

# (EstablishedunderthePresidencyUniversityAct,2013oftheKarnataka Act41of2013)

**Mini Project**

**CSE3035 - R Programming for Data Science**

**2024 - 2025**

**Project Title : Indian Airlines Ticket price analysis**

**Student Name : VIDHYASHREE V**

**Student Roll Number: 20211CAI0081**

**Section : 7CAI3**

**Indian Airlines Ticket price analysis**

**Abstract:**

This project analyzes Indian airline ticket prices using R programming to identify key factors influencing airfare variations. The dataset includes information such as airlines, flight durations, departure times, stops, and ticket classes, allowing for a comprehensive exploration of how these variables impact pricing. Through data pre-processing techniques like handling missing values, encoding categorical data, and normalizing price data, the dataset was prepared for analysis. Visualizations, including bar plots and heatmaps, were employed to reveal trends in ticket availability, price distribution, and the relationship between flight duration and price. The findings provide actionable insights for consumers and airlines, helping optimize travel decisions and pricing strategies.

**Introduction**

Airline ticket prices in India fluctuate due to various factors, including airline choice, travel route, booking time, and demand. Analyzing these price variations can offer insights into how airlines set prices and how passengers can secure better deals. This project focuses on exploring these pricing trends using R programming. The dataset is pre-processed, visualized, and analyzed to reveal key pricing patterns that can benefit both consumers and airlines.

.**Dataset Details**

 **Dataset Name**: Indian Airlines Ticket Price Data

 **Number of Rows**: 300,153

* **Number of Columns**: 12
* **Key columns:** Airline, Flight, Source City, Departure Time, Stops,Arrival Time,Destination City, Class, Duration, Days Left, Price.
* **Source of data:** The dataset was provided as a CSV file named "Indian Airlines.csv," containing flight data from various airlines operating in India.

**Pre-processing Techniques**

* **Technique 1: Dataset Overview**

Initially, the dataset was loaded using the read.csv() function. An overview of the dataset was obtained using functions like head() to view the first few rows and str() to check the structure and data types of the columns.

* **Technique 2: Handling Missing Data**

The dataset was checked for missing values using sapply() to count the number of missing entries in each column. Missing values in the price column were addressed by replacing them with the median price to maintain data integrity.

* **Technique 3: Summarizing Data**

To analyze the relationship between flight duration and ticket price, the data was grouped by duration, and the average price for each duration was calculated.

**Visualization Techniques**

**Chart 1: Bar Plot: Number of Flights by Airlines**

A Box plot was used to visualize the distribution of Total Price for different Product Types. Each box represents the range of prices for each product type, including the median and any outliers. The plot helps visualize how prices vary across product categories.

**Chart 2: Bar Plot: Availability of Tickets w.r.t Class**

bar plot that displays the count of purchases for each Product Type. Each bar represents a different product type, and the height of the bar shows how many purchases were made for that type. The plot helps visualize which product types are most popular based on the number of purchases.

**Chart 3: Bar Plot: Price for Different Airlines Based on Class**

A grouped bar plot where the ticket prices for different airlines are shown, and prices are further categorized by class. The geom\_bar(stat = "identity", position = "dodge") creates a side-by-side comparison of prices between classes.

**Chart 4: Scatter Plot: Average Price Depending on Duration of Flights**

This scatter plot shows how the average price of tickets changes with the duration of flights. The geom\_point() function is used to create the scatter plot, with each point representing the average price for a particular flight duration.

**Chart 4: Heatmap: Average Price by Airlines and Stops**

A heatmap is used to show the relationship between airlines, number of stops, and the average ticket price. The geom\_tile() function creates the heatmap, and scale\_fill\_gradient() is used to provide color scaling based on price.

**Source Code and Screen shots:**

**Pre-processing Technique1: Dataset Overview**

**Source code:**

dataset <- read.csv("C:/Users/DELL/OneDrive/Desktop/Indian Airlines.csv")

# View the first few rows of the dataset

head(dataset)

# Structure of the dataset

str(dataset)

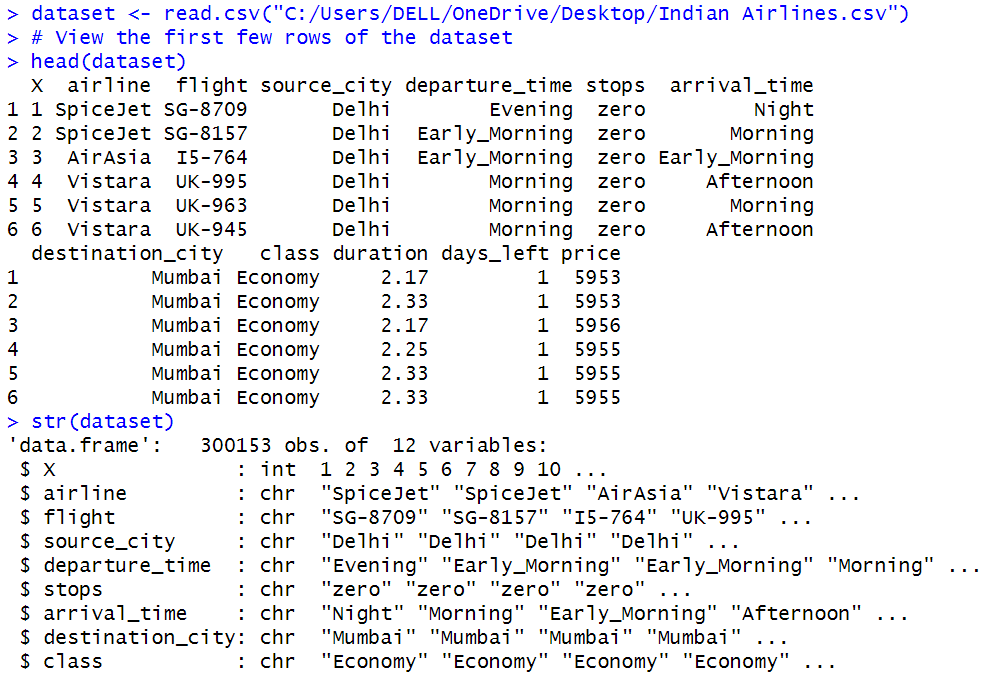
# Get the shape (rows and columns)

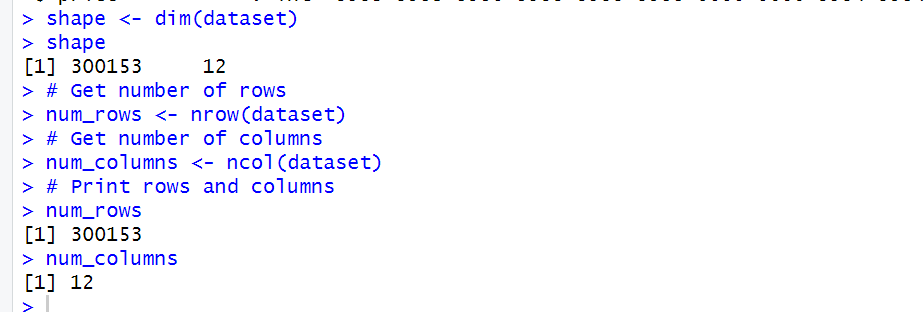
shape <- dim(dataset)

num\_rows <- nrow(dataset)

num\_columns <- ncol(dataset)

**Screen shot:**

****

****

**Pre-processing Technique2:Handle Missing Data**

**Source code:**

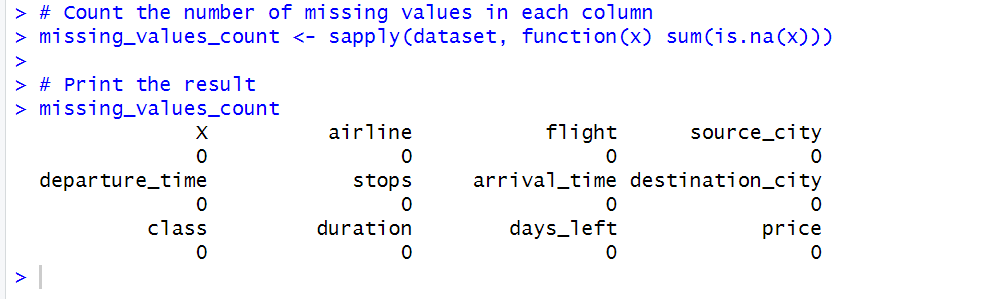
#HANDLE MISSING DATA

# Count the number of missing values in each column

missing\_values\_count <- sapply(dataset, function(x) sum(is.na(x)))

print(missing\_values\_count)

**Screen shot:**

****

**Pre-processing Technique3:** **Normalizing Continuous Variables**

**Source code:**

# Normalize the 'price' column (Min-Max normalization)

dataset$normalized\_price <- (dataset$price - min(dataset$price)) / (max(dataset$price) - min(dataset$price))

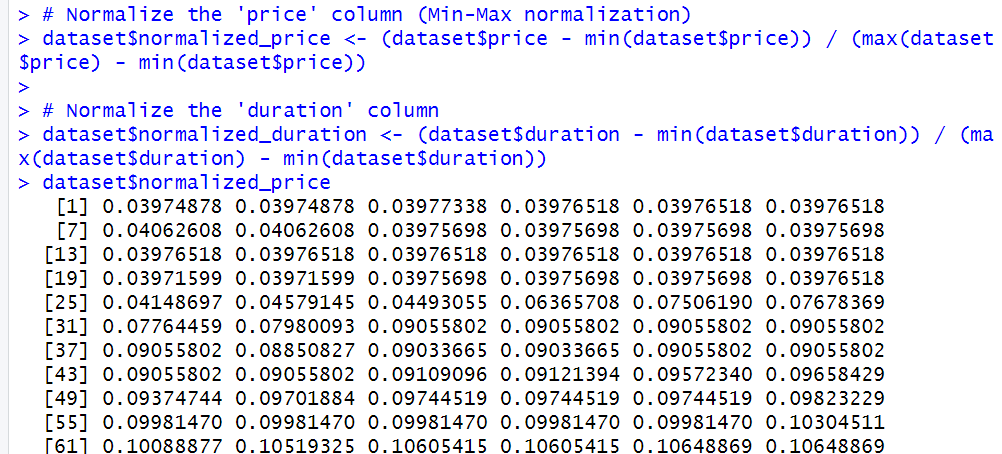
# Normalize the 'duration' column

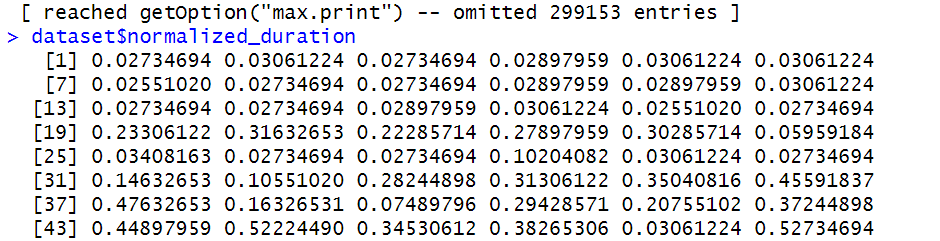
dataset$normalized\_duration <- (dataset$duration - min(dataset$duration)) / (max(dataset$duration) - min(dataset$duration))

dataset$normalized\_price

dataset$normalized\_duration

**Screen shot:**





**Visualization Technique1: Barplot : Number of Flights by Airlines**

# Load required libraries

library(ggplot2)

# Create the plot

ggplot(dataset, aes(x = airline, fill = airline)) +

geom\_bar() +

labs(title = "Number of Flights by Airlines", y = "No. of Flights") +

theme\_minimal() +

theme(

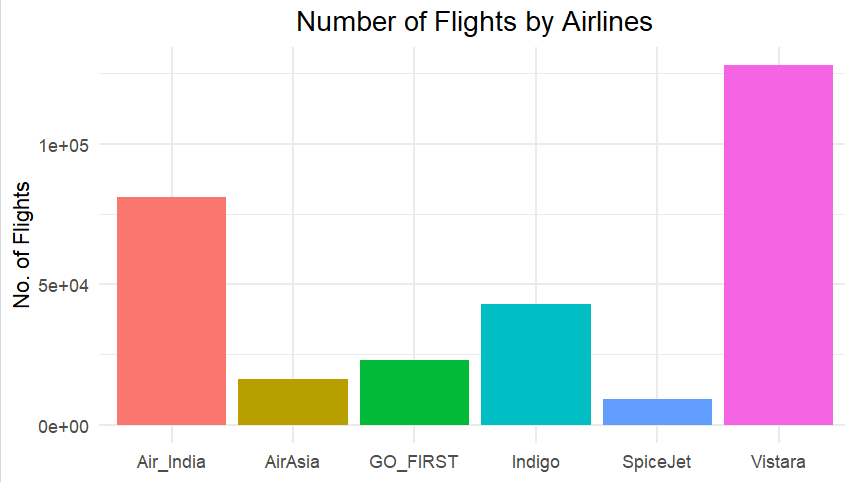
plot.title = element\_text(size = 14),

axis.title.x = element\_blank()

) +

theme(legend.position = "none") + # Remove the hue-like feature as it's redundant

theme(plot.title = element\_text(hjust = 0.5)) # Center the title



**Visualization Technique 2: Barplot:Availability of Tickets w.r.t Class**

# Install and load ggplot2 if not already installed

install.packages("ggplot2")

library(ggplot2)

# Create the bar plot for 'class'

ggplot(dataset, aes(x = class, fill = class)) +

geom\_bar() + # This creates the count plot

labs(title = "Availability of Tickets w.r.t Class", y = "No. of Tickets") + # Title and y-axis label

theme\_minimal() + # Simple clean background

theme(

plot.title = element\_text(hjust = 0.5, size = 14) # Centering and sizing the title

)

****

**Visualization Technique 3: Barplot:Price of Different Airlines Based on Class**

# Install and load necessary packages

install.packages("ggplot2")

library(ggplot2)

install.packages("dplyr")

library(dplyr) # For data manipulation like sorting

# Sort the data by price in ascending order

sorted\_data <- dataset %>% arrange(price)

# Create the bar plot

ggplot(sorted\_data, aes(x = airline, y = price, fill = class)) +

geom\_bar(stat = "identity", position = "dodge") + # Bar plot, separating by class

labs(title = "Price for Different Airlines Based on Class",

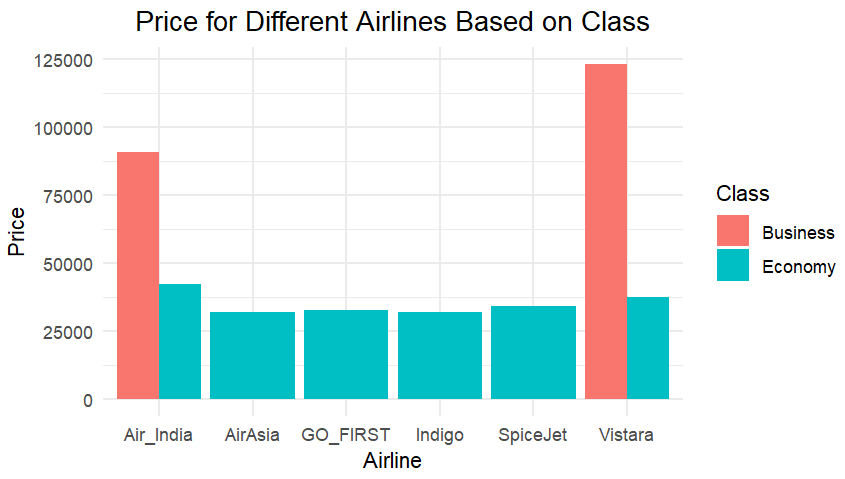
x = "Airline",

y = "Price",

fill = "Class") + # Title and axis labels

theme\_minimal() + # Clean background

theme(plot.title = element\_text(hjust = 0.5, size = 14)) # Center and size the title



**Visualization Technique 4:ScatterPlot:Average Price depending on Duration of Flights**

df\_ticket <- dataset %>%

group\_by(duration) %>%

summarise(price = mean(price, na.rm = TRUE))

# Create the scatter plot

ggplot(df\_ticket, aes(x = duration, y = price)) +

geom\_point(size = 3, color = 'blue') + # Scatter plot points

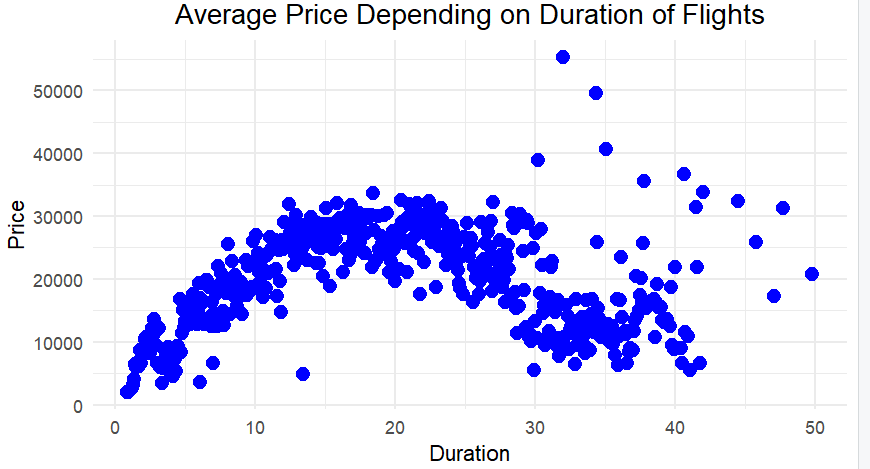
labs(title = "Average Price Depending on Duration of Flights",

x = "Duration",

y = "Price") + # Title and axis labels

theme\_minimal() + # Clean background

theme(plot.title = element\_text(hjust = 0.5, size = 14)) # Center the title.



**Visualization Technique 5: Heatmap : Average Price by Airlines and Stops**

# Average price for airlines and stops

airline\_stop\_stats <- dataset %>%

group\_by(airline, stops) %>%

summarise(average\_price = mean(price, na.rm = TRUE))

# Heatmap

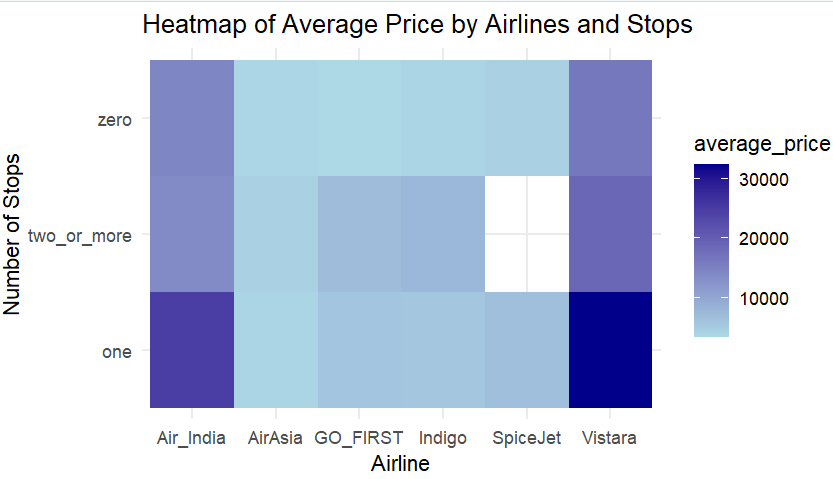
ggplot(airline\_stop\_stats, aes(x = airline, y = stops, fill = average\_price)) +

geom\_tile() +

labs(title = "Heatmap of Average Price by Airlines and Stops", x = "Airline", y = "Number of Stops") +

scale\_fill\_gradient(low = "lightblue", high = "darkblue") +

theme\_minimal()



**CLASSIFICATION TASK**

**Classification task 1**

**Decision Tree for Ticket class and Regression tree for Ticket class**

# Load required libraries

install.packages("rpart")

install.packages("randomForest")

install.packages("caTools")

library(rpart)

library(randomForest)

library(caTools)

# Preprocess dataset (assuming it's already loaded)

dataset$class <- as.factor(dataset$class)

# Split data into training and testing sets

set.seed(123)

split <- sample.split(dataset$class, SplitRatio = 0.7)

train\_data <- subset(dataset, split == TRUE)

test\_data <- subset(dataset, split == FALSE)

# 1. Decision Tree Model

tree\_model <- rpart(class ~ airline + duration + stops + price, data = train\_data, method = "class")

tree\_pred <- predict(tree\_model, test\_data, type = "class")

tree\_accuracy <- mean(tree\_pred == test\_data$class)

# 2. Random Forest Model

rf\_model <- randomForest(class ~ airline + duration + stops + price, data = train\_data, ntree = 100)

rf\_pred <- predict(rf\_model, test\_data)

rf\_accuracy <- mean(rf\_pred == test\_data$class)

# Print accuracies

print(paste("Decision Tree Accuracy:", round(tree\_accuracy, 2)))

print(paste("Random Forest Accuracy:", round(rf\_accuracy, 2)))

# View the first few rows of test\_data to choose an example row

head(test\_data)

#sample\_input <- test\_data[50, ]

sample\_input <- test\_data[nrow(test\_data), ]

# Make a prediction using the Random Forest model

predicted\_class\_1 <- predict(rf\_model, sample\_input)

predicted\_class\_2 <- predict(tree\_model, sample\_input)

# Print the predicted class

print(paste("Predicted Ticket Class for Test Row 2:", predicted\_class\_1))

print(paste("Predicted Ticket Class for Test Row 2:", predicted\_class\_2))

# Compare with the actual class label for random forest model

actual\_class <- test\_data$class[nrow(test\_data)]

print(paste("Actual Ticket Class for Test Row 2:", actual\_class))

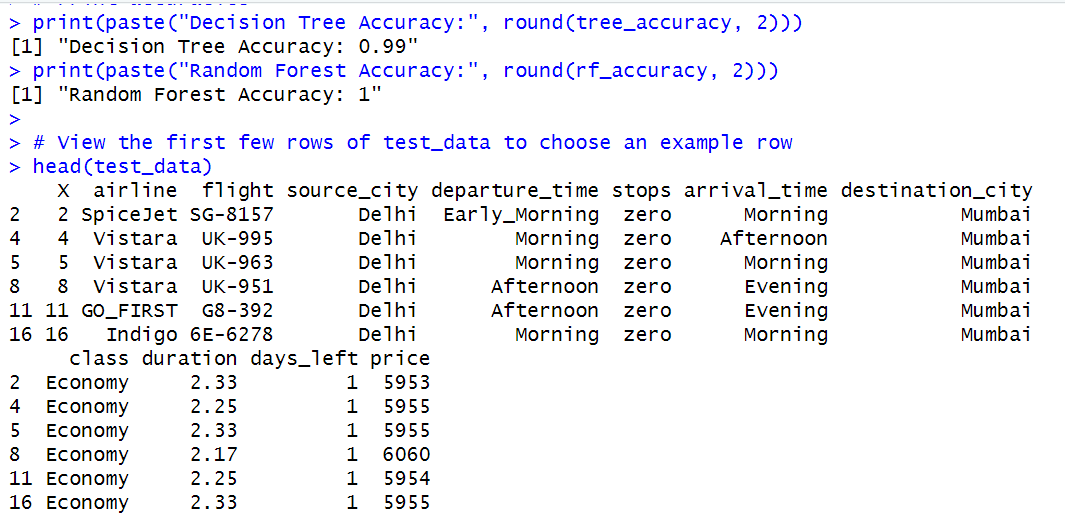
print(paste("Predicted Ticket Class for Test Row 2:", predicted\_class\_1))

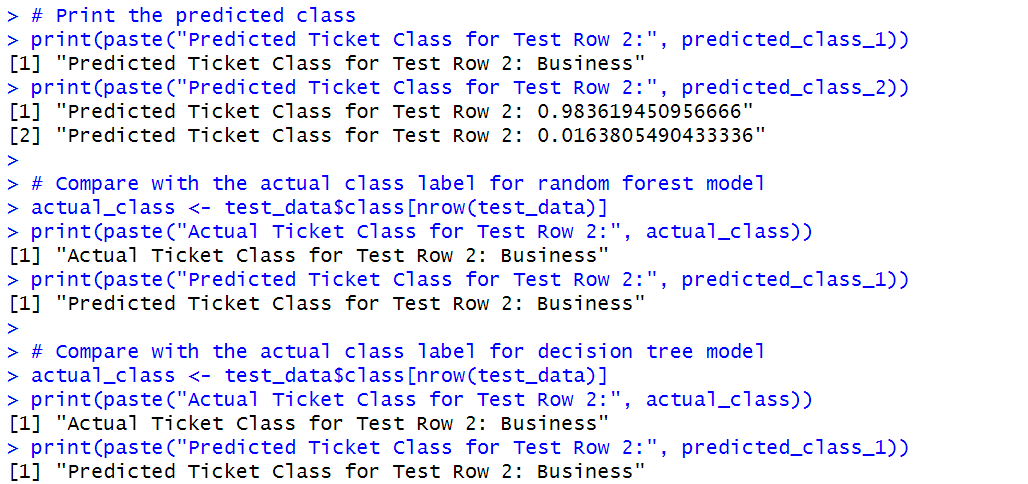
# Compare with the actual class label for decision tree model

actual\_class <- test\_data$class[nrow(test\_data)]

print(paste("Actual Ticket Class for Test Row 2:", actual\_class))

print(paste("Predicted Ticket Class for Test Row 2:", predicted\_class\_1))





**CLASSIFICATION TASK 2**

**Decision Tree for Stops Prediction and Regression tree for stops Prediction**

# Convert stops to a factor for classification

dataset$stops <- as.factor(dataset$stops)

# Split the data into training and testing sets

set.seed(456)

split\_stops <- sample.split(dataset$stops, SplitRatio = 0.7)

train\_data\_stops <- subset(dataset, split\_stops == TRUE)

test\_data\_stops <- subset(dataset, split\_stops == FALSE)

# 1. Decision Tree for Stops Prediction

tree\_model\_stops <- rpart(stops ~ airline + duration + price + class, data = train\_data\_stops, method = "class")

tree\_pred\_stops <- predict(tree\_model\_stops, test\_data\_stops, type = "class")

tree\_accuracy\_stops <- mean(tree\_pred\_stops == test\_data\_stops$stops)

# 2. Random Forest for Stops Prediction

rf\_model\_stops <- randomForest(stops ~ airline + duration + price + class, data = train\_data\_stops, ntree = 200)

rf\_pred\_stops <- predict(rf\_model\_stops, test\_data\_stops)

rf\_accuracy\_stops <- mean(rf\_pred\_stops == test\_data\_stops$stops)

# Print the accuracies

print(paste("Decision Tree Accuracy for Stops Prediction:", round(tree\_accuracy\_stops, 2)))

print(paste("Random Forest Accuracy for Stops Prediction:", round(rf\_accuracy\_stops, 2)))

# Select a sample row from the test dataset

example\_row <- test\_data\_stops[nrow(test\_data\_stops), ] # Last row in test data

# Predict the number of stops using Decision Tree

tree\_pred\_example <- predict(tree\_model\_stops, example\_row, type = "class")

# Predict the number of stops using Random Forest

rf\_pred\_example <- predict(rf\_model\_stops, example\_row)

# Print the predictions

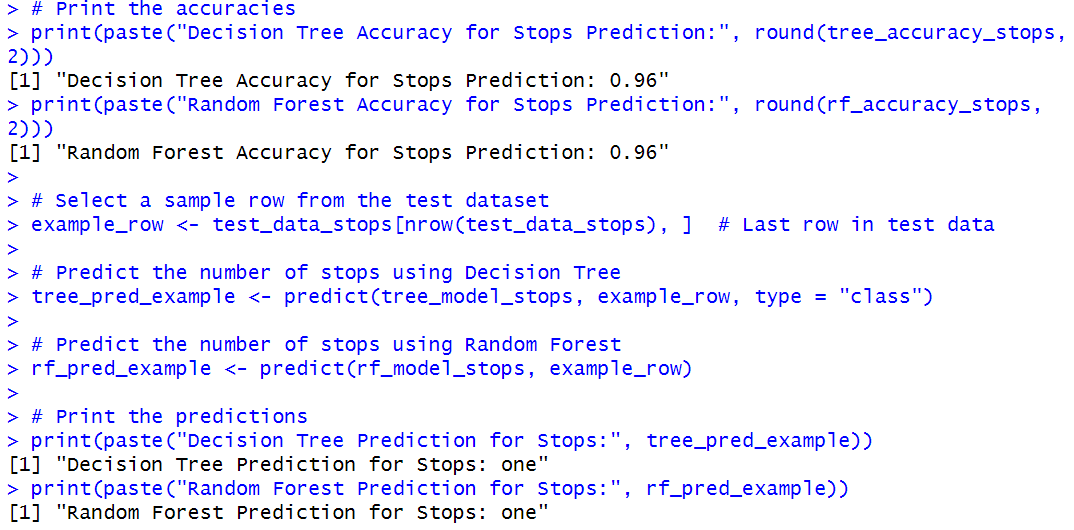
print(paste("Decision Tree Prediction for Stops:", tree\_pred\_example))

print(paste("Random Forest Prediction for Stops:", rf\_pred\_example))

# Compare with the actual value

actual\_stops <- example\_row$stops

print(paste("Actual Number of Stops:", actual\_stops))



**TASK 3**

**Logistic regression**

# Load necessary libraries

library(caret)

library(e1071)

# Load the dataset

flight\_data <- read.csv("C:/Users/DELL/OneDrive/Desktop/Indian Airlines.csv")

# Select relevant columns

flight\_data <- flight\_data[, c("airline", "source\_city", "departure\_time", "stops", "duration", "days\_left", "class")]

# Encode target variable (Economy = 0, Business = 1) and convert to factor

flight\_data$class <- as.factor(ifelse(flight\_data$class == "Economy", "Economy", "Business"))

# Convert categorical variables to factors

categorical\_vars <- c("airline", "source\_city", "departure\_time", "stops")

flight\_data[categorical\_vars] <- lapply(flight\_data[categorical\_vars], as.factor)

# Split the data into training and testing sets

set.seed(42)

train\_index <- createDataPartition(flight\_data$class, p = 0.8, list = FALSE)

train\_data <- flight\_data[train\_index, ]

test\_data <- flight\_data[-train\_index, ]

# Train logistic regression model

logistic\_model <- train(

class ~ .,

data = train\_data,

method = "glm",

family = "binomial",

trControl = trainControl(method = "none")

)

# Make predictions on the test set

predictions <- predict(logistic\_model, test\_data)

# Convert predictions to factors with same levels as test\_data$class

predictions <- factor(predictions, levels = levels(test\_data$class))

# Evaluate the model

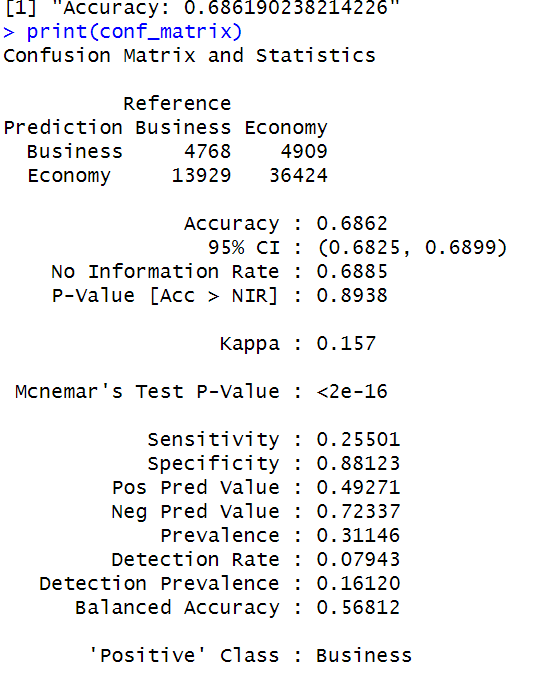
conf\_matrix <- confusionMatrix(predictions, test\_data$class)

accuracy <- conf\_matrix$overall["Accuracy"]

# Print results

print(paste("Accuracy:", accuracy))

print(conf\_matrix)



**Regression Tasks**

# Load necessary libraries

library(caret)

library(e1071)

# Load the dataset

flight\_data <- read.csv("C:/Users/DELL/OneDrive/Desktop/Indian Airlines.csv")

# Select relevant columns for regression

regression\_data <- flight\_data[, c("airline", "source\_city", "departure\_time", "stops", "duration", "days\_left", "price")]

# Convert categorical variables to factors

categorical\_vars <- c("airline", "source\_city", "departure\_time", "stops")

regression\_data[categorical\_vars] <- lapply(regression\_data[categorical\_vars], as.factor)

# Handle missing values (if any)

regression\_data <- na.omit(regression\_data)

**# Regression Task 1: Predict Flight Duration**

set.seed(42)

train\_index\_1 <- createDataPartition(regression\_data$duration, p = 0.8, list = FALSE)

train\_data\_1 <- regression\_data[train\_index\_1, ]

test\_data\_1 <- regression\_data[-train\_index\_1, ]

# Train a Gradient Boosting Model for Flight Duration

gbm\_model\_1 <- train(

duration ~ airline + source\_city + departure\_time + stops + days\_left + price,

data = train\_data\_1,

method = "gbm",

trControl = trainControl(method = "cv", number = 5),

verbose = FALSE

)

# Predict and evaluate

predictions\_1 <- predict(gbm\_model\_1, test\_data\_1)

mse\_1 <- mean((predictions\_1 - test\_data\_1$duration)^2)

r2\_1 <- cor(predictions\_1, test\_data\_1$duration)^2

**# Regression Task 2: Predict Ticket Price**

set.seed(42)

train\_index\_2 <- createDataPartition(regression\_data$price, p = 0.8, list = FALSE)

train\_data\_2 <- regression\_data[train\_index\_2, ]

test\_data\_2 <- regression\_data[-train\_index\_2, ]

# Train a Gradient Boosting Model for Ticket Price

gbm\_model\_2 <- train(

price ~ airline + source\_city + departure\_time + stops + duration + days\_left,

data = train\_data\_2,

method = "gbm",

trControl = trainControl(method = "cv", number = 5),

verbose = FALSE

)

# Predict and evaluate

predictions\_2 <- predict(gbm\_model\_2, test\_data\_2)

mse\_2 <- mean((predictions\_2 - test\_data\_2$price)^2)

r2\_2 <- cor(predictions\_2, test\_data\_2$price)^2

# Print results

cat("Regression Task 1: Predict Flight Duration\n")

cat("MSE:", mse\_1, "\n")

cat("R-squared:", r2\_1, "\n\n")

cat("Regression Task 2: Predict Ticket Price\n")

cat("MSE:", mse\_2, "\n")

cat("R-squared:", r2\_2, "\n")

