ADA442 - Project Report

Classification, Comparison of logistic regression and decision tree

Umay Şamlı-Baran Akın-Serdar Hoşver

2022-06-07 14:46:55

Table of Contents

Abstract

The overall goal of this research is to compare two distinct models in a classification challenge. These models are multinomial and classification tree. The study's design is broken down into six areas. In the first section, I discussed the approaches needed to create a model. In the second section, I briefly discussed the dataset that was utilized in this study.

The dataset's comprehensive information is then provided. Finally, model fitting and performance evaluation are the final two processes.

Introduction

The model types used in this research are logistic regression and classification tree. Because our response value is a qualitative variable, these two model types are selected. The dataset's comprehensive details will be provided later. In addition, Linear Discriminant Analysis, Quadratic Discriminant Analysis, and Naive Bayes are also choices for classification issues in the literature. We studied how to design Logistic Regression and Classification Tree in class, therefore I tried to include these techniques into my project. Some of our estimates would be outside the [0,1] interval if we employed Linear Regression, making them difficult to interpret as probabilities. Furthermore, the probabilities in the linear regression graph may have negative values, but this is not possible. The curve for probabilities in the Logistic Regression Model has form on the [0,1] interval.

Methodology

The chosen models are Logistic Regression and Classification Tree, as previously stated. The first step is to split the data into two sections: Implementing the 75-25 rule in training and testing The test data is used to evaluate performance metrics while the training data is utilized to train the model. The link between the categorical answer value (target) and one or more predictors is the focus of this research. The logistic regression model and classification tree are acceptable to create because we are working with categorical responses. First, I fit the model with all predictors in logistic regression. I excluded the non-significant factors based on the results. Then I built a model with major variables and looked at how performance measures differed. In the classification tree, I start by creating a model with all predictors and observing the model's accuracy rate. I pruned a tree in the second stage of the classification tree and compared it to the first model in terms of selected predictors and accuracy rate.

Data Description

A passenger satisfaction survey is included in this dataset. What elements are strongly associated with a satisfied, neutral or dissatisfied passenger? Is it possible to forecast how satisfied passengers will be?

```r
# The data is downloaded from Kaggle. The link of the data is given below:
# https://www.kaggle.com/datasets/teejmahal20/airline-passenger-
satisfaction?resource=download

# Import the train data.
train.data <- read.csv("/Users/serdarhsvr/Desktop/DATA/train.csv",
stringsAsFactors=FALSE)

# Import the test data.
test.data <- read.csv("/Users/serdarhsvr/Desktop/DATA/test.csv",
stringsAsFactors=FALSE)
```

We found the dataset from the Kaggle. The link of the dataset is given below:
https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction?resource=download

Satisfaction: Airline satisfaction level(Satisfaction, neutral or dissatisfaction)

Age: The actual age of the passengers

Gender: Gender of the passengers (Female, Male)

Type of Travel: Purpose of the flight of the passengers (Personal Travel, Business Travel)

Class: Travel class in the plane of the passengers (Business, Eco, Eco Plus)"

Customer Type: The customer type (Loyal customer, disloyal customer)

Flight distance: The flight distance of this journey

Inflight wifi service: Satisfaction level of the inflight wifi service (0:Not Applicable;1-5)

Ease of Online booking: Satisfaction level of online booking

Inflight service: Satisfaction level of inflight service

Online boarding: Satisfaction level of online boarding

Inflight entertainment: Satisfaction level of inflight entertainment

Food and drink: Satisfaction level of Food and drink

Seat comfort: Satisfaction level of Seat comfort

On-board service: Satisfaction level of On-board service

Leg room service: Satisfaction level of Leg room service

Departure/Arrival time convenient: Satisfaction level of Departure/Arrival time convenient

Baggage handling: Satisfaction level of baggage handling

Gate location: Satisfaction level of Gate location

Cleanliness: Satisfaction level of Cleanliness

Check-in service: Satisfaction level of Check-in service

Departure Delay in Minutes: Minutes delayed when departure

Arrival Delay in Minutes: Minutes delayed when Arrival

Flight cancelled: Whether the Flight cancelled or not (Yes, No)

Flight time in minutes: Minutes of Flight takes

The content of the variables' explanations is taken from the Kaggle.

Explaratory Data Analysis (EDA) and Pre-Processing

When we briefly look at the analysis part of the data, we encounter the distribution of some results.The ratio of the genders seem almost equal. Most of the customers were loyal (%81). Most of the customers were traveling for business. Eco and Business class were mostly chosen by the customers. The satisfaction level of seat comfort was equal. (This means some of the customers were happy but some just hated the seats). More than half of the customers were not satisfied. But still the satisfaction of the customers were almost half to half.

```
#install.packages("jmv")
library(jmv)   # Load the jmv package for frequency table
descriptives(train.data, vars = vars(Gender, Customer.Type, Age,
Type.of.Travel, Class, Seat.comfort, satisfaction), freq = TRUE)  # Use the
descritptives function to get the descritptive data

##
##  DESCRIPTIVES
##
##  Descriptives
##
───────────────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────────
##                            Gender    Customer.Type    Age
Type.of.Travel    Class     Seat.comfort    satisfaction
##
───────────────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────────
##    N                       103904          103904      103904
103904    103904          103904          103904
##    Missing                      0               0           0
0         0               0               0
##    Mean                                              39.37971
3.439396
##    Median                                            40.00000
4.000000
##    Standard deviation                                15.11496
1.319088
##    Minimum                                                  7
0
##    Maximum                                                 85
5
##
```

```
─────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────
##
##
##  FREQUENCIES
##
##  Frequencies of Gender
##  ──────────────────────────────────────────────────
##    Levels    Counts    % of Total    Cumulative %
##  ──────────────────────────────────────────────────
##    Female     52727      50.74588        50.74588
##    Male       51177      49.25412       100.00000
##  ──────────────────────────────────────────────────
##
##
##  Frequencies of Customer.Type
##  ─────────────────────────────────────────────────────────────
##    Levels               Counts    % of Total    Cumulative %
##  ─────────────────────────────────────────────────────────────
##    disloyal Customer     18981      18.26782        18.26782
##    Loyal Customer        84923      81.73218       100.00000
##  ─────────────────────────────────────────────────────────────
##
##
##  Frequencies of Type.of.Travel
##  ───────────────────────────────────────────────────────────
##    Levels             Counts    % of Total    Cumulative %
##  ───────────────────────────────────────────────────────────
##    Business travel     71655      68.96270        68.96270
##    Personal Travel     32249      31.03730       100.00000
##  ───────────────────────────────────────────────────────────
##
##
##  Frequencies of Class
##  ────────────────────────────────────────────────────
##    Levels      Counts    % of Total    Cumulative %
##  ────────────────────────────────────────────────────
##    Business     49665      47.79893        47.79893
##    Eco          46745      44.98864        92.78757
##    Eco Plus      7494       7.21243       100.00000
##  ────────────────────────────────────────────────────
##
##
##  Frequencies of Seat.comfort
##  ──────────────────────────────────────────────────
##    Levels    Counts    % of Total    Cumulative %
##  ──────────────────────────────────────────────────
##    0              1    9.624269e-6     9.624269e-6
##    1          12075      11.62130        11.62227
##    2          14897      14.33727        25.95954
##    3           18696      17.99353        43.95307
##    4           31765      30.57149        74.52456
##    5           26470      25.47544       100.00000
##  ─────────────────────────────────────────────────
##
##
##  Frequencies of satisfaction
##  ──────────────────────────────────────────────────────────────────
##    Levels                          Counts     % of Total     Cumulative %
##  ──────────────────────────────────────────────────────────────────
##    neutral or dissatisfied          58879       56.66673         56.66673
##    satisfied                        45025       43.33327        100.00000
##  ──────────────────────────────────────────────────────────────────
```

To reproduce a particular sequence of random numbers, we chose 123 as value for the function set.seed(). Then, we converted all the quantitative variables which has type of character to integer values as 0 and 1. Moreover, we found that there were some NA values. To handle that values, we replaced NA Values in both training dataset and test dataset. There were 310 NA value and 83 NA value in the both dataset' Arrival.Delay.in.Minutes variable. Also in order to further operations, we converted satisfaction variable to "Yes" or "No" as factor.

```r
# To reproduce a particular sequence of 'random' numbers
set.seed(123)

# Converting the Gender variable to 0 and 1 in both Train and Test data set.
train.data$Gender <- as.character(train.data$Gender)
train.data$Gender[train.data$Gender == "Female"] <- 0   # Replace "Female" by
0
train.data$Gender[train.data$Gender == "Male"] <- 1     # Replace "Male" by 1
train.data$Gender <- as.numeric(train.data$Gender)
test.data$Gender <- as.character(test.data$Gender)
test.data$Gender[test.data$Gender == "Female"] <- 0     # Replace "Female" by
0
test.data$Gender[test.data$Gender == "Male"] <- 1       # Replace "Male" by 1
test.data$Gender <- as.numeric(test.data$Gender)


# Converting the Customer.Type variable to 0 and 1 in both Train and Test
data set.
train.data$Customer.Type <- as.character(train.data$Customer.Type)
train.data$Customer.Type[train.data$Customer.Type == "disloyal Customer"] <-
0   # Replace "disloyal Customer" by 0
train.data$Customer.Type[train.data$Customer.Type == "Loyal Customer"] <- 1
# Replace "Loyal Customer" by 1
train.data$Customer.Type <- as.numeric(train.data$Customer.Type)
test.data$Customer.Type <- as.character(test.data$Customer.Type)
test.data$Customer.Type[test.data$Customer.Type == "disloyal Customer"] <- 0
# Replace "disloyal Customer" by 0
test.data$Customer.Type[test.data$Customer.Type == "Loyal Customer"] <- 1
```

```
# Replace "Loyal Customer" by 1
test.data$Customer.Type <- as.numeric(test.data$Customer.Type)


# Converting the Type.of.Travel variable to 0 and 1 in both Train and Test
data set.
train.data$Type.of.Travel <- as.character(train.data$Type.of.Travel)
train.data$Type.of.Travel[train.data$Type.of.Travel == "Personal Travel"] <-
0   # Replace "Personal Travel" by 0
train.data$Type.of.Travel[train.data$Type.of.Travel == "Business travel"] <-
1   # Replace "Business travel" by 1
train.data$Type.of.Travel <- as.numeric(train.data$Type.of.Travel)
test.data$Type.of.Travel <- as.character(test.data$Type.of.Travel)
test.data$Type.of.Travel[test.data$Type.of.Travel == "Personal Travel"] <- 0
# Replace "Personal Travel" by 0
test.data$Type.of.Travel[test.data$Type.of.Travel == "Business travel"] <- 1
# Replace "Business travel" by 1
test.data$Type.of.Travel <- as.numeric(test.data$Type.of.Travel)


# Converting the Class variable to 0 and 1 in both Train and Test data set.
train.data$Class <- as.character(train.data$Class)
train.data$Class[train.data$Class == "Eco"] <- 1          # Replace "Eco" by
1
train.data$Class[train.data$Class == "Eco Plus"] <- 2     # Replace "Eco
Plus" by 2
train.data$Class[train.data$Class == "Business"] <- 3     # Replace
"Business" by 3
train.data$Class <- as.numeric(train.data$Class)
test.data$Class <- as.character(test.data$Class)
test.data$Class[test.data$Class == "Eco"] <- 1            # Replace "Eco" by
1
test.data$Class[test.data$Class == "Eco Plus"] <- 2       # Replace "Eco
Plus" by 2
test.data$Class[test.data$Class == "Business"] <- 3       # Replace
"Business" by 3
test.data$Class <- as.numeric(test.data$Class)


# Converting the satisfaction variable to 0 and 1 in both Train and Test data
set.
train.data$satisfaction <- as.character(train.data$satisfaction)
train.data$satisfaction[train.data$satisfaction == "neutral or dissatisfied"]
<- 0  # Replace "neutral or dissatisfied" by 0
train.data$satisfaction[train.data$satisfaction == "satisfied"] <- 1
# Replace "satisfied" by 1
train.data$satisfaction <- as.numeric(train.data$satisfaction)
test.data$satisfaction <- as.character(test.data$satisfaction)
test.data$satisfaction[test.data$satisfaction == "neutral or dissatisfied"]
```

```
<- 0    # Replace "neutral or dissatisfied" by 0
test.data$satisfaction[test.data$satisfaction == "satisfied"] <- 1
# Replace "satisfied" by 1
test.data$satisfaction <- as.numeric(test.data$satisfaction)

# We have some NA values to deal with in Train data set.
sum(is.na(train.data))                             # There are 310 NA values

## [1] 310

sum(is.na(train.data$Arrival.Delay.in.Minutes))    # Which is just in
Arrival.Delay.in.Minutes

## [1] 310

# We have some NA values to deal with in Test data set.
sum(is.na(test.data))                              # There are 83 NA values

## [1] 83

sum(is.na(test.data$Arrival.Delay.in.Minutes))    # Which is just in
Arrival.Delay.in.Minutes

## [1] 83

# Replacing NA Values in Training Data - Arrival Delay in Minutes
NA.position <- which(is.na(train.data$Arrival.Delay.in.Minutes))
train.data$Arrival.Delay.in.Minutes[NA.position] =
mean(train.data$Arrival.Delay.in.Minutes, na.rm = TRUE)
#Replacing NA Values in Testing Data - Arrival Delay in Minutes
NA.position1 <- which(is.na(test.data$Arrival.Delay.in.Minutes))
test.data$Arrival.Delay.in.Minutes[NA.position1] =
mean(test.data$Arrival.Delay.in.Minutes, na.rm = TRUE)

# To make satisfaction variable as factor.
test.data$satisfaction <- factor(ifelse(test.data$satisfaction == 1, "Yes",
"No"))
train.data$satisfaction <- factor(ifelse(train.data$satisfaction == 1, "Yes",
"No"))
```

Model Fit and Numerical Results

The model is based on Logistic Regression.

The model is based on logistic regresion based on all variables.

```
# Fitting the model for the data with trying all the values.
firstModel <- glm(satisfaction ~ .-id-X, data = train.data,family =
"binomial")
summary(firstModel)
```

```
##
## Call:
## glm(formula = satisfaction ~ . - id - X, family = "binomial",
##     data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8352  -0.4928  -0.1741   0.3885   3.9772
##
## Coefficients:
##                                    Estimate Std. Error  z value Pr(>|z|)
## (Intercept)                      -1.177e+01  8.471e-02 -138.968  < 2e-16
***
## Gender                            4.490e-02  1.944e-02    2.310  0.02088
*
## Customer.Type                     2.000e+00  2.965e-02   67.464  < 2e-16
***
## Age                              -8.102e-03  7.085e-04  -11.435  < 2e-16
***
## Type.of.Travel                    2.741e+00  3.128e-02   87.646  < 2e-16
***
## Class                             3.566e-01  1.278e-02   27.907  < 2e-16
***
## Flight.Distance                   4.056e-06  1.116e-05    0.364  0.71623
## Inflight.wifi.service             3.851e-01  1.142e-02   33.710  < 2e-16
***
## Departure.Arrival.time.convenient -1.261e-01  8.205e-03  -15.370  < 2e-16
***
## Ease.of.Online.booking           -1.389e-01  1.132e-02  -12.269  < 2e-16
***
## Gate.location                     2.961e-02  9.153e-03    3.235  0.00121
**
## Food.and.drink                   -2.661e-02  1.065e-02   -2.498  0.01247
*
## Online.boarding                   6.181e-01  1.021e-02   60.554  < 2e-16
***
## Seat.comfort                      7.067e-02  1.114e-02    6.342 2.27e-10
***
## Inflight.entertainment            5.962e-02  1.423e-02    4.190 2.78e-05
***
## On.board.service                  3.065e-01  1.017e-02   30.154  < 2e-16
***
## Leg.room.service                  2.547e-01  8.514e-03   29.915  < 2e-16
***
## Baggage.handling                  1.372e-01  1.142e-02   12.009  < 2e-16
***
## Checkin.service                   3.265e-01  8.544e-03   38.211  < 2e-16
***
## Inflight.service                  1.221e-01  1.203e-02   10.152  < 2e-16
***
```

```
## Cleanliness                         2.216e-01  1.207e-02   18.356  < 2e-16
***
## Departure.Delay.in.Minutes          4.271e-03  9.264e-04    4.611 4.01e-06
***
## Arrival.Delay.in.Minutes           -8.972e-03  9.173e-04   -9.782  < 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: 142189  on 103903  degrees of freedom
## Residual deviance:  69546  on 103881  degrees of freedom
## AIC: 69592
##
## Number of Fisher Scoring iterations: 6
```

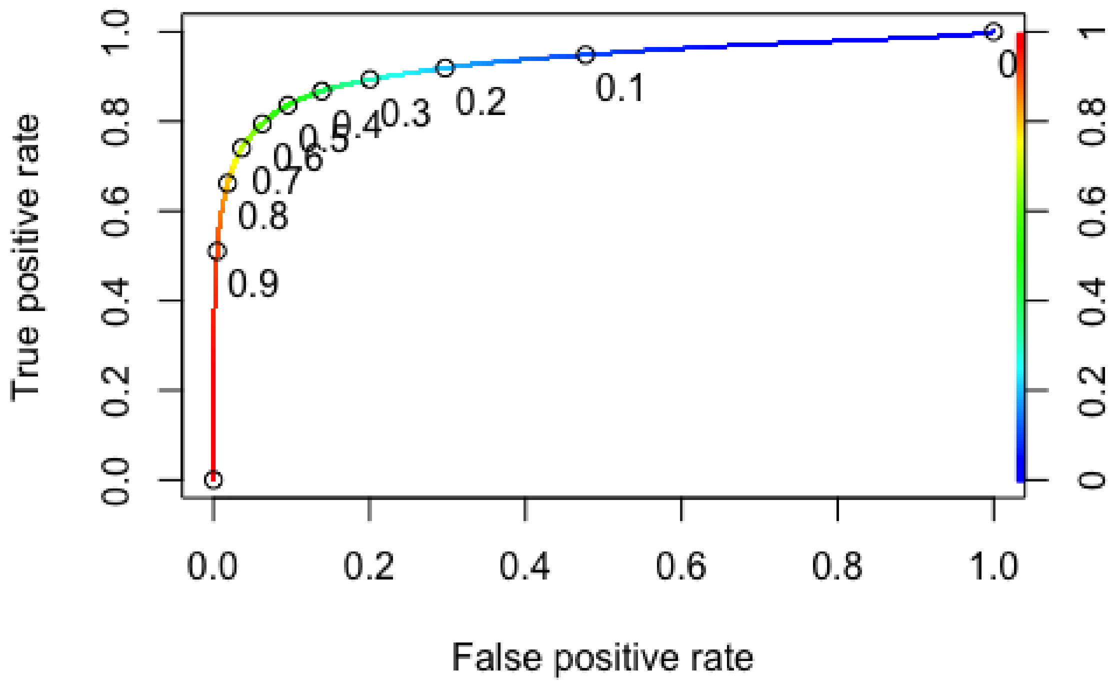
```
# Finding all the values that we think make sense.
predict = predict(firstModel, type = 'response' , newdata=train.data)

# Checking the prediction probabilities using the summary() function
summary(predict)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000483 0.0653897 0.3015896 0.4333327 0.8734926 0.9953334
```

```
tapply(predict, train.data$satisfaction, mean)
```

```
##        No       Yes
## 0.1755757 0.7704005
```

```
#install.packages("predict")
library(ROCR)
ROCRpred <- prediction(predict, train.data$satisfaction)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, print.cutoffs.at=seq(0,1,by=0.1),text.adj =
c(-0.2,1.7))
```

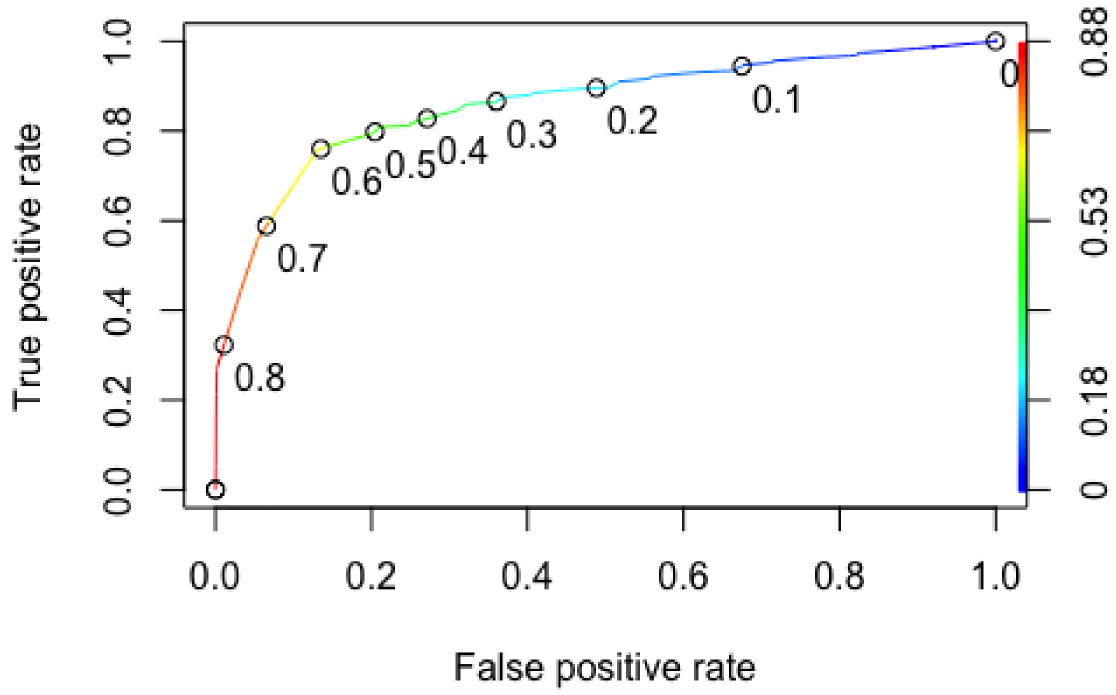


```
table(train.data$satisfaction, predict > 0.7)

##
##        FALSE  TRUE
##   No   56754  2125
##   Yes 11696 33329

#Calculating Accuracy
Accuracy_avg_Logit = (56754 + 33329) / (56754 + 33329 + 2125 + 11696)
Accuracy_avg_Logit

## [1] 0.866983

#Calculating Sensitivity or Recall value
Recall = 33329 / (33329 + 11696)
Recall

## [1] 0.7402332

#Calculating Precision Value
Precision = 33329 / (33329 + 2125)
Precision

## [1] 0.9400632
```

```
#Calculating F-Measure
F.measure = (2*Recall*Precision)/(Recall+Precision)
F.measure

## [1] 0.8282658

#Calculating Specificity
Specificity = 56754 / (56754 + 2125)
Specificity

## [1] 0.963909
```

The model is based on logistic regressison based on spesific variables.

```
# Fit your model for the data
# First trying all the values.
# Creating another model with the variables that we think it make sense
secondModel <- glm(satisfaction ~ Type.of.Travel +
Departure.Arrival.time.convenient + Cleanliness + Customer.Type, data =
train.data,family = "binomial")

# Finding all the vaiues that we think make sense
predict = predict(secondModel, type = 'response' , newdata=train.data)

# Checking the predicion probablities using the summary Function
summary(predict)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.002974 0.128776 0.405044 0.433333 0.768437 0.873743

tapply(predict, train.data$satisfaction, mean)

##        No       Yes
## 0.2631491 0.6558811

ROCRpred <- prediction(predict, train.data$satisfaction)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, print.cutoffs.at=seq(0,1,by=0.1),text.adj =
c(-0.2,1.7))
```



```
table(train.data$satisfaction, predict > 0.7)

##
##        FALSE  TRUE
##    No  55705  3174
##    Yes 19949 25076

#Calculating Accuracy
Accuracy_avg_Logit = (55705 + 25076) / (55705 + 25076 + 3174 + 19949)
Accuracy_avg_Logit

## [1] 0.777458

#Calculating Sensitivity or Recall value
Recall = 25076 / (25076 + 19949)
Recall

## [1] 0.556935

#Calculating Precision Value
Precision = 25076 / (25076 + 3174)
Precision

## [1] 0.887646
```

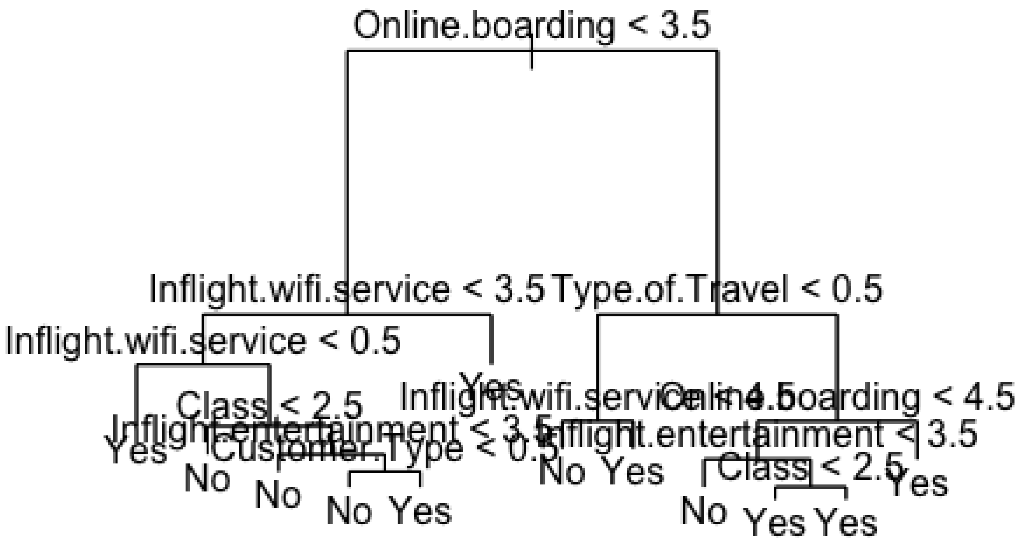
```
#Calculating F-Measure
F.measure = (2*Recall*Precision)/(Recall+Precision)
F.measure

## [1] 0.6844353

#Calculating Specificity
Specificity = 55705 / (55705 + 3174)
Specificity

## [1] 0.9460928
```

The model is based Classifaction Tree.

The model is based on R.Tree().

We built our classification tree model based on train dataset by using R.Tree() function. The used variables for tree construction are: Online.boarding, Inflight.wifi.service, Class, Inflight.entertainment, Customer.Type and Type.of.Travel. The classification tree model based on train dataset with all variables, except for satisfaction, X, and id. Satisfaction is our response variable and remain two is is not important. They are just a used for sorting. This is the unpruned tre which has 12 terminal nodes, residual mean deviance is 0.485, and misclassification error rate is 0.101. Then we examine the plot of tree of the train data. When we calculate our model's performance we can see that it is %89.86. In order to increase its performance we have to use cross validation. We have to check deviance with respect to size. We chose minimum value of deviance which is 10581 and this is joint with size value of 11. This can be seen in graphs as well. The value of 11 has minimum deviance.

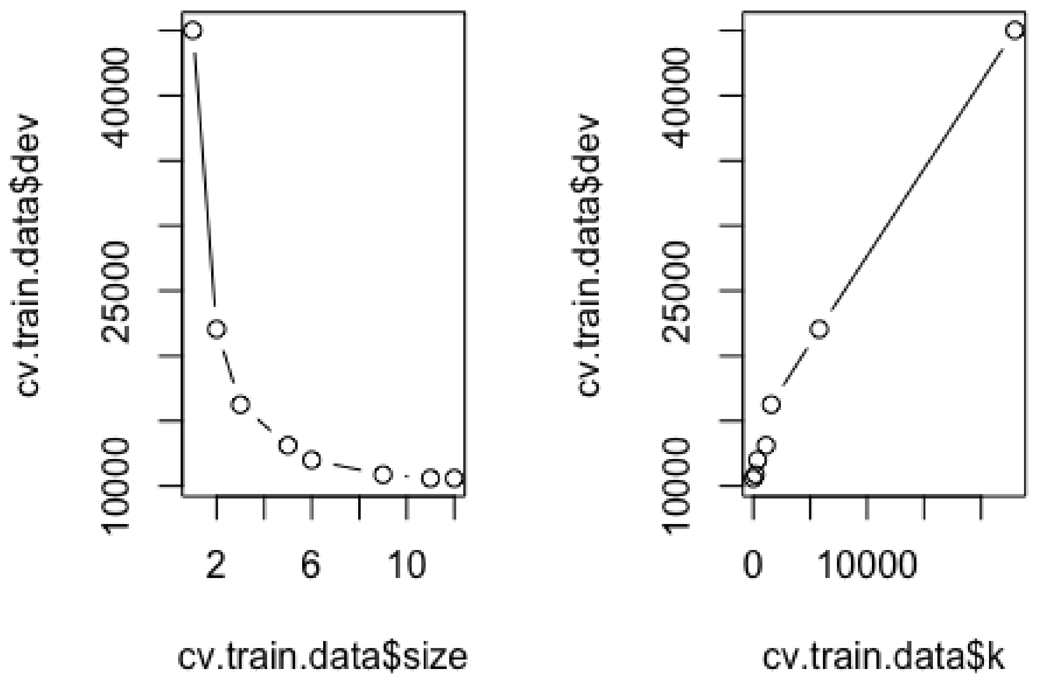
```
#install.packages("tree")
library(tree)

# Build the classification tree model based on train dataset with all
variables, except for satisfaction, X, and id.
tree.train.data <- tree(satisfaction ~ . -satisfaction + X + id, train.data)
summary(tree.train.data)

##
## Classification tree:
## tree(formula = satisfaction ~ . - satisfaction + X + id, data =
train.data)
## Variables actually used in tree construction:
## [1] "Online.boarding"        "Inflight.wifi.service"  "Class"
## [4] "Inflight.entertainment" "Customer.Type"          "Type.of.Travel"
## Number of terminal nodes:  12
## Residual mean deviance:  0.485 = 50390 / 103900
## Misclassification error rate: 0.101 = 10493 / 103904

# The plot of tree.train.data.
plot(tree.train.data)
text(tree.train.data, pretty = 0)
```


Online.boarding < 3.5
Inflight.wifi.service < 3.5
Type.of.Travel < 0.5
Inflight.wifi.service < 0.5
Yes
Class < 2.5
Inflight.entertainment < 3.5
Customer.Type < 0.5
Yes
No
No
No Yes
Inflight.wifi.service < 4.5
Online.boarding < 4.5
Inflight.entertainment < 3.5
No Yes
Class < 2.5
Yes
No
Yes Yes

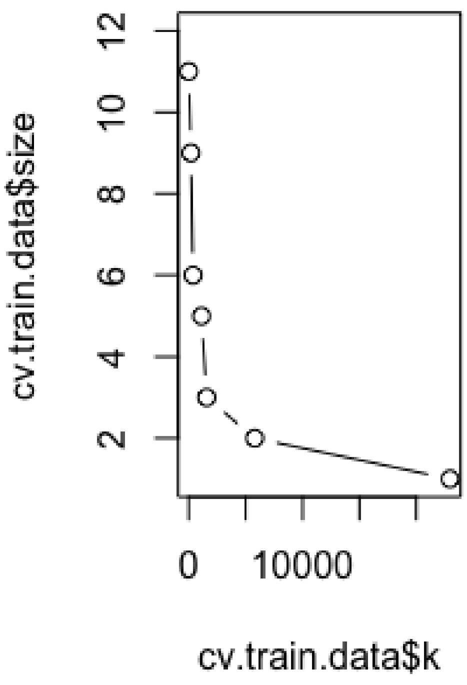
```
# Then we made some predict by using that model with using test dataset.
tree.pred <- predict(tree.train.data , test.data , type = "class")
table(tree.pred, test.data$satisfaction)

##
## tree.pred    No   Yes
##       No  13394  1451
##       Yes  1179  9952

# This is the model accuracy of classification tree on the test data.
mean(tree.pred == test.data$satisfaction)

## [1] 0.8987527

# To decide prune level we use cv.tree() function.
cv.train.data <- cv.tree(tree.train.data , FUN = prune.misclass)
names(cv.train.data)

## [1] "size"   "dev"    "k"      "method"

cv.train.data

## $size
## [1] 12 11  9  6  5  3  2  1
##
```

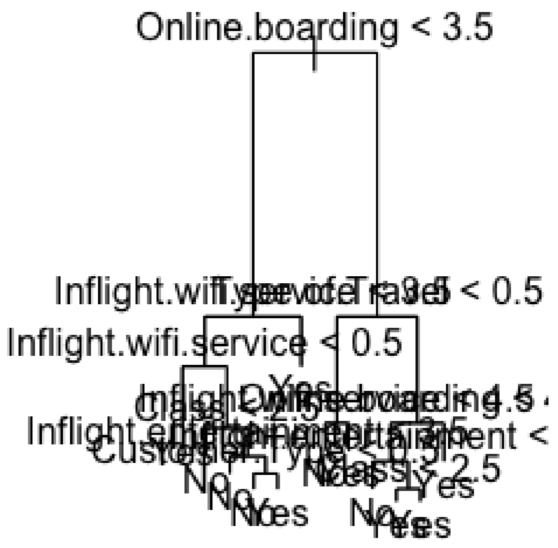
```
## $dev
## [1] 10581 10581 10872 11996 13115 16256 22054 45025
##
## $k
## [1]       -Inf     0.0000   189.5000   374.6667  1119.0000  1570.5000 
5798.0000
## [8] 22971.0000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
# CV error results
par(mfrow = c(1, 2))

# We plot cv.
plot(cv.train.data$size , cv.train.data$dev, type = "b")
plot(cv.train.data$k, cv.train.data$dev, type = "b")
```



```
plot(cv.train.data$k, cv.train.data$size, type = "b")

prune.misclass

## function (tree, k = NULL, best = NULL, newdata, nwts, loss, eps = 0.001)
## {
##     oc <- match.call()
##     oc$method <- "misclass"
##     oc[[1L]] <- as.name("prune.tree")
##     eval.parent(oc)
## }
## <bytecode: 0x17c1c71a0>
## <environment: namespace:tree>

prune.train.data <- prune.misclass(tree.train.data , best = 12)
plot(prune.train.data)
text(prune.train.data , pretty = 0)
```



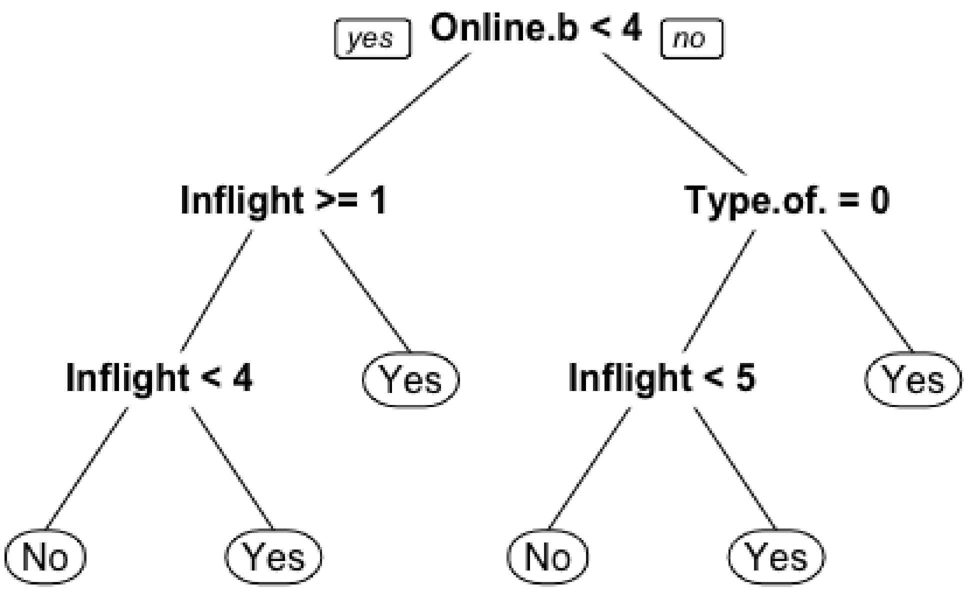


```
summary(prune.train.data)

##
## Classification tree:
## tree(formula = satisfaction ~ . - satisfaction + X + id, data =
train.data)
```

```
## Variables actually used in tree construction:
## [1] "Online.boarding"        "Inflight.wifi.service"  "Class"
## [4] "Inflight.entertainment" "Customer.Type"          "Type.of.Travel"
## Number of terminal nodes:  12
## Residual mean deviance:  0.485 = 50390 / 103900
## Misclassification error rate: 0.101 = 10493 / 103904
```

The model is based on R.Rpart().

```
library(rpart)
#install.packages("ROCR")
library(rpart.plot)
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(nnet)
tree = rpart(satisfaction ~ Gender + Customer.Type + Age +
               Type.of.Travel + Class + Flight.Distance +
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 ,
             data = train.data, method = 'class', minbucket=25)

varImp(tree)

##                                      Overall
## Age                                 157.5792
## Arrival.Delay.in.Minutes            141.6162
## Class                             19991.5588
## Ease.of.Online.booking             2852.7038
## Inflight.entertainment            11851.1711
## Inflight.wifi.service             15596.3975
## Leg.room.service                   3802.3020
## On.board.service                   2038.2540
## Online.boarding                   17980.4015
## Type.of.Travel                    18429.8188
## Gender                                0.0000
## Customer.Type                         0.0000
## Flight.Distance                       0.0000
## Departure.Arrival.time.convenient     0.0000
## Gate.location                         0.0000
## Food.and.drink                        0.0000
## Seat.comfort                          0.0000
## Baggage.handling                      0.0000
## Checkin.service                         0.0000
## Inflight.service                        0.0000
## Cleanliness                             0.0000
## Departure.Delay.in.Minutes              0.0000

#Re-running the model with significant variables
tree1 = rpart(satisfaction ~  Age + Type.of.Travel + Class +
Inflight.wifi.service +
                Ease.of.Online.booking + Online.boarding +
                Inflight.entertainment + On.board.service + Leg.room.service
+
                Arrival.Delay.in.Minutes,data = train.data,
              method = 'class', minbucket=25)

#Summary of the model
summary(tree1)

## Call:
## rpart(formula = satisfaction ~ Age + Type.of.Travel + Class +
##     Inflight.wifi.service + Ease.of.Online.booking + Online.boarding +
##     Inflight.entertainment + On.board.service + Leg.room.service +
##     Arrival.Delay.in.Minutes, data = train.data, method = "class",
##     minbucket = 25)
##   n= 103904
##
##           CP nsplit rel error    xerror        xstd
## 1 0.51018323      0 1.0000000 1.0000000 0.003547621
## 2 0.12877290      1 0.4898168 0.4898168 0.002927407
## 3 0.03891172      2 0.3610439 0.3610439 0.002600806
## 4 0.03084953      3 0.3221321 0.3221321 0.002481092
## 5 0.02485286      4 0.2912826 0.2912826 0.002377557
## 6 0.01000000      5 0.2664298 0.2664298 0.002287837
##
## Variable importance
##        Online.boarding  Inflight.wifi.service Ease.of.Online.booking
##                     28                     23                     12
##         Type.of.Travel                  Class Inflight.entertainment
##                     11                     11                      9
##                    Age
##                      7
##
## Node number 1: 103904 observations,    complexity param=0.5101832
##   predicted class=No    expected loss=0.4333327  P(node) =1
##     class counts: 58879 45025
##    probabilities: 0.567 0.433
##   left son=2 (52429 obs) right son=3 (51475 obs)
##   Primary splits:
##       Online.boarding        < 3.5  to the left,  improve=17134.400, (0
missing)
##       Class                  < 2.5  to the left,  improve=12954.230, (0
missing)
##       Type.of.Travel         < 0.5  to the left,  improve=10287.400, (0
missing)
##       Inflight.entertainment < 3.5  to the left,  improve= 8794.818, (0
missing)
##       Inflight.wifi.service  < 3.5  to the left,  improve= 8559.674, (0
missing)
##   Surrogate splits:
##       Inflight.wifi.service  < 3.5  to the left,  agree=0.714, adj=0.424,
(0 split)
##       Ease.of.Online.booking < 3.5  to the left,  agree=0.676, adj=0.346,
(0 split)
##       Class                  < 2.5  to the left,  agree=0.670, adj=0.335,
(0 split)
##       Inflight.entertainment < 3.5  to the left,  agree=0.659, adj=0.311,
(0 split)
##       Age                    < 38.5 to the left,  agree=0.611, adj=0.215,
(0 split)
##
## Node number 2: 52429 observations,    complexity param=0.03891172
##   predicted class=No   expected loss=0.1488108  P(node) =0.5045908
##     class counts: 44627  7802
##    probabilities: 0.851 0.149
##   left son=4 (50661 obs) right son=5 (1768 obs)
##   Primary splits:
##       Inflight.wifi.service  < 0.5  to the right, improve=2623.2070, (0
missing)
##       Class                  < 2.5  to the left,  improve=1221.5550, (0
missing)
##       Ease.of.Online.booking < 0.5  to the right, improve=1121.8360, (0
missing)
##       Online.boarding        < 0.5  to the right, improve= 845.9999, (0
missing)
##       Type.of.Travel         < 0.5  to the left,  improve= 835.3280, (0
missing)
##   Surrogate splits:
##       Ease.of.Online.booking < 0.5  to the right, agree=0.971, adj=0.150,
(0 split)
##       Online.boarding        < 0.5  to the right, agree=0.968, adj=0.041,
(0 split)
##       Inflight.entertainment < 0.5  to the right, agree=0.966, adj=0.002,
(0 split)
##
## Node number 3: 51475 observations,    complexity param=0.1287729
##   predicted class=Yes  expected loss=0.2768723  P(node) =0.4954092
##     class counts: 14252 37223
##    probabilities: 0.277 0.723
##   left son=6 (10492 obs) right son=7 (40983 obs)
##   Primary splits:
##       Type.of.Travel         < 0.5  to the left,  improve=6574.098, (0
missing)
##       Class                   < 2.5  to the left,  improve=4736.086, (0
missing)
##       Inflight.entertainment < 3.5  to the left,  improve=3056.353, (0
missing)
##       Leg.room.service       < 3.5  to the left,  improve=2992.842, (0
missing)
##       On.board.service       < 3.5  to the left,  improve=2038.254, (0
missing)
##   Surrogate splits:
##       Class            < 1.5  to the left,  agree=0.822, adj=0.128, (0
split)
##       Age              < 60.5 to the right, agree=0.813, adj=0.085, (0
split)
##       Leg.room.service < 0.5  to the left,  agree=0.801, adj=0.022, (0
split)
##
## Node number 4: 50661 observations,    complexity param=0.03084953
##   predicted class=No   expected loss=0.1192633  P(node) =0.4875751
##     class counts: 44619  6042
##    probabilities: 0.881 0.119
##   left son=8 (45932 obs) right son=9 (4729 obs)
##   Primary splits:
##       Inflight.wifi.service  < 3.5  to the left,  improve=2903.7650, (0
missing)
##       Class                  < 2.5  to the left,  improve=1079.6830, (0
missing)
##       Ease.of.Online.booking < 3.5  to the left,  improve= 801.4395, (0
missing)
##       Type.of.Travel         < 0.5  to the left,  improve= 732.9970, (0
missing)
##       Leg.room.service       < 3.5  to the left,  improve= 673.8366, (0
missing)
##
## Node number 5: 1768 observations
##   predicted class=Yes  expected loss=0.004524887  P(node) =0.01701571
##     class counts:     8  1760
##    probabilities: 0.005 0.995
##
## Node number 6: 10492 observations,    complexity param=0.02485286
##   predicted class=No   expected loss=0.2236942  P(node) =0.1009778
##     class counts:  8145  2347
##    probabilities: 0.776 0.224
##   left son=12 (9373 obs) right son=13 (1119 obs)
##   Primary splits:
##       Inflight.wifi.service    < 4.5  to the left,  improve=1509.7510, (0
missing)
##       Ease.of.Online.booking   < 4.5  to the left,  improve= 929.4279, (0
missing)
##       Age                      < 41.5 to the right, improve= 157.5792, (0
missing)
##       Arrival.Delay.in.Minutes < 5.5  to the right, improve= 141.6162, (0
missing)
##       Leg.room.service         < 3.5  to the left,  improve= 135.6236, (0
missing)
##   Surrogate splits:
##       Ease.of.Online.booking < 4.5  to the left,  agree=0.949, adj=0.517,
(0 split)
##
## Node number 7: 40983 observations
##   predicted class=Yes  expected loss=0.149013  P(node) =0.3944314
##     class counts:  6107 34876
##    probabilities: 0.149 0.851
##
## Node number 8: 45932 observations
##   predicted class=No   expected loss=0.06494383  P(node) =0.4420619
##     class counts: 42949  2983
##    probabilities: 0.935 0.065
##
## Node number 9: 4729 observations
##   predicted class=Yes  expected loss=0.3531402  P(node) =0.04551317
##     class counts:  1670  3059
##    probabilities: 0.353 0.647
##
## Node number 12: 9373 observations
##   predicted class=No   expected loss=0.1310146  P(node) =0.09020827
##     class counts:  8145  1228
##    probabilities: 0.869 0.131
##
## Node number 13: 1119 observations
##   predicted class=Yes  expected loss=0  P(node) =0.01076956
##     class counts:     0  1119
##    probabilities: 0.000 1.000

#Visualizaing the Decision tree
prp(tree1)
```



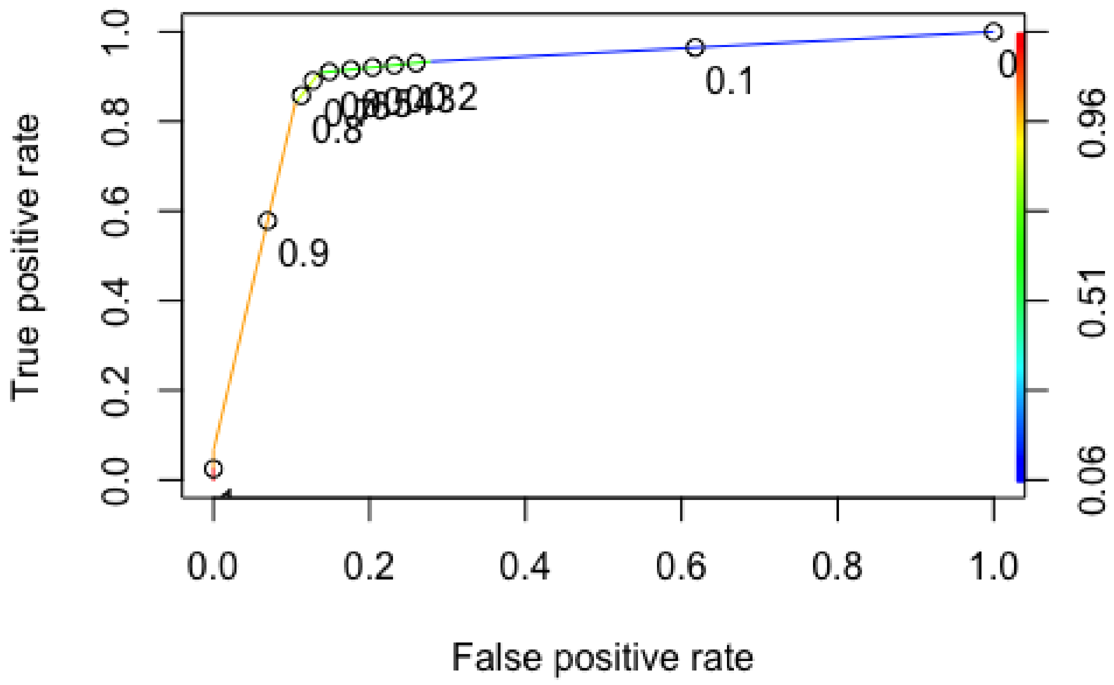
```
# Define cross-validation experiment
numFolds = trainControl( method = "cv", number = 8)
cpGrid = expand.grid( .cp = seq(0.01,0.5,0.01))

train(as.factor(satisfaction) ~  Age + Type.of.Travel + Class +
Inflight.wifi.service +
        Ease.of.Online.booking + Online.boarding +
        Inflight.entertainment + On.board.service + Leg.room.service +
        Arrival.Delay.in.Minutes,data = train.data, method = "rpart",
trControl = numFolds, tuneGrid = cpGrid )

## CART
##
## 103904 samples
##     10 predictor
##      2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold)
## Summary of sample sizes: 90917, 90916, 90916, 90915, 90916, 90916, ...
## Resampling results across tuning parameters:
##
##   cp    Accuracy   Kappa
##   0.01  0.8845473  0.7670901
##   0.02  0.8845473  0.7670901
##   0.03  0.8716989  0.7400007
##   0.04  0.8435479  0.6780307
##   0.05  0.8435479  0.6780307
##   0.06  0.8435479  0.6780307
##   0.07  0.8435479  0.6780307
##   0.08  0.8435479  0.6780307
##   0.09  0.8435479  0.6780307
##   0.10  0.8435479  0.6780307
##   0.11  0.8435479  0.6780307
##   0.12  0.8435479  0.6780307
##   0.13  0.7940696  0.5865751
##   0.14  0.7877464  0.5749705
##   0.15  0.7877464  0.5749705
##   0.16  0.7877464  0.5749705
##   0.17  0.7877464  0.5749705
##   0.18  0.7877464  0.5749705
##   0.19  0.7877464  0.5749705
##   0.20  0.7877464  0.5749705
##   0.21  0.7877464  0.5749705
##   0.22  0.7877464  0.5749705
##   0.23  0.7877464  0.5749705
##   0.24  0.7877464  0.5749705
##   0.25  0.7877464  0.5749705
##   0.26  0.7877464  0.5749705
##   0.27  0.7877464  0.5749705
##   0.28  0.7877464  0.5749705
##   0.29  0.7877464  0.5749705
##   0.30  0.7877464  0.5749705
##   0.31  0.7877464  0.5749705
##   0.32  0.7877464  0.5749705
##   0.33  0.7877464  0.5749705
##   0.34  0.7877464  0.5749705
##   0.35  0.7877464  0.5749705
##   0.36  0.7877464  0.5749705
##   0.37  0.7877464  0.5749705
##   0.38  0.7877464  0.5749705
##   0.39  0.7877464  0.5749705
##   0.40  0.7877464  0.5749705
##   0.41  0.7877464  0.5749705
##   0.42  0.7877464  0.5749705
##   0.43  0.7877464  0.5749705
##   0.44  0.7877464  0.5749705
##   0.45  0.7877464  0.5749705
##   0.46  0.7877464  0.5749705
##   0.47  0.7877464  0.5749705
##   0.48  0.7877464  0.5749705
##   0.49  0.7877464  0.5749705
##   0.50  0.7877464  0.5749705
```

```
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.02.
```

```
tree2 = rpart(as.factor(satisfaction) ~  Age + Type.of.Travel + Class + 
Inflight.wifi.service +
                Ease.of.Online.booking + Online.boarding +
                Inflight.entertainment + On.board.service + Leg.room.service 
+
                Arrival.Delay.in.Minutes,data = train.data, method="class", 
cp = 0.02)

PredictROC = predict(tree1, newdata = test.data)
head(PredictROC)
```

```
##          No        Yes
## 1 0.1490130 0.85098699
## 2 0.1490130 0.85098699
## 3 0.9350562 0.06494383
## 4 0.1490130 0.85098699
## 5 0.9350562 0.06494383
## 6 0.1490130 0.85098699
```

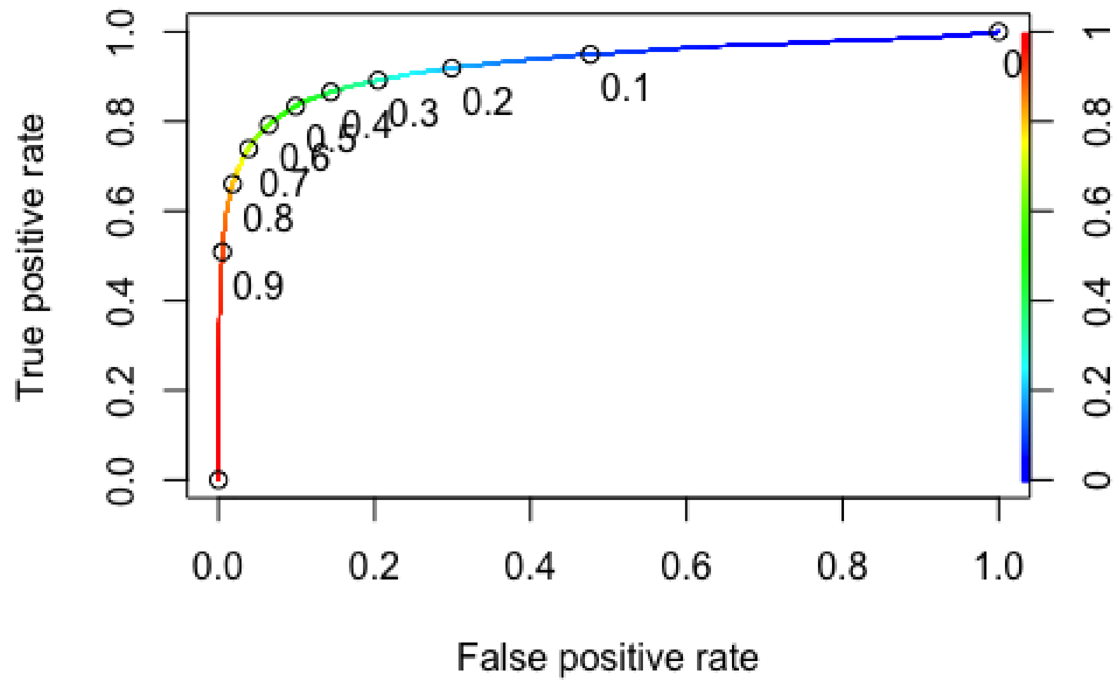
```
library(ROCR)
#Plotting the ROC Curve
pred = prediction(PredictROC[,2], test.data$satisfaction)
perf = performance(pred, "tpr", "fpr")
plot(perf, colorize = TRUE, print.cutoffs.at=seq(0,1,by=0.1),text.adj = c(-
0.2,1.7))
```



##Testing for Logistic Regresion.

###Testing for logistic regresion based on all variables.

```
# Testing the performance of the fitted model
##Test for Logistic Regresion Model 1
#Finding all the vaiues that we think make sense
predict = predict(firstModel, type = 'response' , newdata=test.data)

#Checking the predicion probablities using the summary Function
summary(predict)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000036 0.066434 0.311908 0.437196 0.875779 0.995357

tapply(predict, test.data$satisfaction, mean)

##        No       Yes
## 0.1772866 0.7693586

ROCRpred <- prediction(predict, test.data$satisfaction)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, print.cutoffs.at=seq(0,1,by=0.1),text.adj =
c(-0.2,1.7))
```

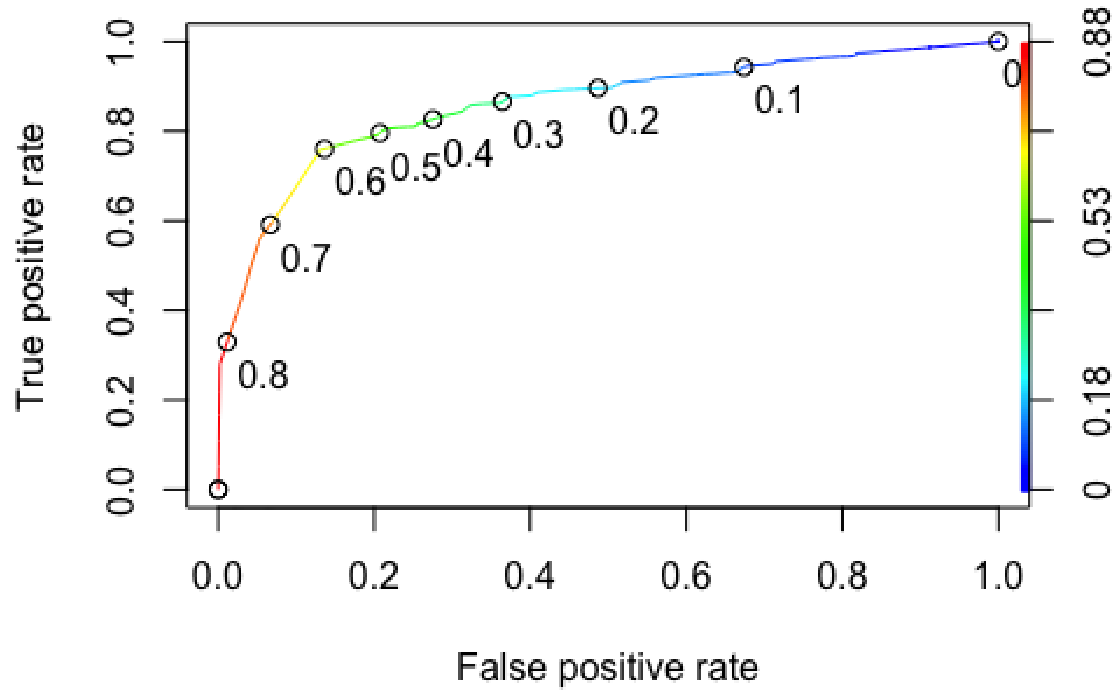


```
table(test.data$satisfaction, predict > 0.7)

##
##        FALSE  TRUE
##    No  14009   564
##    Yes  2981  8422

#Calculating Accuracy
Accuracy_avg_Logit = (14009   + 8422) / (14009 + 8422 + 564 + 2981)
Accuracy_avg_Logit

## [1] 0.8635279

#Calculating Sensitivity or Recall value
Recall = 8422 / (8422 + 3174)
Recall

## [1] 0.7262849

#Calculating Precision Value
Precision = 8422 / (8422 + 771)
Precision

## [1] 0.9161318
```

```
#Calculating F-Measure
F.measure = (2*Recall*Precision)/(Recall+Precision)
F.measure

## [1] 0.8102362

#Calculating Specificity
Specificity = 14009  / (14009  + 771)
Specificity

## [1] 0.9478349

#Testing Data AUC-ROC(Area Under the Curve - Receiver operator
Characteristics) value
AUC1 = as.numeric(performance(ROCRpred, "auc")@y.values)
AUC1

## [1] 0.9255553
```

###Testing for logistic regresion based on spesific variables.

```
##Testing for Logistic Regresion Model 2
#Finding all the vaiues that we think make sense
predict = predict(secondModel, type = 'response' , newdata=test.data)

#Checking the predicion probablities using the summary Function
summary(predict)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
## 0.002974 0.126767 0.428244 0.436143 0.768437 0.873743

tapply(predict, test.data$satisfaction, mean)

##        No       Yes 
## 0.2638526 0.6563296

ROCRpred <- prediction(predict, test.data$satisfaction)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, print.cutoffs.at=seq(0,1,by=0.1),text.adj =
c(-0.2,1.7))
```



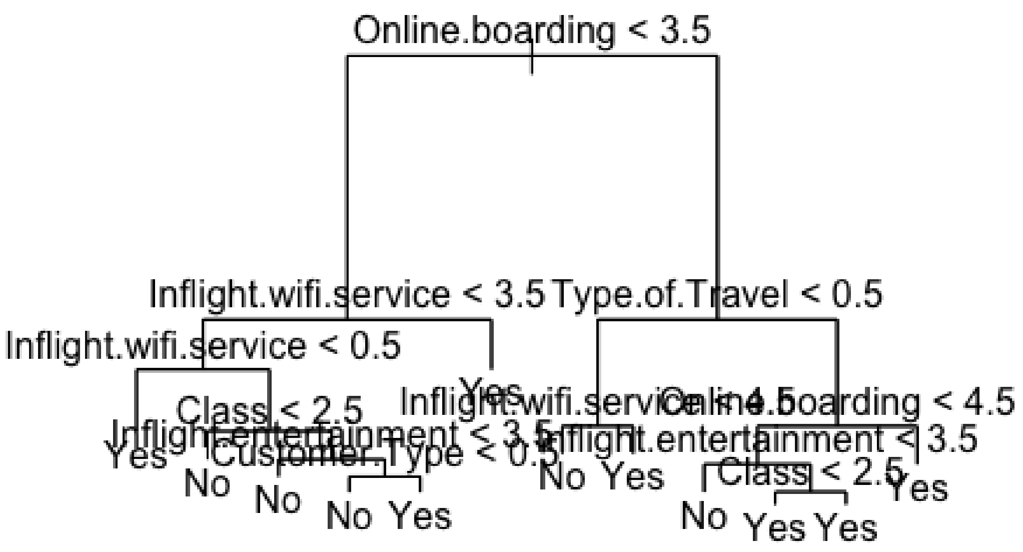
```
table(test.data$satisfaction, predict > 0.7)

##
##        FALSE  TRUE
##   No  13802   771
##   Yes  5021  6382

#Calculating Accuracy
Accuracy_avg_Logit = (13802  + 6382) / (13802  + 6382 + 5021 + 771)
Accuracy_avg_Logit

## [1] 0.7770249

#Calculating Sensitivity or Recall value
Recall = 6382 / (6382 + 5021)
Recall

## [1] 0.5596773

#Calculating Precision Value
Precision = 6382 / (6382 + 771)
Precision

## [1] 0.8922131
```

```
#Calculating F-Measure
F.measure = (2*Recall*Precision)/(Recall+Precision)
F.measure

## [1] 0.6878638

#Calculating Specificity
Specificity = 13802  / (13802  + 771)
Specificity

## [1] 0.9470939

#Testing Data AUC-ROC(Area Under the Curve - Receiver operator
Characteristics) value
AUC2 = as.numeric(performance(ROCRpred, "auc")@y.values)
AUC2

## [1] 0.8607978
```

Testing for Classifaction Tree based on R.Tree().

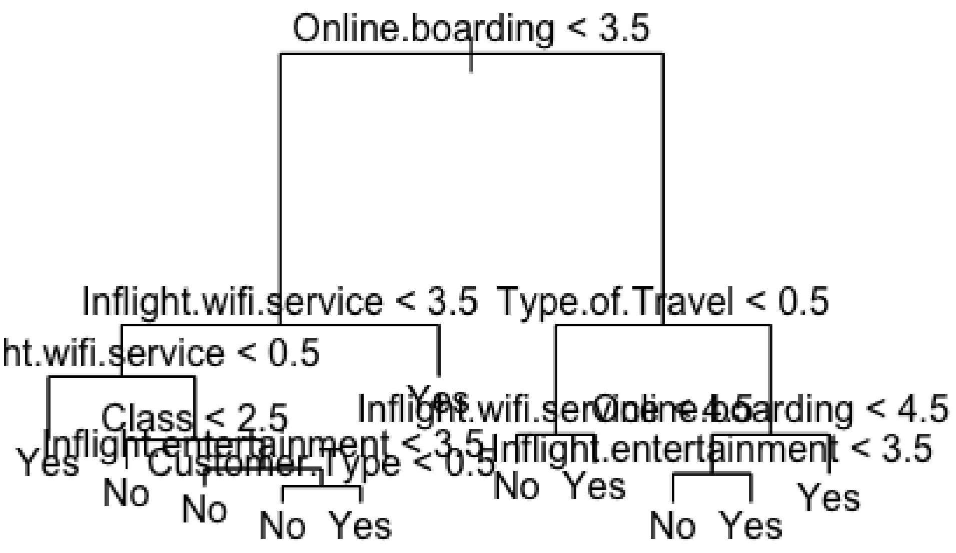
The pruned tree which has 11 terminal nodes, residual mean deviance is 0.485, and misclassification error rate is 0.101. Then we examine the plot of tree of the train data. When we calculate our model's performance we can see that it is again %89.86.We cannot upgrade our models performance by using prune. It is already the best model in terms of its terminal nodes.

```
# This the prune train data.
prune.train.data <- prune.misclass(tree.train.data , best = 12)
plot(prune.train.data)
text(prune.train.data , pretty = 0)
```


Online.boarding < 3.5
Inflight.wifi.service < 3.5
Type.of.Travel < 0.5
Inflight.wifi.service < 0.5
Yes
Class < 2.5
Inflight.wifi.service < 4.5
Online.boarding < 4.5
Inflight.entertainment < 3.5
Customer.Type < 0.5
Inflight.entertainment < 3.5
Yes
No
No
No Yes
No Yes
Class < 2.5
Yes
No
Yes Yes

```
summary(prune.train.data)

##
## Classification tree:
## tree(formula = satisfaction ~ . - satisfaction + X + id, data =
train.data)
## Variables actually used in tree construction:
## [1] "Online.boarding"        "Inflight.wifi.service"  "Class"
## [4] "Inflight.entertainment" "Customer.Type"          "Type.of.Travel"
## Number of terminal nodes:  12
## Residual mean deviance:  0.485 = 50390 / 103900
## Misclassification error rate: 0.101 = 10493 / 103904

tree.pred_pruned <- predict(prune.train.data, test.data, type = "class")
table(tree.pred_pruned , test.data$satisfaction)

##
## tree.pred_pruned    No   Yes
##              No  13394  1451
##              Yes  1179  9952

# This is the model accuracy of classification tree on the test data.
mean(tree.pred_pruned == test.data$satisfaction)
```

```
## [1] 0.8987527

# The pruned model with 11 teminal nodes.
prune.train.data <- prune.misclass(tree.train.data , best = 11)
plot(prune.train.data)
text(prune.train.data , pretty = 0)
```


Online.boarding < 3.5
Inflight.wifi.service < 3.5
Type.of.Travel < 0.5
Inflight.wifi.service < 0.5
Class < 2.5
Inflight.entertainment < 3.5
Customer.Type < 0.5
Inflight.wifi.service < 4.5
Online.boarding < 4.5
Inflight.entertainment < 3.5
Yes
No
No
No
Yes
Yes
No
Yes
No
Yes
Yes

```
tree.pred <- predict(prune.train.data, test.data, type = "class")
table(tree.pred , test.data$satisfaction)

##
## tree.pred    No   Yes
##       No  13394  1451
##       Yes  1179  9952

# Compute model accuracy rate on test data
mean(tree.pred == test.data$satisfaction)

## [1] 0.8987527

#Testing Data AUC-ROC(Area Under the Curve - Receiver operator
Characteristics) value
AUC3 = as.numeric(performance(ROCRpred, "auc")@y.values)
AUC3
```

```
## [1] 0.8607978
```

Testing for Classifaction Tree based on R.Rpart().

```
#Confusion Matrix table to find accuracy
#From the ROC Curve, we found 0.7 is the optimum threshold value for Cut-off.
table(test.data$satisfaction, PredictROC[,2] > 0.65)

##
##        FALSE  TRUE
##   No   13044  1529
##   Yes   1817  9586

#Calculating Accuracy
Accuracy_avg_Tree = (13044+9586)/(13044+9586+1529+1817)
Accuracy_avg_Tree

## [1] 0.8711888

#Calculating Sensitivity or Recall value
Recall = (9586)/(9586+1817)
Recall

## [1] 0.840656

#Calculating Precision Value
Precision = (9586)/(9586+1529)
Precision

## [1] 0.8624381

#Calculating F-Measure
F.measure = (2*Recall*Precision)/(Recall+Precision)
F.measure

## [1] 0.8514078

#Calculating Specificity
Specificity = (14003)/(14003+570)
Specificity

## [1] 0.9608866

#Testing Data AUC-ROC(Area Under the Curve - Receiver operator
Characteristics) value
AUC4 = as.numeric(performance(pred, "auc")@y.values)
AUC4

## [1] 0.9034752
```

Final comparison if you are using more than one model !

Final comparison for Logistic Regression.

```
# This is the logistic regresion based on all variables performance
AUC1

## [1] 0.9255553

# This is the logistic regresion based on spesific variables performance
AUC2

## [1] 0.8607978
```

logistic regresion based on all variables performance heigher than logistic regresion based on spesific variables performance

Final comparison for Classifaction Tree.

```
# This is the R.Tree performance
AUC3

## [1] 0.8607978

# This is the R.Part performance
AUC4

## [1] 0.9034752
```

R.Part is more perfomance than R.Tree

Conclusions

We did two different models for desicon tree and logistic regeresion.

We tested them with the train and test data's. At the end we found that the results that we found with the train and test datas were close. Thats why we satisfied about our data's reliability.

Than we testd our our data's reliabilities with AUC. Acording to the AUC vairables of the models. The most accured model's were: 1- Logistic regresion with all variables calculated. 2- First desicon tree. 3- Second desicon tree. 4- Logistic tree that we spesificaly chosed meaninfull variable.

Now, Based on the comparison between the 2 models of logistic regression and Decision Tree, we could conclude that Logitic regression performed better when compared with Decision Tree on a slightly higher Note. And on compaison with accuracy and Specificity, logitsic regression had a slightly upper hand than Decision Tree. Where the Area under the Curve (AUC) value was found to be almost similar for both the cases.

Thus the Airline customer satisfaction analysis depends on the various factors and satisfaction across each individual category is analyzed to determine the overall satisfaction of the airline passengers. Analysis of data using the logistic regression proved to perform better when compared to Decision tree and results accuracy also proved to quite high.

References

- https://owl.purdue.edu/owl/research_and_citation/apa_style/apa_formatting_and_style_guide/in_text_citations_the_basics.html - https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction?resource=download