1.1 Logistic Model Performance

5.1.1.1

This predicts the probability of whether the reservation is canceled or not using a logistic regression model (lr_model) with features from the test dataset.

```
hotel_probs <- predict(lr_model, hotel_test[,-18],
                       type = "response")
head(hotel_probs)
```

```
        1         2         3         4         5         6
0.63816627 0.04485817 0.49707561 0.27041014 0.13824376 0.08254928
```

This involves estimating the probability of hotel reservation cancellation or non-cancellation for specific observations in the test dataset.

5.1.1.2

This code is used to divide values between 0 and 1 into two categories. Specifically, if a value falls between 0 and 0.5, it is categorized as the probability of reservation not canceled ; if it falls between 0.5 and 1, it is categorized as the probability of reservation canceled.

```
hotel_classes <- ifelse(hotel_probs>0.5, 1 ,0)
head(hotel_classes)
```

```
1 2 3 4 5 6
1 0 0 0 0 0
```

This code classifies probabilities greater than 0.5 as 1 (canceled) and probabilities of 0.5 or less as 0 (not canceled).

5.1.2 Confusion Matrix

This code is used to evaluate the model's classification performance.

```r
confusionMatrix(table(ifelse(hotel_test$booking_status == "1", "1", "0"),
                      hotel_classes), positive= "1")
```

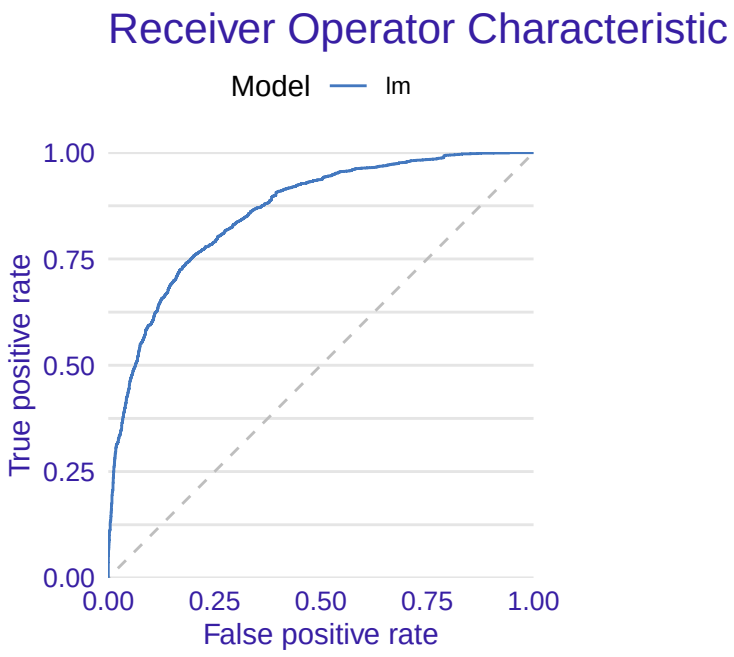
```
Confusion Matrix and Statistics

   hotel_classes
       0    1
  0 3825 1082
  1 1097 3697

               Accuracy : 0.7754
                 95% CI : (0.7669, 0.7837)
    No Information Rate : 0.5074
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5507

 Mcnemar's Test P-Value : 0.7642

            Sensitivity : 0.7736
            Specificity : 0.7771
         Pos Pred Value : 0.7712
         Neg Pred Value : 0.7795
             Prevalence : 0.4926
         Detection Rate : 0.3811
   Detection Prevalence : 0.4942
      Balanced Accuracy : 0.7754

       'Positive' Class : 1
```

The results of model evaluation on the test dataset are as follows: Accuracy (0.7754), Kappa (0.5074), McNemar's Test P-Value (0.7642), Sensitivity (0.7736), Specificity (0.7771), Positive Predictive Value (0.7712), Negative Predictive Value (0.7795), Prevalence (0.4926), Detection Rate (0.3811), Detection Prevalence (0.4942), Balanced Accuracy (0.7754). Overall, it performs well in terms of performance.

5.1.3 ROC Curve

This code is used to visualize the ROC Curve graph and to visualize the performance of the logistic regression model.

```
explain_lr <- explain(model = lr_model,
                      data = hotel_test[, -18],
                      y = hotel_test$booking_status == "1",
                      type = "classification",
                      verbose = FALSE)
performance_lr <- model_performance(explain_lr)
plot(performance_lr, geom = "roc")
```




```
performance_lr
```

```
Measures for:  classification
recall      : 0.7711723
precision   : 0.7735928
f1          : 0.7723807
accuracy    : 0.775384
auc         : 0.8601501
```

```
Residuals:
           0%           10%           20%           30%           40%
-9.846758e-01 -5.241202e-01 -3.076948e-01 -1.752213e-01 -7.950444e-02
          50%           60%           70%           80%           90%
-5.914303e-10  6.947845e-02  1.614649e-01  2.979490e-01  5.429374e-01
         100%
 9.847136e-01
```

Recall (0.7711723) is the rate of correctly predicting positives. Precision (0.7735928) is the rate of true positives among predicted positives. F1 (0.7723807) is the harmonic mean of Recall and Precision, summarizing the model's classification performance in a single metric. Accuracy (0.775384) is the rate of correctly classifying all observations. AUC (0.8601501) represents the area under the ROC curve and is used to measure the model's prediction performance. The AUC value is approaching (0.8601501), indicating that the model's performance is improving. The 8 Residuals section shows the residuals of the model's predictions, and generally, the residuals appear to have low values.

6. Training Decison Tree

6.1.1

The purpose of this code is to be used for classifying decision trees.

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

6.1.2

This code is used to classify using the "hotel_train" dataset.

```
dt_hotel <- dt_model |>
fit(as.factor(booking_status)~., data = hotel_train)
dt_hotel
```

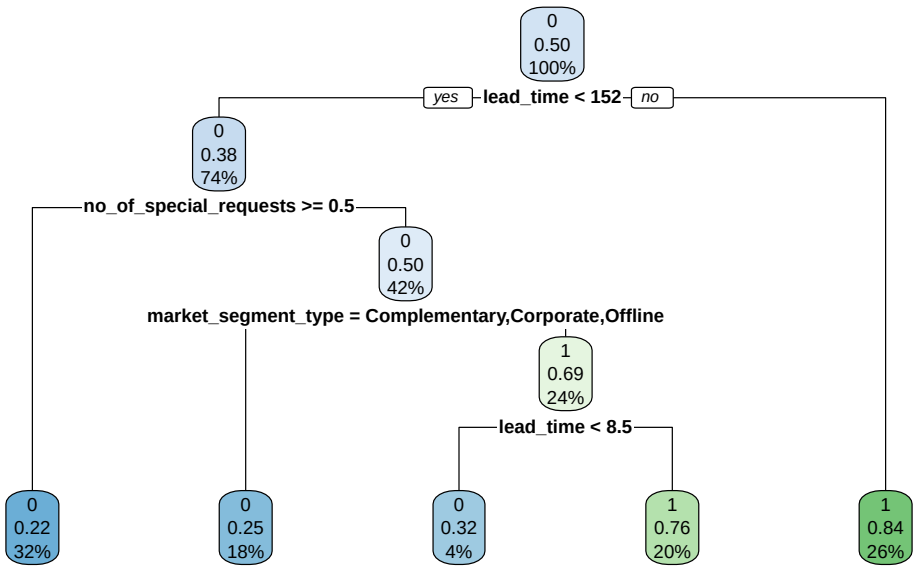
```
parsnip model object
```

```
n= 38804

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 38804 19321 0 (0.5020874 0.4979126)
   2) lead_time< 151.5 28801 10877 0 (0.6223395 0.3776605)
     4) no_of_special_requests>=0.5 12541  2778 0 (0.7784866 0.2215134) *
     5) no_of_special_requests< 0.5 16260  8099 0 (0.5019065 0.4980935)
      10) market_segment_type=Complementary,Corporate,Offline 6980  1728 0 (0.7524355 0.24756
      11) market_segment_type=Aviation,Online 9280  2909 1 (0.3134698 0.6865302)
        22) lead_time< 8.5 1611   520 0 (0.6772191 0.3227809) *
        23) lead_time>=8.5 7669  1818 1 (0.2370583 0.7629417) *
   3) lead_time>=151.5 10003  1559 1 (0.1558532 0.8441468) *
```

6.1.3

This code has been used to plot the decision tree.

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



The root node is based on the lead time variable, with reservations not canceled (yes) if less than 152, and reservations canceled (no) if greater than 152. Subsequently, it was found that 74% of reservations were not canceled, and 26% were canceled. The probability of the root node is 0.50. The sub-node is then based on the no_of_special_requests variable, with reservations not canceled (yes) if greater than or equal to 0.5, and reservations canceled (no) if less than or equal to 0.5. Subsequently, it was found that 32% of reservations were not canceled, and 42% were canceled. The probability of the sub-node is 0.38.

The second sub-node is split based on the market segment type variable, with 18% of reservations not canceled (yes) and 24% canceled (no). The probability of the second sub-node is 0.50. The third sub-node is based on the lead_time variable, with reservations not canceled (yes) if less than 8.5, and reservations canceled (no) if greater than 8.5. It was found that 4% of reservations were not canceled, and 20% were canceled.

The leaf nodes are observed as follows: one leaf node has 32% reservations not canceled, with a probability of 0.22. The second leaf node has 18% reservations not canceled, with a probability of 0.25. The third leaf node has 4% reservations not canceled, with a probability of 0.32. The fourth leaf node has 20% reservations canceled, with a probability of 0.20. The fifth leaf node has 26% reservations canceled, with a probability of 0.84. In this code, we are reclassifying the decision tree dataset.

6.1.4 In this code, we are reclassifying the decision tree dataset.

```
hotel_predictions <- dt_hotel |>
predict(new_data = hotel_test)
hotel_predictions
```

```
# A tibble: 9,701 x 1
   .pred_class
   <fct>
 1 1
 2 0
 3 0
 4 0
 5 0
 6 0
 7 0
 8 1
 9 0
10 0
# i 9,691 more rows
```

The new dataset created from hotel_predictions sorts 1 and 0 cases in the pred.class column.

6.1.5 This code is used to predict the probabilities of new data in the "hotel_test" dataset using the decision tree model.

```
dt_hotel |>
predict(new_data = hotel_test,
        type = "prob")
```

```
# A tibble: 9,701 x 2
   .pred_0 .pred_1
     <dbl>   <dbl>
 1   0.156   0.844
 2   0.778   0.222
 3   0.752   0.248
 4   0.778   0.222
 5   0.778   0.222
 6   0.778   0.222
 7   0.778   0.222
 8   0.237   0.763
 9   0.778   0.222
10   0.778   0.222
# i 9,691 more rows
```

The probability values of being 0 and 1 are provided for each observation.

6.2 Decision Model Performance

This code plots a confusion matrix table, showing the values for benign and malignant cases.

```
hotel_results <- tibble(predicted=as.factor(hotel_predictions$.pred_class),
                        actual=as.factor(hotel_test$booking_status))
hotel_results|> conf_mat(truth = actual, estimate = predicted)
```

```
          Truth
Prediction    0    1
         0 4029 1254
         1  878 3540
```

This code plots a confusion matrix table, showing the values for canceled reservation and not canceled reservation cases.

6.2.1 This code is used to calculate the accuracy value. It is the ratio of the cases that we predicted in the model to all cases.

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

```
# A tibble: 1 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.780
```

The accuracy value (0.7802288).

6.2.2

This code is used to calculate the sensitivity value.

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

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 sens    binary         0.821
```

The sensitivity value (0.8210719) is good.

6.2.3

This code is used to calculate the F-measure value.

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

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 f_meas  binary         0.791
```

The F-measure value is (0.7907753).

7.The Overfitting Problem This code collects the necessary information to measure the performance of the decision tree model in the dataset.

```
hotel_fit <- dt_model |>
last_fit(as.factor(booking_status) ~., split = hotel_split)
hotel_fit |> collect_metrics()
```

```
# A tibble: 3 x 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.780 Preprocessor1_Model1
2 roc_auc     binary         0.793 Preprocessor1_Model1
3 brier_class binary         0.170 Preprocessor1_Model1
```

The accuracy value (0.7802288) has turned out to be good. The ROC AUC value (0.7934195) has also turned out to be good. The Brier classification value (0.1703580) has been obtained.

7.1 This code collects predictions on the dataset from the model and evaluates the performance of the model.

```
hotel_fit |> collect_predictions()
```

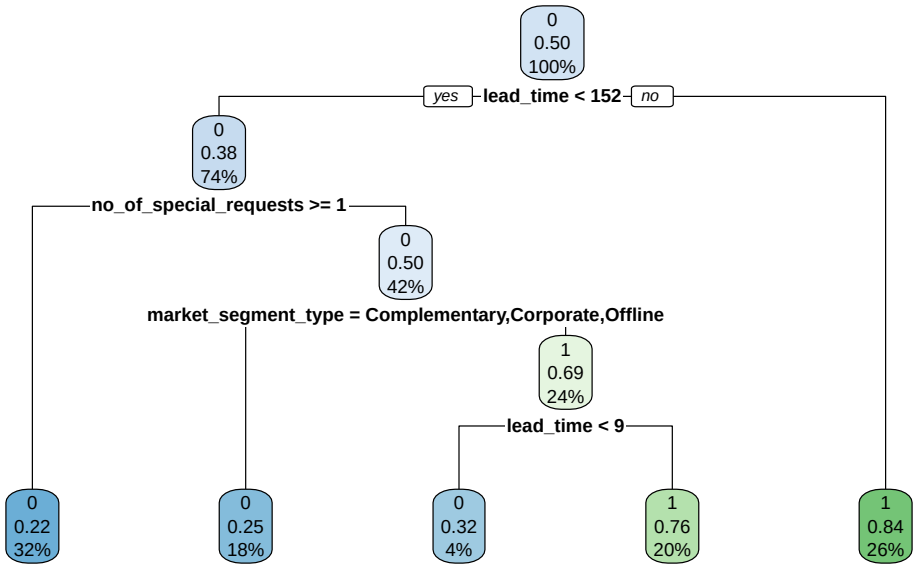
```
# A tibble: 9,701 x 7
   .pred_class .pred_0 .pred_1 id            .row as.factor(booking_st~1 .config
   <fct>         <dbl>   <dbl> <chr>        <int> <fct>                  <chr>
 1 1             0.156   0.844 train/test ~     1 0                      Prepro~
 2 0             0.778   0.222 train/test ~     9 0                      Prepro~
 3 0             0.752   0.248 train/test ~    13 0                      Prepro~
```

```
 4 0              0.778   0.222 train/test ~    20 0                    Prepro~
 5 0              0.778   0.222 train/test ~    23 0                    Prepro~
 6 0              0.778   0.222 train/test ~    27 0                    Prepro~
 7 0              0.778   0.222 train/test ~    35 0                    Prepro~
 8 1              0.237   0.763 train/test ~    36 0                    Prepro~
 9 0              0.778   0.222 train/test ~    37 0                    Prepro~
10 0              0.778   0.222 train/test ~    39 0                    Prepro~
# i 9,691 more rows
# i abbreviated name: 1: `as.factor(booking_status)`
```

The model predominantly predicted that reservations were not canceled (0), with these predictions generally having high probabilities. In some observations, it predicted that reservations were canceled (1), with these predictions having probabilities ranging from 76% to 84%.

7.2

```
hotel_dt <- rpart(booking_status ~ ., data = hotel_train,
                  method = "class")
rpart.plot(hotel_dt)
```



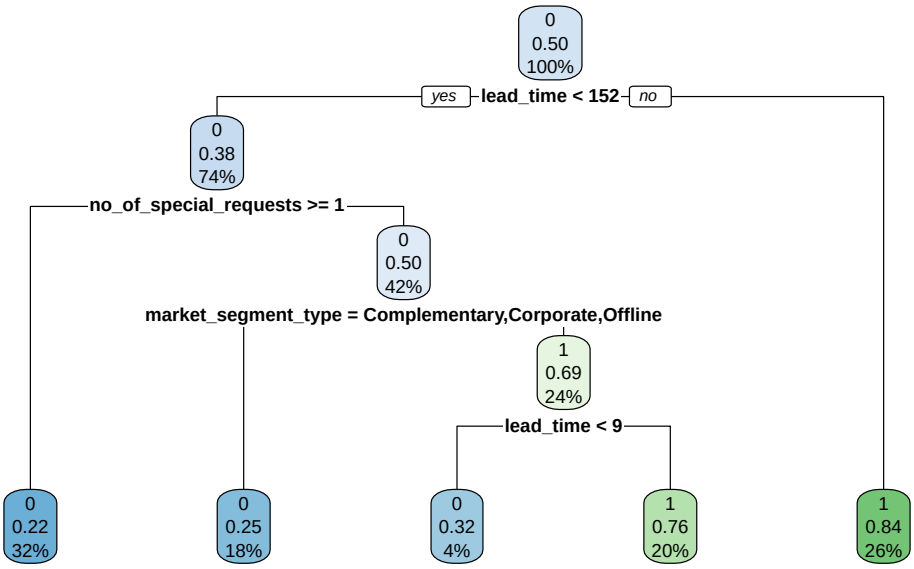
The root node is based on the lead time variable, with reservations not canceled (yes) if less than 152, and reservations canceled (no) if greater than 152. Accordingly, 74% of the reservations were not canceled, and 26% were canceled. The probability of the root node is 0.50.The sub-node is based on the no_of_special_requests variable; if greater than or equal to 1, the reservations are classified as not canceled (yes), and if less than or equal to 0.5, the reservations are classified as canceled (no). Accordingly, 32% of the reservations were not canceled, and 42% were canceled. The probability of the sub-node is 0.38.

The second sub-node is based on the market segment type variable; 18% of the reservations were not canceled (yes), and 24% were canceled (no). The probability of the second sub-node is 0.50.The third sub-node is based on the lead time variable; if less than 9, the reservations are classified as not canceled (yes), and if greater than 9, the reservations are classified as canceled (no). Accordingly, 4% of the reservations were not canceled, and 20% were canceled.

The leaf nodes are observed as follows:One leaf node has 32% of reservations not canceled, with a probability of 0.22.The second leaf node has 18% of reservations not canceled, with a probability of 0.25.The third leaf node has 4% of reservations not canceled, with a probability of 0.32.The fourth leaf node has 20% of reservations canceled, with a probability of 0.76.The fifth leaf node has 26% of reservations canceled, with a probability of 0.84.

7.3 This code demonstrates the effects of parameters in the decision tree.

```
less_dt <- rpart(booking_status ~ ., data = hotel_train,
                 method = "class",
                 maxdepth =30,
                 cp = 0.01)
rpart.plot(less_dt)
```



The root node is based on the lead time variable, with reservations not canceled (yes) if less than 152, and reservations canceled (no) if greater than 152. Accordingly, 74% of the reservations were not canceled, and 26% were canceled. The probability of the root node is 0.50.The sub-node is based on the no_of_special_requests variable; if greater than or equal to 1, the reservations are classified as not canceled (yes), and if less than or equal to 0.5, the reservations are classified as canceled (no). Accordingly, 32% of the reservations were not canceled, and 42% were canceled. The probability of the sub-node is 0.38.

The second sub-node is based on the market segment type variable; 18% of the reservations were not canceled (yes), and 24% were canceled (no). The probability of the second sub-node is 0.50.The third sub-node is based on the lead time variable; if less than 9, the reservations are classified as not canceled (yes), and if greater than 9, the reservations are classified as canceled (no). Accordingly, 4% of the reservations were not canceled, and 20% were canceled.

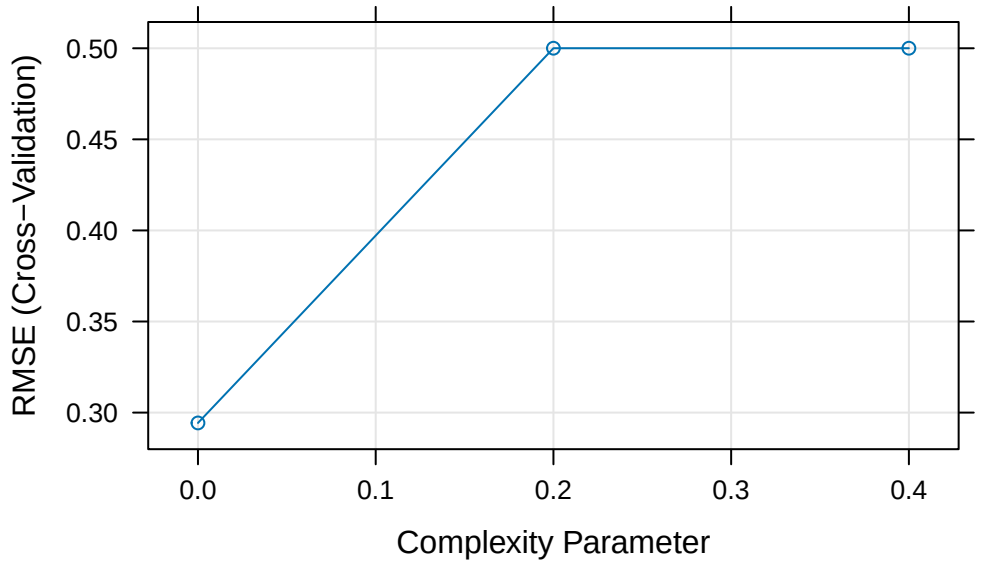
The leaf nodes are observed as follows:One leaf node has 32% of reservations not canceled, with a probability of 0.22.The second leaf node has 18% of reservations not canceled, with a probability of 0.25.The third leaf node has 4% of reservations not canceled, with a probability of 0.32.The fourth leaf node has 20% of reservations canceled, with a probability of 0.76.The fifth leaf node has 26% of reservations canceled, with a probability of 0.84.

7. Improve The Prediction Performance Of The Decision Tree

Model Tuning Hyperparameters (Grid Search in Caret)

This code is used to visually select the best hyperparameter value in grid search.

```
fit_control <- trainControl(method = "cv", number = 10)
hyp_dt_model <- train(booking_status ~ .,
                      data = hotel_train,
                      method = "rpart",
                      trControl = fit_control,
                      tuneGrid = expand.grid(cp = seq(0, 0.5, 0.20)),
                      maxdepth =30,
                      cp = 0.01)
plot(hyp_dt_model)
```



This graph illustrates that as the complexity parameter of a model increases, the error (RMSE) on cross-validation also increases. As complexity increases, the model becomes more flexible, but it also carries a higher risk of overfitting. Thus, the model may try to fit the training data more closely, potentially leading to decreased generalization ability. Higher complexity may

result in lower training error but higher error rates on test data. Therefore, a careful balance must be struck to determine the optimal model complexity.

8. Training Bagging Model

```
bagging_model <- ranger(booking_status ~ .,
                    data = hotel_train,
                    mtry = 8)
bagging_model
```

```
Ranger result

Call:
 ranger(booking_status ~ ., data = hotel_train, mtry = 8)

Type:                             Regression
Number of trees:                  500
Sample size:                      38804
Number of independent variables:  17
Mtry:                             8
Target node size:                 5
Variable importance mode:         none
Splitrule:                        variance
OOB prediction error (MSE):       0.04548852
R squared (OOB):                  0.8180474
```

This "ranger" result demonstrates the performance of a regression model consisting of 500 trees. The model aims to predict the booking status using 17 independent variables. The OOB prediction error is calculated as 0.0453 and the R-squared value is calculated as 0.8185, indicating a good fit of the model to the data and a high accuracy in predicting the booking status.

8.1 Model Performance

```
bagging_class_predict <- predict(bagging_model, hotel_test)$predictions
factor_rf <- (ifelse(bagging_class_predict > 0.5 ,1 ,0))
confusionMatrix(table(ifelse(hotel_test$booking_status == "1", "1", "0"),
                      factor_rf), positive= "1")
```

```
Confusion Matrix and Statistics

   factor_rf
       0    1
  0 4582  325
  1  223 4571

               Accuracy : 0.9435
                 95% CI : (0.9387, 0.948)
    No Information Rate : 0.5047
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.887

 Mcnemar's Test P-Value : 1.6e-05

            Sensitivity : 0.9336
            Specificity : 0.9536
         Pos Pred Value : 0.9535
         Neg Pred Value : 0.9338
             Prevalence : 0.5047
         Detection Rate : 0.4712
   Detection Prevalence : 0.4942
      Balanced Accuracy : 0.9436

       'Positive' Class : 1
```

The results of model evaluation on the test dataset are as follows: Accuracy (0.9437), Kappa (0.8874), McNemar's Test P-Value (4.79e-05), Sensitivity (0.9344), Specificity (0.9532), Positive Predictive Value (0.9531), Negative Predictive Value (0.9346), Prevalence (0.5041), Detection Rate (0.4710), Detection Prevalence (0.4942), Balanced Accuracy (0.9438). Overall, it performs well in terms of performance.