UNIVERSITY OF GLASGOW

DATA SCIENCE FOR

MARKETING ANALYTICS

MGT5372

GUID: 2808473M

ANALYSING THE CUSTOMER REVIEW AND RATING TO GAIN INSIGHTS ON CUSTOMER SERVICE, COMPETITATIVE PERFORMMANCE AND OPERATIONAL EFFICIENCY.

TABLE OF CONTENT

Section – 1

Q 1.1

Answer: The first step in the analysis involved importing the two datasets, "flight_reviews_01.csv" and "flight_reviews_02.csv". This was accomplished using the ‘read_csv’ function from the ‘readr’ library in R. This function reads a file in table format and creates a data frame from it, which is suitable for data manipulation tasks.

```
> # Question 1.1
>
> # Loading the datasets
> df1 <- read_csv("C:/Users/Methil/OneDrive/Documents/Data Science/flight_reviews_01.csv")
Rows: 499 Columns: 17
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (9): title, review_date, review, aircraft, type_of_traveller, seat_type, route, date_flown, food_and_beverages
dbl (6): rating, seat_comfort, cabin_staff_service, ground_service, value_for_money, wifi_and_connectivity
lgl (2): trip_verified, recommend

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
> df2 <- read_csv("C:/Users/Methil/OneDrive/Documents/Data Science/flight_reviews_02.csv")
Rows: 500 Columns: 17
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (9): title, review_date, review, aircraft, type_of_traveller, seat_type, route, date_flown, food_and_beverages
dbl (6): rating, seat_comfort, cabin_staff_service, ground_service, value_for_money, wifi_and_connectivity
lgl (2): trip_verified, recommend

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

After importing the datasets, they were merged into a single dataset named "df" using the ‘bind_rows’ function from the ‘dplyr’ library. This function combines two data frames by binding them row-wise, which is appropriate in this case since both datasets have the same columns.

```
>
> # Merging the two datasets
> df <- bind_rows(df1, df2)
>
> # Exporting the merged dataset
> write.csv(df, "Data.csv", row.names = FALSE)
```

The merged dataset was then exported as a CSV file named "Data.csv" using the “write.csv” function. This function writes an object to a file in table format, which is useful for saving and sharing data.

Q 1 .2

Answer: The presence of missing values in the dataset was checked using the ‘is.na’ function combined with ‘colSums’. This combination of functions allows for the identification of missing values in each column of the dataset. The columns which were found to have missing values are:

Aircraft – 405

Type of traveller – 1

Route – 2

Seat comfort – 61

Cabin staff service – 68

Food and beverage - 442

Ground staff – 43

Wifi and connectivity - 759

```
>
> # Checking for missing values
> missing_values <- colSums(is.na(df))
>
> # Reporting columns with missing values
> print(missing_values[missing_values > 0])
            aircraft      type_of_traveller                  route          seat_comfort   cabin_staff_service    food_and_beverages
                 405                      1                      2                     61                    68                   442
      ground_service wifi_and_connectivity
                  43                   759
```

These missing values were dealt with by replacing them with the mean of the respective column if the column was numeric, and with the most frequent value if the column was categorical. This approach was chosen because it allows for the preservation of the overall distribution and characteristics of the data while providing a reasonable estimate for the missing values.

Q 1.3

Answer: The chosen method for dealing with missing values was to replace them with the mean value for numeric columns and the most frequent value for categorical columns. This approach was chosen because it prevents the loss of data that would occur if rows or columns with missing values were removed. It also provides a reasonable estimate for the missing values that can allow for meaningful analysis to be conducted. This method is often used in data analysis as it is a simple and effective way to handle missing data without introducing too much bias.

```
>
> # Dealing with missing values
> for(col in names(df)) {
+   if(is.numeric(df[[col]])) {
+     df[[col]][is.na(df[[col]])] <- mean(df[[col]], na.rm = TRUE)
+   } else {
+     levels <- unique(df[[col]])
+     df[[col]][is.na(df[[col]])] <- levels[which.max(tabulate(match(df[[col]], levels)))]
+   }
+ }
>
>
```

Q 1.4

Answer: There are six numeric variables in the dataset: "rating", "seat_comfort", "cabin_staff_service", "ground_service", "value_for_money", and "wifi_and_connectivity". These variables represent different aspects of the flight experience, such as the overall rating, comfort of the seat, quality of the cabin staff service, ground service, value for money, and wifi connectivity. A description table was created for these variables using the ‘sapply’ function, which applies a function to each element of a list or vector. In this case, several statistical functions (mean, median, min, max, standard deviation) were applied to these numeric columns to provide a summary of their characteristics.

```
>
> # Getting numeric columns
> numeric_cols <- df %>% select_if(is.numeric)
>
> # Description table for numeric columns
> description <- sapply(numeric_cols, function(x) c(Mean = mean(x), Median = median(x),
+                                                   Min = min(x), Max = max(x),
+                                                   Standard_Deviation = sd(x)))
> print(description)
                    rating seat_comfort cabin_staff_service ground_service value_for_money wifi_and_connectivity
Mean              4.409409     2.795309            3.191192       2.773013       2.529530            2.1333333
Median            3.000000     3.000000            3.191192       3.000000       2.000000            2.1333333
Min               1.000000     1.000000            1.000000       1.000000       1.000000            1.0000000
Max              10.000000     5.000000            5.000000       5.000000       5.000000            5.0000000
Standard_Deviation 3.243546     1.326733            1.471155       1.452571       1.465324            0.7202575
>
```

Description Table

Variable name	Mean	Median	Min	Max	Standard_Deviation
rating	4.409409	3	1	10	3.243546
seat_comfort	2.795309	3	1	5	1.326733
cabin_staff_service	3.191192	3.191192	1	5	1.471155
ground_service	2.773013	3	1	5	1.452571
value_for_money	2.52953	2	1	5	1.465324
wifi_and_connectivity	2.1333333	2.1333333	1	5	0.7202575

Q 1.5

Answer: The 'review_date' and 'date_flown' variables were first converted to Date type using the ‘dmy’ and ‘my’ functions from the lubridate library, respectively. These functions are designed to handle dates in various formats and convert them into a standard Date type that can be easily manipulated in R.

```
>
> # Required library
> library(lubridate)
>
> # Convert 'review_date' and 'date_flown' to Date type
> df$review_date <- dmy(df$review_date) # Using dmy function to convert dates in "day/month/year" format
> df$date_flown <- my(df$date_flown) # Using my function to convert dates in "month/year" format
>
```

After converting these variables to Date type, they were decomposed into 'review_year', 'review_month', 'review_day', 'flown_year', and 'flown_month' using the ‘year’, ‘month’, and ‘day’ functions from the same library. These functions extract the respective components from a Date object.

```
> # Decompose 'review_date'
> df$review_year <- year(df$review_date)
> df$review_month <- month(df$review_date)
> df$review_day <- day(df$review_date)
>
> # Decompose 'date_flown'
> df$flown_year <- year(df$date_flown)
> df$flown_month <- month(df$date_flown)
```

The original 'review_date' and 'date_flown' columns were then removed from the dataset. This step was necessary to facilitate further analysis based on the year, month, and day of the review and flight.

```
>
> # Drop the original columns
> df$review_date <- NULL
> df$date_flown <- NULL
>
```

review_year	review_month	review_day	flown_year	flown_month
2022	11	7	2022	11
2022	11	5	2022	10
2022	10	30	2022	10
2022	10	25	2022	10
2022	10	21	2022	10
2022	10	16	2022	10
2022	9	7	2022	8
2022	9	6	2021	11
2022	8	22	2022	6
2022	8	20	2022	6
2022	7	27	2022	7
2022	7	22	2022	5
2022	7	20	2022	6
2022	7	4	2022	6
2022	6	30	2022	6
2022	6	27	2022	6
2022	6	23	2022	6
2022	6	20	2022	6
2022	5	30	2022	5
2022	5	27	2022	5
2022	5	23	2022	5
2022	5	18	2022	3
2022	5	18	2022	5
2022	5	11	2022	5
2022	5	8	2022	5

The screenshot above shows that the 'review_date' variable is been decomposed in three new variables which are 'review_year', 'review_month', and 'review_day', and 'date_flown' variable is been decomposed in two new variables 'flown_year' and 'flown_month'. The variables are expressed as integers.

Section – 2

Q 2.1

Answer: The percentage of verified and non-verified trips was calculated by creating a table of the 'trip_verified' column using the ‘table’ function and then converting these counts into proportions using the ‘prop.table’ function.

```
> # Question 2.1
>
> # Count and percentages of verified and non-verified trips
> verified_counts <- table(df$trip_verified)
> verified_percentages <- prop.table(verified_counts) * 100
```

The results show that approximately 78.38% of the trips were verified, while 21.62% were not.

```
> print(verified_percentages)

   FALSE     TRUE
21.62162 78.37838
```

The average overall ratings for verified and non-verified trips were then calculated using the ‘tapply’ function, which applies a function to subsets of a vector grouped by some criteria. Then tt applies the ‘mean’ function to the 'rating' column, grouped by the 'trip_verified' column.

```
>
> # Average ratings for verified and non-verified trips
> avg_rating <- tapply(df$rating, df$trip_verified, mean)
```

The results show that the average rating for verified trips was approximately 4.50, while for non-verified trips it was approximately 4.06.

```
> print(avg_rating)
   FALSE     TRUE
4.064815 4.504470
```

Q 2.2

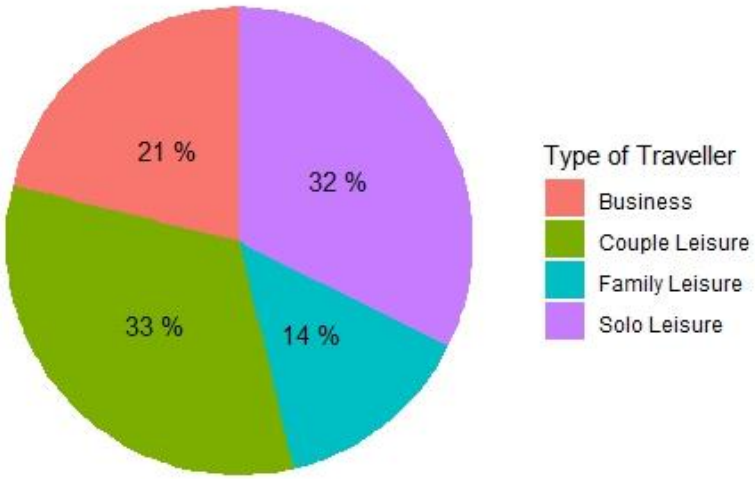
Answer: A pie chart was created to visualize the composition of types of travellers. This was done by first creating a table of the 'type_of_traveller' column and then converting this table into a data frame.

```
>
> # Required library
> library(ggplot2)
>
> # Count of each type of travellers
> traveller_counts <- table(df$type_of_traveller)
>
> # Convert to data frame
> traveller_df <- as.data.frame(table(df$type_of_traveller))
> names(traveller_df) <- c("type_of_traveller", "count")
```

The ‘ggplot’ function from the ‘ggplot2’ library was then used to create the pie chart, with the count of each type of traveller represented as a slice of the pie. The ‘geom_bar’ function creates the pie chart, the ‘coord_polar’ function transforms it into a pie chart, and the ‘geom_text’ function adds the percentage labels to the pie chart . The ‘theme_void’ function removes all non-data ink from the plot.

```
> # Pie chart
> ggplot(traveller_df, aes(x = "", y = count, fill = type_of_traveller)) +
+   geom_bar(width = 1, stat = "identity") +
+   coord_polar("y", start=0) +
+   geom_text(aes(label = paste(round(count/sum(count)*100), "%")), position = position_stack(vjust = 0.5)) +
+   theme_void() +
+   labs(fill = "Type of Traveller")
```

Pie Chart



The pie chart above indicates four types of travellers with percentage:

Business – 21%

Couple Leisure – 33%

Family – 14%

Solo Leisure – 32%

Q 2.3

Answer: The most common type of traveller and seat type were determined by creating tables of the 'type_of_traveller' and 'seat_type' columns using the ‘table’ fucntion, respectively, and then finding the names of the maximum values in these tables using the ‘which.max’ function. The results show that the most common type of traveller was "Couple Leisure" and the most common seat type was "Economy Class".

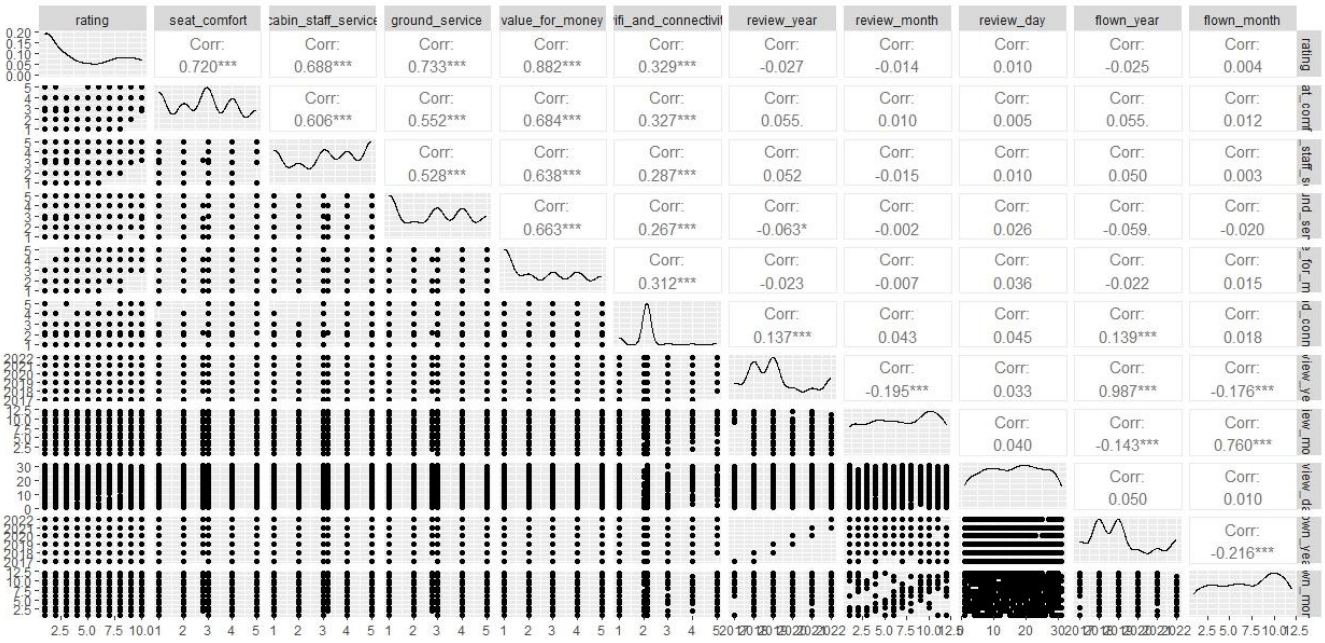
```
>
> # Most common type of traveller
> common_traveller <- names(which.max(traveller_counts))
>
> # Most common seat type
> seat_counts <- table(df$seat_type)
> common_seat <- names(which.max(seat_counts))
>
> print(common_traveller)
[1] "Couple Leisure"
> print(common_seat)
[1] "Economy Class"
```

Q 2.4

Answer: The code first selects the numeric columns from the dataset using the 'select_if' and 'is.numeric' functions from the 'dplyr' library. The 'ggpairs' function from the 'GGally' library is then used to create a grid of plots for these numeric variables. This function creates a matrix of scatterplots for each pair of numeric variables, with Pearson correlation coefficients in the upper right corner , scatterplots, and variable distributions om the diagonal. This type of plot is useful for exploring the relationships between multiple variables at once and can provide insights into potential correlations and patterns in the data.

```
>
> # Required library
> library(GGally)
>
> # Selecting numeric columns
> numeric_cols <- df %>% select_if(is.numeric)
>
> # ggpairs plot
> ggpairs(numeric_cols)
```

Numeric Plot



Section – 3

Q 3.1

Answer:

Logistic Regression:

Logistic Regression is a statistical method used for binary classification problems. It is a type of regression analysis where the dependent variable is binary or dichotomous, i.e., it can take only two possible outcomes. The logistic regression model uses a logistic function to model a binary dependent variable. The logistic function, also known as the sigmoid function, is an S-shaped curve that maps any real-valued number into another value between 0 and 1. In machine learning, this is useful for transforming an arbitrary-valued function into a function better suited for binary classification (Hosmer Jr, D. W.,

Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression. John Wiley & Sons).

The logistic regression model estimates the probability that a given input point belongs to a certain class. The output of logistic regression is a probability that the given input point belongs to a certain class. The central premise of Logistic Regression is the assumption that your input space can be separated into two nice 'regions', one for each class, by a linear (read: straight) boundary. If the probability is greater than a threshold value, typically 0.5, the model predicts the class label to be positive; otherwise, it predicts the class label to be negative.

In the context of this project, the logistic regression model is used to predict whether a customer would recommend a flight based on various features such as rating, seat comfort, cabin staff service, food and beverages, ground service, value for money, and wifi and connectivity. The logistic regression model outputs probabilities that are then converted to binary outcomes (recommend or not recommend) using a threshold, typically 0.5.

Decision Tree:

Decision Trees are a type of supervised learning algorithm that is mostly used for classification problems, but it is capable of performing both classification and regression tasks. It works for both categorical and continuous input and output variables. In this project, a decision tree model is used to predict the same outcome as the logistic regression model. The decision tree model makes predictions by learning simple decision rules inferred from the features of the data. The topmost node in a decision tree is known as the root node. It learns to partition based on the attribute value. It partitions the tree recursively in a manner called recursive partitioning (Rokach, L., & Maimon, O. (2014). Data mining with decision trees: theory and applications. World scientific).

A decision tree is a flowchart-like structure in which each internal node represents a feature(or attribute), each branch represents a decision rule, and each leaf node represents an outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree recursively in a manner called recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.

Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.

Q 3.2

Answer: Library Installation and Loading: The necessary libraries are installed and loaded. The ‘caTools’ library is used for its ‘sample.split’ function, which helps in splitting the data into training and testing sets. The ‘glmnet’ library is used for logistic regression, and the ‘rpart’ library is used for decision tree modeling. The ‘lattice’, ‘caret’, and ‘ggplot2’ libraries are used for data visualization and model evaluation.

Data Preparation: The response and predictor variables are selected from the dataframe ‘df’ and stored in ‘df_model’. The response variable is "recommend", and the predictor variables are various customer ratings.

```
> # Select the response and predictor variables
> df_model <- df[c("rating", "seat_comfort", "cabin_staff_service", "food_and_beverages",
"ground_service", "value_for_money", "wifi_and_connectivity", "recommend")]
>
```

Data Splitting: The data is split into training and testing sets using the ‘sample.split’ function from the ‘caTools’ library. A seed is set for reproducibility of results. The split ratio is set to 0.75, meaning 75% of the data goes into the training set and the remaining 25% goes into the testing set.

```
> # Split the data into training and testing sets
> set.seed(123) # For reproducibility
> split <- sample.split(df_model$recommend, SplitRatio = 0.75)
> training_set <- subset(df_model, split == TRUE)
> testing_set <- subset(df_model, split == FALSE)
>
```

Model Building - Logistic Regression: The ‘glm’ function from the ‘glmnet’ library is used to build the logistic regression model. The model is trained on the training set with "recommend" as the response variable and all other variables as predictors. The ‘family’ argument is set to ‘binomial’, indicating that the response variable is binary.

```
> #Implementing logistic regression model
>
> # Required library
> library(glmnet)
>
> # Build the model
> log_model <- glm(recommend ~ ., family = binomial, data = training_set)
```

Model Prediction - Logistic Regression: The ‘predict’ function is used to make predictions on the testing set using the logistic regression model. The ‘type’ argument is set to "response" to get probabilities.

```
> # Make predictions
> log_pred <- predict(log_model, newdata = testing_set, type = "response")
>
> # Convert probabilities to binary outcome
> log_pred <- ifelse(log_pred > 0.5, 1, 0)
```

Outcome Conversion - Logistic Regression: The probabilities are converted to binary outcomes (1 or 0) using the ‘ifelse’ function. If the predicted probability is greater than 0.5, it is considered as 1 (recommend); otherwise, it is 0 (not recommend).

```
#Implementing the decision tree model

# Required library
library(rpart)

# Build the model
tree_model <- rpart(recommend ~ ., data = training_set, method = "class")
```

Model Building - Decision Tree: The ‘rpart’ function from the ‘rpar’t library is used to build the decision tree model. The model is trained on the training set with "recommend" as the response variable and all other variables as predictors. The ‘method’ argument is set to "class", indicating that the response variable is categorical.

```
> # Make predictions
> tree_pred <- predict(tree_model, newdata = testing_set, type = "class")
>
```

Model Prediction - Decision Tree: The ‘predict’ function is used to make predictions on the testing set using the decision tree model. The ‘type’ argument is set to "class" to get class labels.

```
>
> # Convert the variables to factors with "FALSE" and "TRUE" levels
> log_pred <- as.factor(log_pred)
> levels(log_pred) <- c("FALSE", "TRUE")
> tree_pred <- as.factor(tree_pred)
> testing_set$recommend <- as.factor(testing_set$recommend)
> levels(testing_set$recommend) <- c("FALSE", "TRUE")
>
> # Compute accuracy of logistic regression model
> log_accuracy <- sum(diag(confusionMatrix(log_pred, testing_set$recommend)$table)) /
sum(confusionMatrix(log_pred, testing_set$recommend)$table)
>
> # Compute accuracy of decision tree model
> tree_accuracy <- sum(diag(confusionMatrix(tree_pred, testing_set$recommend)$table))
/ sum(confusionMatrix(tree_pred, testing_set$recommend)$table)
>
```

Model Evaluation: The ‘confusionMatrix’ function from the ‘caret’ library is used to compute the confusion matrix for both models. The accuracy of each model is calculated as the sum of the diagonal elements (true positives and true negatives) divided by the sum of all elements in the confusion matrix. The accuracies of the two models are compared.

```
> # Compute importance of features in logistic regression model
> log_imp <- abs(coef(log_model))
>
> # Sort in descending order
> log_imp <- sort(log_imp, decreasing = TRUE)
```

Feature Importance - Logistic Regression: The ‘coef’ function is used to get the coefficients of the logistic regression model. The absolute values of the coefficients are taken as the importance of the features. The features are sorted in descending order of importance.

Each of these steps is crucial in the process of building, predicting, and evaluating the models.

Q 3.3

Answer: The best model is selected based on the accuracy of the predictions on the testing data. In this case, the logistic regression model has an accuracy of

approximately 0.9466%, and the decision tree model has an accuracy of approximately 0.9317%. Therefore, the logistic regression model is selected as the best model due to its higher accuracy (Raschka, 2018).

```
> # Compare accuracies
> log_accuracy
[1] 0.9466667
> tree_accuracy
[1] 0.9317269
```

The logistic regression model provides insights into the importance of different features in predicting whether a customer would recommend a service. The importance of features is computed based on the absolute values of the coefficients in the logistic regression model. The larger the absolute value of a coefficient, the more important the corresponding feature is in the prediction.

The most important feature in the prediction is the overall rating given by the customer, followed by the rating for food and beverages. This suggests that the overall experience and the quality of food and beverages are crucial factors that influence a customer's decision to recommend a service. This aligns with the marketing theory that customer satisfaction is a key driver of customer loyalty and word-of-mouth recommendations (Kotler et al., 2019).

The ratings for wifi and connectivity, ground service, cabin staff service, and value for money also play significant roles in the prediction. This indicates that these aspects of the service also contribute to a customer's overall satisfaction and their decision to recommend the service. This is consistent with the service quality model, which suggests that different dimensions of service quality, including reliability, responsiveness, assurance, empathy, and tangibles, influence customer satisfaction (Parasuraman et al., 1988).

The seat comfort has the least importance in the prediction. This might be because seat comfort is a basic expectation that customers have for a service,

and it does not significantly differentiate a service in the eyes of the customers (Kotler et al., 2019).

In conclusion, the logistic regression model provides valuable business insights into what factors are important in predicting customer recommendation. These insights can guide business strategies to improve customer satisfaction and increase word-of-mouth recommendations.

Q.4 Discuss two feasible data science methods or steps that can be taken to improve your analysis. For example, can any information be extracted through text mining of reviews? (10%)

Answer: One way to improve the analysis is to use text mining techniques on the reviews. This could provide additional insights into why customers recommend or do not recommend a flight. For example, sentiment analysis could be used to determine the overall sentiment of the reviews, and topic modeling could be used to identify common themes in the reviews.

Another way to improve the analysis is to use more advanced machine learning models such as random forests or gradient boosting. These models can capture more complex relationships between the features and the target variable and could potentially improve the accuracy of the predictions (Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794)).

Referencing

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression. John Wiley & Sons.

Rokach, L., & Maimon, O. (2014). Data mining with decision trees: theory and applications. World scientific.

Kotler, P., Keller, K. L., Manceau, D., & Hémonnet-Goujot, A. (2019). Marketing management (15th global ed.). Pearson.

Parasuraman, A., Zeithaml, V. A., & Berry, L. L. (1988). SERVQUAL: A multiple-item scale for measuring consumer perceptions of service quality. Journal of Retailing, 64(1), 12.

Raschka, S. (2018). Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. arXiv preprint arXiv:1811.12808.