T00711122_ML_FINAL

2023-04-14

1.IMPORTING NECESSARY LIBRARIES

```
library(ggplot2)      #Importing necessary libraries for my classification methods and related procedur
library(tidyverse)
library(class)
library(caret)
library(e1071)
library(dplyr)
library(rpart)
library(rpart.plot)
library(randomForest)
```

2.LOADING AIRLINE DATASET

```
airline <- read.csv("/Users/keerthanasenthilkumar/Downloads/10K_Airline Passenger Satisfaction - V4.csv
```

(2.1) CHECKING THE COLUMN NAMES IN THE DATASET

```
names(airline)
```

```
##  [1] "Gender"
##  [2] "Age"
##  [3] "Customer.Type"
##  [4] "Type.of.Travel"
##  [5] "Class"
##  [6] "Flight.Distance"
##  [7] "Departure.Delay.in.Minutes"
##  [8] "Arrival.Delay.in.Minutes"
##  [9] "Departure.and.Arrival.Time.Convenience"
## [10] "Ease.of.Online.Booking"
## [11] "Check.in.Service"
## [12] "Online.Boarding"
## [13] "Gate.Location"
## [14] "On.board.Service"
## [15] "Seat.Comfort"
## [16] "Leg.Room.Service"
## [17] "Cleanliness"
## [18] "Food.and.Drink"
## [19] "In.flight.Service"
## [20] "In.flight.Wifi.Service"
## [21] "In.flight.Entertainment"
## [22] "Baggage.Handling"
## [23] "Satisfaction.Rating"
```

```r
#To know the column names of the airline data set.
```

(2.2) SUMMARY OF THE DATASET

```r
summary(airline)
```

```
##     Gender               Age        Customer.Type      Type.of.Travel    
##  Length:9342        Min.   : 7.00   Length:9342        Length:9342       
##  Class :character   1st Qu.:28.00   Class :character   Class :character  
##  Mode  :character   Median :41.00   Mode  :character   Mode  :character  
##                     Mean   :40.47                                        
##                     3rd Qu.:52.00                                        
##                     Max.   :85.00                                        
##     Class           Flight.Distance  Departure.Delay.in.Minutes
##  Length:9342        Min.   :  67.0   Min.   :   0.0            
##  Class :character   1st Qu.: 309.0   1st Qu.:   0.0            
##  Mode  :character   Median : 587.5   Median :   0.0            
##                     Mean   :1082.6   Mean   :  15.6            
##                     3rd Qu.:1633.0   3rd Qu.:  12.0            
##                     Max.   :3997.0   Max.   :1017.0            
##  Arrival.Delay.in.Minutes Departure.and.Arrival.Time.Convenience
##  Min.   :   0.0           Min.   :0.000                         
##  1st Qu.:   0.0           1st Qu.:2.000                         
##  Median :   0.0           Median :3.000                         
##  Mean   :  18.5           Mean   :3.088                         
##  3rd Qu.:  18.0           3rd Qu.:4.000                         
##  Max.   :1011.0           Max.   :5.000                         
##  Ease.of.Online.Booking Check.in.Service Online.Boarding Gate.Location
##  Min.   :1.000          Min.   :1.000    Min.   :1.000   Min.   :1.00 
##  1st Qu.:2.000          1st Qu.:3.000    1st Qu.:3.000   1st Qu.:2.00 
##  Median :3.000          Median :4.000    Median :4.000   Median :3.00 
##  Mean   :2.906          Mean   :3.397    Mean   :3.424   Mean   :2.98 
##  3rd Qu.:4.000          3rd Qu.:4.000    3rd Qu.:4.000   3rd Qu.:4.00 
##  Max.   :5.000          Max.   :5.000    Max.   :5.000   Max.   :5.00 
##  On.board.Service  Seat.Comfort   Leg.Room.Service  Cleanliness  
##  Min.   :1.000    Min.   :1.000   Min.   :1.00     Min.   :1.00  
##  1st Qu.:3.000    1st Qu.:3.000   1st Qu.:2.00     1st Qu.:2.00  
##  Median :4.000    Median :4.000   Median :4.00     Median :3.00  
##  Mean   :3.471    Mean   :3.524   Mean   :3.45     Mean   :3.33  
##  3rd Qu.:4.000    3rd Qu.:5.000   3rd Qu.:5.00     3rd Qu.:4.00  
##  Max.   :5.000    Max.   :5.000   Max.   :5.00     Max.   :5.00  
##  Food.and.Drink  In.flight.Service In.flight.Wifi.Service
##  Min.   :1.000   Min.   :1.000     Min.   :1.000         
##  1st Qu.:2.000   1st Qu.:3.000     1st Qu.:2.000         
##  Median :3.000   Median :4.000     Median :3.000         
##  Mean   :3.205   Mean   :3.748     Mean   :2.771         
##  3rd Qu.:4.000   3rd Qu.:5.000     3rd Qu.:4.000         
##  Max.   :5.000   Max.   :5.000     Max.   :5.000         
##  In.flight.Entertainment Baggage.Handling Satisfaction.Rating
##  Min.   :1.00            Min.   :1.000    Length:9342        
##  1st Qu.:2.00            1st Qu.:3.000    Class :character   
##  Median :4.00            Median :4.000    Mode  :character   
##  Mean   :3.39            Mean   :3.758
```

```
##   3rd Qu.:5.00            3rd Qu.:5.000
##   Max.   :5.00            Max.   :5.000
```

```
#This function displays the minimum, maximum, median, mean, and quartiles for each continuous variable.
```

(2.3) STRUCTURE OF THE DATASET

```
str(airline)
```

```
## 'data.frame':    9342 obs. of  23 variables:
##  $ Gender                                : chr  "Male" "Female" "Male" "Male" ...
##  $ Age                                   : int  48 35 41 50 49 43 43 60 50 38 ...
##  $ Customer.Type                         : chr  "First-time" "Returning" "Returning" "Returning" ...
##  $ Type.of.Travel                        : chr  "Business" "Business" "Business" "Business" ...
##  $ Class                                 : chr  "Business" "Business" "Business" "Business" ...
##  $ Flight.Distance                       : int  821 821 853 1905 3470 3788 1963 853 2607 2822 ...
##  $ Departure.Delay.in.Minutes            : int  2 26 0 0 0 0 0 0 0 13 ...
##  $ Arrival.Delay.in.Minutes              : int  5 39 0 0 1 0 0 3 0 0 ...
##  $ Departure.and.Arrival.Time.Convenience: int  3 2 4 2 3 4 3 3 1 2 ...
##  $ Ease.of.Online.Booking                : int  3 2 4 2 3 4 3 4 1 5 ...
##  $ Check.in.Service                      : int  4 3 4 3 3 3 4 3 3 3 ...
##  $ Online.Boarding                       : int  3 5 5 4 5 5 4 4 2 5 ...
##  $ Gate.Location                         : int  3 2 4 2 3 4 3 4 1 2 ...
##  $ On.board.Service                      : int  3 5 3 5 3 4 5 3 4 5 ...
##  $ Seat.Comfort                          : int  5 4 5 5 4 4 5 4 3 4 ...
##  $ Leg.Room.Service                      : int  2 5 3 5 4 4 5 4 4 5 ...
##  $ Cleanliness                           : int  5 5 5 4 5 3 4 4 3 4 ...
##  $ Food.and.Drink                        : int  5 3 5 4 4 3 5 4 3 2 ...
##  $ In.flight.Service                     : int  5 5 3 5 3 4 5 3 4 5 ...
##  $ In.flight.Wifi.Service                : int  3 2 4 2 3 4 3 4 4 2 ...
##  $ In.flight.Entertainment               : int  5 5 3 5 3 4 5 3 4 5 ...
##  $ Baggage.Handling                      : int  5 5 3 5 3 4 5 3 4 5 ...
##  $ Satisfaction.Rating                   : chr  "Satisfied" "Satisfied" "Satisfied" "Satisfied" ...
```

```
#It displays the number of observations and variables in the dataset,names of the variables and the dat
```

(2.4) CHECKING THE NULL VALUES

```
sum(is.na(airline))
```

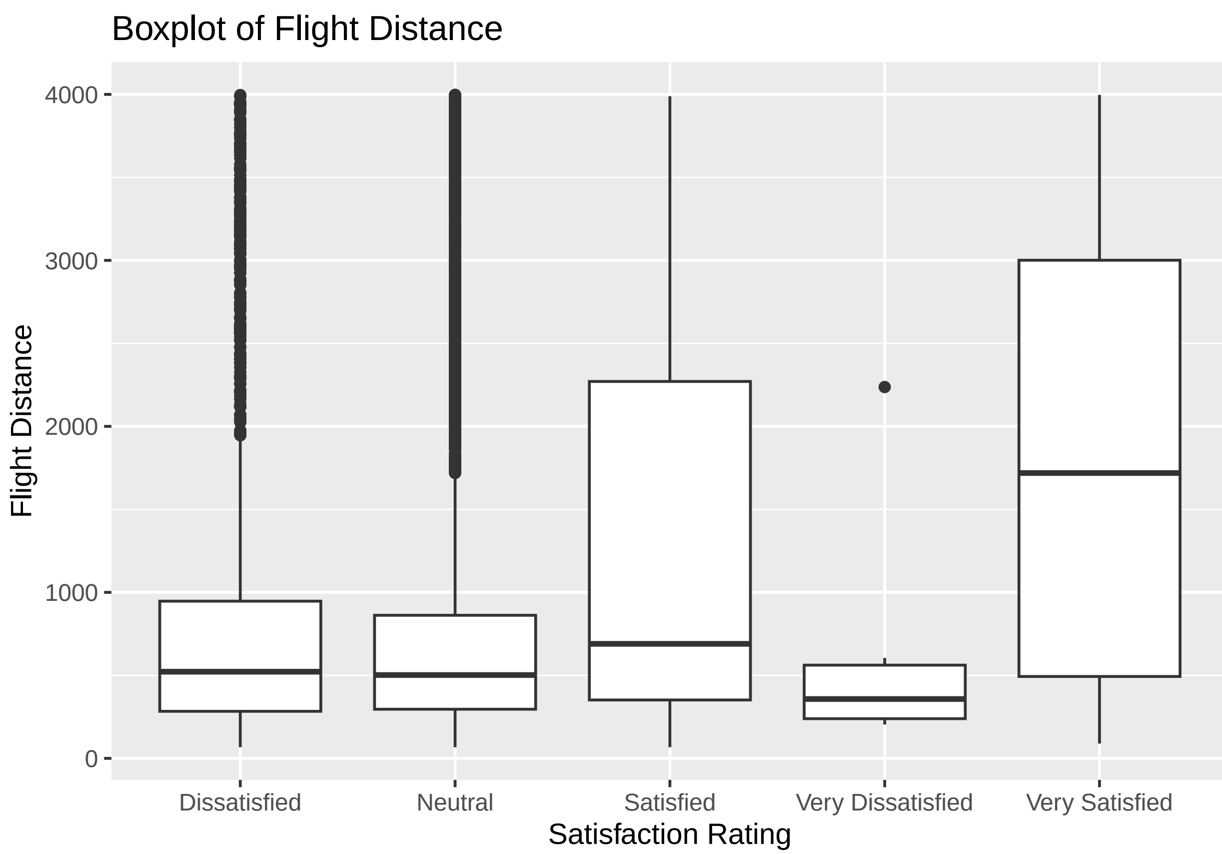
```
## [1] 0
```

```
#To check whether airline dataset have any missing or null values.
```

3.EXPLORATORY DATA ANALYSIS

(3.1) BOXPLOT FOR CHECKING THE OUTLIERS IN CONTINUOUS VARIABLES

```
ggplot(airline, aes(x = Satisfaction.Rating  , y = Flight.Distance)) +
  geom_boxplot() +
  labs(y = "Flight Distance", x = "Satisfaction Rating" ) +
  ggtitle("Boxplot of Flight Distance")
```



(3.2) REMOVE THE OUTLIERS

```
identify_outliers <- function(x) {
  q1 <- quantile(x, 0.25)
  q3 <- quantile(x, 0.75)
  iqr <- q3 - q1
  upper_fence <- q3 + 1.5*iqr
  lower_fence <- q1 - 1.5*iqr
  outlier_indices <- which(x < lower_fence | x > upper_fence)
  return(outlier_indices)
}

outliers <- identify_outliers(airline$Flight.Distance)
if (length(outliers) > 0) {
  cat("Outliers identified in Flight Distance. \n")

# Remove the outliers:
  airline <- airline[!airline$Flight.Distance %in% outliers,]
  cat("Outliers removed from the dataset.\n")
} else {
  cat("No outliers identified in Flight Distance.\n")
}
```

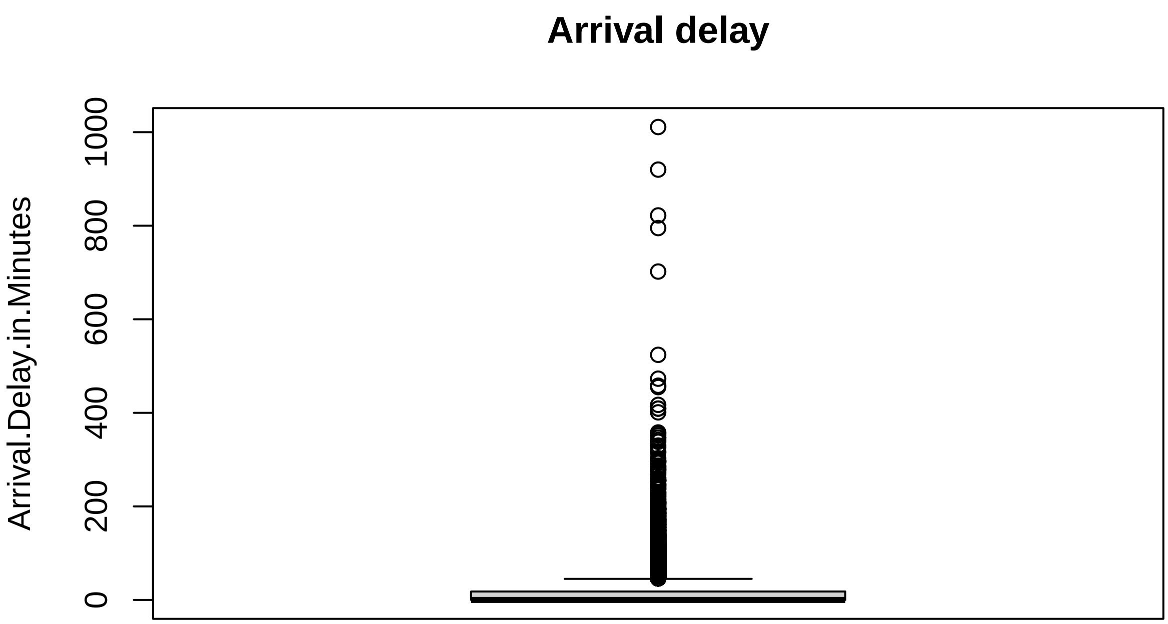
```
## Outliers identified in Flight Distance.
## Outliers removed from the dataset.
```

SCALING

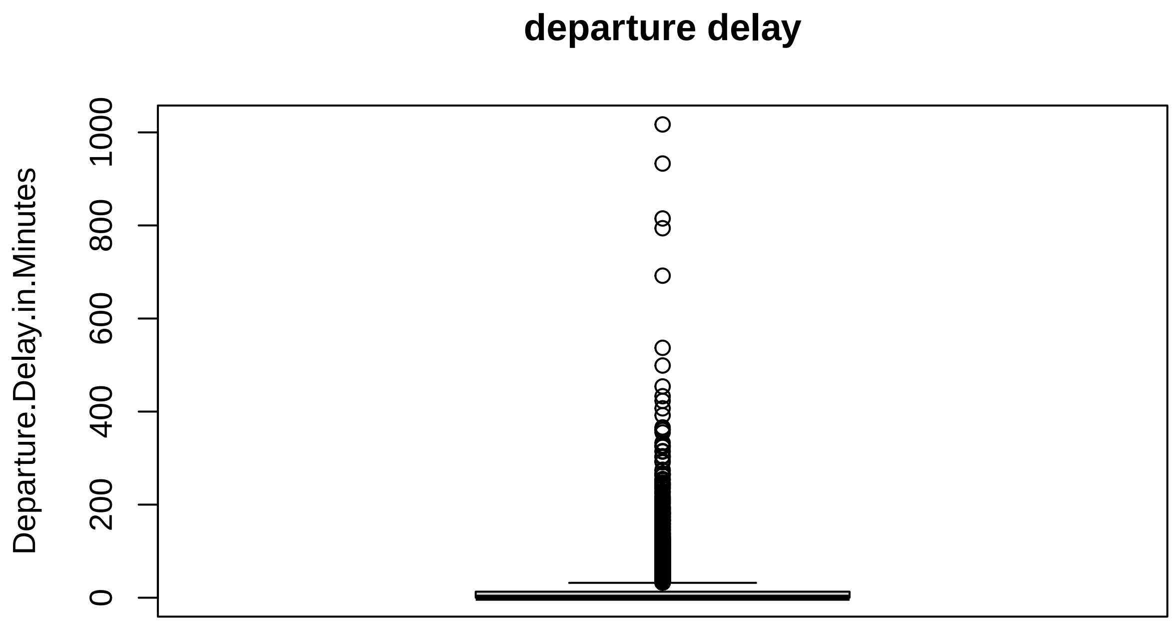
```
#scaling the continuous variable:
airline$Flight.Distance <- scale(airline$Flight.Distance)
```

(3.3) BOXPLOT FOR OTHER CONTINUOUS VARIABLES

```
boxplot(airline$Arrival.Delay.in.Minutes,main="Arrival delay",ylab="Arrival.Delay.in.Minutes")
```




```
boxplot(airline$Departure.Delay.in.Minutes,main="departure delay",ylab="Departure.Delay.in.Minutes")
```



(3.4) REMOVING OUTLIERS

```
# Identify and remove outliers:
identify_outliers <- function(x) {
  q1 <- quantile(x, 0.25)
  q3 <- quantile(x, 0.75)
  iqr <- q3 - q1
  upper_fence <- q3 + 1.5*iqr
  lower_fence <- q1 - 1.5*iqr
  outlier_indices <- which(x < lower_fence | x > upper_fence)
  return(outlier_indices)
}

outliers <- identify_outliers(airline$Arrival.Delay.in.Minutes)
if (length(outliers) > 0) {
  cat("Outliers identified in Arrival delay. \n")

  # Remove the outliers from the dataset
  airline <- airline[!airline$Arrival.Delay.in.Minutes %in% outliers,]
  cat("Outliers removed from the dataset.\n")
} else {
  cat("No outliers identified in Arrival delay.\n")
}
```

```
## Outliers identified in Arrival delay.
## Outliers removed from the dataset.
```

```
outliers <- identify_outliers(airline$Departure.Delay.in.Minutes)
if (length(outliers) > 0) {
  cat("Outliers identified in departure delay. \n")

# Remove the outliers :
  airline <- airline[!airline$Departure.Delay.in.Minutes %in% outliers,]
  cat("Outliers removed from the dataset.\n")
} else {
  cat("No outliers identified in departure delay.\n")
}
```

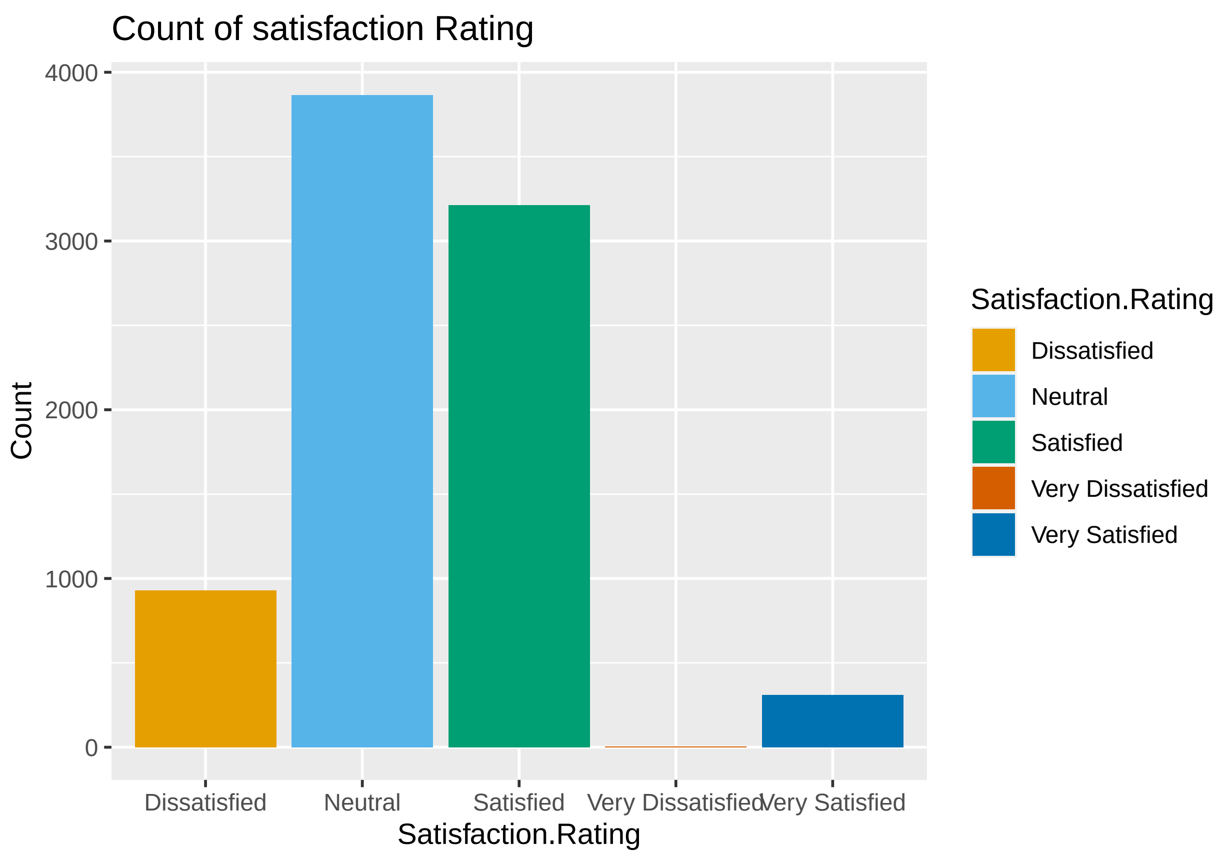
```
## Outliers identified in departure delay.
## Outliers removed from the dataset.
```

SCALING

```
airline$Arrival.Delay.in.Minutes <- scale(airline$Arrival.Delay.in.Minutes)
airline$Departure.Delay.in.Minutes <-scale(airline$Departure.Delay.in.Minutes)
```

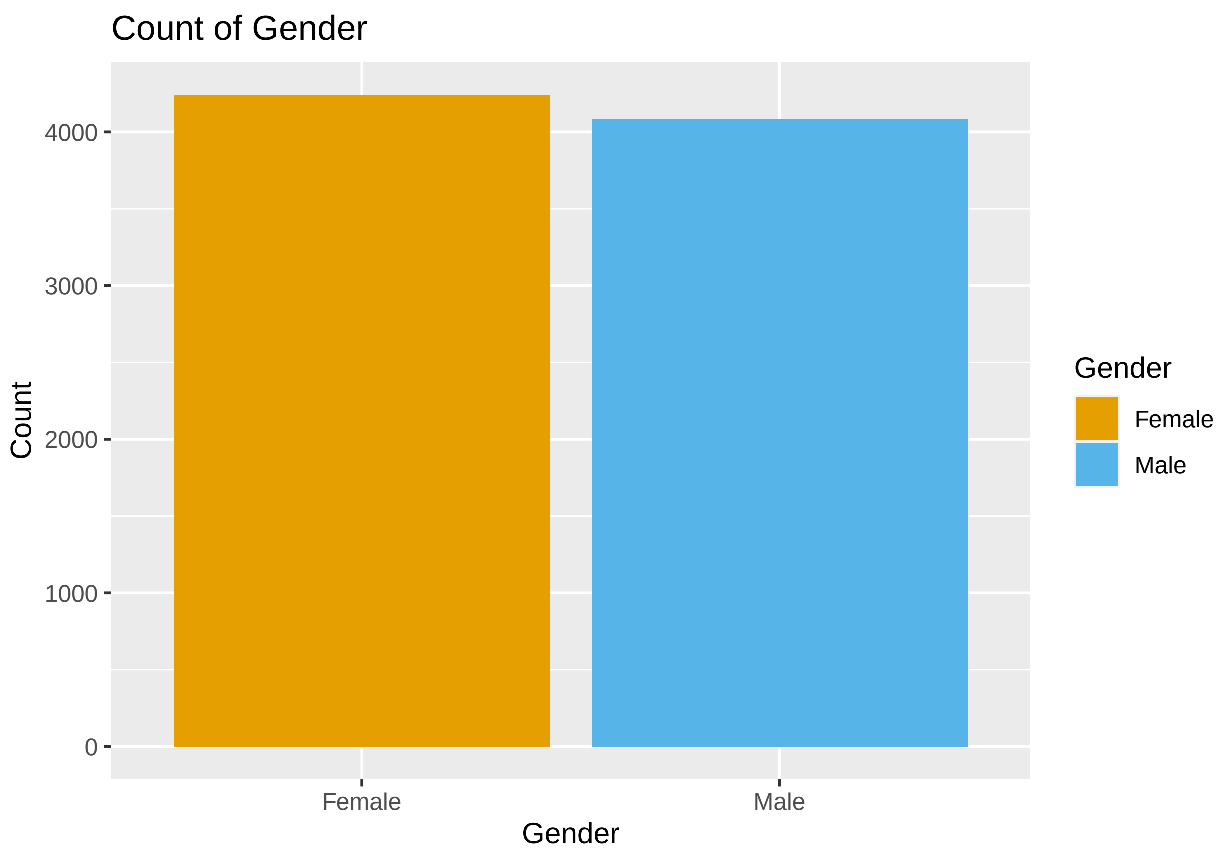
(3.5) DISTRIBUTION PLOT FOR MY TARGET VARIABLE(Satisfaction Rating):

```
colors <- c("#E69F00", "#56B4E9", "#009E73", "#D55E00", "#0072B2")

ggplot(airline, aes(x = Satisfaction.Rating, fill = Satisfaction.Rating)) +
  geom_bar() +
  scale_fill_manual(values = colors) +
  labs(title = "Count of satisfaction Rating", x = "Satisfaction.Rating", y = "Count")
```

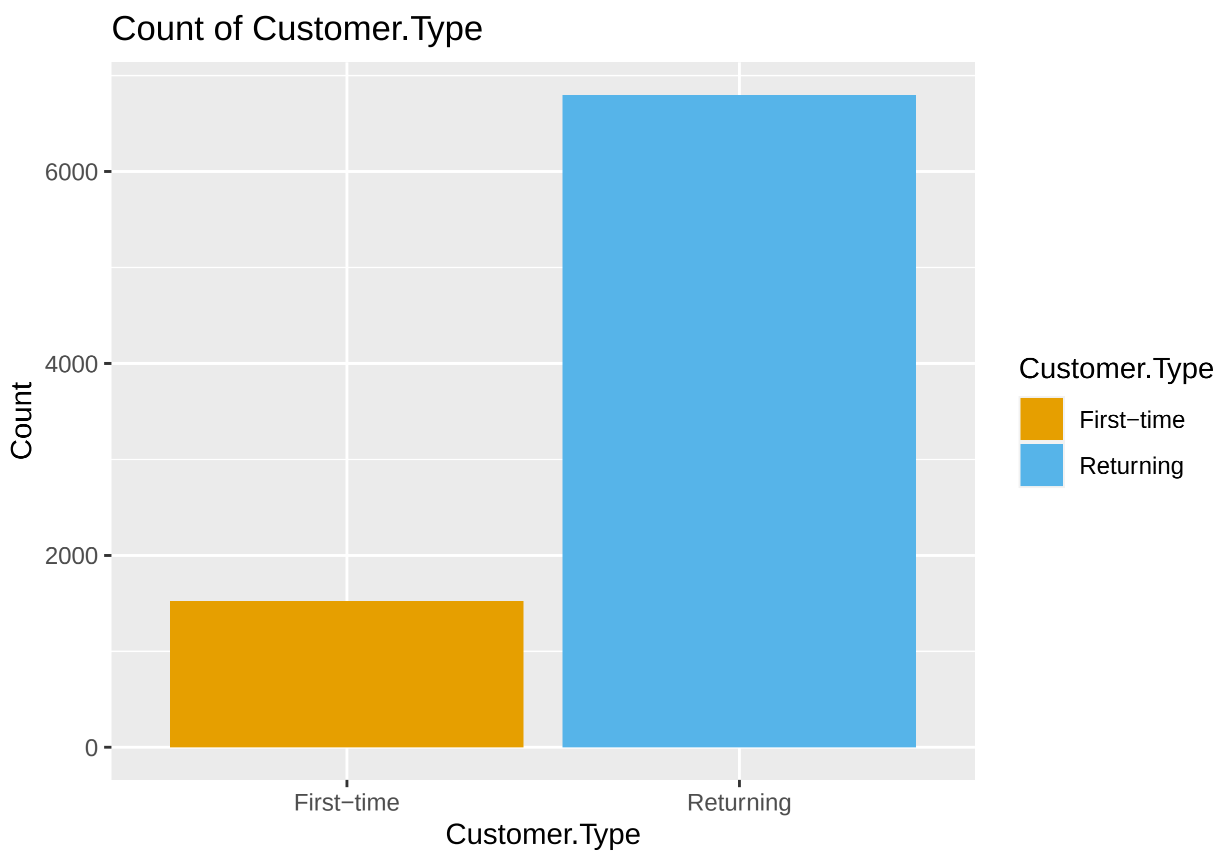


(3.6) BARCHART FOR OTHER CATEGORICAL VARIABLES (Gender,Customer type,Type of travel,class)

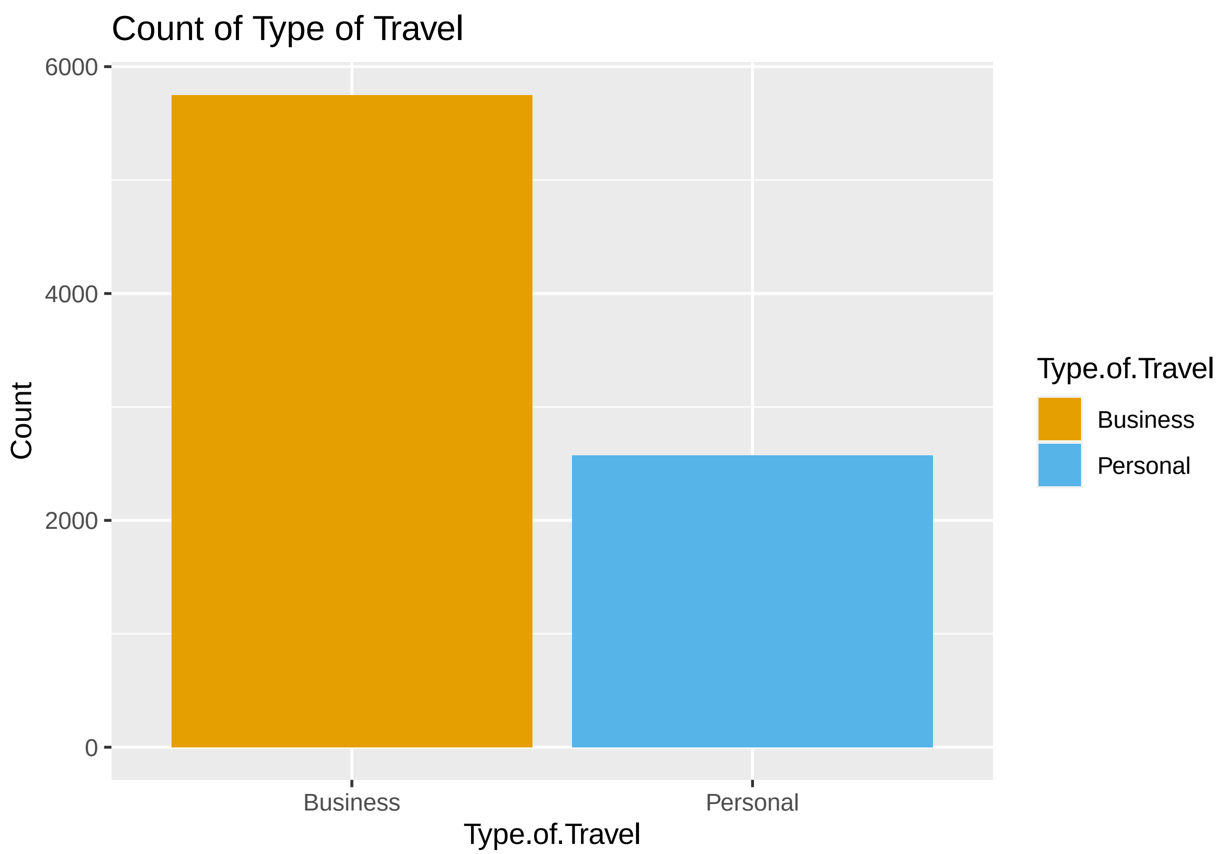
```
#setting the colors for each group:
colors <- c("#E69F00", "#56B4E9", "#009E73", "#D55E00", "#0072B2")

#creating bar chart for gender variable:
ggplot(airline, aes(x = Gender, fill = Gender)) +
  geom_bar() +
  scale_fill_manual(values = colors) +
  labs(title = "Count of Gender", x = "Gender", y = "Count")
```



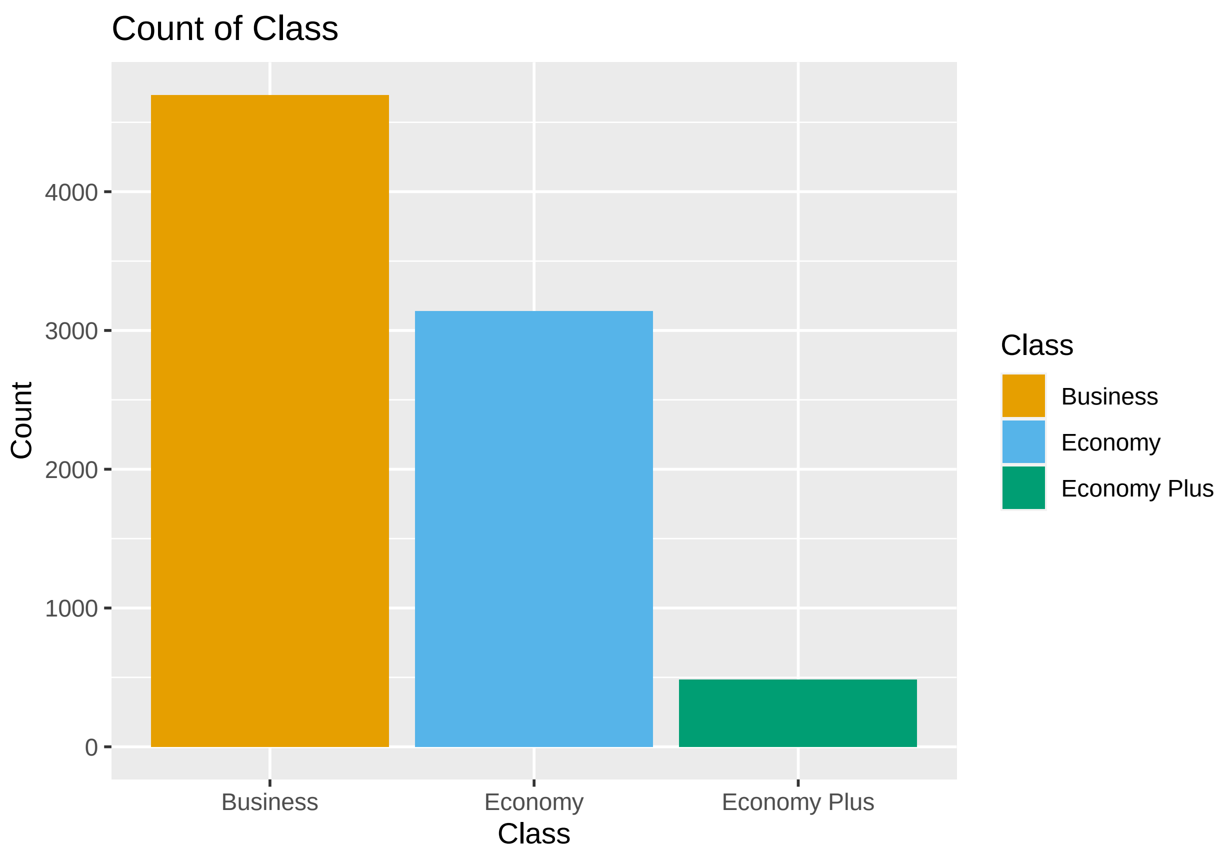

```
#creating bar chart for variable customer type:
ggplot(airline, aes(x = Customer.Type , fill = Customer.Type)) +
  geom_bar() +
  scale_fill_manual(values = colors) +
  labs(title = "Count of Customer.Type", x = "Customer.Type", y = "Count")
```




```r
#creating bar chart for variable travel type:
ggplot(airline, aes(x =Type.of.Travel , fill = Type.of.Travel)) +
  geom_bar() +
  scale_fill_manual(values = colors) +
  labs(title = "Count of Type of Travel", x = "Type.of.Travel", y = "Count")
```




```
#creating bar chart for variable class:
ggplot(airline, aes(x = Class, fill = Class)) +
  geom_bar() +
  scale_fill_manual(values = colors) +
  labs(title = "Count of Class", x = "Class", y = "Count")
```



4.CLASSIFICATION MODELS

```
set.seed(123)
#It ensures that the random numbers generated in subsequent commands are reproducible.

#converting categorical variable to factor:
airline$Satisfaction.Rating <- as.factor(airline$Satisfaction.Rating)

#splitting dataset into train and test data set:
train_index <- createDataPartition(airline$Satisfaction.Rating, p = 0.6, list = FALSE)
train <- airline[train_index, ]
test <- airline[-train_index, ]
#The train data set contains 60% of the observations based on the Satisfaction.Rating variable from the
#it can be used to train the model, while the test data set contains the remaining 40%
#it will be used to evaluate the performance of the trained model.
```

(4.1) SUPPORT VECTOR MACHINE

```
# Training the SVM model:
svm_model <- svm(Satisfaction.Rating ~ ., data = train, kernel = "linear")

# Making predictions on test data:
svm_pred <- predict(svm_model, test)

# Calculating accuracy:
accuracy <- mean(svm_pred == test$Satisfaction.Rating)
```

```r
# Printing accuracy:
cat("Accuracy of the SVM model on test data:", round(accuracy * 100, 2), "%\n")
```

```
## Accuracy of the SVM model on test data: 99.97 %
```

STATISTICS OF MODEL

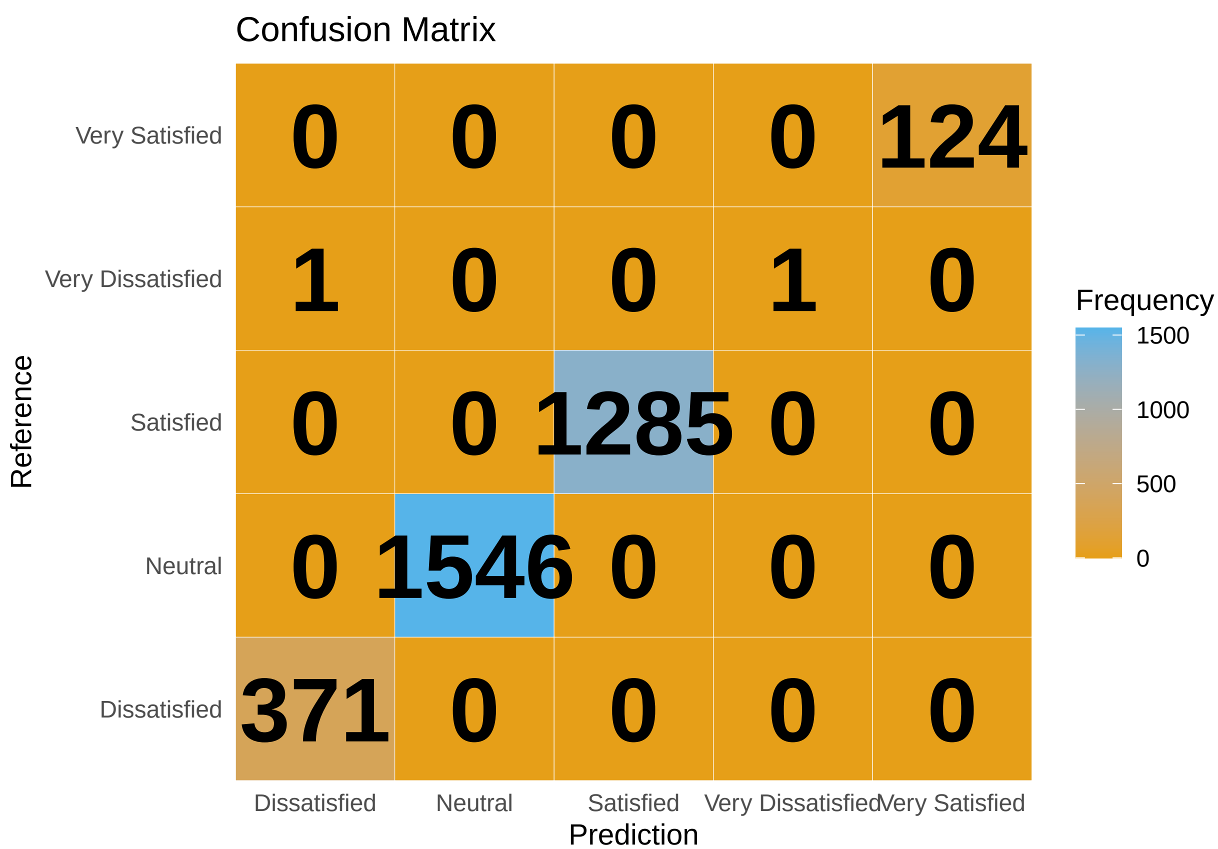
```r
# Create confusion matrix for the model:
confusionMatrix(svm_pred, test$Satisfaction.Rating)
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Dissatisfied Neutral Satisfied Very Dissatisfied
##   Dissatisfied               371       0         0                 1
##   Neutral                      0    1546         0                 0
##   Satisfied                    0       0      1285                 0
##   Very Dissatisfied            0       0         0                 1
##   Very Satisfied               0       0         0                 0
##                    Reference
## Prediction          Very Satisfied
##   Dissatisfied                   0
##   Neutral                        0
##   Satisfied                      0
##   Very Dissatisfied              0
##   Very Satisfied               124
##
## Overall Statistics
##
##                Accuracy : 0.9997
##                  95% CI : (0.9983, 1)
##     No Information Rate : 0.4645
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9995
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Dissatisfied Class: Neutral Class: Satisfied
## Sensitivity                       1.0000         1.0000           1.0000
## Specificity                       0.9997         1.0000           1.0000
## Pos Pred Value                    0.9973         1.0000           1.0000
## Neg Pred Value                    1.0000         1.0000           1.0000
## Prevalence                        0.1115         0.4645           0.3861
## Detection Rate                    0.1115         0.4645           0.3861
## Detection Prevalence              0.1118         0.4645           0.3861
## Balanced Accuracy                 0.9998         1.0000           1.0000
##                      Class: Very Dissatisfied Class: Very Satisfied
## Sensitivity                         0.5000000               1.00000
## Specificity                         1.0000000               1.00000
```

```
## Pos Pred Value                0.0000000           1.00000
## Neg Pred Value                0.9996994           1.00000
## Prevalence                    0.0006010           0.03726
## Detection Rate                0.0003005           0.03726
## Detection Prevalence          0.0003005           0.03726
## Balanced Accuracy             0.7500000           1.00000
```

CONFUSION MATRIX FOR SVM

```
cm <- confusionMatrix(svm_pred, test$Satisfaction.Rating)

# create the confusion matrix plot
cm_plot <- ggplot(data = as.data.frame(cm$table),
                  aes(x = Prediction, y = Reference, fill = as.numeric(Freq))) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "#E69F18", high = "#56B4E9") +
  theme_minimal() +
  labs(title = "Confusion Matrix", x = "Prediction", y = "Reference") +
  geom_text(aes(label = Freq), size = 12, fontface = "bold") +
  scale_x_discrete(expand = c(0, 0.1)) +
  scale_y_discrete(expand = c(0, 0.1))+
  guides(fill = guide_colorbar(title = "Frequency"))
cm_plot
```



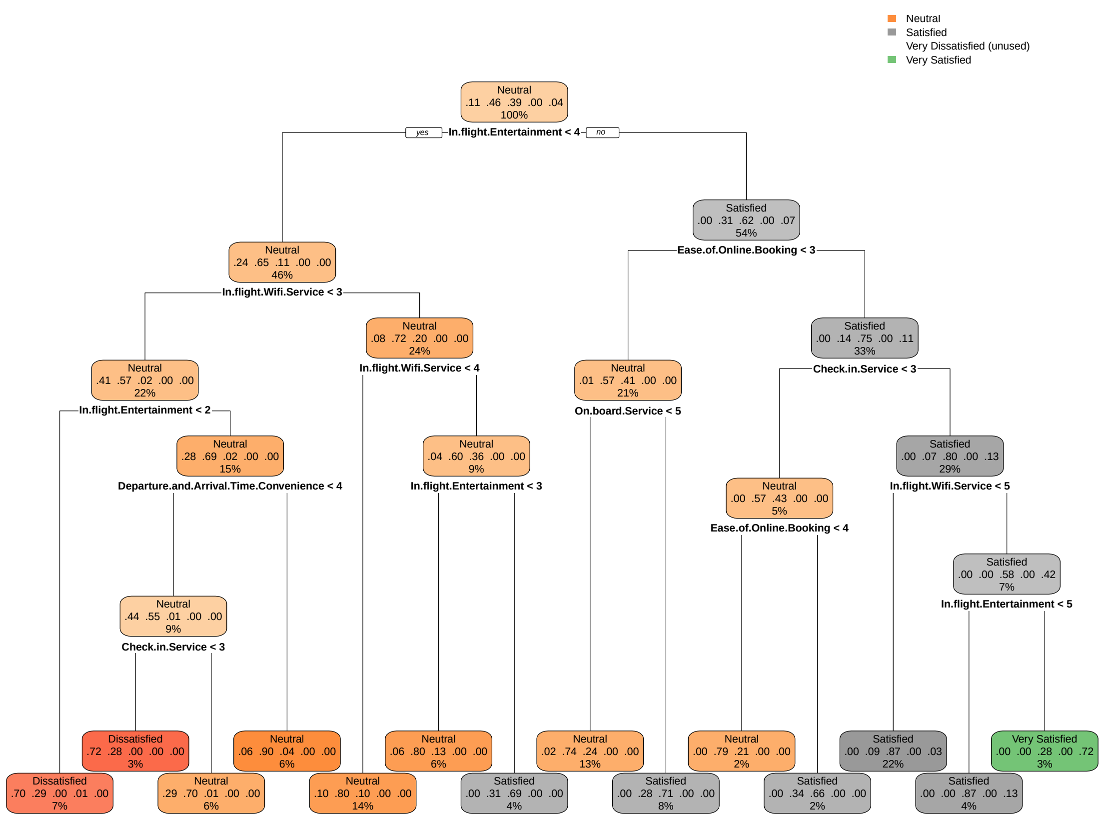
ACCURACY OF SVM MODEL ON TRAINING DATASET

```r
#Making prediction on train data set:
svm_pred <- predict(svm_model, train)

# Calculating accuracy:
accuracy <- mean(svm_pred == train$Satisfaction.Rating)

# Printing accuracy:
cat("Accuracy of the SVM model on train data:", round(accuracy * 100, 2), "%\n")
```

```
## Accuracy of the SVM model on train data: 99.98 %
```

(4.2) DECISION TREE

```r
# creating the decision tree:
d_model <- rpart(Satisfaction.Rating ~ ., data = train, method = "class")

# View the decision tree:
rpart.plot(d_model)
```




```r
# Making predictions on test data set:
d_pred <- predict(d_model, test, type = "class")

# Calculating accuracy:
accuracy <- mean(d_pred == test$Satisfaction.Rating)
```

```r
# Printing accuracy:
cat("Accuracy of the decision tree model on test data:", round(accuracy * 100, 2), "%\n")
```

```
## Accuracy of the decision tree model on test data: 77.55 %
```

STATISTICS OF MODEL

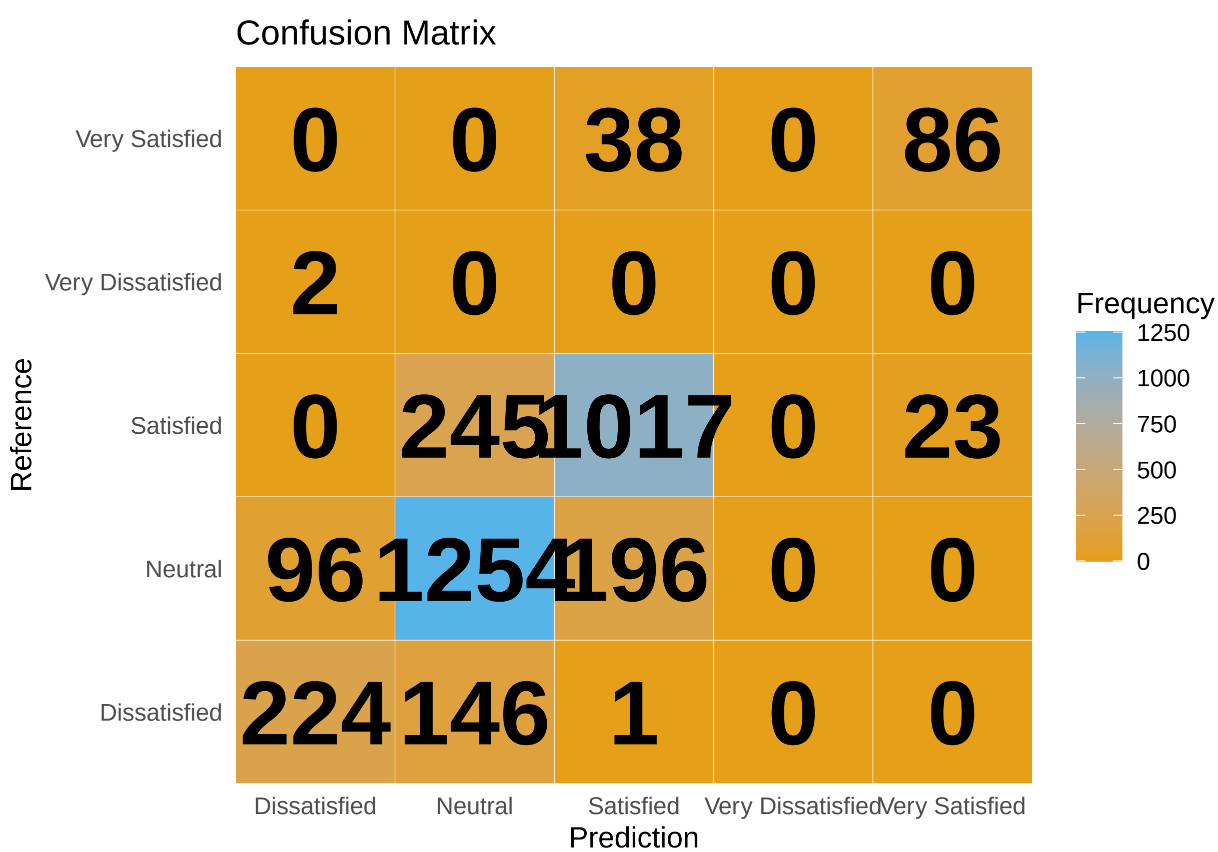
```r
# Evaluate the performance of model:
confusionMatrix(d_pred, test$Satisfaction.Rating)
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Dissatisfied Neutral Satisfied Very Dissatisfied
##   Dissatisfied               224      96         0                 2
##   Neutral                    146    1254       245                 0
##   Satisfied                    1     196      1017                 0
##   Very Dissatisfied            0       0         0                 0
##   Very Satisfied               0       0        23                 0
##                    Reference
## Prediction          Very Satisfied
##   Dissatisfied                   0
##   Neutral                        0
##   Satisfied                     38
##   Very Dissatisfied              0
##   Very Satisfied                86
##
## Overall Statistics
##
##                Accuracy : 0.7755
##                  95% CI : (0.761, 0.7896)
##     No Information Rate : 0.4645
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6339
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Dissatisfied Class: Neutral Class: Satisfied
## Sensitivity                      0.60377         0.8111           0.7914
## Specificity                      0.96686         0.7806           0.8850
## Pos Pred Value                   0.69565         0.7623           0.8123
## Neg Pred Value                   0.95110         0.8265           0.8709
## Prevalence                       0.11148         0.4645           0.3861
## Detection Rate                   0.06731         0.3768           0.3056
## Detection Prevalence             0.09675         0.4943           0.3762
## Balanced Accuracy                0.78532         0.7959           0.8382
##                      Class: Very Dissatisfied Class: Very Satisfied
## Sensitivity                          0.000000               0.69355
## Specificity                          1.000000               0.99282
```

```
## Pos Pred Value                         NaN          0.78899
## Neg Pred Value                    0.999399          0.98820
## Prevalence                        0.000601          0.03726
## Detection Rate                    0.000000          0.02584
## Detection Prevalence              0.000000          0.03275
## Balanced Accuracy                 0.500000          0.84318
```

CONFUSION MATRIX FOR DECISION TREE

```
cm <- confusionMatrix(d_pred, test$Satisfaction.Rating)

# create the confusion matrix plot

cm_plot <- ggplot(data = as.data.frame(cm$table),
                  aes(x = Prediction, y = Reference, fill = as.numeric(Freq))) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "#E69F18", high = "#56B4E9") +
  theme_minimal() +
  labs(title = "Confusion Matrix", x = "Prediction", y = "Reference") +
  geom_text(aes(label = Freq), size = 12, fontface = "bold") +
  scale_x_discrete(expand = c(0, 0.1)) +
  scale_y_discrete(expand = c(0, 0.1))+
  guides(fill = guide_colorbar(title = "Frequency"))
cm_plot
```



ACCURACY OF DECISION TREE MODEL ON TRAINING DATASET

```r
# Making predictions on train data set:
d_pred <- predict(d_model, train, type = "class")

# Calculating accuracy:
accuracy <- mean(d_pred == train$Satisfaction.Rating)

# Printing accuracy:
cat("Accuracy of the decision tree model on train data:", round(accuracy * 100, 2), "%\n")
```

```
## Accuracy of the decision tree model on train data: 78.44 %
```

(4.3) RANDOM FOREST

```r
# Build the random forest model:
rf_model <- randomForest(Satisfaction.Rating ~ ., data = train, ntree = 100)

# Evaluating the test performance of the model:
rf_pred <- predict(rf_model, test)

# Calculating accuracy:
accuracy <- mean(rf_pred == test$Satisfaction.Rating)

# Printing accuracy:
cat("Accuracy of the random forest model on test data:", round(accuracy * 100, 2), "%\n")
```

```
## Accuracy of the random forest model on test data: 90.38 %
```

STATISTICS OF MODEL

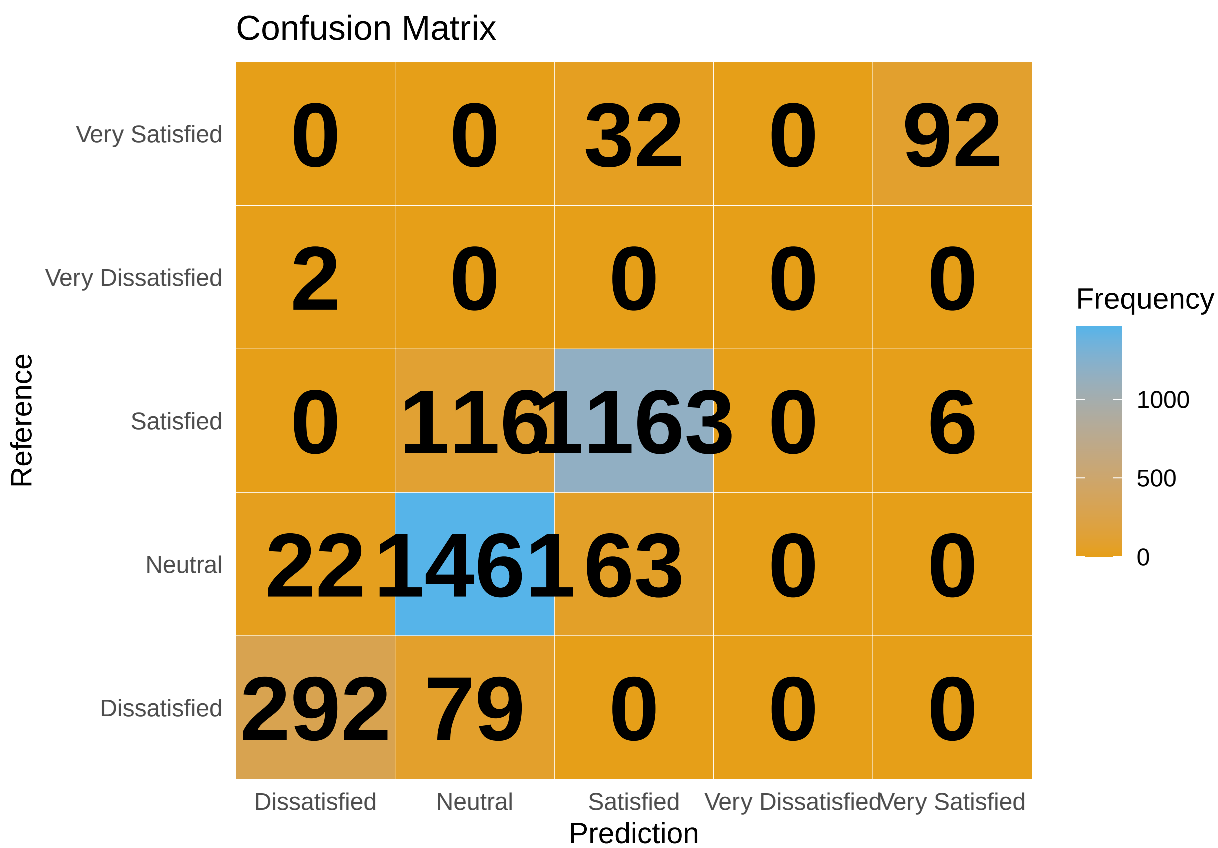
```r
# Evaluate the performance of the model
confusionMatrix(rf_pred, test$Satisfaction.Rating)
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction            Dissatisfied Neutral Satisfied Very Dissatisfied
##   Dissatisfied                 292      22         0                 2
##   Neutral                       79    1461       116                 0
##   Satisfied                      0      63      1163                 0
##   Very Dissatisfied              0       0         0                 0
##   Very Satisfied                 0       0         6                 0
##                     Reference
## Prediction            Very Satisfied
##   Dissatisfied                     0
##   Neutral                          0
##   Satisfied                       32
##   Very Dissatisfied                0
##   Very Satisfied                  92
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9038
##                  95% CI : (0.8933, 0.9137)
##     No Information Rate : 0.4645
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8427
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Dissatisfied Class: Neutral Class: Satisfied
## Sensitivity                      0.78706         0.9450           0.9051
## Specificity                      0.99188         0.8906           0.9535
## Pos Pred Value                   0.92405         0.8822           0.9245
## Neg Pred Value                   0.97377         0.9492           0.9411
## Prevalence                       0.11148         0.4645           0.3861
## Detection Rate                   0.08774         0.4390           0.3495
## Detection Prevalence             0.09495         0.4976           0.3780
## Balanced Accuracy                0.88947         0.9178           0.9293
##                      Class: Very Dissatisfied Class: Very Satisfied
## Sensitivity                          0.000000               0.74194
## Specificity                          1.000000               0.99813
## Pos Pred Value                            NaN               0.93878
## Neg Pred Value                       0.999399               0.99009
## Prevalence                           0.000601               0.03726
## Detection Rate                       0.000000               0.02764
## Detection Prevalence                 0.000000               0.02945
## Balanced Accuracy                    0.500000               0.87003
```

CONFUSION MATRIX FOR RANDOM FOREST

```r
cm <- confusionMatrix(rf_pred, test$Satisfaction.Rating)

# create the confusion matrix plot
cm_plot <- ggplot(data = as.data.frame(cm$table),
                  aes(x = Prediction, y = Reference, fill = as.numeric(Freq))) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "#E69F18", high = "#56B4E9") +
  theme_minimal() +
  labs(title = "Confusion Matrix", x = "Prediction", y = "Reference") +
  geom_text(aes(label = Freq), size = 12, fontface = "bold") +
  scale_x_discrete(expand = c(0, 0.1)) +
  scale_y_discrete(expand = c(0, 0.1))+
  guides(fill = guide_colorbar(title = "Frequency"))
cm_plot
```



ACCURACY OF RANDOM FOREST MODEL ON TRAINING DATASET

```
#Making prediction on train data set:
rf_pred <- predict(rf_model, train)

# Calculating accuracy:
accuracy <- mean(rf_pred == train$Satisfaction.Rating)

# Printing accuracy:
cat("Accuracy of the random forest model on train data:", round(accuracy * 100, 2), "%\n")
```

```
## Accuracy of the random forest model on train data: 99.96 %
```

(4.4) NAIVE BAYES CLASSIFIER

```
# Train the Naive Bayes model
nb <- naiveBayes(Satisfaction.Rating ~ ., data = train)

# Make predictions on the test set
pred <- predict(nb, newdata = test[, -which(names(test) == "Satisfaction.Rating")])

# Calculating accuracy:
accuracy <- mean(pred == test$Satisfaction.Rating)

# Printing accuracy:
cat("Accuracy of the naive bayes model on test data:", round(accuracy * 100, 2), "%\n")
```

```
## Accuracy of the naive bayes model on test data: 78.94 %
```

‘

STATISTICS OF MODEL

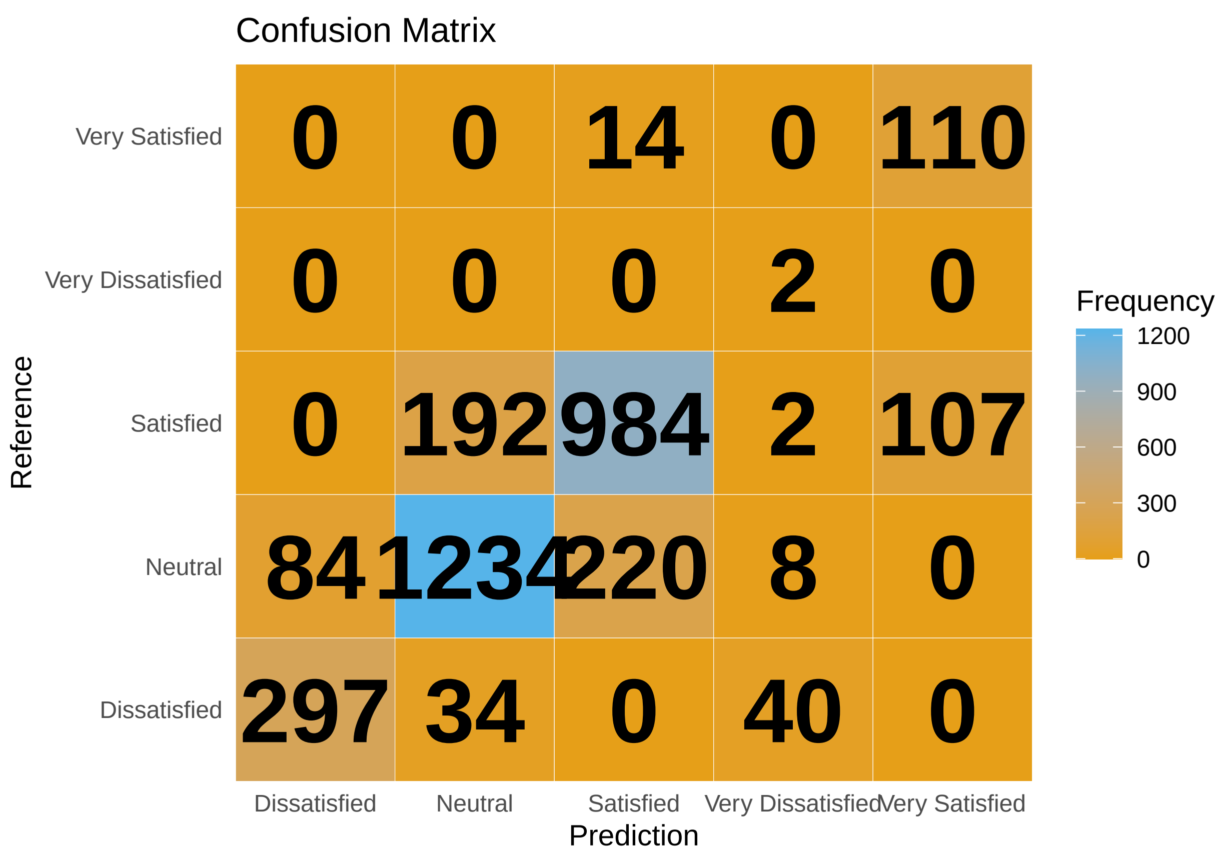
```
# Evaluate the performance of the model
confusionMatrix(pred, test$Satisfaction.Rating)
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Dissatisfied Neutral Satisfied Very Dissatisfied
##   Dissatisfied               297      84         0                 0
##   Neutral                     34    1234       192                 0
##   Satisfied                    0     220       984                 0
##   Very Dissatisfied           40       8         2                 2
##   Very Satisfied               0       0       107                 0
##                    Reference
## Prediction          Very Satisfied
##   Dissatisfied                   0
##   Neutral                        0
##   Satisfied                     14
##   Very Dissatisfied              0
##   Very Satisfied               110
##
## Overall Statistics
##
##                Accuracy : 0.7894
##                  95% CI : (0.7751, 0.8031)
##     No Information Rate : 0.4645
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6707
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Dissatisfied Class: Neutral Class: Satisfied
## Sensitivity                      0.80054         0.7982           0.7658
## Specificity                      0.97159         0.8732           0.8855
## Pos Pred Value                   0.77953         0.8452           0.8079
## Neg Pred Value                   0.97489         0.8330           0.8573
## Prevalence                       0.11148         0.4645           0.3861
## Detection Rate                   0.08924         0.3708           0.2957
## Detection Prevalence             0.11448         0.4387           0.3660
## Balanced Accuracy                0.88607         0.8357           0.8256
##                      Class: Very Dissatisfied Class: Very Satisfied
## Sensitivity                          1.000000               0.88710
## Specificity                          0.984967               0.96660
## Pos Pred Value                       0.038462               0.50691
## Neg Pred Value                       1.000000               0.99550
## Prevalence                           0.000601               0.03726
```

```
## Detection Rate                    0.000601          0.03305
## Detection Prevalence              0.015625          0.06520
## Balanced Accuracy                 0.992483          0.92685
```

CONFUSION MATRIX FOR NAIVE BAYES

```
cm <- confusionMatrix(pred, test$Satisfaction.Rating)

# create the confusion matrix plot

cm_plot <- ggplot(data = as.data.frame(cm$table),
                  aes(x = Prediction, y = Reference, fill = as.numeric(Freq))) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "#E69F18", high = "#56B4E9") +
  theme_minimal() +
  labs(title = "Confusion Matrix", x = "Prediction", y = "Reference") +
  geom_text(aes(label = Freq), size = 12, fontface = "bold") +
  scale_x_discrete(expand = c(0, 0.1)) +
  scale_y_discrete(expand = c(0, 0.1))+
  guides(fill = guide_colorbar(title = "Frequency"))
cm_plot
```



ACCURACY OF NAIVE BAYES MODEL ON TRAINING DATASET

```
# Make predictions on the train set:
pred <- predict(nb, newdata = train[, -which(names(train) == "Satisfaction.Rating")])
```

```r
# Calculating accuracy:
accuracy <- mean(pred == train$Satisfaction.Rating)

# Printing accuracy:
cat("Accuracy of the naive bayes model on train data:", round(accuracy * 100, 2), "%\n")
```

```
## Accuracy of the naive bayes model on train data: 79.28 %
```

(4.5) KNN

```r
#Train the KNN model:
knn_model <- train(Satisfaction.Rating~., data = train, method = "knn")

#Make prediction on test data set:
knn_predict <- predict(knn_model, newdata = test)

# Calculating accuracy:
accuracy <- mean(knn_predict == test$Satisfaction.Rating)

# Printing accuracy:
cat("Accuracy of the knn model on test data:", round(accuracy * 100, 2), "%\n")
```

```
## Accuracy of the knn model on test data: 86 %
```

STATISTICS OF MODEL

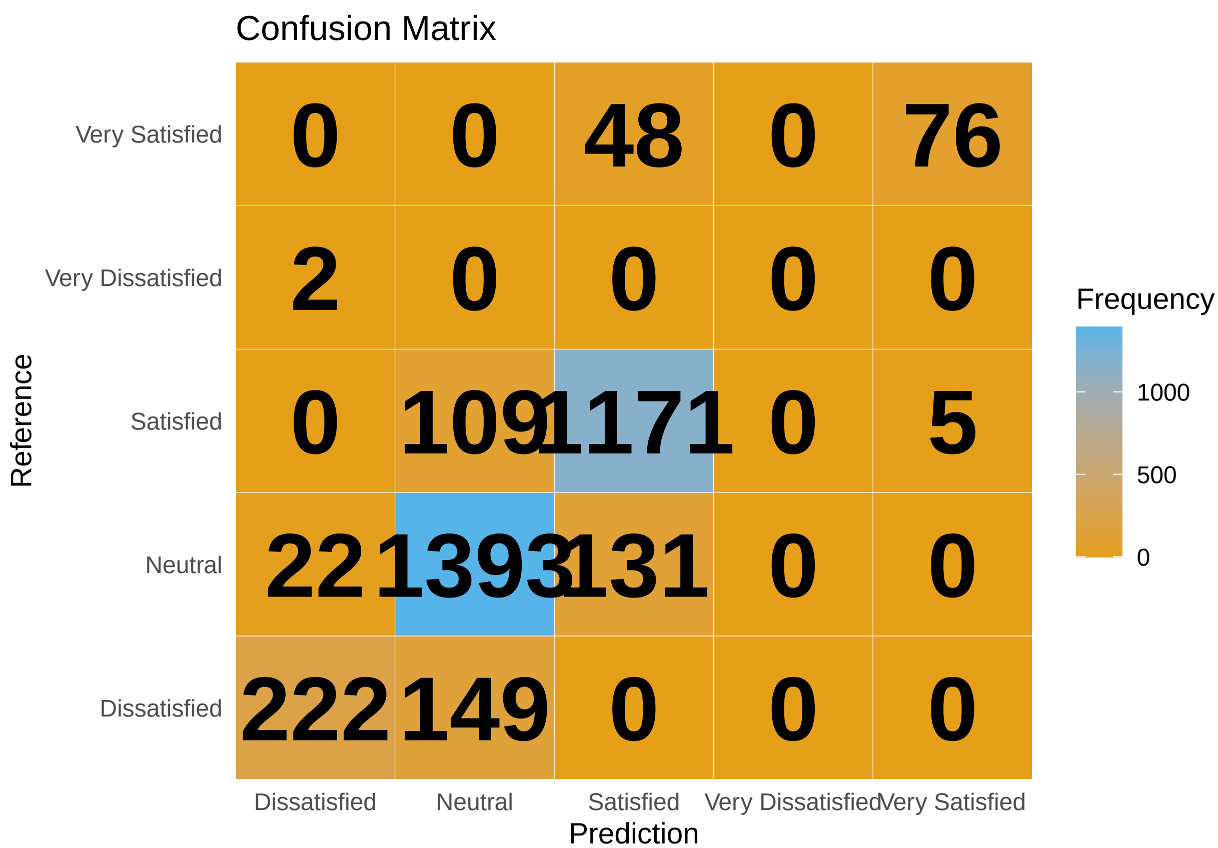
```r
confusionMatrix(knn_predict, test$Satisfaction.Rating)
```

```
## Confusion Matrix and Statistics
##
##                      Reference
## Prediction            Dissatisfied Neutral Satisfied Very Dissatisfied
##   Dissatisfied                 222      22         0                 2
##   Neutral                      149    1393       109                 0
##   Satisfied                      0     131      1171                 0
##   Very Dissatisfied              0       0         0                 0
##   Very Satisfied                 0       0         5                 0
##                      Reference
## Prediction            Very Satisfied
##   Dissatisfied                     0
##   Neutral                          0
##   Satisfied                       48
##   Very Dissatisfied                0
##   Very Satisfied                  76
##
## Overall Statistics
##
##                Accuracy : 0.86
##                  95% CI : (0.8477, 0.8716)
##     No Information Rate : 0.4645
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                    Kappa : 0.7681
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Dissatisfied Class: Neutral Class: Satisfied
## Sensitivity                      0.59838         0.9010           0.9113
## Specificity                      0.99188         0.8552           0.9124
## Pos Pred Value                   0.90244         0.8437           0.8674
## Neg Pred Value                   0.95165         0.9088           0.9424
## Prevalence                       0.11148         0.4645           0.3861
## Detection Rate                   0.06671         0.4186           0.3519
## Detection Prevalence             0.07392         0.4961           0.4056
## Balanced Accuracy                0.79513         0.8781           0.9118
##                      Class: Very Dissatisfied Class: Very Satisfied
## Sensitivity                          0.000000               0.61290
## Specificity                          1.000000               0.99844
## Pos Pred Value                            NaN               0.93827
## Neg Pred Value                       0.999399               0.98522
## Prevalence                           0.000601               0.03726
## Detection Rate                       0.000000               0.02284
## Detection Prevalence                 0.000000               0.02434
## Balanced Accuracy                    0.500000               0.80567
```

CONFUSION MATRIX

```r
cm <- confusionMatrix(knn_predict, test$Satisfaction.Rating)

# create the confusion matrix plot

cm_plot <- ggplot(data = as.data.frame(cm$table),
                  aes(x = Prediction, y = Reference, fill = as.numeric(Freq))) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "#E69F18", high = "#56B4E9") +
  theme_minimal() +
  labs(title = "Confusion Matrix", x = "Prediction", y = "Reference") +
  geom_text(aes(label = Freq), size = 12, fontface = "bold") +
  scale_x_discrete(expand = c(0, 0.1)) +
  scale_y_discrete(expand = c(0, 0.1))+
  guides(fill = guide_colorbar(title = "Frequency"))
cm_plot
```



ACCURACY OF KNN MODEL ON TRAINING DATASET

```
knn_predict <- predict(knn_model, newdata = train)

# Calculating accuracy:
accuracy <- mean(knn_predict == train$Satisfaction.Rating)

# Printing accuracy:
cat("Accuracy of the knn model on train data:", round(accuracy * 100, 2), "%\n")
```

```
## Accuracy of the knn model on train data: 89.35 %
```

5.COMPARISON OF FIVE CLASSIFICATION MODELS BASED ON TRAINING AND TESTING'S ACCURACY

```
model_data <- data.frame(Model = c("SVM", "Decision Tree", "Random Forest", "Naive Bayes","KNN"),
                         Training_Accuracy = c(99.98, 78.44, 99.96, 79.28, 89.35 ),
                         Testing_Accuracy = c(99.97, 77.55, 90.38, 78.94, 86))

# Converting the data frame to long format:
model_data_long <- tidyr::gather(model_data, key = "Accuracy_Type", value = "Accuracy", -Model)

# Plotting a clustered bar chart:
ggplot(model_data_long, aes(x = Model, y = Accuracy, fill = Accuracy_Type)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Model", y = "Accuracy", title = "Comparison of Model Accuracies") +
```

```
scale_fill_manual(values = c("blue", "red"), name = "Accuracy Type") +
theme_minimal()
```