The Report

Süleyman Erim, Giacomo Schiavo, Mattia Varagnolo

2023-07-25

Introduction to data

This section introduces the purpose of the exploratory data analysis (EDA) and sets up the necessary libraries and data files.

```
# import libraries
library(tidyverse)
library(tinytex)
library(dplyr)
library(corrplot)
library(ggplot2)
library(gridExtra)
library(correlation)
library(reshape)
library(reshape2)
library(tidyverse) # for data manipulation
library(ggplot2) # for plotting
library(gridExtra) # for grid.arrange
library(regclass) # for VIF package
library(MLmetrics) # to create confusion matrix
library(pROC) # for ROC Curve
library(e1071) # for Naive Bayes Classifier
library(class)
library(caret)
library(corrr)
library(ppcor)
library(glmnet) # for Lasso Regression
```

```
data_train = read.csv("C:/Users/matti/Documents/GitHub/stat_project/train.csv")
data_test = read.csv("C:/Users/matti/Documents/GitHub/stat_project/test.csv")
```

```
# merge train and test data
data = rbind(data_train, data_test)
attach(data)
```

Data Analysis

In this project, we will develop a predictive model to determine whether a passenger will be satisfied or dissatisfied with the services offered by an airline company. The dataset used for this project is a survey on

airline passenger satisfaction, which contains information about passengers' demographics, travel preferences, and satisfaction with various aspects of their flights.

The dataset: https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction

Description of our variables

Here are the variables in the dataset:

- Gender: Gender of the passengers (Female, Male)
- Customer Type: The customer type (Loyal customer, disloyal customer)
- Age: The actual age of the passengers
- 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)
- Flight distance: The flight distance of this journey
- Inflight wifi service: Satisfaction level of the inflight wifi service (0: Not Applicable; 1-5)
- Departure/Arrival time convenient: Satisfaction level of Departure/Arrival time convenient
- Ease of Online booking: Satisfaction level of online booking
- Gate location: Satisfaction level of Gate location
- Food and drink: Satisfaction level of Food and drink
- Online boarding: Satisfaction level of online boarding
- Seat comfort: Satisfaction level of Seat comfort
- Inflight entertainment: Satisfaction level of inflight entertainment
- On-board service: Satisfaction level of On-board service
- Leg room service: Satisfaction level of Leg room service
- Baggage handling: Satisfaction level of baggage handling
- Check-in service: Satisfaction level of Check-in service
- Inflight service: Satisfaction level of inflight service
- Cleanliness: Satisfaction level of Cleanliness
- Departure Delay in Minutes: Minutes delayed when departure
- Arrival Delay in Minutes: Minutes delayed when Arrival
- Satisfaction: Airline satisfaction level (Satisfaction, neutral or dissatisfaction)

The objective of our report is to predict passenger satisfaction with airline services based on the provided dataset, which includes various demographic and satisfaction-related variables such as gender, age, travel type, flight class, and satisfaction levels with different aspects of the journey. The dataset represents a survey on airline passenger satisfaction and will be used to develop a predictive model to determine whether passengers will be satisfied or dissatisfied with the airline services.

Now we're going to get a summary of all the features in our dataset:

```
summary(data)
```

```
##        X                id            Gender          Customer.Type
##  Min.   :     0   Min.   :     1   Length:129880      Length:129880
##  1st Qu.: 16235   1st Qu.: 32471   Class :character   Class :character
##  Median : 38964   Median : 64941   Mode  :character   Mode  :character
##  Mean   : 44159   Mean   : 64941
##  3rd Qu.: 71433   3rd Qu.: 97410
##  Max.   :103903   Max.   :129880
##
##       Age        Type.of.Travel        Class           Flight.Distance
##  Min.   : 7.00   Length:129880      Length:129880      Min.   :  31
##  1st Qu.:27.00   Class :character   Class :character   1st Qu.: 414
##  Median :40.00   Mode  :character   Mode  :character   Median : 844
##  Mean   :39.43                                         Mean   :1190
##  3rd Qu.:51.00                                         3rd Qu.:1744
```

```
##  Max.   :85.00                                        Max.   :4983
##
##  Inflight.wifi.service Departure.Arrival.time.convenient Ease.of.Online.booking
##  Min.   :0.000         Min.   :0.000                     Min.   :0.000
##  1st Qu.:2.000         1st Qu.:2.000                     1st Qu.:2.000
##  Median :3.000         Median :3.000                     Median :3.000
##  Mean   :2.729         Mean   :3.058                     Mean   :2.757
##  3rd Qu.:4.000         3rd Qu.:4.000                     3rd Qu.:4.000
##  Max.   :5.000         Max.   :5.000                     Max.   :5.000
##
##  Gate.location   Food.and.drink  Online.boarding  Seat.comfort
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:2.000   1st Qu.:2.000   1st Qu.:2.000   1st Qu.:2.000
##  Median :3.000   Median :3.000   Median :3.000   Median :4.000
##  Mean   :2.977   Mean   :3.205   Mean   :3.253   Mean   :3.441
##  3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:5.000
##  Max.   :5.000   Max.   :5.000   Max.   :5.000   Max.   :5.000
##
##  Inflight.entertainment On.board.service Leg.room.service Baggage.handling
##  Min.   :0.000          Min.   :0.000    Min.   :0.000    Min.   :1.000
##  1st Qu.:2.000          1st Qu.:2.000    1st Qu.:2.000    1st Qu.:3.000
##  Median :4.000          Median :4.000    Median :4.000    Median :4.000
##  Mean   :3.358          Mean   :3.383    Mean   :3.351    Mean   :3.632
##  3rd Qu.:4.000          3rd Qu.:4.000    3rd Qu.:4.000    3rd Qu.:5.000
##  Max.   :5.000          Max.   :5.000    Max.   :5.000    Max.   :5.000
##
##  Checkin.service Inflight.service  Cleanliness    Departure.Delay.in.Minutes
##  Min.   :0.000   Min.   :0.000    Min.   :0.000   Min.   :   0.00
##  1st Qu.:3.000   1st Qu.:3.000    1st Qu.:2.000   1st Qu.:   0.00
##  Median :3.000   Median :4.000    Median :3.000   Median :   0.00
##  Mean   :3.306   Mean   :3.642    Mean   :3.286   Mean   :  14.71
##  3rd Qu.:4.000   3rd Qu.:5.000    3rd Qu.:4.000   3rd Qu.:  12.00
##  Max.   :5.000   Max.   :5.000    Max.   :5.000   Max.   :1592.00
##
##  Arrival.Delay.in.Minutes satisfaction
##  Min.   :   0.00          Length:129880
##  1st Qu.:   0.00          Class :character
##  Median :   0.00          Mode  :character
##  Mean   :  15.09
##  3rd Qu.:  13.00
##  Max.   :1584.00
##  NA's   :393
```

Calculate the proportion of satisfied and dissatisfied customers in the dataset.

```
prop.table(table(data$satisfaction))
```

```
##
## neutral or dissatisfied               satisfied
##               0.5655374               0.4344626
```

From the summary, it is evident that many features represent ratings on the services provided by the airline agency, and these ratings range from 0 to 5. Additionally, we noticed that the "Arrival Delay in Minutes" feature contains some missing values (NA).

Next, we will examine the distribution of all nominal features. Specifically, we have categorical data for

Gender, Customer Type, Type of Travel, and Class, while all the rating features are ordinal.

```
table(data$Gender)
```

```
##
## Female   Male
##  65899  63981
```

The "Gender" feature appears to be well-balanced, meaning that it has an approximately equal number of occurrences for each category, likely male and female. This balance can be beneficial for modeling as it prevents any significant bias towards a particular gender in the analysis and predictions.

```
table(data$Customer.Type)
```

```
##
## disloyal Customer    Loyal Customer
##             23780            106100
```

The "Customer Type" feature contains only two values, "disloyal customer" and "loyal customer." The distribution of values is imbalanced, with one category potentially having significantly more occurrences than the other.

```
table(data$Type.of.Travel)
```

```
##
## Business travel Personal Travel
##           89693           40187
```

The "Type of Travel" feature consists of only two values: "personal travel" and "business travel." The distribution of values is imbalanced, with "business travel" occurring twice as much as "personal travel."

```
table(data$Class)
```

```
##
## Business      Eco Eco Plus
##    62160    58309     9411
```

The "Class" feature contains three values: "business," "eco plus," and "eco." The distribution of values is imbalanced. "Business" and "eco" classes appear to be relatively balanced, while "eco plus" is significantly underrepresented compared to the other two classes.

```
table(data$satisfaction)
```

```
##
## neutral or dissatisfied               satisfied
##                   73452                   56428
```

The "satisfaction" feature, which serves as our target variable, is an important aspect of the analysis. The values for this feature are not perfectly balanced, meaning that there is an unequal distribution of satisfied and dissatisfied passengers in the dataset.

Data preprocessing

In this section of data preprocessing, several steps are performed to prepare the dataset for further analysis and modeling. The specific actions taken include:

1. Renaming columns: the names of the features (columns) are modified to improve their clarity and usability.
2. Dropping unnecessary columns: two columns, "X" and "id," are removed from the dataset. The "X" column likely represents the index of the row, which does not carry any meaningful information for

analysis. The "id" column is presumed to be an unknown indexing number, which may not contribute to the predictive modeling process.

3. Converting categorical variables to factors: categorical variables, such as "Gender", "Customer Type", "Type of Travel" and "Class" are converted into factors.

By performing these data preprocessing steps, the dataset is cleaned and transformed into a more suitable format for the subsequent analysis, making it easier to build a predictive model for passenger satisfaction.

Renaming columns and removing features

```
# replace dots with underscores in column names
names(data) = gsub("\\.", "_", names(data))
# drop X and id column
data <- data %>% dplyr::select(-X, -id)
names(data)
```

```
##  [1] "Gender"                          "Customer_Type"
##  [3] "Age"                             "Type_of_Travel"
##  [5] "Class"                           "Flight_Distance"
##  [7] "Inflight_wifi_service"           "Departure_Arrival_time_convenient"
##  [9] "Ease_of_Online_booking"          "Gate_location"
## [11] "Food_and_drink"                  "Online_boarding"
## [13] "Seat_comfort"                    "Inflight_entertainment"
## [15] "On_board_service"                "Leg_room_service"
## [17] "Baggage_handling"                "Checkin_service"
## [19] "Inflight_service"                "Cleanliness"
## [21] "Departure_Delay_in_Minutes"      "Arrival_Delay_in_Minutes"
## [23] "satisfaction"
```

Convert categorical features to factors

```
# convert categorical features to factor
data$Gender = as.factor(data$Gender)
data$Customer_Type = as.factor(data$Customer_Type)
data$Type_of_Travel = as.factor(data$Type_of_Travel)
data$Class = as.factor(data$Class)
data$satisfaction = as.factor(data$satisfaction)
data$Arrival_Delay_in_Minutes <- as.numeric(data$Arrival_Delay_in_Minutes)


ratings_fts_names = c("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", "On_board_service")

for (col in ratings_fts_names) {
  data[[col]] = as.factor(data[[col]])
}
```

Handling na values

In this section, we analyze the dataset to identify variables with missing values, particularly focusing on the "Arrival_Delay_in_Minutes" variable. We calculate the proportion of missing values for this variable and subsequently remove the examples or rows with missing values from the dataset.

```r
# list features with na values
prop.table(colSums(is.na(data)))
```

```
##                            Gender                     Customer_Type
##                                 0                                 0
##                               Age                    Type_of_Travel
##                                 0                                 0
##                             Class                    Flight_Distance
##                                 0                                 0
##             Inflight_wifi_service Departure_Arrival_time_convenient
##                                 0                                 0
##            Ease_of_Online_booking                     Gate_location
##                                 0                                 0
##                   Food_and_drink                    Online_boarding
##                                 0                                 0
##                     Seat_comfort            Inflight_entertainment
##                                 0                                 0
##                On_board_service                   Leg_room_service
##                                 0                                 0
##                Baggage_handling                    Checkin_service
##                                 0                                 0
##                Inflight_service                       Cleanliness
##                                 0                                 0
##        Departure_Delay_in_Minutes          Arrival_Delay_in_Minutes
##                                 0                                 1
##                      satisfaction
##                                 0
```

To determine the proportion of missing values for the "Arrival_Delay_in_Minutes" variable, we can count the number of instances where this variable has missing values (commonly denoted as "NaN" or "NA") and divide it by the total number of examples in the dataset.

```r
# Arrival_Delay_in_Minutes has na values, proportion of na values
prop.table(table(is.na(data$Arrival_Delay_in_Minutes)))
```

```
##
##      FALSE       TRUE
## 0.99697413 0.00302587
```

Indeed, since the proportion of missing values for the "Arrival_Delay_in_Minutes" variable is very low (less than 3% of the entire dataset), it is reasonable to proceed with dropping these missing values from the dataset.

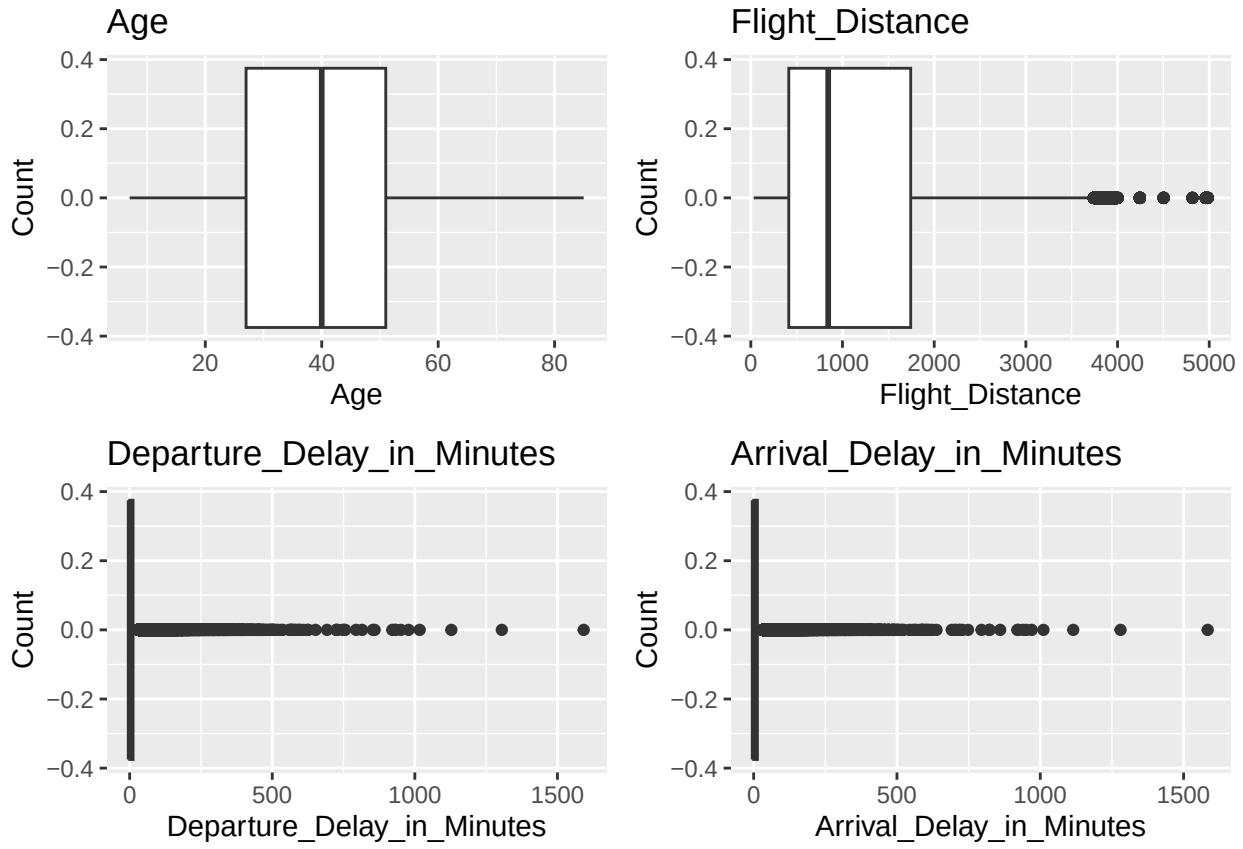
```r
# drop na values in Arrival_Delay_in_Minutes
data = data[!is.na(data$Arrival_Delay_in_Minutes),]

any(is.na(data))
```

```
## [1] FALSE
```

Outliers

In this section, box plots are created for each numeric variable present in the dataset. Box plots are a powerful visualization tool used to identify the presence of outliers in the data. For each numeric variable, the box plot displays a box that represents the interquartile range (IQR), with the median indicated by a line inside the box. The "lines" extending from the box show the range of the data, and any data points beyond the lines are considered potential outliers.

```
# plot boxplot of each numeric variable excluding ratings features
plots = list()
for (col in names(data)[sapply(data, is.numeric)]) {
  if (col %in% ratings_fts_names) {
    next
  }
  plot = ggplot(data, aes(x = .data[[col]])) +
  geom_boxplot() +
  labs(title = col, x = col, y = "Count")
  plots[[col]] = plot
}

grid.arrange(grobs = plots, ncol = 2)
```



We can see that there are outliers in Departure_Delay_in_Minutes, Arrival_Delay_in_Minutes and Flight_Distance. Considering the presence of both near-zero and very large values in the dataset, alternative distributions like the log-normal distribution may be more appropriate for modeling the "Departure_Delay_in_Minutes" and "Arrival_Delay_in_Minutes" variables, as they can better capture the variability in delay times. Since we have decided to keep the outliers in the dataset, it's essential to understand their potential impact on our analysis and modeling. Outliers are data points that deviate significantly from the rest of the data and can introduce noise or bias in statistical analyses and machine learning models.

In the context of "Departure_Delay_in_Minutes," "Arrival_Delay_in_Minutes," and "Flight_Distance," outliers might represent extreme values that could be caused by various factors, such as severe weather conditions, operational disruptions, or exceptional circumstances. By retaining these outliers, we acknowledge that such extreme situations can occur and affect the overall flight delay and distance patterns.

Visualization

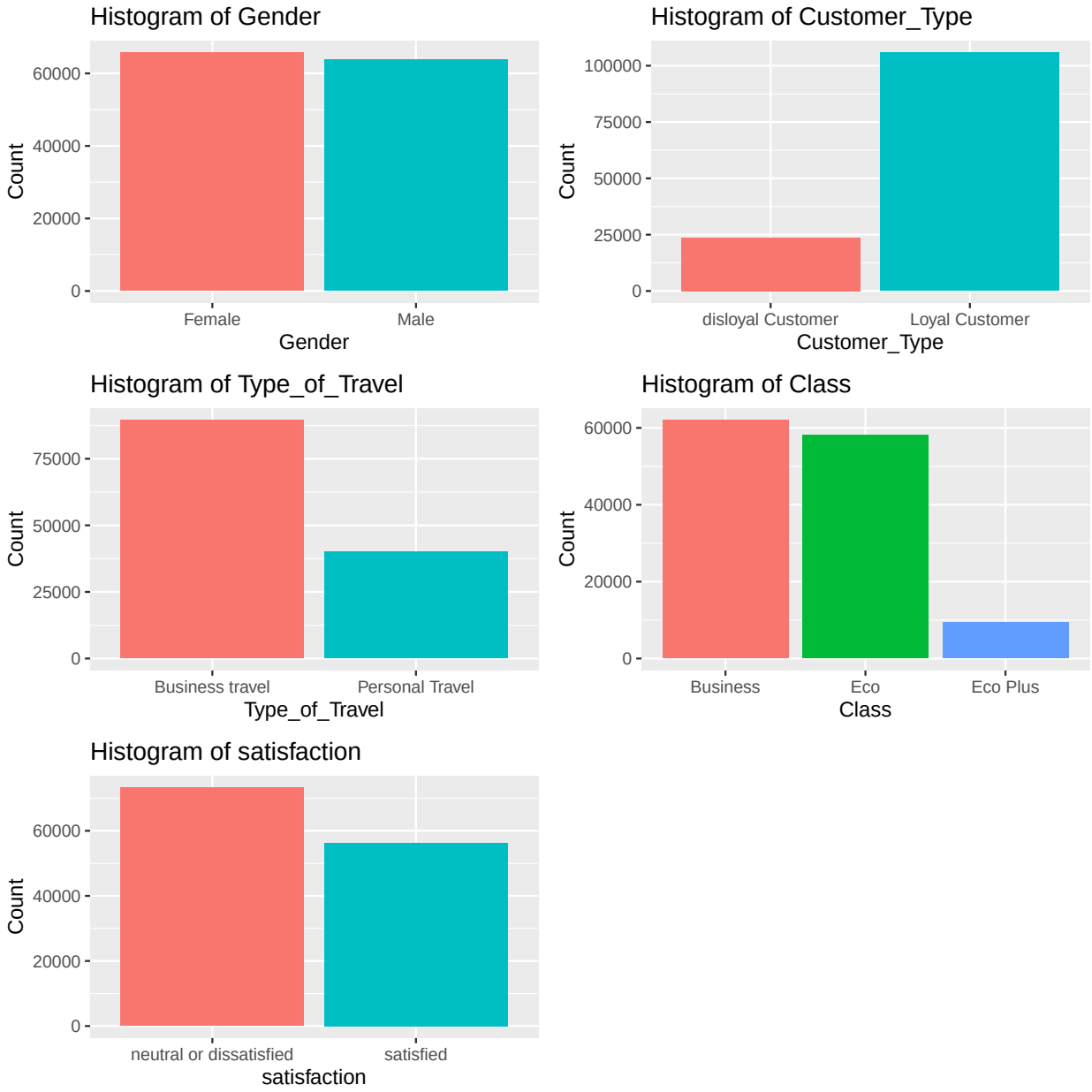
In this section, histograms are used to visualize the distribution of the variables in the dataset, starting with the nominal features so that we can gain insights into the distribution of categories within each feature.

Upon visualizing the nominal features, it becomes apparent that some features exhibit heavily unbalanced distributions. This means that certain categories within these features have significantly higher frequencies compared to others. The presence of such imbalanced distributions could have implications for analysis and modeling, as it may lead to biased results or difficulties in predicting less frequent categories accurately.

```
# plot distribution of categorical variables
plots = list()
for (col in names(data)[sapply(data, is.factor)]) {
  if (col %in% ratings_fts_names) {
    next
  }
  plot = ggplot(data, aes(x = .data[[col]], fill = .data[[col]])) +
  geom_bar() +
  labs(title = paste("Histogram of", col), x = col, y = "Count") +
  guides(fill = FALSE)

  plots[[col]] = plot
}
```

```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
## of ggplot2 3.3.4.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
grid.arrange(grobs = plots, ncol = 2)
```



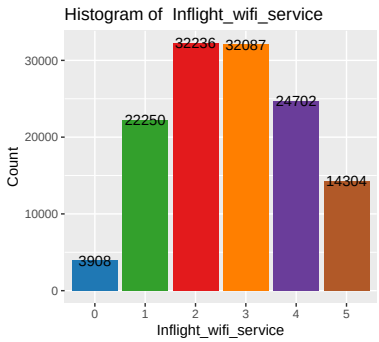
From the analysis of the nominal features, we can observe the following regarding their balance:

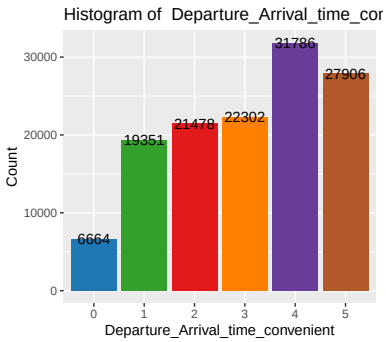
1. Gender: it is almost perfectly balanced, meaning that there is a relatively equal representation of both genders in the dataset.
2. Satisfaction: the target feature appears to be slightly imbalanced, with fewer instances of "satisfied" compared to the other class ("dissatisfied").
3. Type of Travel: it shows an imbalance, with more instances of "business travel" compared to "personal travel." We can assume that the majority of people in this dataset travel for business.
4. Class: it also exhibits imbalance, with "business" and "eco" classes having relatively balanced representations, while the "Eco Plus" class is underrepresented.
5. Customer Type: it shows an imbalance, with a higher number of "loyal customer" instances compared to "disloyal customer."

When dealing with imbalanced data, we need to take specific measures during model training and evaluation to ensure that the model performs well and doesn't exhibit bias towards the majority class. Appropriate techniques such as resampling, using different evaluation metrics or employing specialized algorithms can help address the imbalance and lead to a more accurate and fair predictive model.

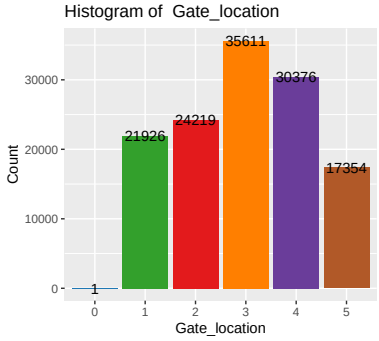
Now we plot the distribution of ratings features.

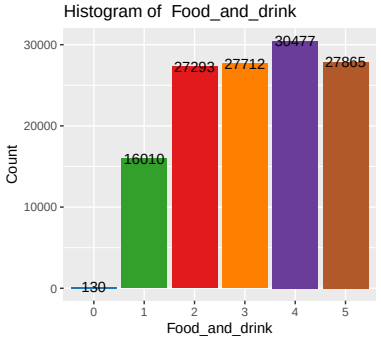
```
# plot distribution of ratings features
plots = list()
my_palette <- c("#1f78b4", "#33a02c", "#e31a1c", "#ff7f00", "#6a3d9a", "#b15928")

for (col in names(data)[sapply(data, is.factor)]) {
  if (!col %in% ratings_fts_names) {
    next
  }
  plot <- ggplot(data, aes(x = .data[[col]], fill = factor(.data[[col]]))) +
    geom_bar() +
    geom_text(stat = 'count', aes(label = after_stat(count))) +
    labs(title = paste("Histogram of ", col), x = col, y = "Count") +
    scale_fill_manual(values = my_palette) +
    guides(fill = FALSE)

  plots[[col]] <- plot
}
grid.arrange(grobs = plots, ncol = 3)
```

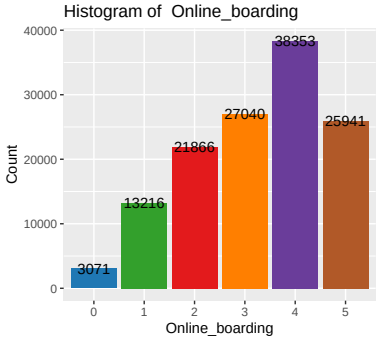

Histogram of Inflight_wifi_service
Count
Inflight_wifi_service
3908
22250
32236
32087
24702
14304

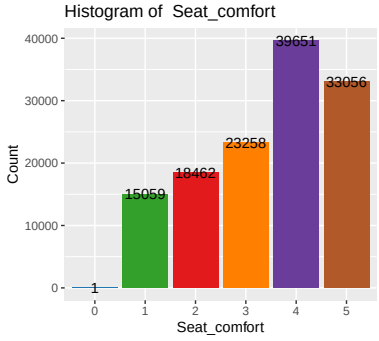

Histogram of Departure_Arrival_time_cor
Count
Departure_Arrival_time_convenient
6664
19351
21478
22302
31786
27906

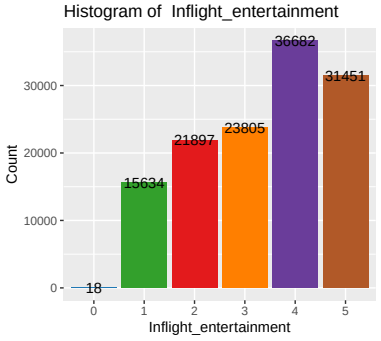

Histogram of Ease_of_Online_booking
Count
Ease_of_Online_booking
5666
21808
29983
30297
24362
17371

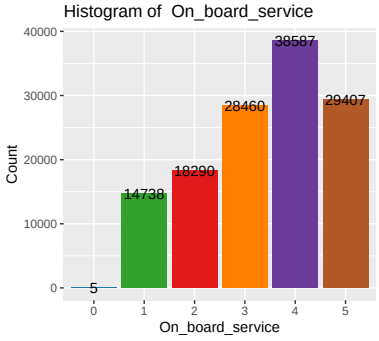

Histogram of Gate_location
Count
Gate_location
1
21926
24219
35611
30376
17354

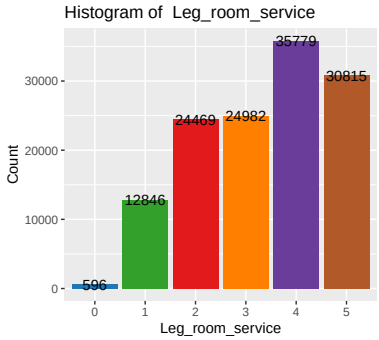

Histogram of Food_and_drink
Count
Food_and_drink
130
16010
27293
27712
30477
27865

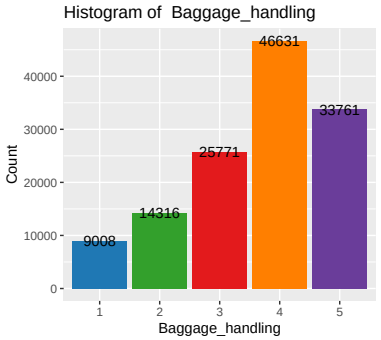

Histogram of Online_boarding
Count
Online_boarding
3071
13216
21866
27040
38353
25941

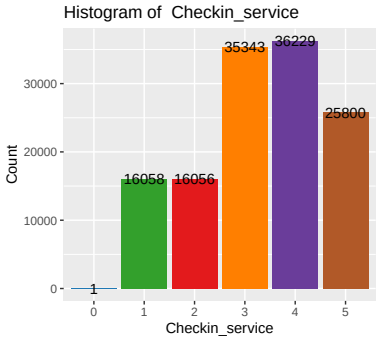

Histogram of Seat_comfort
Count
Seat_comfort
1
15059
18462
23258
39651
33056

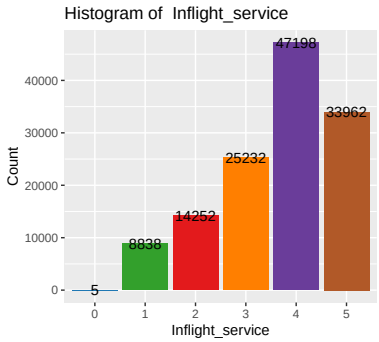

Histogram of Inflight_entertainment
Count
Inflight_entertainment
18
15634
21897
23805
36682
31451

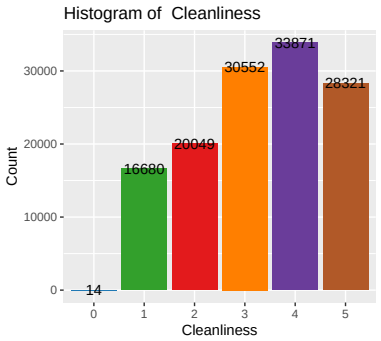

Histogram of On_board_service
Count
On_board_service
5
14738
18290
28460
38587
29407


Histogram of Leg_room_service
Count
Leg_room_service
596
12846
24469
24982
35779
30815


Histogram of Baggage_handling
Count
Baggage_handling
9008
14316
25771
46631
33761


Histogram of Checkin_service
Count
Checkin_service
1
16058
16056
35343
36229
25800


Histogram of Inflight_service
Count
Inflight_service
5
8838
14252
25232
47198
33962


Histogram of Cleanliness
Count
Cleanliness
14
16680
20049
30552
33871
28321

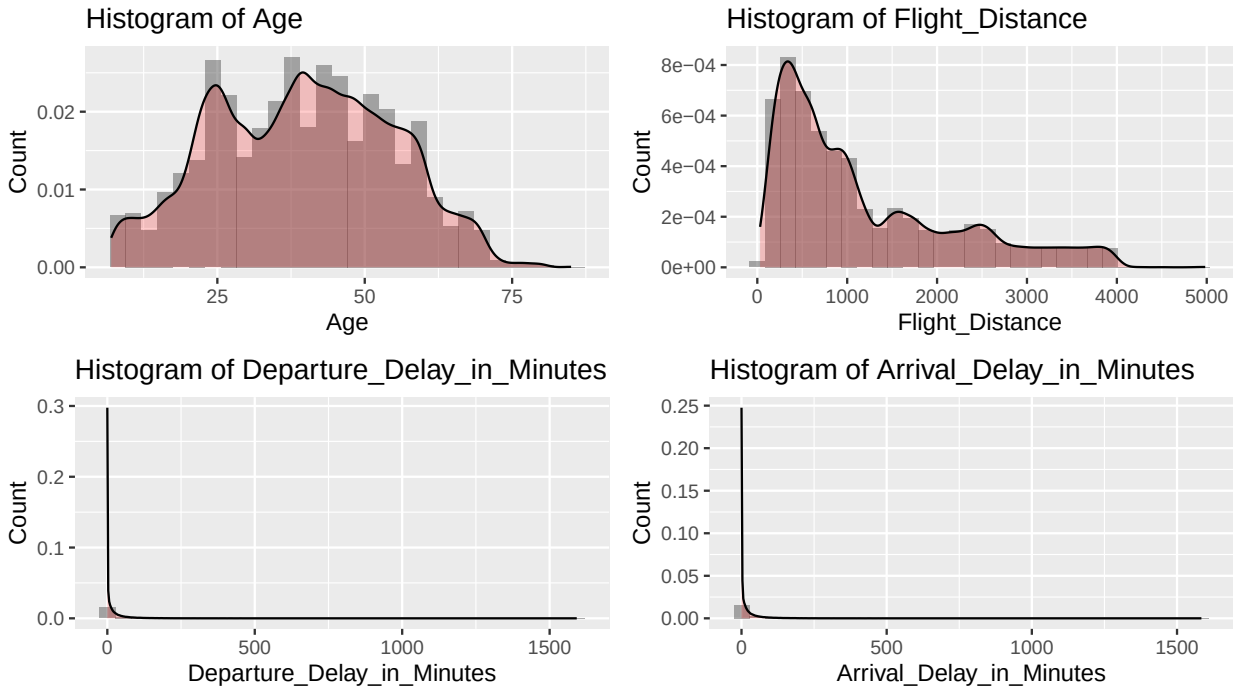
Based on the graphs showing the histograms of the ratings, we can observe that the majority of them tend to fall between 3 and 4. This conclusion is drawn from the visual representation of the data, where the histogram bars are higher in the range of 3 to 4, indicating a higher frequency of ratings in that range.

```
# compute the mean value of all the ratings
ratings_data = data[, c(ratings_fts_names)]
ratings_data <- apply(ratings_data, 2, as.numeric)
ratings_mean = colMeans(ratings_data)
ratings_mean
```

```
##             Inflight_wifi_service Departure_Arrival_time_convenient
##                          2.728544                          3.057349
##           Ease_of_Online_booking                      Gate_location
##                          2.756786                          2.976909
##                    Food_and_drink                   Online_boarding
##                          3.204685                          3.252720
##                      Seat_comfort            Inflight_entertainment
##                          3.441589                          3.358067
##                  On_board_service                 Leg_room_service
##                          3.383204                          3.351078
##                  Baggage_handling                  Checkin_service
##                          3.631886                          3.306239
##                  Inflight_service                      Cleanliness
##                          3.642373                          3.286222
##                On_board_service.1
##                          3.383204
```

This section presents histograms to visualize the distribution of numeric variables in the dataset.

```
# plot distribution and density of numeric variables excluding ratings features
plots = list()
for (col in names(data)[sapply(data, is.numeric)]) {
  if (col %in% ratings_fts_names) {
    next
  }
  plot = ggplot(data, aes(x = .data[[col]])) +
  geom_histogram(aes(y = after_stat(density)), bins = 30, alpha = 0.5) +
  geom_density(alpha = 0.2, fill = "red") +
  labs(title = paste("Histogram of", col), x = col, y = "Count")

  plots[[col]] = plot
}

grid.arrange(grobs = plots, ncol = 2)
```



The histograms provide a useful overview of the distribution of the numeric variables in the dataset. This information can be used to identify potential outliers, and to choose appropriate statistical methods for analyzing the data:

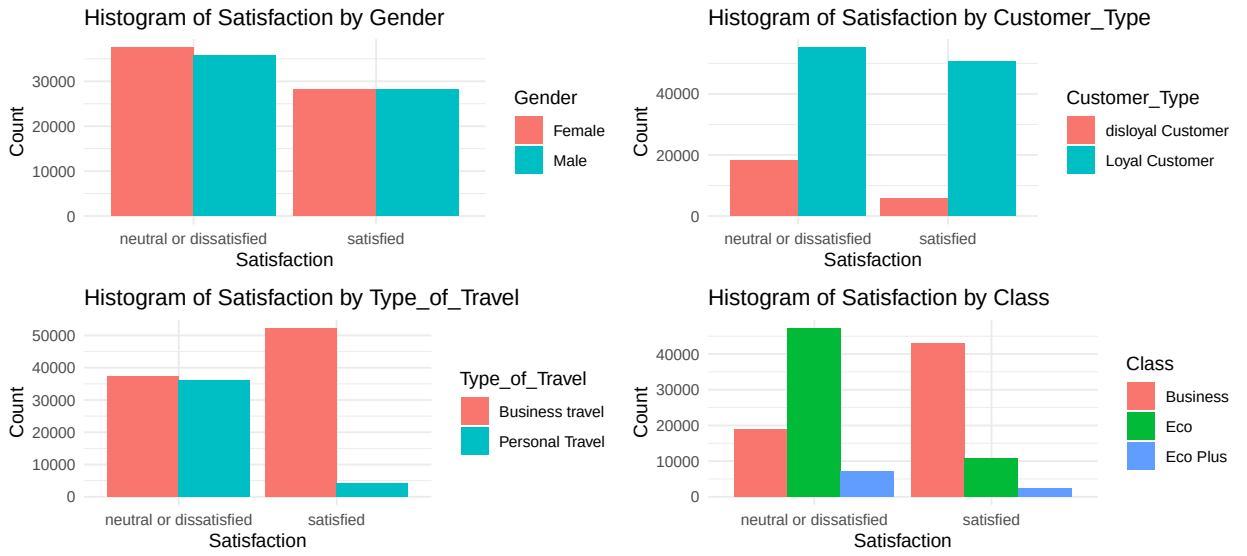
1. The age distribution is bimodal, which means that it has two distinct peaks. This could be due to a number of factors, such as the airline's target customer base, or the typical age of people who travel by air.
2. The flight distance distribution is log-normal, which means that it is skewed to the left. This is likely due to the fact that there are a few very long flights, which skew the distribution.
3. The departure delay and arrival delay distributions are very similar, which suggests that they are caused by the same factors. These factors could include weather conditions, air traffic control delays, or mechanical problems with the aircraft.

Variables vs Target

Categorical Variables vs Target

```
# plots categorical variables vs satisfaction
plots = list()
for (col in names(data)[sapply(data, is.factor)]) {
  if (col == "satisfaction" || col %in% ratings_fts_names) {
    next
  }
  plot = ggplot(data, aes(x = satisfaction, fill = .data[[col]])) +
  theme_minimal() +
  geom_bar(position = "dodge") +
  labs(title = paste("Histogram of Satisfaction by", col), x = "Satisfaction", y = "Count")

  plots[[col]] = plot

}
```

```
grid.arrange(grobs = plots, ncol = 2)
```



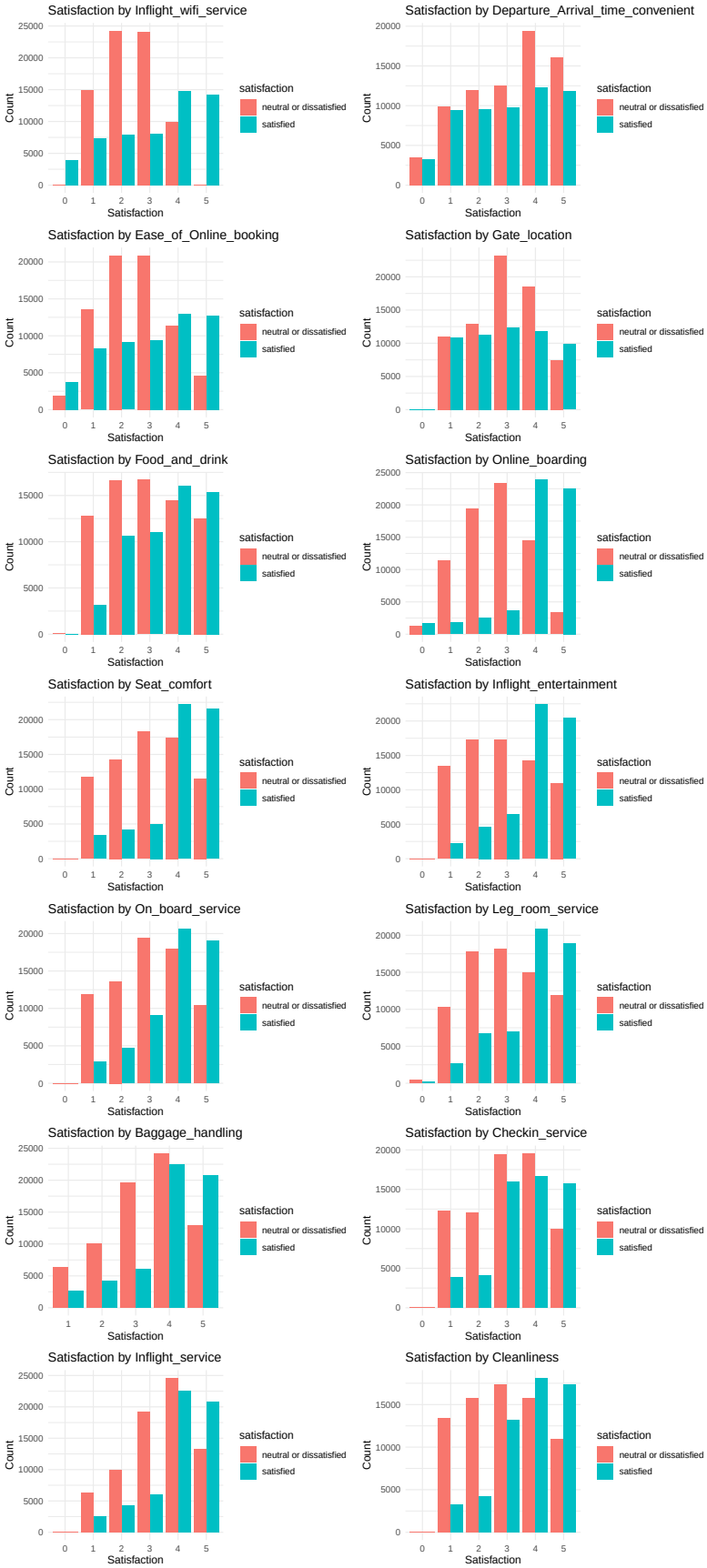
The observations from the graph analysis provide valuable insights into how different nominal features relate to passenger satisfaction:

1. Gender: both males and females appear to have similar distributions in terms of satisfaction, indicating that gender may not be a strong predictor of passenger satisfaction. The distributions closely resemble the overall distribution of the satisfaction feature.
2. Customer type: the graph suggests that "disloyal customers" are more likely to be unsatisfied or neutral compared to "loyal customers." This indicates that customer loyalty may play a role in passenger satisfaction, with loyal customers tending to be more satisfied.
3. Type of travel: The graph indicates that "personal travelers" are more likely to be unsatisfied compared to "business travelers." Conversely, "business travelers" tend to have a higher proportion of satisfied passengers. This finding suggests that the purpose of travel may have an influence on passenger satisfaction.
4. Class: The graph shows that "business class" passengers are more satisfied than unsatisfied, whereas "eco" and "eco plus" passengers tend to have a higher proportion of unsatisfied passengers. This suggests that the class of service provided by the airline may be a significant factor affecting passenger satisfaction.

Rating Features vs Target

```
# plots ratings features vs satisfaction
plots = list()
for (col in names(data)[sapply(data, is.factor)]) {
  if (!col %in% ratings_fts_names) {
    next
  }
  plot = ggplot(data, aes(x = .data[[col]], fill = satisfaction)) +
  theme_minimal() +
  geom_bar(position = "dodge") +
  labs(title = paste("Satisfaction by", col), x = "Satisfaction", y = "Count")

  plots[[col]] = plot
```

```
}

grid.arrange(grobs = plots, ncol = 2)
```


Satisfaction by Inflight_wifi_service
Satisfaction by Departure_Arrival_time_convenient
Satisfaction by Ease_of_Online_booking
Satisfaction by Gate_location
Satisfaction by Food_and_drink
Satisfaction by Online_boarding
Satisfaction by Seat_comfort
Satisfaction by Inflight_entertainment
Satisfaction by On_board_service
Satisfaction by Leg_room_service
Satisfaction by Baggage_handling
Satisfaction by Checkin_service
Satisfaction by Inflight_service
Satisfaction by Cleanliness
Count
Satisfaction
satisfaction
neutral or dissatisfied
satisfied

Based on the observed distribution of the ratings, we can draw the following conclusion:

1. For most of the ratings, the mean value for unsatisfied/neutral consumers is around 3, except for "Inflight Service" and "Baggage Handling."

```
# calculate the mean of the ratings of unsatisfied/neutral consumers
ratings_data = data[data$satisfaction == "neutral or dissatisfied", c(ratings_fts_names)]
ratings_data <- apply(ratings_data, 2, as.numeric)
ratings_mean = colMeans(ratings_data)
ratings_mean
```

```
##             Inflight_wifi_service Departure_Arrival_time_convenient
##                          2.398470                          3.130229
##            Ease_of_Online_booking                     Gate_location
##                          2.549512                          2.980184
##                    Food_and_drink                  Online_boarding
##                          2.958525                          2.658846
##                       Seat_comfort            Inflight_entertainment
##                          3.038525                          2.892236
##                  On_board_service                 Leg_room_service
##                          3.019570                          2.990495
##                  Baggage_handling                  Checkin_service
##                          3.374681                          3.043045
##                  Inflight_service                     Cleanliness
##                          3.389662                          2.932851
##                 On_board_service.1
##                          3.019570
```

2. For most of the ratings given by satisfied consumers, the mean value is around 4. However, it's important to note that we cannot generalize from this information alone.

```
# calculate the mean of the ratings of unsatisfied/neutral consumers
ratings_data = data[data$satisfaction == "satisfied", c(ratings_fts_names)]
ratings_data <- apply(ratings_data, 2, as.numeric)
ratings_mean = colMeans(ratings_data)
ratings_mean
```

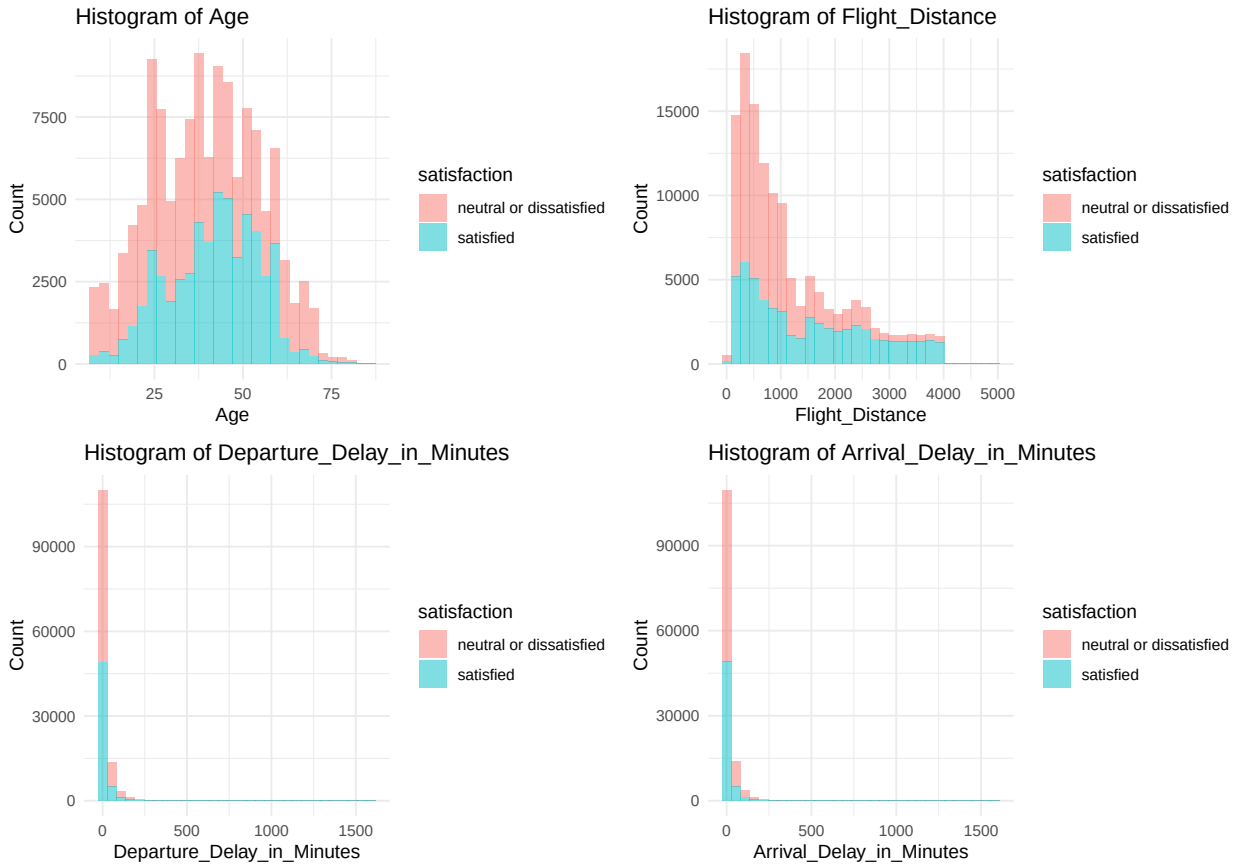
```
##             Inflight_wifi_service Departure_Arrival_time_convenient
##                          3.158135                          2.962497
##            Ease_of_Online_booking                     Gate_location
##                          3.026554                          2.972646
##                    Food_and_drink                  Online_boarding
##                          3.525061                          4.025648
##                       Seat_comfort            Inflight_entertainment
##                          3.966176                          3.964345
##                  On_board_service                 Leg_room_service
##                          3.856475                          3.820376
##                  Baggage_handling                  Checkin_service
##                          3.966638                          3.648786
##                  Inflight_service                     Cleanliness
##                          3.971277                          3.746134
##                 On_board_service.1
##                          3.856475
```

3. we can see that certain ratings, such as "Online_boarding," "Seat_comfort," "Inflight_entertainment," "On_board_service," and "Leg_room_service," exhibit similar distributions. The neutral/unsatisfied votes appear to follow a normal curve centered around a mean of approximately 3, while the satisfied

votes tend to concentrate on grades 4 and 5. This pattern strongly suggests that these ratings are not independent but rather correlated with each other.Through correlation matrix we'll have further considerations.

Numerical Variables vs Target

```
# plots numeric variables vs satisfaction excluding ratings features with histograms of different color
plots = list()
for (col in names(data)[sapply(data, is.numeric)]) {
  if (col %in% ratings_fts_names) {
    next
  }
  plot = ggplot(data, aes(x = .data[[col]], fill = satisfaction, group = satisfaction)) +
    theme_minimal() +
    geom_histogram(alpha = 0.5, bins = 30) +
    labs(title = paste("Histogram of", col), x = col, y = "Count")

  plots[[col]] = plot
}

grid.arrange(grobs = plots, ncol = 2)
```



Based on the boxplots and the information provided, we can make the following observations about the distributions of the numerical features:

1. Age: the boxplot for "Age" suggests that the distribution is approximately normal. Furthermore, it appears that neutral or dissatisfied customers tend to be slightly younger on average compared to

satisfied customers.

2. Flight Distance: the boxplot does not exhibit a normal distribution. Instead, it appears to have a right-skewed distribution. This is evident from the longer whisker on the right side, indicating that there are some outliers with larger flight distances.
3. Departure Delay in Minutes: the boxplot for "Departure Delay in Minutes" also shows a right-skewed distribution. The majority of the data appears to be concentrated towards the lower values, with a lot of outliers representing longer departure delays.
4. Arrival Delay in Minutes: similar to the "Departure Delay in Minutes," the boxplot for "Arrival Delay in Minutes" exhibits a right-skewed distribution. The bulk of the data is clustered towards the lower values, with a lot of outliers indicating longer arrival delays.

Based on these observations, it's essential to consider the appropriate data transformations or use different distribution models when working with "Flight Distance," "Departure Delay in Minutes," and "Arrival Delay in Minutes." For example, logarithmic transformations may be suitable to handle the skewed nature of these variables in statistical analyses and modeling tasks.

Correlation matrix

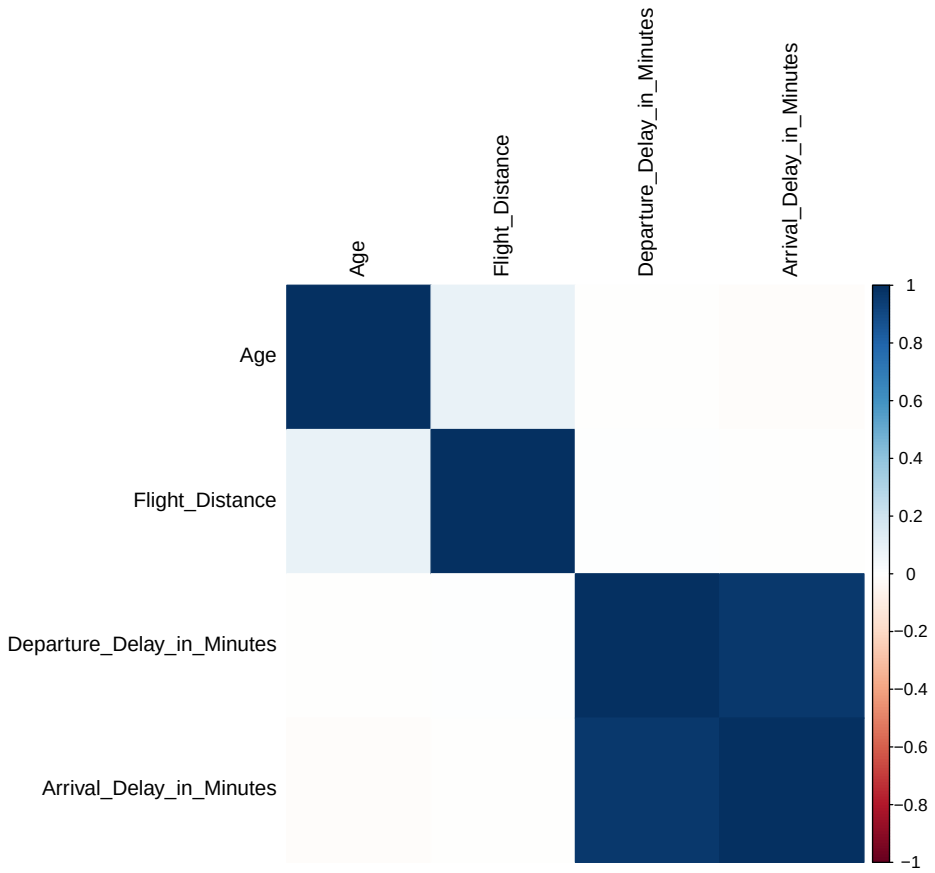
A correlation matrix is a table that shows how multiple variables in a dataset are related to each other. It provides a simple and concise way to see if there are any connections between the different variables.

In a correlation matrix, each row and column represent a variable, and the cells display the correlation coefficients between pairs of variables. The correlation coefficient is a value between -1 and 1 that indicates the strength and direction of the relationship:

A correlation coefficient close to 1 suggests a strong positive relationship (as one variable goes up, the other tends to go up as well). A correlation coefficient close to -1 indicates a strong negative relationship (as one variable goes up, the other tends to go down). A correlation coefficient close to 0 suggests little to no linear relationship between the variables. By examining the correlation matrix, we can quickly identify which variables are positively correlated, negatively correlated, or have no significant correlation. This helps us understand how different factors in the dataset might be connected or affect each other.

```
## CORRELATION MATRIX FOR NUMERICAL VARIABLES

ds_cor1 <- cor(subset(data,select = c(Age,Flight_Distance,Departure_Delay_in_Minutes,Arrival_Delay_in_M
```

```
options(repr.plot.width = 14, repr.plot.height = 8)
corrplot(ds_cor1, na.label = " ", method="color", tl.col = "black", tl.cex = 1)
```



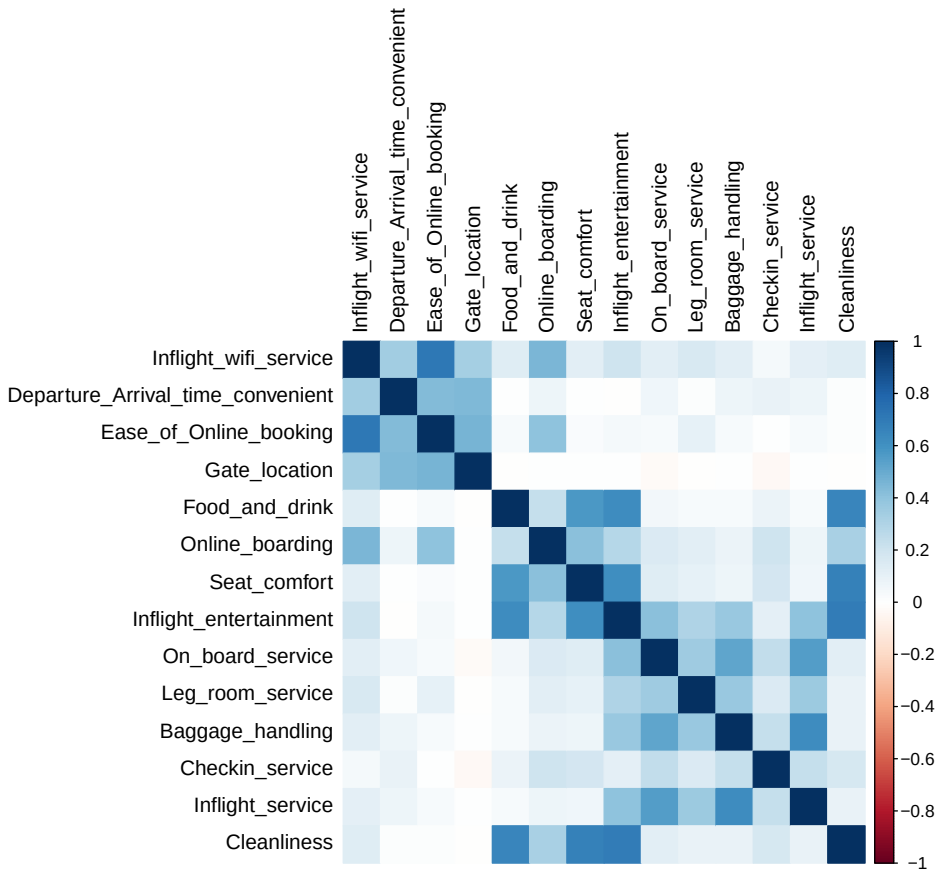

```r
ds_cor1_df <- data.frame()

# Loop through the rows and columns of 'ds_cor1'
for (row in rownames(ds_cor1)) {
  for (col in colnames(ds_cor1)) {
    # Check if the correlation is greater than 0.3 or less than -0.3
    if (ds_cor1[row, col] > 0.3 | ds_cor1[row, col] < -0.3) {
      # Create a new row with the feature names and correlation value
      new_row <- c(row, col, ds_cor1[row, col])
      # Append the new row to the 'ds_cor1_df'
      ds_cor1_df <- rbind(ds_cor1_df, new_row)
    }
  }
}

# Set appropriate column names for the result dataframe
colnames(ds_cor1_df) <- c("Feature1", "Feature2", "CorrelationValue")

# Delete correlation values for features with themselves
ds_cor1_df <- ds_cor1_df[ds_cor1_df$Feature1 != ds_cor1_df$Feature2, ]

# Remove duplicate rows based on the CorrelationValue column
ds_cor1_df <- ds_cor1_df[!duplicated(ds_cor1_df$CorrelationValue), ]

# Print the resulting dataframe
print(ds_cor1_df)
```

```
##                    Feature1                  Feature2  CorrelationValue
## 4 Departure_Delay_in_Minutes Arrival_Delay_in_Minutes 0.965291183546321
```

1. **Departure Delay in Minutes and Arrival Delay in Minutes: 0.965** The correlation coefficient of 0.965 between "Departure_Delay_in_Minutes" and "Arrival_Delay_in_Minutes" indicates a strong positive relationship between these two variables. This suggests that there is a high tendency for flights with longer departure delays to also have longer arrival delays

```
## CORRELATION MATRIX FOR ORDINAL VARIABLES
ordinal_variables = c('Inflight_wifi_service', 'Departure_Arrival_time_convenient', 'Ease_of_Online_book

# for each ordinal variables, change the values to numeric
for (col in ordinal_variables) {
  data[[col]] = as.numeric(data[[col]])
}

ds_cor2 <- cor(subset(data, select = ordinal_variables))
```

```
options(repr.plot.width = 14, repr.plot.height = 8)
corrplot(ds_cor2, na.label=" ", tl.cex=1, tl.col="black", method="color")
```




```
ds_cor2_df = data.frame()

for (row in rownames(ds_cor2)) {
  for (col in colnames(ds_cor2)) {
    # Check if the partial correlation is greater than 0.5 or less than -0.5
    if (ds_cor2[row, col] > 0.5 | ds_cor2[row, col] < -0.5) {
```

```
      # Create a new row with the feature names and correlation value
      new_row <- c(row, col, ds_cor2[row, col])
      # Append the new row to the 'partial_corr_cat_dataframe'
      ds_cor2_df <- rbind(ds_cor2_df, new_row)
    }
  }
}

# Set appropriate column names for the result dataframe
colnames(ds_cor2_df)= c("Feature1", "Feature2", "CorrelationValue")

# delete correlation value for feature with themselves
ds_cor2_df = ds_cor2_df[ds_cor2_df$Feature1 != ds_cor2_df$Feature2,]

ds_cor2_df <- ds_cor2_df[!duplicated(ds_cor2_df$CorrelationValue), ]
# Print the resulting dataframe (only the head)
print(ds_cor2_df)
```

```
##                  Feature1               Feature2  CorrelationValue
## 2   Inflight_wifi_service Ease_of_Online_booking 0.714888463225064
## 8          Food_and_drink           Seat_comfort 0.575993384494181
## 9          Food_and_drink Inflight_entertainment  0.62336587526467
## 10         Food_and_drink            Cleanliness 0.658026031662384
## 14           Seat_comfort Inflight_entertainment 0.611949136293315
## 15           Seat_comfort            Cleanliness 0.679656973056977
## 19 Inflight_entertainment            Cleanliness 0.692491066657791
## 21       On_board_service       Baggage_handling 0.520399703885874
## 22       On_board_service       Inflight_service 0.551459657463911
## 26       Baggage_handling       Inflight_service 0.629492431026368
```

Considerations

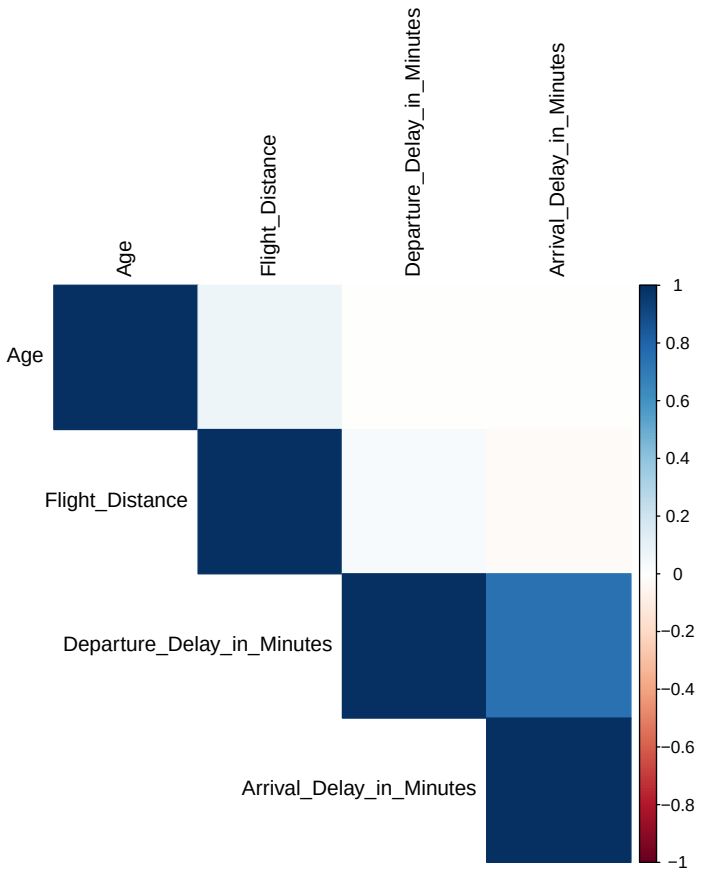
1. **Inflight Wifi Service and Ease of Online Booking (Correlation: 0.715):** There is a strong positive correlation between inflight wifi service and the ease of online booking. Passengers who find online booking convenient are more likely to appreciate the availability and quality of inflight wifi.
2. **Food and Drink and Seat Comfort (Correlation: 0.576):** A moderate positive correlation exists between food and drink options and seat comfort. Passengers who enjoy comfortable seating are likely to have a favorable perception of the available food and drink offerings.
3. **Food and Drink and Inflight Entertainment (Correlation: 0.623):** There is a moderate positive correlation between food and drink services and inflight entertainment. Passengers who have a satisfying dining experience may also be more likely to enjoy the available inflight entertainment options.
4. **Food and Drink and Cleanliness (Correlation: 0.658):** Food and drink services exhibit a moderate positive correlation with cleanliness. Passengers who are pleased with the cleanliness of the cabin are likely to have a positive dining experience as well.
5. **Seat Comfort and Inflight Entertainment (Correlation: 0.612):** There is a moderate positive correlation between seat comfort and inflight entertainment. Passengers who find their seats comfortable are more likely to engage and enjoy the available entertainment options.
6. **Seat Comfort and Cleanliness (Correlation: 0.680):** A moderate positive correlation exists between seat comfort and cleanliness. Passengers who experience clean cabins are likely to perceive the seating as more comfortable.

7. **Inflight Entertainment and Cleanliness (Correlation: 0.692):** There is a moderate positive correlation between inflight entertainment and cleanliness. Passengers who have access to enjoyable inflight entertainment are more likely to appreciate the cleanliness of the aircraft.
8. **On-board Service and Baggage Handling (Correlation: 0.520):** On-board service and baggage handling exhibit a moderate positive correlation. Passengers who receive good on-board service are likely to have positive experiences with baggage handling.
9. **On-board Service and Inflight Service (Correlation: 0.551):** There is a moderate positive correlation between on-board service and inflight service. Passengers who receive satisfactory on-board service may also have positive interactions with the inflight service.
10. **Baggage Handling and Inflight Service (Correlation: 0.629):** Baggage handling and inflight service show a moderate positive correlation. Passengers who have a positive experience with baggage handling are likely to have favorable interactions with the inflight service staff as well.

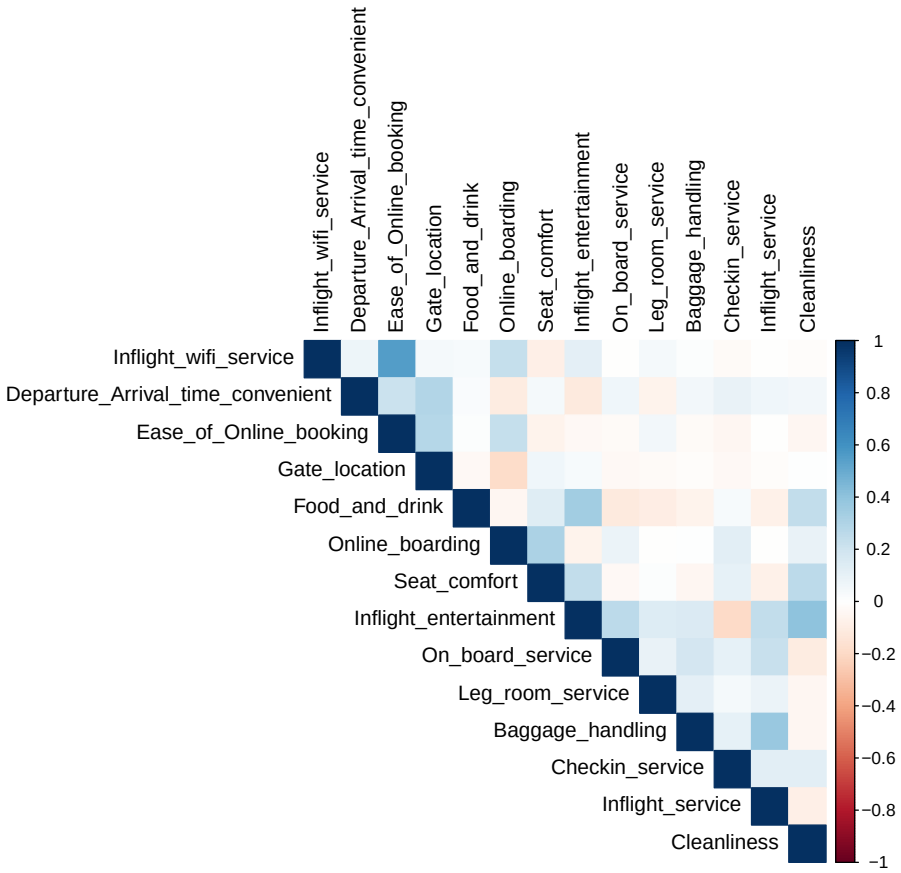
Partial correlation matrix

Partial correlations are used to explore and understand the direct relationship between two variables while controlling for the influence of other variables. In simpler terms, it helps us figure out the true connection between two factors by removing the effects of other factors that might be influencing them both Using partial correlations allows us to dig deeper and find the direct links between variables, giving us a more accurate understanding of their relationships in complex datasets.

```
numerical_features <- c("Age", "Flight_Distance", "Departure_Delay_in_Minutes", "Arrival_Delay_in_Minut
partial_correlations_num <- pcor(data[, numerical_features], method = "spearman")

# make the matrix
partial_corr_matrix_num <- partial_correlations_num$estimate

# Customize the plot size
options(repr.plot.width = 200, repr.plot.height = 200)

# Create the correlation plot
corrplot(partial_corr_matrix_num,
         method = "color",  # Choose "color" to visualize the correlations using colors
         type = "upper",    # Only plot the upper triangular part of the matrix (to avoid duplication)
         tl.col = "black",  # Label color
         tl.cex = 1,        # Label font size
)
```




```r
#PARTIAL CORRELATION MATRIX FOR CATEGORICAL/ORDINAL
categorical_features = c('Inflight_wifi_service', 'Departure_Arrival_time_convenient', 'Ease_of_Online_
partial_correlations_cat <- pcor(data[, categorical_features], method = "spearman")

# make the matrix
partial_corr_matrix_cat <- partial_correlations_cat$estimate

# Customize the plot size
options(repr.plot.width = 200, repr.plot.height = 200)

# Create the correlation plot
corrplot(partial_corr_matrix_cat,
         method = "color",  # Choose "color" to visualize the correlations using colors
         type = "upper",    # Only plot the upper triangular part of the matrix (to avoid duplication)
         tl.col = "black",  # Label color
         tl.cex = 1,        # Label font size
)
```




```
partial_correlations_num_dataframe = as.data.frame(partial_correlations_num$estimate)

partial_corr_num_dataframe = data.frame()

for (row in rownames(partial_correlations_num_dataframe)) {
  for (col in colnames(partial_correlations_num_dataframe)) {
    # Check if the partial correlation is greater than 0.5 or less than -0.5
    if (partial_correlations_num_dataframe[row, col] > 0.5 | partial_correlations_num_dataframe[row, co
      # Create a new row with the feature names and correlation value
      new_row <- c(row, col, partial_correlations_num_dataframe[row, col])
      # Append the new row to the 'partial_corr_num_dataframe'
      partial_corr_num_dataframe <- rbind(partial_corr_num_dataframe, new_row)
    }
  }
}

# Set appropriate column names for the result dataframe
colnames(partial_corr_num_dataframe)= c("Feature1", "Feature2", "CorrelationValue")

# delete correlation value for feature with themselves
partial_corr_num_dataframe = partial_corr_num_dataframe[partial_corr_num_dataframe$Feature1 != partial_


partial_corr_num_dataframe <- partial_corr_num_dataframe[!duplicated(partial_corr_num_dataframe$Correla

# Print the resulting dataframe
```

```r
print(partial_corr_num_dataframe)
```

```
##                      Feature1                 Feature2 CorrelationValue
## 4 Departure_Delay_in_Minutes Arrival_Delay_in_Minutes 0.74052816105061
```

```r
partial_correlations_cat_dataframe = as.data.frame(partial_correlations_cat$estimate)

partial_corr_cat_dataframe = data.frame()

for (row in rownames(partial_correlations_cat_dataframe)) {
  for (col in colnames(partial_correlations_cat_dataframe)) {
    # Check if the partial correlation is greater than 0.5 or less than -0.5
    if (partial_correlations_cat_dataframe[row, col] > 0.3 | partial_correlations_cat_dataframe[row, co
      # Create a new row with the feature names and correlation value
      new_row <- c(row, col, partial_correlations_cat_dataframe[row, col])
      # Append the new row to the 'partial_corr_cat_dataframe'
      partial_corr_cat_dataframe <- rbind(partial_corr_cat_dataframe, new_row)
    }
  }
}

# Set appropriate column names for the result dataframe
colnames(partial_corr_cat_dataframe)= c("Feature1", "Feature2", "CorrelationValue")

# delete correlation value for feature with themselves
partial_corr_cat_dataframe = partial_corr_cat_dataframe[partial_corr_cat_dataframe$Feature1 != partial_

partial_corr_cat_dataframe <- partial_corr_cat_dataframe[!duplicated(partial_corr_cat_dataframe$Correla
# Print the resulting dataframe
print(partial_corr_cat_dataframe)
```

```
##                  Feature1               Feature2  CorrelationValue
## 2   Inflight_wifi_service Ease_of_Online_booking  0.55209292676152
## 8          Food_and_drink Inflight_entertainment 0.349220795334936
## 10        Online_boarding           Seat_comfort 0.318097785008979
## 15 Inflight_entertainment            Cleanliness 0.400565997111643
## 19       Baggage_handling       Inflight_service 0.371817567906577
```

We computed the partial correlations for each pair of categorical features and identified correlations with an absolute value greater than 0.5. We then removed rows where Feature1 and Feature2 are equal and removed any duplicate correlations to obtain the final results.

Below are the significant partial correlations between categorical features:

1. **Inflight WiFi Service and Ease of Online Booking: 0.552** There is a moderate positive correlation between passengers' satisfaction with inflight WiFi service and the ease of the online booking process.
2. **Food and Drink and Inflight Entertainment: 0.349** Passengers who are satisfied with the food and drink options onboard are more likely to enjoy the inflight entertainment.
3. **Online Boarding and Seat Comfort: 0.318** There is a positive correlation between passengers' satisfaction with the online boarding process and their comfort with the seating arrangements.
4. **Inflight Entertainment and Cleanliness: 0.401** Passengers who find the inflight entertainment enjoyable are more likely to rate the cleanliness of the aircraft positively.
5. **Baggage Handling and Inflight Service: 0.372** There is a positive correlation between passengers' satisfaction with baggage handling and their perception of inflight service quality.

For numerical features, we have only one significant partial correlation:

6. **Departure Delay in Minutes and Arrival Delay in Minutes** There is a correlation between departure delay and arrival delay. If a flight experiences a departure delay, it is very likely to have an arrival delay as well.

Convert categorical to numerical

In this section we convert the categorical variables to numeric representation for modelling.

```
gender_map = c("Male" = 0, "Female" = 1)
data$Gender = gender_map[as.numeric(data$Gender)]

customer_type_map = c("Loyal Customer" = 0, "disloyal Customer" = 1)
data$Customer_Type = customer_type_map[as.numeric(data$Customer_Type)]

type_of_travel_map = c("Personal Travel" = 0, "Business travel" = 1)
data$Type_of_Travel = type_of_travel_map[as.numeric(data$Type_of_Travel)]

class_map = c("Business" = 0, "Eco" = 1, "Eco Plus" = 2)
data$Class = class_map[as.numeric(data$Class)]

satisfaction_map = c("neutral or dissatisfied" = 0, "satisfied" = 1)
data$satisfaction = satisfaction_map[as.numeric(data$satisfaction)]

for (col in ratings_fts_names) {
  data[[col]] <- as.numeric(data[[col]])
}
```

Data Preparation

Train test split

This section splits the data into training and testing sets, prints the proportion of satisfied and dissatisfied customers in each set, and saves the true values of the target variable for the test set.

```
set.seed(123)

train_index = sample(1:nrow(data), 0.8*nrow(data))
# 80% of data is used for training
train_data = data[train_index,]
# 20% of data is used for testing
test_data = data[-train_index,]
```

Features and Outputs

```
# Seperate y_train and y_test for further use
X_train = as.matrix(train_data %>% dplyr::select(-satisfaction))
y_train = train_data$satisfaction

X_test = as.matrix(test_data %>% dplyr::select(-satisfaction))
y_test = test_data$satisfaction
```

Number of Samples

```
# Number of samples in train data
train_rows <- nrow(train_data)
print(train_rows)
```

```
## [1] 103589
```

```
# Number of samples in test data
test_rows <- nrow(test_data)
print(test_rows)
```

```
## [1] 25898
```

Data Balance

```
# Proportion of satisfied and unsatisfied customers for train data
prop.table(table(y_train))
```

```
## y_train
##         0         1
## 0.5668845 0.4331155
```

```
# Proportion of satisfied and unsatisfied customers for test data
prop.table(table(y_test))
```

```
## y_test
##        0        1
## 0.559966 0.440034
```

As we can see the proportion of binary classes in train and test are similar. We can say that our test data is representative of train data.

In this project we want to find unsatisfied customers (class 0). We hypothesis that if we find the unsatisfied customers, then we arrange our customer satisfaction campaign accordingly.

Classification Models

Classification, a form of supervised learning, involves predicting the qualitative response of an observation by assigning it to a specific category or class. Multivariate techniques excel in constructing prediction models, forming the foundation of statistical classification. In the realm of machine learning, this process is referred to as supervised learning.

To build a classifier, we utilize a set of training observations. From a geometric perspective, an allocation rule determines how the input space is divided into regions, each labeled according to its classification. The boundaries of these regions can vary based on the assumptions made when constructing the classification model.

Logistic Regression

In logistic model observations are assumed to be realizations of independent Bernoulli random variables. It is like if a customer will be satisfied with flight or not.

Basic Logistic Classifier

```
# Model definition with all features:
glm_full<- glm(data = train_data,satisfaction ~ .,
```

```
                family = "binomial")
# summary of full model
summary(glm_full)
```

```
##
## Call:
## glm(formula = satisfaction ~ ., family = "binomial", data = train_data)
##
## Coefficients:
##                                    Estimate Std. Error  z value Pr(>|z|)
## (Intercept)                      -1.009e+01  9.391e-02 -107.446  < 2e-16 ***
## Gender                            6.254e-02  1.947e-02    3.212  0.00132 **
## Customer_Type                     2.097e+00  2.952e-02   71.047  < 2e-16 ***
## Age                              -8.023e-03  7.113e-04  -11.279  < 2e-16 ***
## Type_of_Travel                   -2.847e+00  3.006e-02  -94.735  < 2e-16 ***
## Class                            -5.084e-01  1.903e-02  -26.717  < 2e-16 ***
## Flight_Distance                  -4.382e-06  1.123e-05   -0.390  0.69637
## Inflight_wifi_service             4.003e-01  1.150e-02   34.797  < 2e-16 ***
## Departure_Arrival_time_convenient -1.308e-01  8.157e-03  -16.034  < 2e-16 ***
## Ease_of_Online_booking           -1.567e-01  1.135e-02  -13.804  < 2e-16 ***
## Gate_location                     2.172e-02  9.170e-03    2.369  0.01785 *
## Food_and_drink                   -2.931e-02  1.072e-02   -2.734  0.00626 **
## Online_boarding                   6.238e-01  1.029e-02   60.605  < 2e-16 ***
## Seat_comfort                      5.770e-02  1.121e-02    5.148 2.63e-07 ***
## Inflight_entertainment            5.297e-02  1.424e-02    3.719  0.00020 ***
## On_board_service                  3.075e-01  1.015e-02   30.294  < 2e-16 ***
## Leg_room_service                  2.533e-01  8.525e-03   29.710  < 2e-16 ***
## Baggage_handling                  1.359e-01  1.139e-02   11.924  < 2e-16 ***
## Checkin_service                   3.290e-01  8.554e-03   38.455  < 2e-16 ***
## Inflight_service                  1.270e-01  1.201e-02   10.575  < 2e-16 ***
## Cleanliness                       2.323e-01  1.214e-02   19.140  < 2e-16 ***
## Departure_Delay_in_Minutes        4.940e-03  9.854e-04    5.014 5.34e-07 ***
## Arrival_Delay_in_Minutes         -9.763e-03  9.735e-04  -10.029  < 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: 141746  on 103588  degrees of freedom
## Residual deviance:  69319  on 103566  degrees of freedom
## AIC: 69365
##
## Number of Fisher Scoring iterations: 6
```

As we can see from model statistics, the p value of "flight distance" feature is not lower than 0.5 which indicates that the effect of "flight distance" on prediction of satisfaction is insignificant. However, we would like to keep all features in the model for now. Then we will full model with feature selection models.

In the code snippet below, we can observe the calculation goodness of fit, also known as R-squared (R^2). R^2 serves as a metric to assess how well the GLM (Generalized Linear Model) model aligns with the given data in comparison to a null model, which solely consists of an intercept without any predictors. The R^2 value is derived using a simple formula: 1 minus the ratio of the deviance of the fitted model to the deviance of the null model.

R^2 value provides insights into the amount of variation in the data that can be accounted for by the model,

thus serving as a measure of its effectiveness.

```
r2<- 1 - (summary(glm_full)$deviance/summary(glm_full)$null.deviance)
r2
```

```
## [1] 0.5109608
```

In the provided line of code, the calculation of the variance inflation factor (VIF) is performed by taking the reciprocal of 1 minus the R^2 value. The VIF serves as a metric to evaluate the presence of multicollinearity, which refers to the correlation between predictor variables within a model. High levels of multicollinearity can introduce issues regarding the reliability and interpretability of the GLM (Generalized Linear Model).

```
1/(1-r2)
```

```
## [1] 2.044826
```

Logistic Regression with Backward Variable Selection

Collinearity refers to a situation where two or more predictor variables in a statistical model are closely related to each other. The presence of collinearity can introduce instability to the model and make it more challenging to accurately estimate the effect of each predictor variable.

Ideally, we aim for low collinearity among predictor variables. When collinearity is low, it implies that the predictor variables are independent or weakly correlated, allowing for more reliable estimation of their individual effects on the model.

The Variance Inflation Factor (VIF) is a measure used to assess collinearity among predictor variables within a multiple regression model. It is computed by taking the ratio of the variance of all the weights in the model divided by the variance of a single weight if that particular predictor variable were fitted alone.

A VIF of 1 indicates no collinearity, while a VIF between 1 and 5 suggests moderate collinearity. On the other hand, a VIF greater than 5 indicates high collinearity, which is undesirable and should be avoided.

To address high collinearity, we may choose to remove certain predictor variables from the model and observe the impact on the corresponding R-squared value. By iteratively adjusting the predictor variables, we can determine a suitable set of variables that minimize collinearity and yield a more robust model.

Variable selection is crucial in regression models to ensure interpretability, computational efficiency, and generalization. It involves removing irrelevant variables, reducing redundancy, and combating overfitting. By choosing relevant variables, the model becomes easier to interpret, algorithms work faster with high-dimensional data, and overfitting is reduced. The focus is on low test error rather than training error. Variable selection is especially important for high-dimensional datasets with more features than observations. Here, we are gonna use backward selection starting from full model and reducing number of variables in order of high VIF factor to lower VIF factors to decrease multi-collinearity and increase robustness of model.

```
# VIF Iteration 0
# The process is done iteratively where we delete one variable at time
vif_values <- VIF(glm_full)

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

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

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

```
##                            Variable       VIF
```

```
## 1           Arrival_Delay_in_Minutes 14.215472
## 2         Departure_Delay_in_Minutes 14.179745
## 3             Inflight_entertainment  3.259516
## 4             Ease_of_Online_booking  2.605648
## 5                         Cleanliness  2.477654
## 6              Inflight_wifi_service  2.234296
## 7                       Seat_comfort  2.056168
## 8                     Food_and_drink  2.028994
## 9                   Inflight_service  2.003726
## 10                  Baggage_handling  1.815847
## 11 Departure_Arrival_time_convenient  1.710988
## 12                    Type_of_Travel  1.701605
## 13                  On_board_service  1.630928
## 14                     Customer_Type  1.586551
## 15                     Gate_location  1.521229
## 16                   Online_boarding  1.493912
## 17                             Class  1.456328
## 18                   Flight_Distance  1.336138
## 19                  Leg_room_service  1.218179
## 20                   Checkin_service  1.208464
## 21                               Age  1.183984
## 22                            Gender  1.007103
```

```r
# VIF Iteration 1
# The process is done iteratively where we delete one variable at time
# Model definition:
glm_vif1<- glm(data = train_data,
               satisfaction ~ .-Arrival_Delay_in_Minutes,
               family = "binomial")
vif_values <- VIF(glm_vif1)

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

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

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

```
##                            Variable      VIF
## 1             Inflight_entertainment 3.255757
## 2             Ease_of_Online_booking 2.605184
## 3                         Cleanliness 2.473880
## 4              Inflight_wifi_service 2.233429
## 5                       Seat_comfort 2.055698
## 6                     Food_and_drink 2.024997
## 7                   Inflight_service 2.002608
## 8                   Baggage_handling 1.816225
## 9  Departure_Arrival_time_convenient 1.712667
## 10                    Type_of_Travel 1.699870
## 11                  On_board_service 1.631063
## 12                     Customer_Type 1.584320
## 13                     Gate_location 1.521679
```

```
## 14                   Online_boarding 1.492315
## 15                             Class 1.456741
## 16                   Flight_Distance 1.336573
## 17                  Leg_room_service 1.218233
## 18                   Checkin_service 1.208269
## 19                               Age 1.183542
## 20        Departure_Delay_in_Minutes 1.019193
## 21                            Gender 1.007146
```

```
# VIF Iteration 2
# The process is done iteratively where we delete one variable at time
# Model definition:
glm_vif2<- glm(data = train_data,
               satisfaction ~ .
              -Arrival_Delay_in_Minutes
              -Inflight_entertainment,
               family = "binomial")
vif_values <- VIF(glm_vif2)

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

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

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

```
##                            Variable      VIF
## 1             Ease_of_Online_booking 2.598560
## 2              Inflight_wifi_service 2.186179
## 3                        Cleanliness 2.077800
## 4                       Seat_comfort 1.926523
## 5                  Inflight_service 1.865661
## 6                  Baggage_handling 1.774342
## 7                    Food_and_drink 1.756367
## 8  Departure_Arrival_time_convenient 1.709160
## 9                    Type_of_Travel 1.669340
## 10                    Customer_Type 1.540963
## 11                 On_board_service 1.526698
## 12                    Gate_location 1.521864
## 13                   Online_boarding 1.474095
## 14                             Class 1.447649
## 15                   Flight_Distance 1.336219
## 16                  Leg_room_service 1.203120
## 17                               Age 1.179147
## 18                   Checkin_service 1.167757
## 19        Departure_Delay_in_Minutes 1.017485
## 20                            Gender 1.005707
```

```
# VIF Iteration 3
# The process is done iteratively where we delete one variable at time
# Model definition:
glm_vif3<- glm(data = train_data,
```

```
                satisfaction ~ .
                -Arrival_Delay_in_Minutes
                -Inflight_entertainment
                -Ease_of_Online_booking ,
                family = "binomial")
vif_values <- VIF(glm_vif3)

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

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

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

```
##                            Variable      VIF
## 1                       Cleanliness 2.065788
## 2                      Seat_comfort 1.912316
## 3                  Inflight_service 1.866483
## 4                  Baggage_handling 1.775338
## 5                    Food_and_drink 1.753620
## 6                    Type_of_Travel 1.647770
## 7  Departure_Arrival_time_convenient 1.579042
## 8              Inflight_wifi_service 1.550964
## 9                   On_board_service 1.526237
## 10                     Customer_Type 1.519908
## 11                             Class 1.439914
## 12                  Online_boarding 1.414436
## 13                    Gate_location 1.391271
## 14                  Flight_Distance 1.335326
## 15                 Leg_room_service 1.198439
## 16                               Age 1.172301
## 17                  Checkin_service 1.166160
## 18       Departure_Delay_in_Minutes 1.016243
## 19                           Gender 1.005019
```

```
# VIF Iteration 4
# The process is done iteratively where we delete one variable at time
# Model definition:
glm_vif4<- glm(data = train_data,
                satisfaction ~ .
               -Arrival_Delay_in_Minutes
               -Inflight_entertainment
               -Ease_of_Online_booking
               -Cleanliness ,
                family = "binomial")
vif_values <- VIF(glm_vif4)

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

# Sort the data frame in decreasing order of VIF values
```

```r
sorted_df <- vif_df[order(-vif_df$VIF), ]
rownames(sorted_df) <- NULL

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

```
##                            Variable      VIF
## 1                  Inflight_service 1.876230
## 2                  Baggage_handling 1.785063
## 3                    Type_of_Travel 1.638313
## 4                      Seat_comfort 1.589209
## 5  Departure_Arrival_time_convenient 1.583651
## 6              Inflight_wifi_service 1.552265
## 7                   On_board_service 1.532209
## 8                      Customer_Type 1.511725
## 9                              Class 1.437882
## 10                   Food_and_drink 1.436115
## 11                  Online_boarding 1.409432
## 12                    Gate_location 1.399444
## 13                  Flight_Distance 1.335607
## 14                 Leg_room_service 1.202751
## 15                              Age 1.169812
## 16                  Checkin_service 1.163605
## 17       Departure_Delay_in_Minutes 1.013200
## 18                           Gender 1.004492
```

We deleted all the features with VIF greater than 2 to decrease multicollinearity. Let's check the current model statistics.

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

```
##
## Call:
## glm(formula = satisfaction ~ . - Arrival_Delay_in_Minutes - Inflight_entertainment -
##     Ease_of_Online_booking - Cleanliness, family = "binomial",
##     data = train_data)
##
## Coefficients:
##                                  Estimate Std. Error  z value Pr(>|z|)
## (Intercept)                    -1.019e+01  9.050e-02 -112.653  < 2e-16 ***
## Gender                          6.586e-02  1.928e-02    3.415 0.000637 ***
## Customer_Type                   2.125e+00  2.894e-02   73.426  < 2e-16 ***
## Age                            -8.147e-03  7.027e-04  -11.595  < 2e-16 ***
## Type_of_Travel                 -2.850e+00  2.952e-02  -96.543  < 2e-16 ***
## Class                          -4.662e-01  1.866e-02  -24.977  < 2e-16 ***
## Flight_Distance                -4.647e-06  1.110e-05   -0.419 0.675402
## Inflight_wifi_service           3.193e-01  9.487e-03   33.657  < 2e-16 ***
```

```
## Departure_Arrival_time_convenient -1.661e-01  7.799e-03  -21.303  < 2e-16 ***
## Gate_location                     -1.518e-02  8.700e-03   -1.745 0.080960 .
## Food_and_drink                     8.647e-02  8.923e-03    9.691  < 2e-16 ***
## Online_boarding                    6.119e-01  9.991e-03   61.239  < 2e-16 ***
## Seat_comfort                       1.868e-01  9.779e-03   19.099  < 2e-16 ***
## On_board_service                   3.223e-01  9.728e-03   33.130  < 2e-16 ***
## Leg_room_service                   2.524e-01  8.399e-03   30.056  < 2e-16 ***
## Baggage_handling                   1.543e-01  1.115e-02   13.839  < 2e-16 ***
## Checkin_service                    3.337e-01  8.313e-03   40.142  < 2e-16 ***
## Inflight_service                   1.480e-01  1.147e-02   12.907  < 2e-16 ***
## Departure_Delay_in_Minutes        -4.332e-03  2.620e-04  -16.538  < 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: 141746  on 103588  degrees of freedom
## Residual deviance:  70189  on 103570  degrees of freedom
## AIC: 70227
##
## Number of Fisher Scoring iterations: 5
```

"Flight_Distance" and "Gate_location" features are insignificant in prediction of satisfaction. Let's drop them from model.

```
# Drop Flight_Distance and Gate_location from model.

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

summary(glm_backward)
```

```
##
## Call:
## glm(formula = satisfaction ~ . - Arrival_Delay_in_Minutes - Inflight_entertainment -
##     Ease_of_Online_booking - Cleanliness - Flight_Distance -
##     Gate_location, family = "binomial", data = train_data)
##
## Coefficients:
##                                    Estimate Std. Error  z value Pr(>|z|)
## (Intercept)                       -1.024e+01  8.602e-02 -119.103  < 2e-16 ***
## Gender                             6.591e-02  1.928e-02    3.418  0.00063 ***
## Customer_Type                      2.124e+00  2.777e-02   76.477  < 2e-16 ***
## Age                               -8.133e-03  7.012e-04  -11.599  < 2e-16 ***
## Type_of_Travel                    -2.843e+00  2.900e-02  -98.031  < 2e-16 ***
## Class                             -4.629e-01  1.762e-02  -26.269  < 2e-16 ***
## Inflight_wifi_service              3.166e-01  9.320e-03   33.971  < 2e-16 ***
## Departure_Arrival_time_convenient -1.723e-01  6.973e-03  -24.705  < 2e-16 ***
```

```
## Food_and_drink                     8.683e-02  8.920e-03    9.734  < 2e-16 ***
## Online_boarding                    6.142e-01  9.880e-03   62.172  < 2e-16 ***
## Seat_comfort                       1.859e-01  9.766e-03   19.035  < 2e-16 ***
## On_board_service                   3.228e-01  9.719e-03   33.209  < 2e-16 ***
## Leg_room_service                   2.527e-01  8.388e-03   30.131  < 2e-16 ***
## Baggage_handling                   1.550e-01  1.114e-02   13.906  < 2e-16 ***
## Checkin_service                    3.341e-01  8.307e-03   40.216  < 2e-16 ***
## Inflight_service                   1.485e-01  1.146e-02   12.953  < 2e-16 ***
## Departure_Delay_in_Minutes        -4.336e-03  2.619e-04  -16.555  < 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: 141746  on 103588  degrees of freedom
## Residual deviance:  70192  on 103572  degrees of freedom
## AIC: 70226
##
## Number of Fisher Scoring iterations: 5
```

Interestingly, model with backward elimination variable selection has less R^2 value than full model. We will discuss the reason at the end of Logistic Regression part.

```
r2<- 1 - (summary(glm_backward)$deviance/summary(glm_backward)$null.deviance)
r2
```

```
## [1] 0.5047998
```

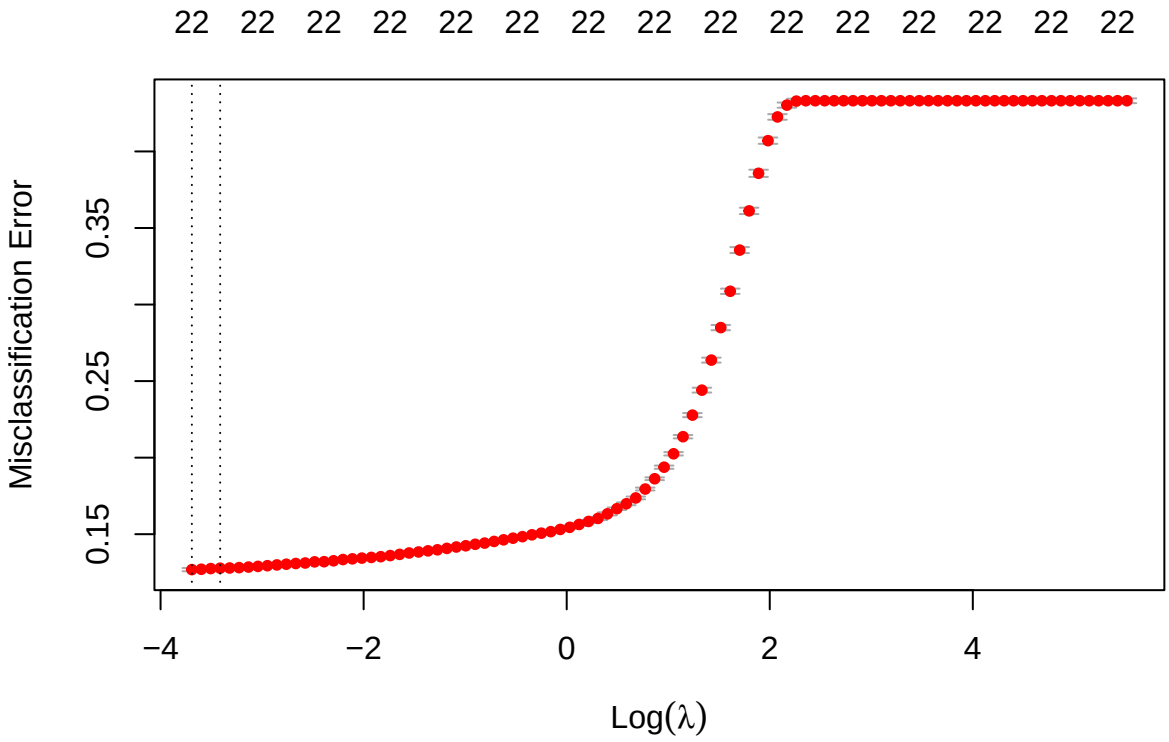
Logistic Regression with Shrinkage Method

Shrinkage methods, such as Ridge and Lasso regression, are techniques used in linear modeling to control model complexity and reduce overfitting. Instead of selecting a subset of predictors or setting some coefficients to zero, these methods constrain or regularize the coefficient estimates, effectively shrinking them towards zero. Ridge regression achieves this by using quadratic shrinking, while Lasso regression uses absolute-value shrinking. Other hybrid approaches, like the elastic net, combine elements of both methods.

One disadvantage of ridge regression is that it includes all predictors in the final model, unlike subset selection methods which typically select a subset of variables. To overcome this drawback, the lasso regression is used as an alternative. The lasso also shrinks the coefficient estimates towards zero, and it tends to perform better.

Lambda is the Tuning Parameter that controls the bias-variance tradeoff and we estimate its best value via cross-validation.

```
# We look for the best value for lambda
# We use cross validation glmnet
glm_lasso <- cv.glmnet(X_train, as.factor(y_train),
                       alpha = 0, family = "binomial", type.measure = "class")
plot(glm_lasso)
```




```
# We identify th best lambda value
best_lambda <- glm_lasso$lambda.min
best_lambda
```

```
## [1] 0.02491896
```

ROC Curve

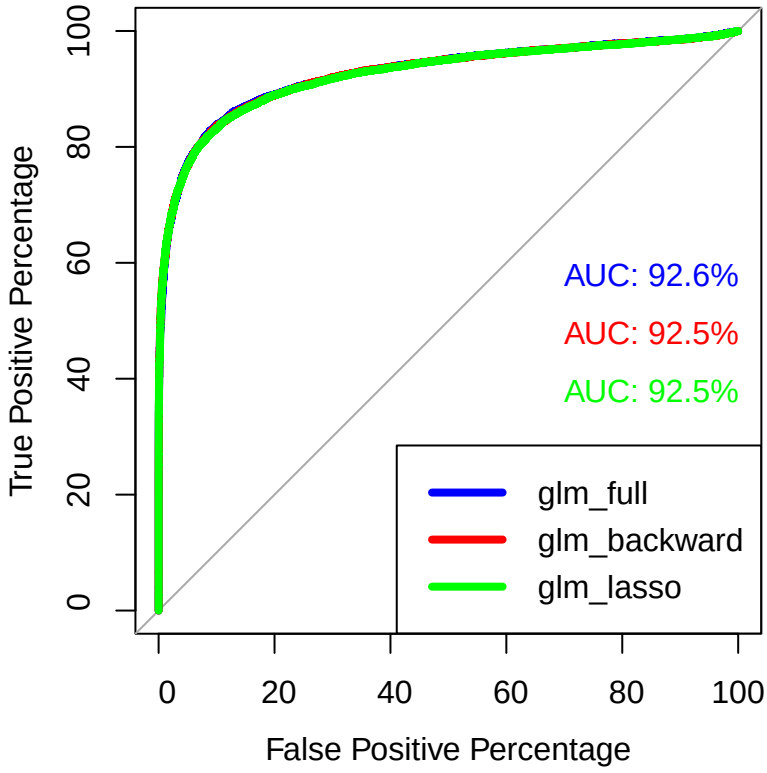
We created our 3 different logistic regression model. Now, it is time to compare them on test sets to see their robustness on generalization and power on predicting satisfaction of customer.

```
# Full model
# Computing the predictions with the model on the test set:
pred_glm_full<- predict(glm_full, data.frame(X_test), type = "response")

# Backward elimination selection
# Computing the predictions with the model on the test set:
pred_glm_backward<- predict(glm_backward, data.frame(X_test), type = "response")

# Lasso regresssion
# Computing the predictions with the model on the test set:
pred_glm_lasso<- predict(glm_lasso, X_test, type= "response", s = best_lambda)
```

The Receiver Operating Characteristics (ROC) curve illustrates the relationship between the True positive rate and the False positive rate across various thresholds. In an ideal scenario, the ROC curve would closely follow the top left corner. The overall effectiveness of a classifier, considering all thresholds, is quantified by the Area Under the Curve (AUC). A larger AUC value indicates a superior classifier performance.

```
##
## Call:
## roc.default(response = y_test, predictor = pred_glm_full, percent = TRUE,     quiet = TRUE, plot = TR
##
## Data: pred_glm_full in 14502 controls (y_test 0) < 11396 cases (y_test 1).
## Area under the curve: 92.56%
```

```
## Warning in roc.default(x, predictor, plot = TRUE, ...): Deprecated use a matrix
## as predictor. Unexpected results may be produced, please pass a numeric vector.
```



Ideally, we expect backward and lasso model to have better generaliation; however, in this scenario, the results yield opposite situation.

Comparison of Logistic Classifiers

We have 3 different Logistic Regression models 1- Basic Logistic Classifier : glm_full 2- Logistic Regression with Backward Variable Selection: glm_backward 3- Logistic Regression with Lasso Shrinkage: glm_lasso

Now, it is time to make predictions and compare metric results. But, first we should decide best thresholds for models.

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

```
    Precision = numeric(length(thresholds)),
    Recall = numeric(length(thresholds))
  )

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

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

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

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

  return(results_df)
}
```

```
evaluate_on_thresholds <- function(predictions,y_test, thresholds) {
  # Converting the prediction in {0,1} according to the chosen threshold:
  output_list <- list()

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

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


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

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

```
thresholds <- c(0.4, 0.5, 0.6, 0.7)
evaluate_on_thresholds(pred_glm_full, y_test, thresholds)
```

Threshold	Accuracy	F1_Score	Precision	Recall
0.4	0.8614	0.8739	0.8907	0.8577
0.5	0.8725	0.8880	0.8738	0.9028
0.6	0.8735	0.8925	0.8519	0.9370
0.7	0.8633	0.8874	0.8236	0.9619

```
thresholds <- c(0.4, 0.5, 0.6, 0.7)
evaluate_on_thresholds(pred_glm_backward, y_test, thresholds)
```

Threshold	Accuracy	F1_Score	Precision	Recall
0.4	0.8589	0.8715	0.8893	0.8544
0.5	0.8714	0.8870	0.8730	0.9015
0.6	0.8734	0.8924	0.8515	0.9375
0.7	0.8628	0.8871	0.8227	0.9624

```
thresholds <- c(0.4, 0.5, 0.6, 0.7)
evaluate_on_thresholds(pred_glm_lasso, y_test, thresholds)
```

Threshold	Accuracy	F1_Score	Precision	Recall
0.4	0.8565	0.8689	0.8896	0.8492
0.5	0.8710	0.8872	0.8694	0.9058
0.6	0.8708	0.8914	0.8419	0.9470
0.7	0.8536	0.8815	0.8058	0.9730

All 3 logistic regression models have highest F1 score with 0.6 threshold. Eventhough the expectation that backward variable selection and lasso regression model would suprass full_model, the results shows opposite. F1 score with 0.6 threshold for glm_full is higher than other models on the test set. We can conclude that generalizability of full model is higher. We may conclude that we have many samples but we do not have enough features to increase model generalizability, so decreasing number of variables or shrinking their weights do not make positive effect. We take pred_glm_full as a winner prediction from logistic regression part to further compare it with other models.

Naive Bayes

The Naive Bayes Classifier is a probabilistic algorithm used for binary classification. It assumes that features are independent of each other and calculates prior probabilities and likelihoods during the training phase. The prior probabilities represent the occurrence of each class, while the likelihoods determine the probability of a feature given a class. By applying Bayes' theorem, the algorithm calculates posterior probabilities for each class and assigns the instance to the class with the highest probability.

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

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

# Evaluate accuracy of classifier
mean(pred_naive_bayes == y_test)
```

```
## [1] 0.8647
```

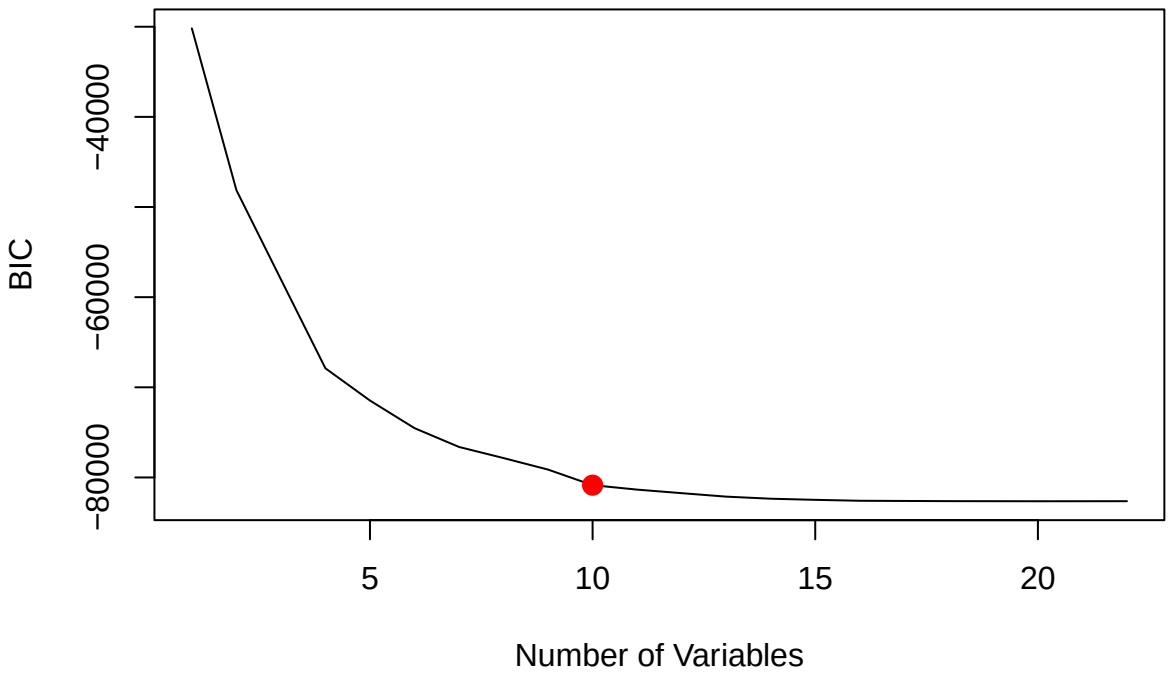
BIC (Bayesian Information Criterion) is a commonly used model selection criteria that help in selecting the best model among a set of competing models. It takes into account the goodness of fit of the model and penalize the complexity of the model to avoid overfitting.

BIC (Bayesian Information Criterion): It balances the trade-off between model fit and model complexity. Formula: BIC = -2 * log-likelihood + p * log(n)

- log-likelihood: The log-likelihood of the model, which measures how well the model fits the data.
- p: The number of parameters in the model.
- n: The sample size.

BIC penalizes model complexity more heavily than the Akaike Information Criterion (AIC). The lower the BIC value, the better the model is considered to be. Therefore, when comparing models, the model with the lowest BIC is preferred.

```
# Best Subset Selection

# The regsubsets() function (part of the leaps library)
# performs best subset selection by identifying the best
# model that contains a given number of predictors,
# where best is quantified using bic

n <- dim(X_train)[1]
regfit.full <- regsubsets(y_train~.,data=data.frame(X_train),nvmax=n)
reg.summary <- summary(regfit.full)

# Plotting BIC
# BIC with its smallest value
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
min <- which.min(reg.summary$bic)
points(10,reg.summary$bic[10],col="red",cex=2,pch=20)
```



Since BIC (Bayesian Information Criterion) does not change dramatically after 10th variable. We can use only 10 variables to decide satisfaction.

```
# choose 10 variable model
best_model <- coef(regfit.full, id = 10)
abs_coefficients <- abs(best_model)
sorted_variables <- names(sort(abs_coefficients, decreasing = TRUE))
column_names <- sorted_variables[-1]
new_X_train <- X_train[, column_names, drop = FALSE]
new_X_test <- X_test[, column_names, drop = FALSE]

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

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

# Evaluate accuracy
mean(pred_naive_bayes == y_test)
```

```
## [1] 0.8770948
```

The accuracy of model with 10 variables is higher. We can continue with this model.

K-Nearest Neigbors (KNN)

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for classification and regression tasks. It is a non-parametric algorithm, which means it doesn't make any assumptions about the underlying

distribution of the data. KNN is a simple yet powerful algorithm that is widely used for its intuitive concept and easy implementation.

The main idea behind KNN is to classify a new data point based on the majority vote of its neighbors. The algorithm assumes that similar data points tend to belong to the same class or have similar numerical values. The "K" in KNN refers to the number of nearest neighbors that are considered for classification or regression.

In this model, we used K-fold cross validation. Cross-validation is a valuable technique in statistics and machine learning that helps assess the performance of a model and select optimal hyperparameters. Specifically, the K-fold cross-validation method is commonly employed for this purpose.

During K-fold cross-validation, the training set is divided into K equally sized subsets or "folds." The model is then trained K times, each time using K-1 of the folds as the training data and leaving one fold as the validation set. This process is repeated K times, ensuring that each fold serves as the validation set exactly once.

We can mention two benefits of k-fold cross validation. First, it allows us to leverage the entire training dataset for both training and validation. By iteratively rotating the folds, every data point gets an opportunity to be in the validation set, providing a more comprehensive evaluation of the model's performance.

Secondly, K-fold cross-validation helps to mitigate the potential bias introduced by using a single validation set. When we reserve a separate validation set, there is a risk that the performance estimation becomes overly influenced by the specific data points in that set. By repeatedly shuffling and partitioning the data into different folds, we obtain a more robust estimate of the model's performance, as it is evaluated on multiple distinct subsets of the data.

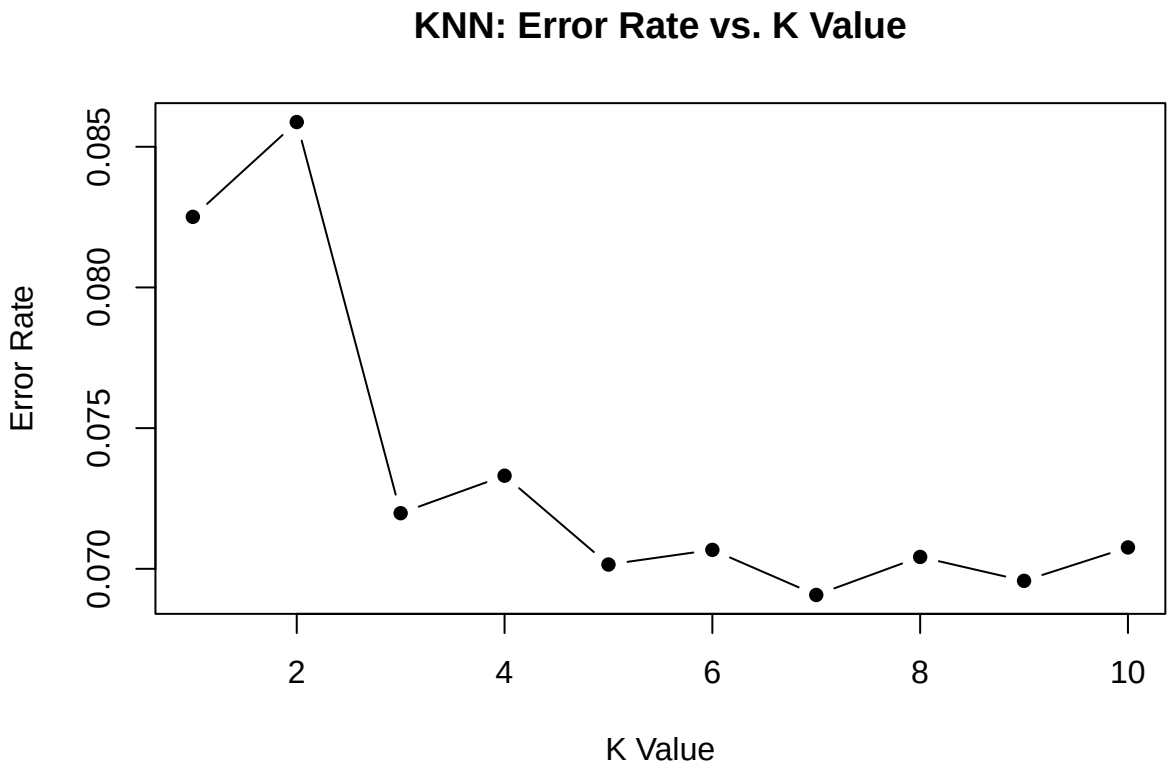
The final evaluation of the model is typically based on the average performance across all K iterations. This averaging process helps to reduce the variance in the performance estimate and provides a more stable measure of the model's effectiveness.

We need to scale our data for two reasons. Firstly, scaling increases the speed of training process. Secondly, if we do not scale the data, the features with higher value will have more impact on predictions, however, the features should have same weight and the change only should be in range of 0-1.

```
# Function for feature scaling
min_max_norm <- function(x) {
(x - min(x)) / (max(x) - min(x))
}
# We normalize the columns
train_scaled <- as.data.frame(lapply(train_data, min_max_norm))
test_scaled<- as.data.frame(lapply(test_data, min_max_norm))
```

```
# KNN with K-fold cross validation

# Define a range of K values
k_values <- 1:10

# Perform cross-validation and calculate error rates
error_rates <- sapply(k_values, function(k) {
  set.seed(123)  # For reproducibility
  model <- train(as.factor(satisfaction)~., data = train_scaled,
                 method = "knn",
                 trControl = trainControl(method = "cv", number = 5),
                 tuneGrid = data.frame(k = k))
  1 - model$results$Accuracy
})

# Plot the Error Rates
```

```r
plot(k_values, error_rates, type = "b", pch = 16, xlab = "K Value", ylab = "Error Rate",
     main = "KNN: Error Rate vs. K Value")
```




```r
# find the k giving minimum error rate
k_min <- which.min(error_rates)
k_min
```

```
## [1] 7
```

```r
# make predictions with k = k_min
pred_knn<- knn(train_scaled[,-23], test_scaled[,-23],
               cl = train_scaled$satisfaction,
               k = k_min)
```

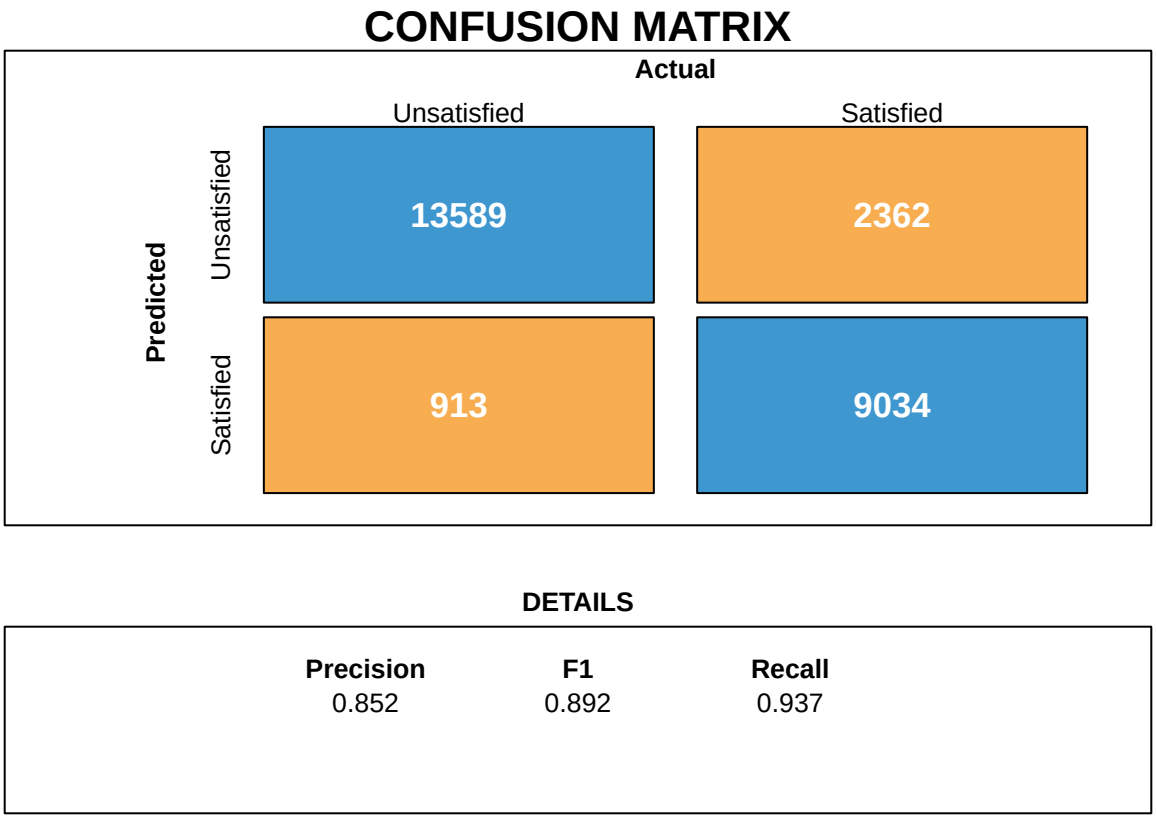
Classification Results

Confusion Matrix and Metrics

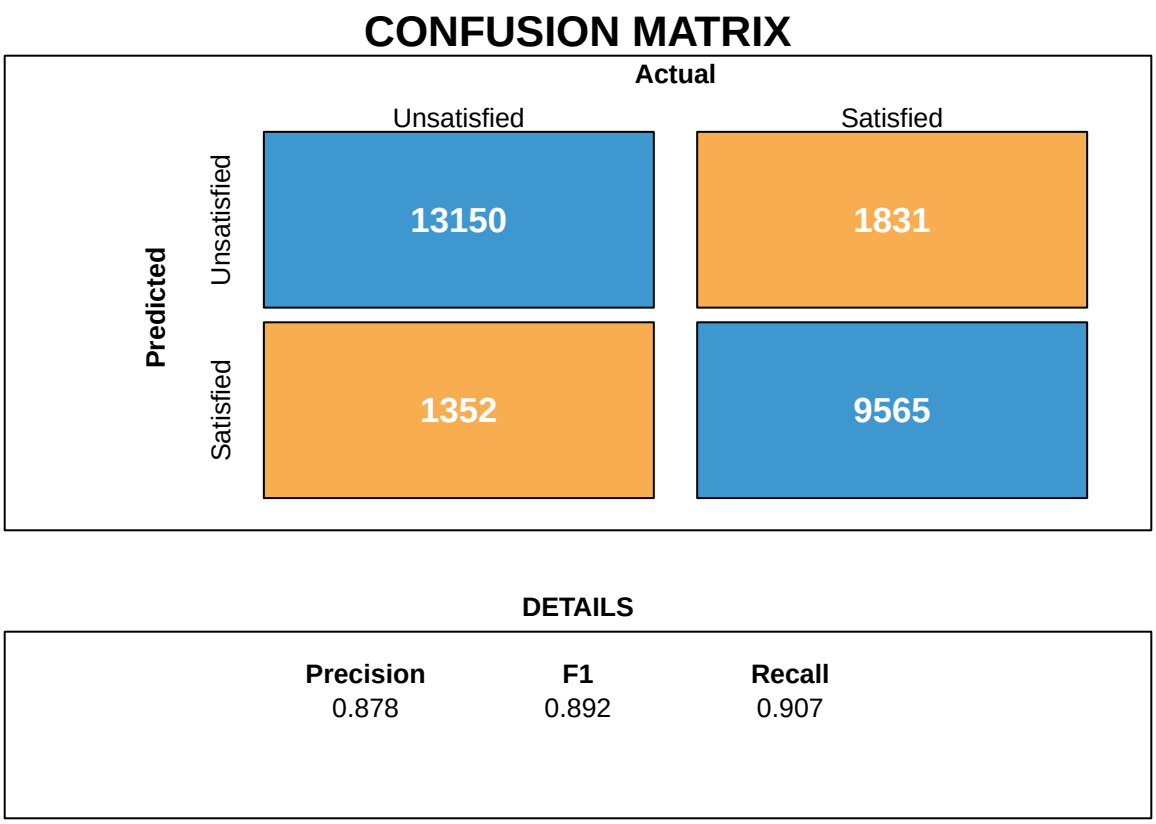
```r
# Create a function for confusion matrix and other metrics
draw_confusion_matrix <- function(cm) {

  layout(matrix(c(1,1,2)))
  par(mar=c(2,2,2,2))
  plot(c(100, 345), c(300, 450), type = "n", xlab="", ylab="", xaxt='n', yaxt='n')
  title('CONFUSION MATRIX', cex.main=2)
```

```
  # create the matrix
  rect(150, 430, 240, 370, col='#3F97D0')
  text(195, 435, 'Unsatisfied', cex=1.2)
  rect(250, 430, 340, 370, col='#F7AD50')
  text(295, 435, 'Satisfied', cex=1.2)
  text(125, 370, 'Predicted', cex=1.3, srt=90, font=2)
  text(245, 450, 'Actual', cex=1.3, font=2)
  rect(150, 305, 240, 365, col='#F7AD50')
  rect(250, 305, 340, 365, col='#3F97D0')
  text(140, 400, 'Unsatisfied', cex=1.2, srt=90)
  text(140, 335, 'Satisfied', cex=1.2, srt=90)

  # add in the cm results
  res <- as.numeric(cm$table)
  text(195, 400, res[1], cex=1.6, font=2, col='white')
  text(195, 335, res[2], cex=1.6, font=2, col='white')
  text(295, 400, res[3], cex=1.6, font=2, col='white')
  text(295, 335, res[4], cex=1.6, font=2, col='white')

  # add in the specifics
  plot(c(100, 0), c(50, 0), type = "n", xlab="", ylab="", main = "DETAILS", xaxt='n', yaxt='n')
  text(30, 40, names(cm$byClass[5]), cex=1.2, font=2)
  text(30, 30, round(as.numeric(cm$byClass[5]), 3), cex=1.2)
  text(50, 40, names(cm$byClass[7]), cex=1.2, font=2)
  text(50, 30, round(as.numeric(cm$byClass[7]), 3), cex=1.2)
  text(70, 40, names(cm$byClass[6]), cex=1.2, font=2)
  text(70, 30, round(as.numeric(cm$byClass[6]), 3), cex=1.2)


}
```

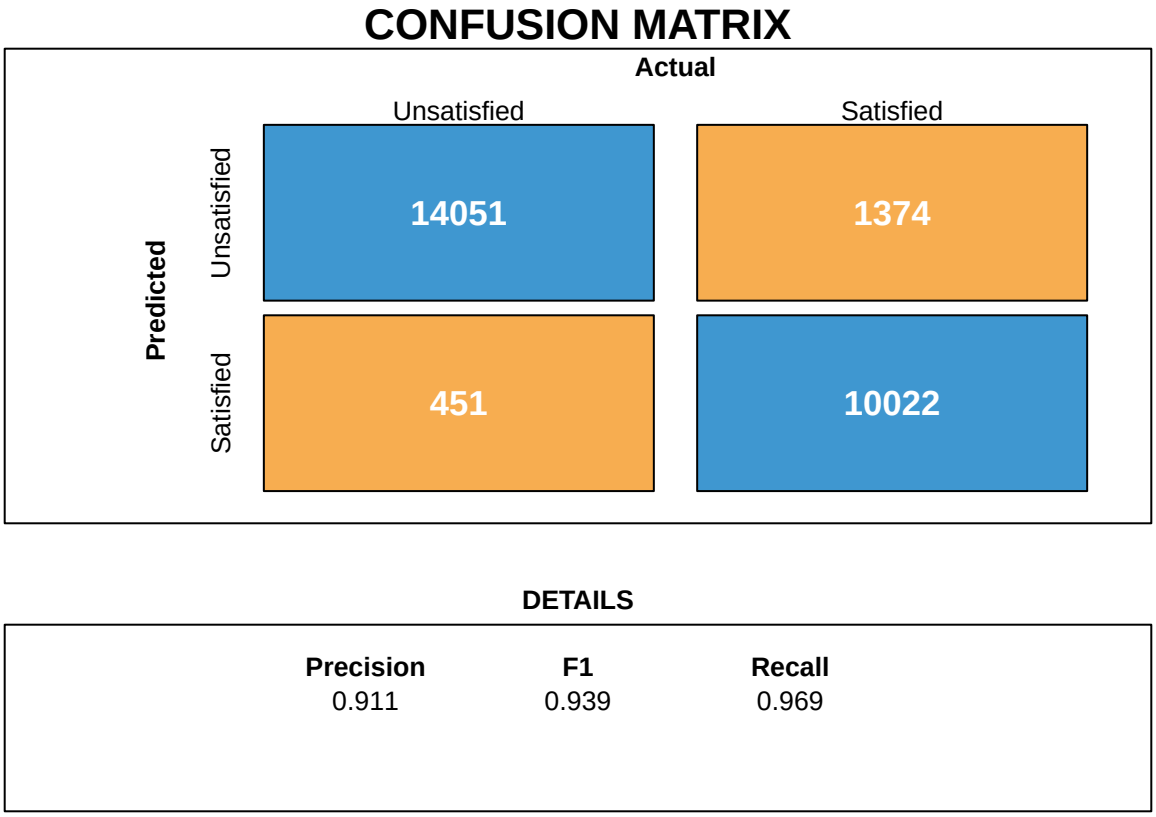
```
# Confusion matrix for glm_full_model

# Use best threshold for prediction (0 or 1)
Threshold <- 0.6
pred_glm_full_factor <- as.factor(ifelse(pred_glm_full >= Threshold , 1,0))


conf_matrix_glm <- confusionMatrix(data = pred_glm_full_factor,
                                   reference = as.factor(y_test))
draw_confusion_matrix(conf_matrix_glm)
```




```
# Confusion matrix for Naive Bayes
conf_matrix_naive <- confusionMatrix(data = pred_naive_bayes,
                                     reference = as.factor(y_test))
draw_confusion_matrix(conf_matrix_naive)
```




```r
# Confusion matrix for KNN
conf_matrix_knn <- confusionMatrix(data = pred_knn, reference = as.factor(y_test))
draw_confusion_matrix(conf_matrix_knn)
```



Conclusion

In this customer satisfaction prediction project, we aimed to develop a machine learning model that could accurately identify unsatisfied customers. We explored three different classification algorithms: Logistic Regression with different variable selection techniques, Naive Bayes, and K-Nearest Neighbors (KNN), to compare their performance.

Model Evaluation

After extensive evaluation using various metrics, including accuracy, F1 score, precision, and recall, we made the following observations:

1. Logistic Regression with the full features model demonstrated the best generalizability among all the tested models. Despite the expectation that backward variable selection and lasso regression might improve performance, the full features model outperformed them on the test set.
2. Naive Bayes showed promising results with a respectable accuracy score. However, it had slightly lower precision and F1 score compared to the best-performing model, Logistic Regression with the full features model.
3. K-Nearest Neighbors (KNN) achieved the highest precision score, making it a compelling choice for identifying unsatisfied customers with high accuracy. However, it is essential to consider the potential increase in model complexity with a growing number of samples, which can impact training time.

Considering the main objective of this project to maximize precision, we select the best-performing model: Logistic Regression with the full features model. This model demonstrated superior performance in terms of

precision, recall, and F1 score. Moreover, its generalizability suggests that it can be effectively applied to new and unseen data.