```
knitr::opts_chunk$set(echo = TRUE)
```

Libraries

```
# import libraries
library(tidyverse)
library(corrplot)
library(ggplot2)
library(gridExtra)
library(correlation)
library(reshape)
library(reshape2)

data_train = read.csv("train.csv")
data_test = read.csv("test.csv")

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

DATA PREPROCESSING

```
# replace dots with underscores in column names
names(data) = gsub("\\.", "_", names(data))

# drop X and id column
data = data %>% select(-X, -id)

# convert categorical features to factor
data$Gender = factor(data$Gender, levels = c("Male", "Female"))
data$Customer_Type = factor(data$Customer_Type, levels = c("Loyal Customer",
"disloyal Customer"))
data$Type_of_Travel = factor(data$Type_of_Travel, levels = c("Personal Travel",
"Business travel"))
data$Class = factor(data$Class, levels = c("Business", "Eco Plus", "Eco"))
data$satisfaction = factor(data$satisfaction, levels = c("neutral or
dissatisfied", "satisfied"))

#######################################################
```

HANDLING NA VALUES

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

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

# na values are only 0.03% of the data -> drop na values
data = data %>% drop_na(Arrival_Delay_in_Minutes)



######################################################
# Print summary for each variable grouped by satisfaction, including the name of
the variable
for (col in names(data)) {
  print(col)
  print(by(data[[col]], data$satisfaction, summary))
}
######################################################
```

OUTLIERS

```
# plot boxplot of numeric variables
plots = list()
for (col in names(data)[sapply(data, is.numeric)]) {
  plot = ggplot(data, aes(x = .data[[col]])) +
  geom_boxplot() +
  labs(title = col, x = col, y = "Count")
  plots[[col]] = plot
}

grid.arrange(grobs = plots, ncol = 3)

# plot boxplot against satisfaction with colors
plots = list()
for (col in names(data)[sapply(data, is.numeric)]) {
  plot = ggplot(data, aes(x = satisfaction, y = .data[[col]], fill =
satisfaction)) +
  geom_boxplot() +
  labs(title = col, x = "Satisfaction", y = col)
  plots[[col]] = plot
}

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

###################################################################

# select examples of departure delay greater than 500
examples=data[data$Departure_Delay_in_Minutes > 800,]
# and print table of satisfaction by departure delay
table(examples$satisfaction)
```

```
# count the number of examples with departure delay = 0
sum(data$Departure_Delay_in_Minutes > 0)
sum(data$Departure_Delay_in_Minutes <= 0)

sum(data$Arrival_Delay_in_Minutes > 0)
sum(data$Arrival_Delay_in_Minutes <= 0)

summary(data)
```

VISUALIZATION

```
# plot pie chart for each variable
plots = list()
for (col in names(data)[sapply(data, is.factor)]) {
  plot = ggplot(data, aes(x = "", fill = .data[[col]])) +
  geom_bar(width = 1) +
  coord_polar("y", start = 0) +
  labs(title = paste("Pie Chart of", col))
  plots[[col]] = plot
}

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


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

  plots[[col]] = plot
}

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

##################

# plot distribution of numeric variables
plots = list()
for (col in names(data)[sapply(data, is.numeric)]) {
  plot = ggplot(data, aes(x = .data[[col]])) +
  geom_histogram() +
  labs(title = col, x = col, y = "Count")
  plots[[col]] = plot
}

grid.arrange(grobs = plots, ncol = 3)
```

```
##################

# plots categorical variables vs satisfaction
plots = list()
for (col in names(data)[sapply(data, is.factor)]) {
  if (col == "satisfaction") {
    next
  }
  plot = ggplot(data, aes(x = satisfaction, fill = .data[[col]])) +
  geom_bar(position = "dodge") +
  scale_fill_manual(values = rainbow(length(unique(data[[col]]))),
                    labels = unique(data[[col]]),
                    name = col) +
  labs(title = paste("Histogram of Satisfaction by", col), x = "Satisfaction", y =
"Count")

  plots[[col]] = plot

}

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

# Create density plots for Age and Flight_Distance
plots = list()
for (col in c("Age", "Flight_Distance")) {
  plot = ggplot(data, aes(x = .data[[col]], fill = satisfaction)) +
  geom_density(alpha = 0.4) +
  labs(title = paste("Density Plot of", col), x = col, y = "Density")
  plots[[col]] = plot
}

# Arrange the density plots in a grid
grid.arrange(grobs = plots)

######################################################

# plots numeric variables vs satisfaction
plots = list()
for (col in names(data)[sapply(data, is.numeric)]) {
  if (col == "satisfaction") {
    next
  }
  plot = ggplot(data, aes(x = satisfaction, y = .data[[col]])) +
  geom_boxplot() +
  labs(x = "Satisfaction", y = col)

  plots[[col]] = plot

}

grid.arrange(grobs = plots, ncol = 4)

#############################################
```

CONVERT CATEGORICAL TO NUMERIC

```
data$Gender = as.numeric(data$Gender) - 1
data$Customer_Type = as.numeric(data$Customer_Type) - 1
data$Type_of_Travel = as.numeric(data$Type_of_Travel) - 1
data$Class = as.numeric(data$Class) - 1
data$satisfaction = as.numeric(data$satisfaction) - 1

#############################################
```

DATA BALANCE

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

##############################################
```

TRAIN TEST SPLIT

```
set.seed(123)
train_index = sample(1:nrow(data), 0.8*nrow(data))
# 80% of data is used for training
train = data[train_index,]
# 20% of data is used for testing
test = data[-train_index,]

# merge train and test data
data = rbind(train, test)
#save on cvs
write.csv(data, file = "data.csv")

# save true values of test satisfaction column
test_true = test$satisfaction

# drop satisfaction column from test data
test = test %>% select(-satisfaction)

# print proportion of satisfied and dissatisfied customers in train and test data
prop.table(table(train$satisfaction))
prop.table(table(test_true))

##############################################
```

CORRELATION MATRIX

```
# correlation matrix only for numeric variables
correlation_matrix = cor(data[, sapply(data, is.numeric)])

# Plot a heatmap of the correlation matrix
ggplot(data = reshape2::melt(correlation_matrix)) +
  geom_tile(aes(x = Var1, y = Var2, fill = value)) +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Correlation") +
  theme(axis.text.x = element_text(angle = 90, vjust = 1,
                                    size = 10, hjust = 1)) +
  coord_fixed()

# Find high correlated features with satisfaction
# TODO: do the same with different threshold to find differences
# NOTE: i decided to use 0.3 as threshold
satisfaction_corr <- correlation_matrix['satisfaction',]
high_corr_satis <- names(satisfaction_corr[abs(satisfaction_corr) > 0.3 |
abs(satisfaction_corr) < -0.3])
high_corr_satis <- high_corr_satis[high_corr_satis != "satisfaction"]
high_corr_satis


# Compute the correlations between the high correlation features and satisfaction
correlations <- data.frame(
  feature = high_corr_satis,
  correlation = sapply(high_corr_satis, function(x) cor(data[,x],
data$satisfaction))
)
correlations

# plot the correlations
ggplot(correlations, aes(x = reorder(feature, correlation), y = correlation)) +
  geom_bar(stat = "identity", fill = "blue", alpha = 0.4) +
  ggtitle("Correlation between features and satisfaction") +
  xlab('Features') +
  ylab('Correlation')


#save on cvs
write.csv(correlations, file = "correlations.csv")

##############################################
```

PLOT ALL DISTRIBUTIONS (with numerical categories)

```
sat = data$satisfaction
features_names = names(data)

num_cols = 4
```

```
num_rows = ceiling(ncol(data)/num_cols)
par(mfrow = c(num_rows, num_cols))

# for each feature plot the density of satisfied and dissatisfied customers
for(col in features_names) {
  # calculate number of breaks
  num_breaks = length(unique(data[[col]]))
  num_breaks = min(num_breaks, 20)
  hist(data[[col]], breaks = num_breaks,
    main = paste("Histogram of ", col), xlab = col, ylab = "Frequency",
    col = "lightblue"
  )
}

#plot the density of columns of data which names are in correlations. Use
barplots.

#TODO: make them visually right.
#DONE

# Count frequence of value of Type_of_Travel

frequency <- table(train$Type_of_Travel)

# Create a barplot
barplot(frequency, col = "blue", xlab = "Type_of_Travel", ylab = "Frequency", main
= "Type_of_Travel - Frequency Plot")




a = ggplot(train, aes(x = Type_of_Travel, fill = sat)) +
  geom_histogram(fill = 'Blue', alpha = 0.4, bins = 2)



b = ggplot(train, aes(x = Class, fill = sat)) +
  geom_bar(fill = 'Blue', alpha = 0.4) +
  ggtitle("Class - Density Plot") +
  xlab('Class')

c = ggplot(train, aes(x = Online_boarding, fill = sat)) +
  geom_bar(fill = 'Blue', alpha = 0.4) +
  ggtitle("Online_boarding - Density Plot") +
  xlab('Online_boarding')

d = ggplot(train, aes(x = Seat_comfort, fill = sat)) +
  geom_bar(fill = 'Blue', alpha = 0.4) +
  ggtitle("Seat_comfort - Density Plot") +
  xlab('Seat_comfort')

e = ggplot(train, aes(x = Inflight_entertainment, fill = sat)) +
  geom_bar(fill = 'Blue', alpha = 0.4) +
  ggtitle("Inflight_entertainment - Density Plot") +
  xlab('Inflight_entertainment')
```

```
f = ggplot(train, aes(x = On_board_service, fill = sat)) +
  geom_bar(fill = 'Blue', alpha = 0.4) +
  ggtitle("On_board_service - Density Plot") +
  xlab('On_board_service')


g = ggplot(train, aes(x = Leg_room_service, fill = sat)) +
  geom_bar(fill = 'Blue', alpha = 0.4) +
  ggtitle("Leg_room_service - Density Plot") +
  xlab('Leg_room_service')

h = ggplot(train, aes(x = Cleanliness, fill = sat)) +
  geom_bar(fill = 'Blue', alpha = 0.4) +
  ggtitle("Cleanliness - Density Plot") +
  xlab('Cleanliness')


grid.arrange(a, b, c, d, e, f, g, h, ncol = )



#CORRELATION MATRIX again but now we are interested in partial correlation
#So we look for all the correlations between variables
#We pick the highest, setting a treshold of our choice

#build a dataframe where for each variable we look the partial correlation with
all the others
#we pick the highest and we save it in a dataframe
#we set a treshold of 0


#correlation(train, partial=TRUE, method='pearson')
#save the partial correlation matrix result in a dataframe and output a file for
further analysis


#partial_corr <- correlation(train, partial=TRUE, method='pearson')
#write.csv(partial_corr, file = "partial_corr.csv")

partial_correlations = read.csv("partial_corr.csv", header = TRUE, sep = ",")

#make the first column the row names
rownames(partial_correlations) = partial_correlations[,1]

colnames(partial_correlations)
#drop the first  (X) column
partial_correlations = partial_correlations[,-1]

# Create a new matrix with rounded partial correlations
partial_correlations_rounded <- round(partial_correlations, digits = 3)


# Initialize empty data frame with 0 rows
# We need it to create a data frame with the results and
# so to show better the correlations.
df <- data.frame(variable1 = character(),
                 variable2 = character(),
                 value = numeric(),
                 stringsAsFactors = FALSE)

# Loop over rows and columns of matrix
for (i in 1:nrow(partial_correlations_rounded)) {
  for (j in 1:ncol(partial_correlations_rounded)) {
    print(partial_correlations_rounded[i,j])
    # Check if value meets criterion
    if ((partial_correlations_rounded[i,j] > 0.300 |
partial_correlations_rounded[i,j] < -0.300)& i != j) {
      print('it is true')
      # Add row to data frame
      df <- rbind(df, data.frame(variable1 =
rownames(partial_correlations_rounded)[i],
                                 variable2 =
colnames(partial_correlations_rounded)[j],
                                 value = partial_correlations_rounded[i,j],
                                 stringsAsFactors = FALSE))
    }
  }
}


# Group the data frame by variable1 and extract top 3 values for each group
df_top3 <- df %>% group_by(variable1) %>% top_n(4, value) %>% ungroup()

#order by variable1
df_top3 <- df_top3[order(df_top3$variable1),]


#delete duplicates in the dataframe if variable1 is equal to variable2
df_top3 <- df_top3[!(df_top3$variable1 == df_top3$variable2),]

print(df_top3, n = nrow(df_top3))
#save on cvs
write.csv(df_top3, file = "df_top3.csv")


#TODO:EXPLAIN THE CORRELATIONS AND PLOTTING THE RESULTS

par(mfrow = c(1, 1))
```

Relation between Arrival_Delay_in_Minutes and Departure_Delay_in_Minutes (linear)

```
# standardize Arrival_Delay_in_Minutes and Departure_Delay_in_Minutes
arrival_std = scale(data$Arrival_Delay_in_Minutes)
departure_std = scale(data$Departure_Delay_in_Minutes)
```

```
# scatter plot of Arrival_Delay_in_Minutes and Departure_Delay_in_Minutes
plot(arrival_std, departure_std, xlab = "Arrival_Delay_in_Minutes", ylab =
"Departure_Delay_in_Minutes")
# plot line y = x
abline(0, 1, col = "red")
```

Relationship between Class (Business, Eco, Eco Plus) and Type of travel (Personal Travel, Business Travel)

```
# plot barplot coloured for each type of class vs type of travel
ggplot(data, aes(x = Class, fill = Type_of_Travel)) +
  geom_bar(position = "dodge") +
  labs(title = "Histogram of Class by Type of Travel", x = "Class", y = "Count")

#######################################################àà
```

VARIABLE DESCRIPTION

```
# Print summary for each variable grouped by satisfaction, including the name of
the variable
for (col in names(data)) {
  print(col)
  print(by(data[[col]], data$satisfaction, summary))
}



# print table of type of travel by satisfaction
table(data$Type_of_Travel, data$satisfaction)
```