Contents

List of Figures

List of Tables

1

Introduction and Background

1.1 Executive Summary

The aviation company, specializing in domestic and international trips, seeks to optimize its marketing strategy by leveraging digital platforms and social media. Formerly depending on cold calling to connect with prospective clients, they are now hoping to make the shift to a more focused strategy by working with a social media platform. In order to position accurate digital advertisements on the user pages of these targeted customers, this move necessitates the creation of predictive algorithms to identify clients with a high propensity to purchase tickets. Furthermore, distinct models need to be created for laptops and mobile devices in order to ensure accuracy because purchase behavior varies across different login devices. The company wants to reduce expenses while maximizing the impact of digital advertising. Predictive models will be created to find potential clients that have a high chance of buying tickets in order to accomplish this. The ultimate goal in the field of digital marketing is to maximize the effectiveness of advertising while keeping expenditures under control.

1.2 Problem Statement

An aviation company that provides domestic as well as international trips to the customers now wants to apply a targeted approach instead of reaching out to each of the customers. This time they want to do it digitally instead of tele calling. Hence they have collaborated with a social networking platform, so they can learn the digital and social behaviour of the customers and provide the digital advertisement on the user page of the targeted customers who have a high propensity to take up the product.

Propensity of buying tickets is different for different login devices. Hence, you have to create 2 models separately for Laptop and Mobile. [Anything which is not a laptop can be considered as mobile phone usage.]

The advertisements on the digital platform are a bit expensive; hence, you need to be very accurate while creating the models. Objective of Study

The principal objective of the project is to leverage digital and social behavior data obtained from the partnership with the social media platform to acquire more pro-

found understanding of consumer inclinations and actions, hence enabling more customized marketing strategies.

- The predictive model can accurately forecast whether a user plans to purchase a ticket during the next month by using the dataset as a guide.
- The number of likes a user has posted overall on their out-of-station check-ins over the course of the previous year is estimated by total likes received on the out-of-town check-in.
- Examine the user's purchasing patterns to ascertain whether or not he is interested in the ticket. This helps us understand why the user is interested in going on the trip.
- Predict the user's inclination for out-of-town travel based on their previous travel experiences.

1.3 Company and Industry Overview

The aviation industry is a critical component of the global economy, facilitating the movement of people and goods across vast distances. It plays a pivotal role in tourism, trade, and economic development. The industry is characterized by intense competition, fluctuating fuel prices, regulatory challenges, and the need for continuous innovation to meet changing customer expectations. In recent years, the industry has seen significant advancements in technology, particularly in digital marketing and customer engagement strategies. Airlines are increasingly leveraging data analytics, social media, and digital advertising to enhance customer outreach and improve operational efficiency.

1.4 Implementation and Expected Outcomes of Company

By integrating sophisticated data analytics and techniques, Improve Marketing Efficiency: Reduce wastage of advertising spend by targeting only high-propensity customers.

Enhance Customer Engagement: Deliver personalized content that resonates with users travel preferences and behaviors. Increase Ticket Sales: Boost conversion rates by focusing on users most likely to purchase tickets.

Gain Competitive Advantage: Stay ahead in a competitive industry by leveraging advanced digital marketing strategies. This targeted digital advertising initiative represents a significant step towards modernizing Airlines' marketing approach, aligning with industry trends, and meeting the evolving needs of its customers.

1.5 Overview of Theoretical Concepts

To effectively implement the targeted digital advertising strategy for Airlines, several theoretical concepts and principles from various fields, including marketing, data science, and consumer behavior, must be understood and applied. This overview highlights key concepts that underpin the development and deployment of predictive models for personalized marketing.

1.5.1 Customer Segmentation

Customer segmentation involves dividing a broad consumer or business market, normally consisting of existing and potential customers, into sub-groups of consumers based on some type of shared characteristics. By understanding these segments, Airlines can tailor marketing efforts to meet the specific needs and preferences of different groups, enhancing the effectiveness of targeted advertisements.

1.5.2 Propensity Modeling

Propensity modeling is a statistical approach used to predict the likelihood of a certain event occurring, such as a customer purchasing a ticket. This technique uses historical data to identify patterns and correlations that indicate a higher propensity for specific behaviors. For Airlines, separate propensity models for laptop and mobile users will help predict which users are most likely to buy tickets, allowing for more precise targeting.

1.5.3 Digital Marketing

Digital marketing encompasses all marketing efforts that use an electronic device or the internet. Leveraging digital channels such as search engines, social media, email, and other websites, businesses can connect with current and prospective customers. For Airlines, this means using the social networking platform to reach customers with personalized advertisements based on their digital and social behavior.

1.5.4 Feature Engineering

Feature engineering involves the process of using domain knowledge to create new features (variables) that make machine learning algorithms work better. This can include combining existing features, creating new variables, or transforming data into a more useful format. Effective feature engineering is crucial for improving the accuracy and performance of predictive models for ticket purchasing behavior.

1.5.5 Machine Learning

Machine learning is a subset of artificial intelligence (AI) that involves the use of algorithms and statistical models to perform tasks without using explicit instructions, relying on patterns and inference instead. In the context of Airlines, machine

learning algorithms will analyze user data to build predictive models, identifying the likelihood of ticket purchases based on various factors.

1.5.6 Evaluation Metrics

To assess the performance of the predictive models, several evaluation metrics are used, including accuracy, precision, recall, F1-score, and ROC(receiver operating characteristic curve). These metrics help determine how well the model is performing in terms of correctly identifying users who are likely to purchase tickets and minimizing false positives and false negatives.

1.5.7 Behavioral Targeting

Behavioral targeting is a technique used in digital marketing to increase the effectiveness of campaigns by using user behavior information to tailor advertisements. By understanding users browsing habits, social media activity, and other online behaviors, Airlines can deliver more relevant advertisements to users with a higher propensity to buy tickets.

2

Research Methodology

2.1 Scope of the Study

This study aims to investigate the effectiveness of using precision propensity models to target marketing campaigns on social media platforms for the aviation tourism industry and seeks to provide a comprehensive analysis of how social media can be leveraged to enhance market targeting and improve business strategies within the aviation tourism industry.

2.2 Methodology

The methodology for this study involves a structured approach combining data collection, preprocessing, model development, and evaluation. The key steps are outlined below:

2.2.1 Data Collection

The data collection process involved collaborating with a social networking platform to acquire user data, including behavioral and engagement metrics. The data acquisition strategy encompassed monitoring user behavior across various digital channels, such as social media, corporate channels, and travel-related websites. The dataset comprises the following variables:

- **UserID:** Unique identifier for each user.
- **Buy_ticket:** Indicates whether the user will buy a ticket in the next month (target variable).
- **Yearly_avg_view_on_travel_page:** Average yearly views on travel-related pages.
- **preferred_device:** Device used for logging in (Laptop or Mobile).
- **total_likes_on_outstation_checkin_given:** Total likes given by the user on outstation check-ins.
- **yearly_avg_Outstation_checkins:** Average number of outstation check-ins done by the user.

- **member_in_family:** Total number of family members mentioned in the user's account.
- **preferred_location_type:** Preferred type of travel location.
- **Yearly_avg_comment_on_travel_page:** Average yearly comments on travel-related pages.
- **total_likes_on_outofstation_checkin_received:** Total likes received by the user on outstation check-ins.
- **week_since_last_outstation_checkin:** Weeks since the last outstation check-in.
- **following_company_page:** Whether the user follows the company's page (Yes or No).
- **montly_avg_comment_on_company_page:** Average monthly comments on the company's page.
- **working_flag:** Whether the user is working or not.
- **travelling_network_rating:** Rating indicating if the user has close friends who like traveling (1 highest, 4 lowest).
- **Adult_flag:** Whether the user is an adult.
- **Daily_Avg_mins_spend_on_traveling_page:** Average daily time spent on the company's page.

The dataset exhibits a longitudinal collection strategy, with variables such as counts, yearly averages, and monthly averages, indicating different frequencies of data collection. Attributes like 'preferred_device' and 'following_company_page' may have originated from platform settings or user preferences, while categorical indicators like 'preferred_location_type' and 'working_flag' could have been gathered from surveys or user profiles. Numerical metrics like 'total_likes_on_outstation_checkin_given' and 'total_likes_on_outofstation_checkin_received' were likely derived from social media interactions.

The data collection process aimed to capture a comprehensive view of user behavior and engagement across various digital channels, enabling the development of propensity models for targeted digital advertising campaigns.

First few rows of the DataFrame:

	UserID	Taken_product	Yearly_avg_view_on_travel_page	preferred_device	total_likes_on_outstation_checkin_given	yearly_avg_Outstation_checkins	member_in_family	preferred_location_type
0	1000001	Yes	307.0	iOS and Android	38570.0	1	2	Financial
1	1000002	No	367.0	iOS	9765.0	1	1	Financial
2	1000003	Yes	277.0	iOS and Android	48055.0	1	2	Other
3	1000004	No	247.0	iOS	48720.0	1	4	Financial
4	1000005	No	202.0	iOS and Android	20685.0	1	1	Medical

Figure 2.1: Sample of data displaying first few rows and attributes

```
Shape of the DataFrame:
(11760, 17)
```

Figure 2.2: Shape of the dataset

	count	mean	std	min	25%	50%	75%	max
Yearly_avg_view_on_travel_page	11179.0	280.830844	68.182958	35.0	232.00	271.0	324.00	464.0
total_likes_on_outstation_checkin_given	11379.0	28170.481765	14385.032134	3570.0	16380.00	28076.0	40525.00	252430.0
Yearly_avg_comment_on_travel_page	11554.0	74.790029	24.026650	3.0	57.00	75.0	92.00	815.0
total_likes_on_outofstation_checkin_received	11760.0	6531.699065	4706.613785	1009.0	2940.75	4948.0	8393.25	20065.0
week_since_last_outstation_checkin	11760.0	3.203571	2.616365	0.0	1.00	3.0	5.00	11.0
montly_avg_comment_on_company_page	11760.0	28.661565	48.660504	11.0	17.00	22.0	27.00	500.0
travelling_network_rating	11760.0	2.712245	1.080887	1.0	2.00	3.0	4.00	4.0
Adult_flag	11760.0	0.793878	0.851823	0.0	0.00	1.0	1.00	3.0
Daily_Avg_mins_spend_on_traveling_page	11760.0	13.817432	9.070657	0.0	8.00	12.0	18.00	270.0

Figure 2.3: Summary of Numerical Attributes in the Dataset

2.2.2 Data Cleaning

Our first step in data cleaning involved dropping columns that were completely empty, ensuring we were working with meaningful data. While it's common practice to handle missing values by replacing them with the mean or median, we chose to drop these entries entirely to avoid introducing potential noise into our ML model. Next, we tackled inconsistencies in the 'member_in_family' column by replacing 'Three' with '3', reflecting the number of members in the family. We then converted this column to integers for consistency. Furthermore, we addressed anomalies in the 'yearly_avg_Outstation_checkins' column by filtering out entries with '*' values.

```
df_cleaned = df.dropna()
df_cleaned.reset_index(drop=True, inplace=True)

# Replace 'three' with '3'
df_cleaned['member_in_family'] = df_cleaned['member_in_family'].
    replace('Three', 3).astype(int)
df_cleaned = df_cleaned[df_cleaned['yearly_avg_Outstation_checkins'
    ] != '*']
df_cleaned.head()
```

Listing 2.1: Code for Data cleaning

```
Information about the DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11760 entries, 0 to 11759
Data columns (total 17 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   UserID                                      11760 non-null  int64
 1   Taken_product                               11760 non-null  object
 2   Yearly_avg_view_on_travel_page              11179 non-null  float64
 3   preferred_device                            11707 non-null  object
 4   total_likes_on_outstation_checkin_given     11379 non-null  float64
 5   yearly_avg_Outstation_checkins              11685 non-null  object
 6   member_in_family                            11760 non-null  object
 7   preferred_location_type                     11729 non-null  object
 8   Yearly_avg_comment_on_travel_page           11554 non-null  float64
 9   total_likes_on_outofstation_checkin_received 11760 non-null  int64
 10  week_since_last_outstation_checkin          11760 non-null  int64
 11  following_company_page                      11657 non-null  object
 12  montly_avg_comment_on_company_page          11760 non-null  int64
 13  working_flag                                11760 non-null  object
 14  travelling_network_rating                   11760 non-null  int64
 15  Adult_flag                                  11760 non-null  int64
 16  Daily_Avg_mins_spend_on_traveling_page      11760 non-null  int64
dtypes: float64(3), int64(7), object(7)
memory usage: 1.5+ MB
```

Figure 2.4: Info of dataset before cleaning

```
<class 'pandas.core.frame.DataFrame'>
Index: 10455 entries, 0 to 10455
Data columns (total 17 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   UserID                                      10455 non-null  int64
 1   Taken_product                               10455 non-null  object
 2   Yearly_avg_view_on_travel_page              10455 non-null  float64
 3   preferred_device                            10455 non-null  object
 4   total_likes_on_outstation_checkin_given     10455 non-null  float64
 5   yearly_avg_Outstation_checkins              10455 non-null  object
 6   member_in_family                            10455 non-null  int64
 7   preferred_location_type                     10455 non-null  object
 8   Yearly_avg_comment_on_travel_page           10455 non-null  float64
 9   total_likes_on_outofstation_checkin_received 10455 non-null  int64
 10  week_since_last_outstation_checkin          10455 non-null  int64
 11  following_company_page                      10455 non-null  object
 12  montly_avg_comment_on_company_page          10455 non-null  int64
 13  working_flag                                10455 non-null  object
 14  travelling_network_rating                   10455 non-null  int64
 15  Adult_flag                                  10455 non-null  int64
 16  Daily_Avg_mins_spend_on_traveling_page      10455 non-null  int64
dtypes: float64(3), int64(8), object(6)
memory usage: 1.4+ MB
```

Figure 2.5: Info of dataset after cleaning

2.2.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyze and summarize the main characteristics of a dataset. It involves techniques and methods to gain insights into the data, understand its structure, detect patterns, identify relationships between variables, and prepare for further analysis. EDA helps in understanding the data better, uncovering patterns, detecting anomalies, testing hypotheses, and checking assumptions with the help of summary statistics and graphical representations. The three different EDA techniques are Univariate, Bivariate, and Multivariate.

- **Univariate analysis:** Univariate analysis is a statistical method used to analyze and describe the characteristics of a single variable without considering the relationship with other variables. It focuses on understanding the distribution, central tendency, variability, and shape of the data for that particular variable.
- **Bivariate analysis:** Bivariate analysis focuses on exploring relationships between two variables in a dataset. It helps in understanding how one variable (independent variable) is related to another variable (dependent variable) or how two variables are related to each other. By conducting a bi-variate analysis, you gain insights into how one attribute of data changes with respect to other attributes of data, helping to understand which attributes are more important while creating an ML model.
- **Multivariate analysis:** Multivariate analysis involves analyzing the relationships among three or more variables simultaneously. It helps in understanding the complex interactions and patterns that exist within the data, taking into account multiple factors and their combined effects. Multivariate analysis techniques include regression analysis, factor analysis, cluster analysis, and discriminant analysis, etc.

We apply these EDA techniques to uncover insights and prepare the data for further modeling and analysis. The detailed analysis of the data can be seen in Chapter 2.3.

2.2.4 Data Preprocessing

Preprocessing is a crucial step in the machine learning pipeline as it ensures that the data is clean, well-formatted, and suitable for analysis. Below are the steps performed on our dataset.

2.2.4.1 Normalization of Numerical Attributes

To standardize the range of the continuous features, we applied the MinMaxScaler from the scikit-learn library. MinMaxScaler transforms features by scaling each feature to a given range, usually between 0 and 1. This transformation is essential because working with larger values requires more computational resources, and normalizing the data helps make computations more efficient.

This normalization ensures that each feature contributes equally to the analysis and helps in speeding up the convergence of algorithms.

```
columns_to_normalize = ['Yearly_avg_view_on_travel_page',
                        'total_likes_on_outstation_checkin_given',
                        'yearly_avg_Outstation_checkins',
                        'member_in_family',
                        'Yearly_avg_comment_on_travel_page',
                        '
   total_likes_on_outofstation_checkin_received',
                        'week_since_last_outstation_checkin',
                        'montly_avg_comment_on_company_page',
                        'travelling_network_rating',
                        'number_of_adults_in_family',
                        'Daily_Avg_mins_spend_on_traveling_page']

# Using MinMaxScaler to normalize between 0 and 1
min_max_scaler = MinMaxScaler()
df_cleaned[columns_to_normalize] = min_max_scaler.fit_transform(
   df_cleaned[columns_to_normalize])
```

Listing 2.2: Code for Normalization of Numerical Attributes

2.2.4.2 Handling Categorical Variable

There are several categorical features in the dataset. These features need to be converted into numerical representations for the machine learning algorithms to process.

- **Simplification of Categorical Values:** For the preferred_device attribute, we simplified the categories by merging all values except 'Laptop' into a single category 'Mobile'. This step helps in reducing the complexity of the feature.
- **Label Encoding:** We used the LabelEncoder from scikit-learn to convert the categorical attributes into numerical values. Label encoding assigns a unique integer to each category.

```
# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Apply Label Encoding to the categorical columns
df_cleaned['preferred_device_encoded'] = label_encoder.
   fit_transform(df_cleaned['preferred_device'])
df_cleaned['preferred_location_type_encoded'] = label_encoder.
   fit_transform(df_cleaned['preferred_location_type'])
df_cleaned['following_company_page_encoded'] = label_encoder.
   fit_transform(df_cleaned['following_company_page'])
df_cleaned['working_flag_encoded'] = label_encoder.fit_transform(
   df_cleaned['working_flag'])
df_cleaned['Taken_product_encoded'] = label_encoder.fit_transform(
   df_cleaned['Taken_product'])
```

Listing 2.3: Code for encoding categorical attributes into numerical values using LabelEncoder

2.2.5 Model Development

Predicting whether a user will purchase a product based on their social media activities is a binary classification problem. Given a labeled dataset, we can employ various machine learning models [1] to construct a better classification model. These models include:

- **Logistic Regression:**Logistic regression is a statistical model that is widely applied to binary classification problems. It calculates the likelihood that a specific input is a member of a particular class. Airlines can utilize logistic regression to forecast, based on digital behavior and other characteristics, the binary outcome of a user's decision to purchase a ticket or not.
- **Decision Trees:** Decision Tree is a supervised machine learning technique that can do both classification and regression problems. It generates a tree-like structure of decisions based on data attributes, allowing the algorithm to predict using a succession of if-else conditions.
- **Random Forest:** Random Forest is an ensemble learning method that uses numerous decision trees during training to output the mode of the classes for classification tasks. It is renowned for its predictability and accuracy in prediction tasks. This technique can be used to develop propensity models for laptop and mobile users, assisting in the very accurate prediction of ticket buying behavior.
- **Gradient Boosting:**
- Gradient Boosting is an ensemble learning method that builds a strong predictive model by combining several weak models (like decision trees). By repeatedly training new models on the residual errors of the prior models, progressively increasing overall performance is achieved. The excellent accuracy and versatility of gradient boosting models allow them to manage intricate, non-linear relationships in data.
- **Support Vector Machines (SVM):** Support Vector Machines (SVMs) are supervised learning models used for classification and regression. SVMs create a hyperplane or group of hyperplanes in a high-dimensional space to distinguish between data points of various classes.
- **Naive Bayes:** Naive Bayes is a family of probabilistic classifiers based on Bayes' theorem that assumes feature independence. Naive Bayes models frequently work well in practice, especially for text classification applications, despite the naive assumption. They need comparatively minimal training data and are quick and easy to use.

- **AdaBoost:** Another ensemble learning method called AdaBoost (Adaptive Boosting) combines several weak classifiers to produce a strong classifier. It operates by repeatedly training new models on the incorrectly classified examples from the prior models, assigning greater weight to the cases that are challenging to categorize. AdaBoost is well-known for improving the performance of weak classifiers.
- **KNN (K-Nearest Neighbors):** K-Nearest Neighbors is a non-parametric technique that may be applied to both classification and regression tasks. It classifies a new data point using the majority class of its k nearest neighbors from the training data. Although KNN is an easy-to-understand technique, it can be computationally costly for large datasets and is dependent on the distance measure and k value that are chosen.

While various algorithms were initially mentioned, given the dataset's multiple features, we can reasonably assume that decision tree classifier or ensemble methods, such as Random Forest, Gradient Boosting, and AdaBoost, are likely to be the best fit for this classification task. We can exclude particular algorithms and train only a selection of models based on parameters such as the dataset's structure and size.

Naive Bayes classifiers and K-Nearest Neighbors (KNN) are two methods that are not considered appropriate for this purpose. Naive Bayes assumes that characteristics are conditionally independent based on the class label, which may not be true in real-world datasets with strongly correlated features. Furthermore, KNN can be computationally expensive with large datasets because it must calculate distances to all training points for each prediction, and it may struggle with categorical features if not properly encoded.

Given these considerations, Naive Bayes and KNN have been excluded from further consideration. The remaining ML models will be trained and evaluated using cross-validation, to determine the model most suitable for our task. The Detailed steps for training the ML models and selection of ML models suitable for our task can be seen in Section3.3.

Cross-validation: Cross-validation [2] is a fundamental technique in machine learning for evaluating model performance and generalizability. It involves partitioning the dataset into multiple subsets, training the model on a portion of these subsets, and then assessing its performance on the remaining subsets. A high cross-validation score signifies that the model consistently performs well across different data subsets, indicating its reliability and ability to generalize effectively. This suggests that the model can capture underlying patterns in the data without overfitting.

2.2.6 Model Evaluation

Evaluating the performance of the predictive models is crucial to ensure their reliability and effectiveness. Various evaluation metrics [3] can be used to assess model performance, including:

Accuracy: The ratio of correctly predicted instances to the total instances.

Precision and Recall: Metrics that provide insight into the model's performance in identifying positive instances correctly.

F1-Score: The harmonic mean of precision and recall, gives a single metric to evaluate the model's performance.

ROC Curve: The Receiver Operating Characteristic curve, plots the true positive rate against the false positive rate, providing a comprehensive view of the model's performance across different thresholds.

The detailed evaluation of the model's test output using the above-mentioned metrics can be seen in Section 3.4.

2.3 Utility of Research

The research conducted in this study holds significant utility for the aviation company in several aspects:

First of all, it will offer a thorough examination of consumer behavior, preferences, and degrees of engagement with regard to travel and tourism-related activities. Through an understanding of variables like preferred location types, connections with others, and interactions on travel-related pages, the company may customize its products and marketing strategies to better meet the demands and preferences of its customers. This information is critical for creating targeted ads and customized vacation packages that are beneficial to different customer categories.

Second, the research highlights how crucial it is to use digital and social media channels for efficient marketing and acquiring customers. By employing data-driven methods and utilizing social connections, the organization may reach a larger audience and increase ticket sales. The examination of user engagement indicators, such as likes, comments, and time spent on travel sites, can assist the organization in identifying areas for improvement and developing strategies to improve customer engagement and loyalty.

Furthermore, the business may be able to maximize its marketing initiatives and resource allocation through the creation and assessment of machine learning models for predicting the tendency of customers to purchase tickets. Data on user demographics, device preferences, and work status can be used to segment the customer base and provide customized advertisements that appeal to particular target audiences. In the fast-paced travel sector, this focused strategy can result in higher productivity, profitability, and a competitive advantage.

3

Data Analysis and Interpretation

3.1 Univariate Analysis

- **Taken_product**

  The histogram explaining how many users are buying the ticket in the next month whereas the `Taken_product` is considered on the X-axis and the number of users is on the Y-axis can be seen in Figure 3.1.

```
counts = df_cleaned['Taken_product'].value_counts()
total_users = counts.sum()
# Plot the bar plot
plt.figure(figsize=(10, 6))
counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.xlabel('Taken_product')
plt.ylabel('Number of Users')
plt.title('Distribution of Taken_product')
plt.grid(axis='y', linestyle='--', alpha=0.7)
for i, count in enumerate(counts):
    percentage = '{:.2f}%'.format(100 * count / total_users)
    plt.text(i, count, percentage, ha='center', va='bottom')
plt.show()
```

Listing 3.1: Code for UniVariable analysis Taken_Product

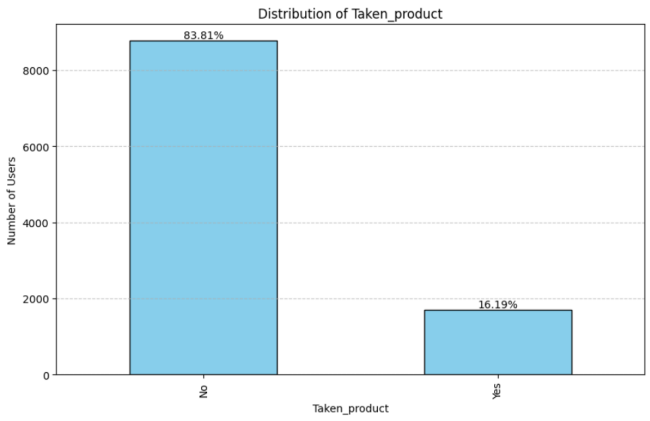

Figure 3.1: Bar graph representing the Taken_product vs Number of Users

The `Taken_product` attribute represents whether users have purchased a ticket for the next month. The histogram in Figure 3.1 shows that over 80% of users have not purchased a ticket, indicating that the dataset is unbalanced.

- **Yearly_avg_view_on_travel_page**

  The histogram provides a visual representation of the distribution of 'Yearly_avg_view_on_travel_page' values and the annotations on the bars indicate the percentage of users in each bin, which can be seen in Figure 3.2.

```
plt.figure(figsize=(10, 6))
hist, bins, _ = plt.hist(df_cleaned['
    Yearly_avg_view_on_travel_page'], bins=20, color='skyblue',
     edgecolor='black', density=False)
plt.xlabel('Yearly_avg_view_on_travel_page')
plt.ylabel('Number of Users')
plt.title('Distribution of Yearly_avg_view_on_travel_page')
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Add count and percentage labels on top of each bar
total_users = len(df_cleaned['Yearly_avg_view_on_travel_page'
    ])
for i in range(len(bins) - 1):
    count = int(hist[i])
    percentage = (count / total_users) * 100
    plt.text(bins[i] + (bins[i+1] - bins[i]) / 2, hist[i], f"{
    percentage:.1f}%", ha='center', va='bottom', rotation=25)

plt.show()
```

Listing 3.2: Code for UniVariable analysis Yearly_avg_view_on_travel_page

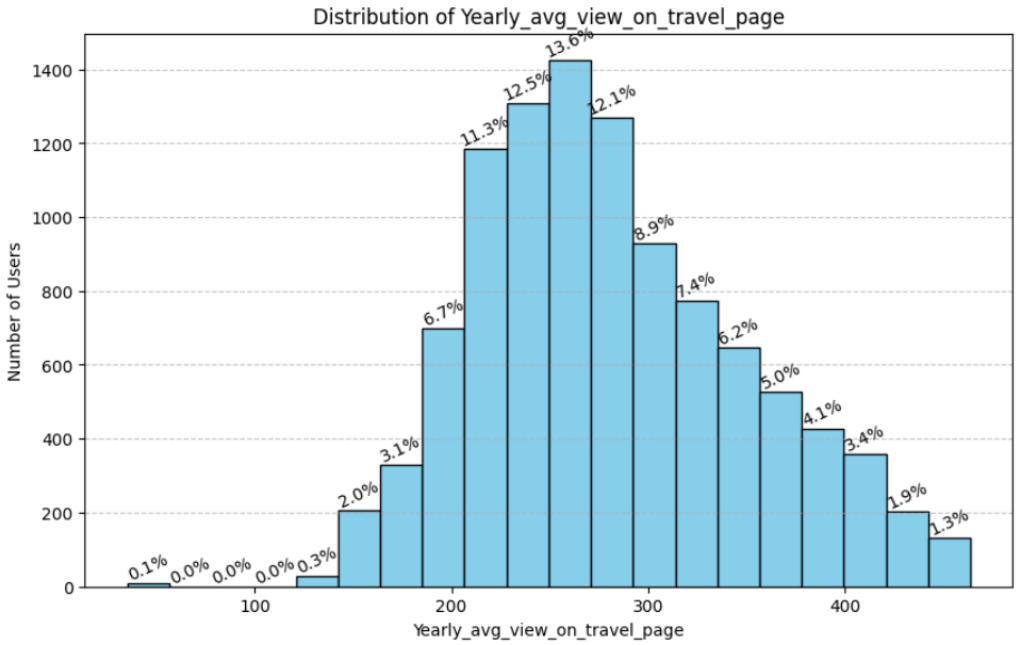

Figure 3.2: Bar graph representing the Yearly_avg_view_on_travel_page vs Number of Users

- **preferred_device**

  The table summarizes the devices users preferred to log in with (iOS, Android, Laptop, Tablet). Based on the data, it is clear that iOS and Android are the most preferred devices. This can be seen in Table 3.1.

```
# Count the occurrences of each device type
device_counts = Counter(df_cleaned['preferred_device'])

# Convert the device_counts dictionary into a DataFrame
df_device_counts = pd.DataFrame.from_dict(device_counts,
    orient='index', columns=['Count'])

# Calculate the percentage for each device type
df_device_counts['Percentage'] = (df_device_counts['Count'] /
    df_device_counts['Count'].sum()) * 100

# Print the DataFrame
df_device_counts
```

Listing 3.3: Code for UniVariable analysis preferred_device

	Count	Percentage
iOS and Android	3246	31.047346
iOS	859	8.216165
ANDROID	97	0.927786
Android	244	2.333812
Android OS	126	1.205165
Other	2	0.019130
Others	1	0.009565
Tab	4172	39.904352
Laptop	1108	10.597800
Mobile	600	5.738881

Table 3.1: Summary of Users' Preferred Devices for Logging In

In the problem statement, we have asked to consider all the devices other than the laptop. The following is the distribution of laptops and mobile phones in Table 3.2.

```
# Subtract the count of 'laptop' from the total count to
    obtain the count of 'mobile'
total_count = sum(device_counts.values())
```

```
laptop_count = device_counts.get('Laptop', 0)
mobile_count = total_count - laptop_count

# Create a DataFrame with the device type, count, and
    percentage
df_device_counts = pd.DataFrame({'Device Type': ['Laptop', '
    Mobile'],
                                 'Count': [laptop_count,
    mobile_count],
                                 'Percentage': [laptop_count /
     total_count * 100, mobile_count / total_count * 100]})

# Print the DataFrame
print(df_device_counts)
```

Listing 3.4: Code for outlier treatment of UniVariable analysis preferred_device

Device Type	Count	Percentage
Laptop	1108	10.5978
Mobile	9347	89.4022

Table 3.2: Summary of Users' Preferred Devices for Logging In, Considering All Devices Except Laptops as Mobile

Table 3.2 indicates that the majority of users prefer logging in with mobile devices, particularly iOS and Android, as opposed to laptops. This highlights the importance of optimizing the user experience for mobile platforms, which collectively account for approximately 89.4% of logins. Investing in mobile-friendly features and ensuring seamless access across iOS and Android devices will likely enhance user engagement and satisfaction.

- **total_likes_on_outstation_checkin_given**

  The histogram explaining total_likes_on _outstation_checkin_given by users can be seen in Figure 3.3.

```
plt.figure(figsize=(10, 6))
hist, bins, _ = plt.hist(df_cleaned['
    total_likes_on_outstation_checkin_given'], bins=20, color='
    skyblue', edgecolor='black', density=False)
plt.xlabel('total_likes_on_outstation_checkin_given')
plt.ylabel('Number of Users')
plt.title('Distribution of
    total_likes_on_outstation_checkin_given')
plt.grid(axis='y', linestyle='--', alpha=0.7)
total_users = len(df_cleaned['Yearly_avg_view_on_travel_page'
    ])
for i in range(len(bins) - 1):
     count = int(hist[i])
     percentage = (count / total_users) * 100
     plt.text(bins[i] + (bins[i+1] - bins[i]) / 2, hist[i], f"{
    percentage:.2f}%", ha='center', va='bottom', rotation=70)
plt.show()
```

Listing 3.5: Code for UniVariable analysis total_likes_on_outstation _checkin_given

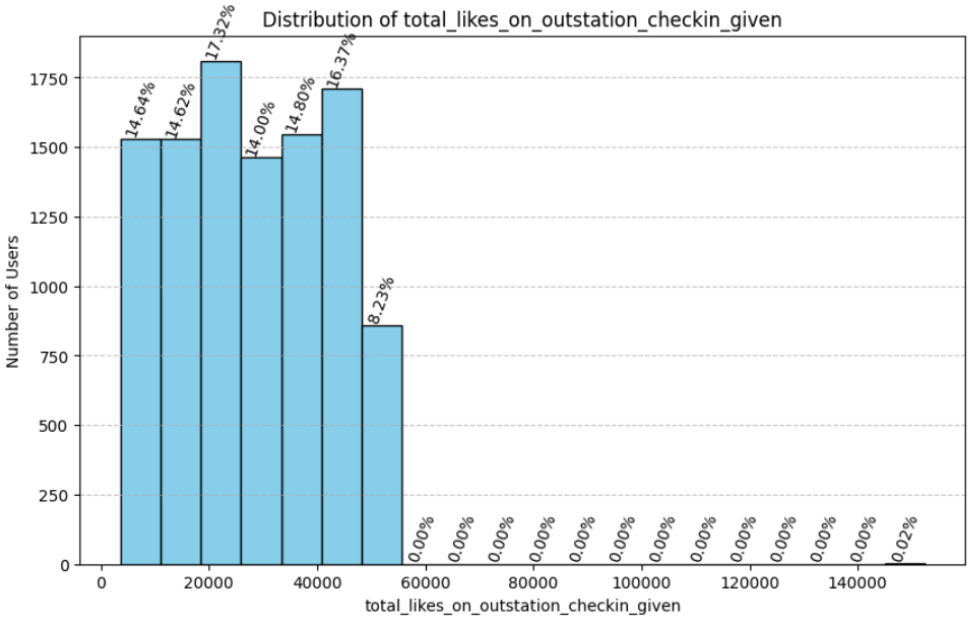

Figure 3.3: Bar graph representing the total _likes_on_outstation_checkin_given vs Number of Users

Very few users have the number of total_likes_on_outstation_checkin_given are more than 56000. So we are treating a value of more than 56000 as 56000. The histogram with outliers treated can be seen in figure 3.4.

```
# Define the condition for handling outliers
condition =df_cleaned['total_likes_on_outstation_checkin_given
    '] > 56000
# Replace values based on the condition
df_cleaned.loc[condition, '
    total_likes_on_outstation_checkin_given'] = 56000
plt.figure(figsize=(10, 6))
hist, bins, _ = plt.hist(df_cleaned['
    total_likes_on_outstation_checkin_given'], bins=20, color='
    skyblue', edgecolor='black', density=False)
plt.xlabel('total_likes_on_outstation_checkin_given')
plt.ylabel('Number of Users')
plt.title('Distribution of
    total_likes_on_outstation_checkin_given')
plt.grid(axis='y', linestyle='--', alpha=0.7)
total_users = len(df_cleaned['Yearly_avg_view_on_travel_page'
    ])
for i in range(len(bins) - 1):
     count = int(hist[i])
     percentage = (count / total_users) * 100
     plt.text(bins[i] + (bins[i+1] - bins[i]) / 2, hist[i], f"{
    percentage:.2f}%", ha='center', va='bottom')

plt.show()
```

Listing 3.6: Code for outlier treatment of UniVariable analysis total_likes_on_outstation _checkin_given

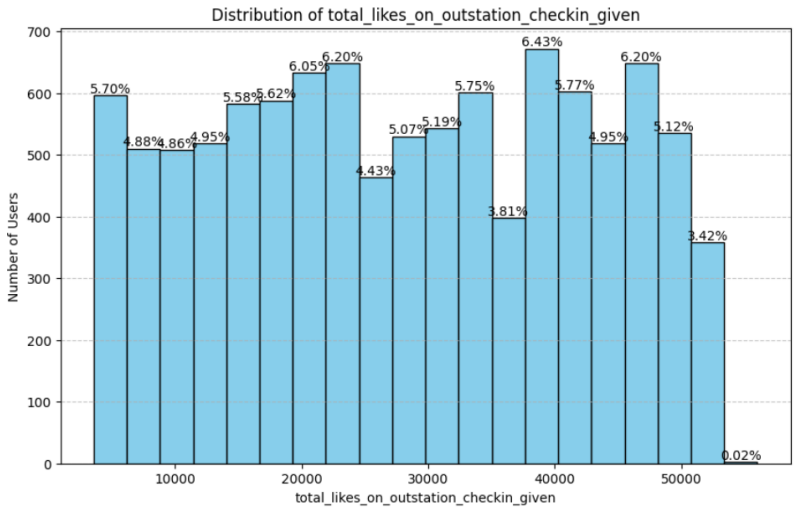

Figure 3.4: Bar graph representing the total _likes_on_outstation_checkin_given vs Number of Users with outlier treatment

- **yearly_avg_Outstation_checkins**

  The histogram illustrates the distribution of yearly average outstation check-ins among users. The x-axis represents the yearly_avg_Outstation_checkins, while the y-axis indicates the number of users can be seen in Figure 3.5. Each bar represents a range of yearly average outstation check-ins, and the percentage label at the top of each bar signifies the proportion of users falling within that range.

```
plt.figure(figsize=(10, 6))
hist, bins, _ = plt.hist(df_cleaned['
    yearly_avg_Outstation_checkins'], bins=20, color='skyblue',
     edgecolor='black', density=False)
plt.xlabel('yearly_avg_Outstation_checkins')
plt.ylabel('Number of Users')
plt.title('Distribution of yearly_avg_Outstation_checkins')
plt.grid(axis='y', linestyle='--', alpha=0.7)
total_users = len(df_cleaned['Yearly_avg_view_on_travel_page'
    ])
for i in range(len(bins) - 1):
    count = int(hist[i])
    percentage = (count / total_users) * 100
    plt.text(bins[i] + (bins[i+1] - bins[i]) / 2, hist[i], f"{
    percentage:.2f}%", ha='center', va='bottom', rotation=70)

plt.show()
```

Listing 3.7: Code for UniVariable analysis yearly_avg_Outstation_checkins

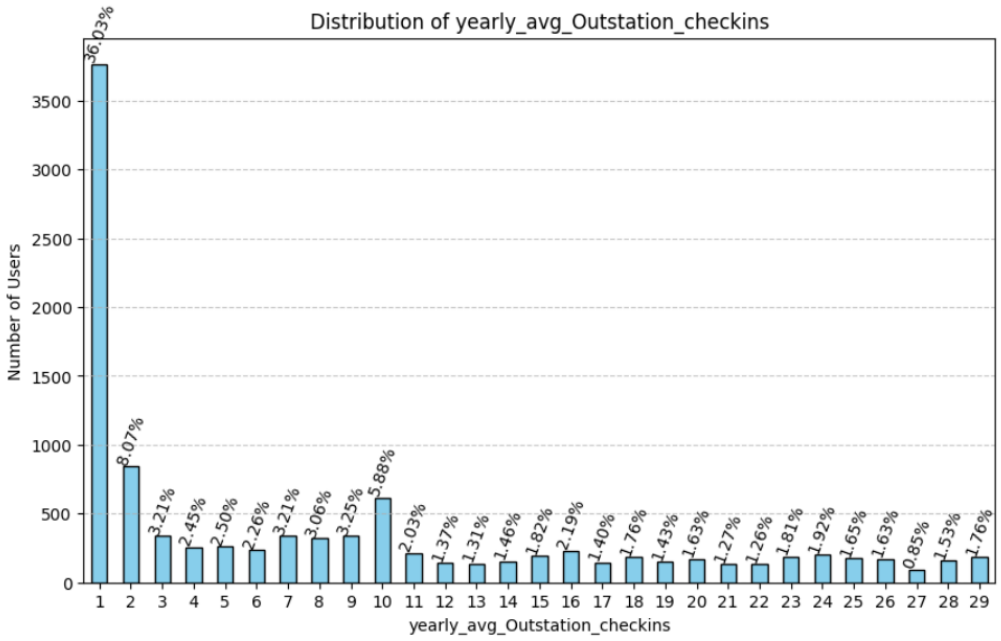

Figure 3.5: Bar graph representing the yearly_avg_Outstation_checkins vs Number of Users

Based on the histogram depicting the distribution of yearly average outstation check-ins among users, it is evident that a significant portion of the users engage in outstation check-ins. Notably, 37.8% of the users participate in at least one outstation check-in per year. Furthermore, the histogram reveals that there are no users with a yearly average outstation check-in count of zero, indicating that all users have engaged in some level of outstation travel. This insight underscores the active participation of users in outstation activities, which is a critical factor for businesses focusing on travel-related services and offerings.

- **member_in_family**

  The histogram illustrates the distribution of members in the family among users. The x-axis represents the member_in_family, while the y-axis indicates the number of users, as depicted in Figure 3.6. Each bar represents the Number of Members in the Family, and the percentage label at the top of each bar signifies the proportion of users having that number.

```
counts = df_cleaned['member_in_family'].value_counts().
    sort_index()
total_users = counts.sum()
# Plot the bar plot
plt.figure(figsize=(10, 6))
counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.xlabel('Number of Members in Family')
plt.ylabel('Number of Users')
plt.title('Distribution of Number of Members in Family')
plt.xticks(rotation=0)  # Rotate x-axis labels if needed
plt.grid(axis='y', linestyle='--', alpha=0.7)
for i, count in enumerate(counts):
    percentage = '{:.2f}%'.format(100 * count / total_users)
    plt.text(i, count, percentage, ha='center', va='bottom')

plt.show()
```

Listing 3.8: Code for UniVariable analysis member_in_family

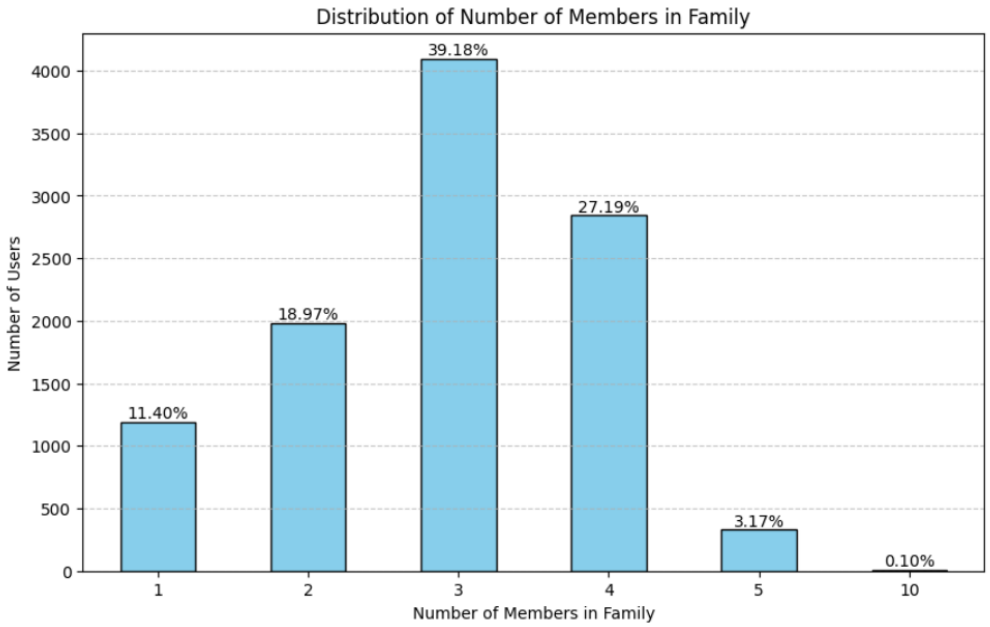

Figure 3.6: Bar graph representing the member_in_family vs Number of Users

Based on the histogram we can see that most of families have 3 members in family.

- **preferred_location_type**

  The histogram illustrates the distribution of preferred location types among users. The x-axis represents the preferred_location_type, while the y-axis indicates the number of users, as depicted in Figure 3.7. Each bar represents the preferred location type, and the percentage label atop each bar signifies the proportion of users who prefer that location.

```
counts = df_cleaned['preferred_location_type'].value_counts()
total_users = counts.sum()
# Plot the bar plot
plt.figure(figsize=(10, 6))
counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.xlabel('preferred_location_type')
plt.ylabel('Number of Users')
plt.title('Distribution of preferred_location_type')
plt.xticks(rotation=90)  # Rotate x-axis labels if needed
plt.grid(axis='y', linestyle='--', alpha=0.7)
for i, count in enumerate(counts):
    percentage = '{:.2f}%'.format(100 * count / total_users)
    plt.text(i, count, percentage, ha='center', va='bottom')

plt.show()
```

Listing 3.9: Code for UniVariable analysis preferred_location_type

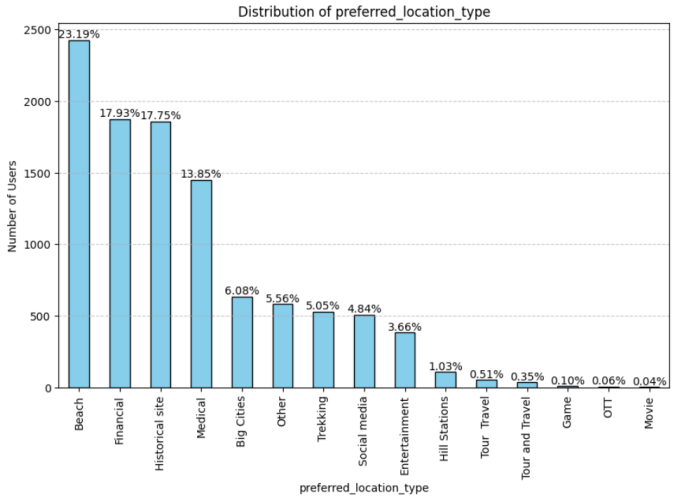

Figure 3.7: Bar graph representing the member_in_family vs Number of Users

Based on the histogram the top three preferred location types are beaches (23.19%), financial hubs (17.93%), and historical sites (17.75%). This information can guide the company in curating travel packages, promotional campaigns, and partnerships tailored to these popular destinations, catering to the preferences of a significant portion of its customer base. By focusing on these high-demand locations, the company can enhance customer satisfaction and potentially drive increased ticket sales.

- **Yearly_avg_comment_on_travel_page**

  The histogram illustrates the distribution of Yearly average comments on travel pages among users. The x-axis represents the Yearly_avg_comment_on_travel_page, while the y-axis indicates the number of users, as depicted in Figure 3.8. Each bar represents the range Yearly_avg_comment_on_travel_page, and the percentage label at the top of each bar signifies the proportion of users that fall in that range.

```
plt.figure(figsize=(10, 6))
hist, bins, _ = plt.hist(df_cleaned['
    Yearly_avg_comment_on_travel_page'], bins=20, color='
    skyblue', edgecolor='black', density=False)
plt.xlabel('Yearly_avg_comment_on_travel_page')
plt.ylabel('Number of Users')
plt.title('Distribution of Yearly_avg_comment_on_travel_page')
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Add count and percentage labels on top of each bar
total_users = len(df_cleaned['
    Yearly_avg_comment_on_travel_page'])
for i in range(len(bins) - 1):
     count = int(hist[i])
     percentage = (count / total_users) * 100
     plt.text(bins[i] + (bins[i+1] - bins[i]) / 2, hist[i], f"{
    percentage:.2f}%", ha='center', va='bottom', rotation=25)

plt.show()
```

Listing 3.10: Code for UniVariable analysis Yearly_avg_comment_on_travel_page

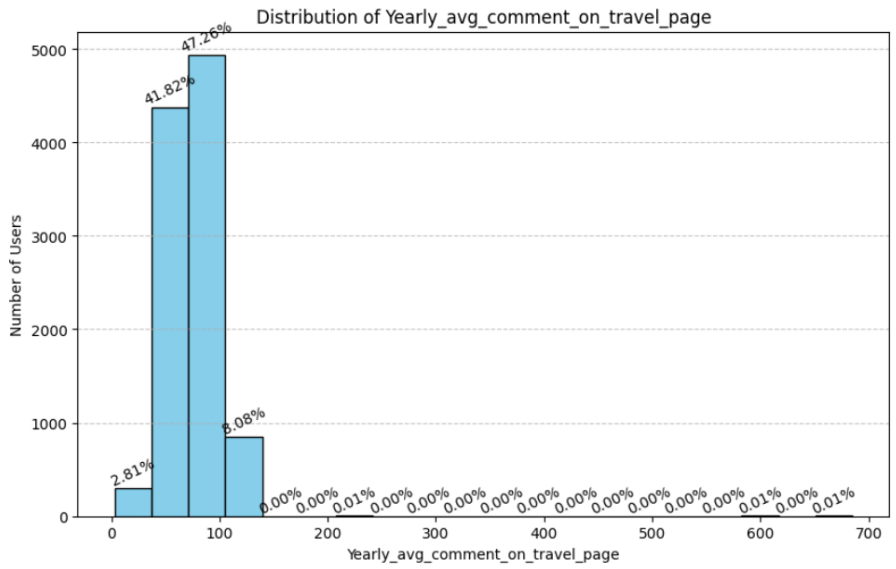

Figure 3.8: Bar graph representing the Yearly_avg_comment_on_travel_page vs Number of Users

Very few users have the number of Yearly_avg_comment_on_travel_page are more than 130. So we are treating a value of more than 130 as 130. The histogram with outliers treated can be seen in figure 3.9.

```
plt.figure(figsize=(10, 6))
hist, bins, _ = plt.hist(df_cleaned['
    Yearly_avg_comment_on_travel_page'], bins=20, color='
    skyblue', edgecolor='black', density=False)
plt.xlabel('Yearly_avg_comment_on_travel_page')
plt.ylabel('Number of Users')
plt.title('Distribution of Yearly_avg_comment_on_travel_page')
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Add count and percentage labels on top of each bar
total_users = len(df_cleaned['
    Yearly_avg_comment_on_travel_page'])
for i in range(len(bins) - 1):
     count = int(hist[i])
     percentage = (count / total_users) * 100
     plt.text(bins[i] + (bins[i+1] - bins[i]) / 2, hist[i], f"{
    percentage:.2f}%", ha='center', va='bottom', rotation=25)

plt.show()
```

Listing 3.11: Code for outlier treatment of UniVariable analysis Yearly_avg_comment_on_travel_page

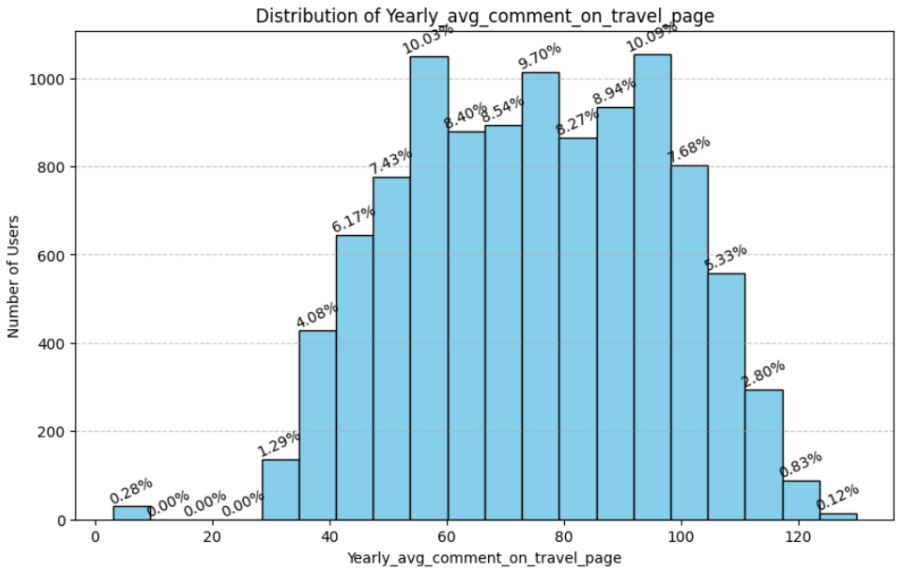

Figure 3.9: Bar graph representing the Yearly_avg_comment_on_travel_page vs Number of Users with outliers handled

- **total_likes_on_outofstation_checkin_received**

  The histogram illustrates the distribution of total likes on out-of-station check-ins received among users. The x-axis represents the total_likes_on_outofstation _checkin_received, while the y-axis indicates the number of users, as depicted in Figure 3.10. Each bar represents the range total_likes_on_outofstation_check in_received, and the percentage label at the top of each bar signifies the proportion of users that fall in that range.

```
plt.figure(figsize=(10, 6))
hist, bins, _ = plt.hist(df_cleaned['
    total_likes_on_outofstation_checkin_received'], bins=20,
    color='skyblue', edgecolor='black', density=False)
plt.xlabel('total_likes_on_outofstation_checkin_received')
plt.ylabel('Number of Users')
plt.title('Distribution of
    total_likes_on_outofstation_checkin_received')
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Add count and percentage labels on top of each bar
total_users = len(df_cleaned['
    total_likes_on_outofstation_checkin_received'])
for i in range(len(bins) - 1):
     count = int(hist[i])
     percentage = (count / total_users) * 100
     plt.text(bins[i] + (bins[i+1] - bins[i]) / 2, hist[i], f"{
    percentage:.2f}%", ha='center', va='bottom', rotation=25)

plt.show()
```

Listing 3.12: Code for UniVariable analysis total_likes_on_outofstation _checkin_received

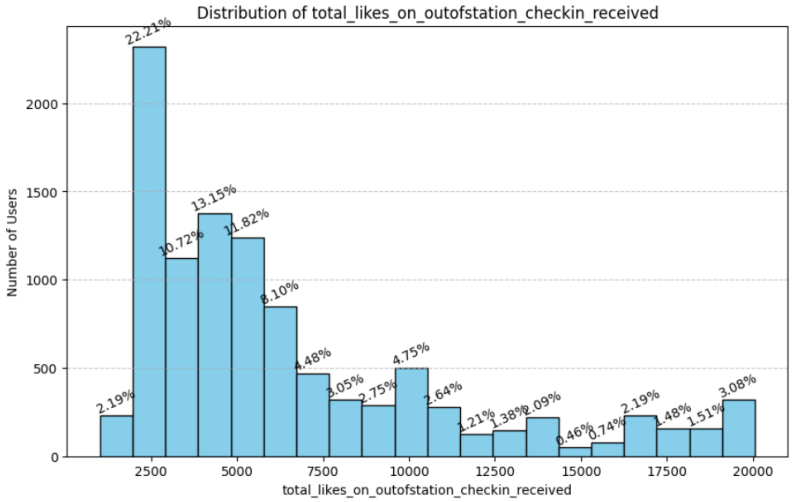

Figure 3.10: Bar graph representing the total_likes_on_outofstation_checkin_received vs Number of Users

- **week_since_last_outstation_checkin**

  The histogram illustrates the distribution of the week since the last outstation check-in among users. The x-axis represents the "week_since_last_outstation_checkin," while the y-axis indicates the number of users, as depicted in Figure 3.11. Each bar represents the week_since_last_outstation_checkin number, and the percentage label atop each bar signifies the proportion of users who prefer that location.

```
counts = df_cleaned['week_since_last_outstation_checkin'].
    value_counts().sort_index()
total_users = counts.sum()
# Plot the bar plot
plt.figure(figsize=(10, 6))
counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.xlabel('week_since_last_outstation_checkin')
plt.ylabel('Number of Users')
plt.title('Distribution of week_since_last_outstation_checkin'
    )
plt.xticks(rotation=0)  # Rotate x-axis labels if needed
plt.grid(axis='y', linestyle='--', alpha=0.7)
for i, count in enumerate(counts):
    percentage = '{:.2f}%'.format(100 * count / total_users)
    plt.text(i, count, percentage, ha='center', va='bottom')

plt.show()
```

Listing 3.13: Code for UniVariable analysis week_since_last _outstation_checkin

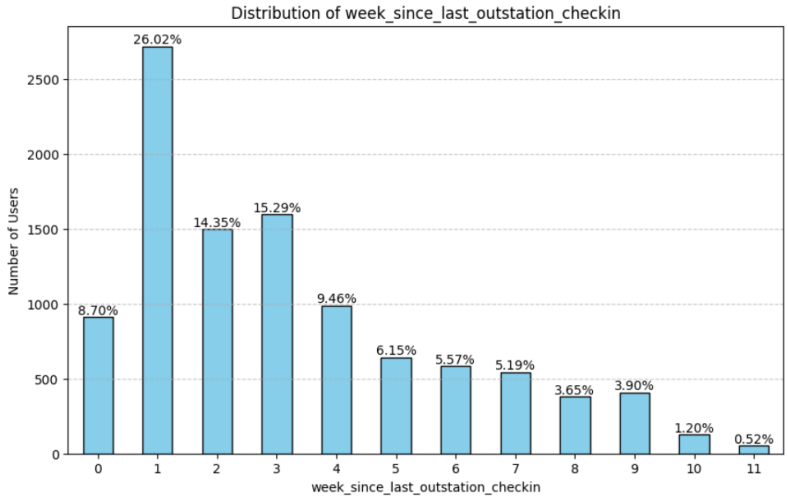

Figure 3.11: Bar graph representing the week_since_last_outstation_checkin vs Number of Users

Based on the 'week_since_last_outstation_checkin' histogram we can see that more than 60% of users have gone for outstations in the last three weeks. Users who went outstations recently may not be interested in going for outstations again so marketing the company packages to these users is not beneficial.

- **following_company_page**

  The histogram Figure 3.12 illustrates the distribution of users based on whether they follow company pages. The x-axis represents the attribute "following_company_page," indicating whether users follow company pages (yes or no). The y-axis represents the number of users. Each bar in the histogram corresponds to either "yes" or "no" for the following company pages. The percentage label at the top of each bar signifies the proportion of users in that category.

```
counts = df_cleaned['following_company_page'].value_counts()
total_users = counts.sum()
# Plot the bar plot
plt.figure(figsize=(10, 6))
counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.xlabel('following_company_page')
plt.ylabel('Number of Users')
plt.title('Distribution of following_company_page')
# plt.xticks(rotation=90)  # Rotate x-axis labels if needed
plt.grid(axis='y', linestyle='--', alpha=0.7)
for i, count in enumerate(counts):
    percentage = '{:.2f}%'.format(100 * count / total_users)
    plt.text(i, count, percentage, ha='center', va='bottom')

plt.show()
```

Listing 3.14: Code for UniVariable analysis following_company_page

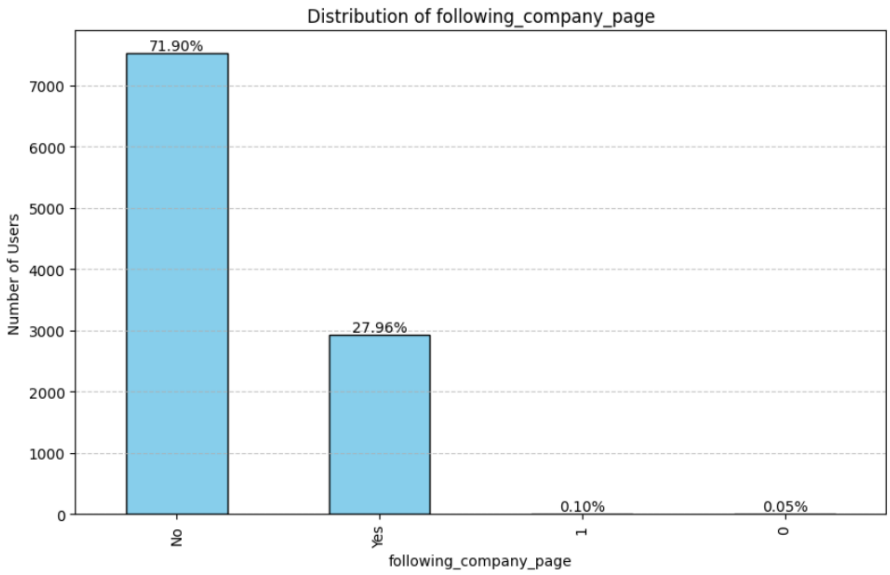

Figure 3.12: Bar graph representing the following_company_page vs Number of Users

The following company page attribute is boolean and must have the values 'Yes' or 'No' but it also has values '1' or '0'. The distribution of the following company page with outliers handled ('0' and '1') can be seen in Figure 3.13.

```
counts = df_cleaned['following_company_page'].value_counts()
total_users = counts.sum()
# Plot the bar plot
plt.figure(figsize=(10, 6))
counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.xlabel('following_company_page')
plt.ylabel('Number of Users')
plt.title('Distribution of following_company_page')
# plt.xticks(rotation=90)  # Rotate x-axis labels if needed
plt.grid(axis='y', linestyle='--', alpha=0.7)
for i, count in enumerate(counts):
    percentage = '{:.2f}%'.format(100 * count / total_users)
    plt.text(i, count, percentage, ha='center', va='bottom')

plt.show()
```

Listing 3.15: Code for UniVariable analysis following_company_page

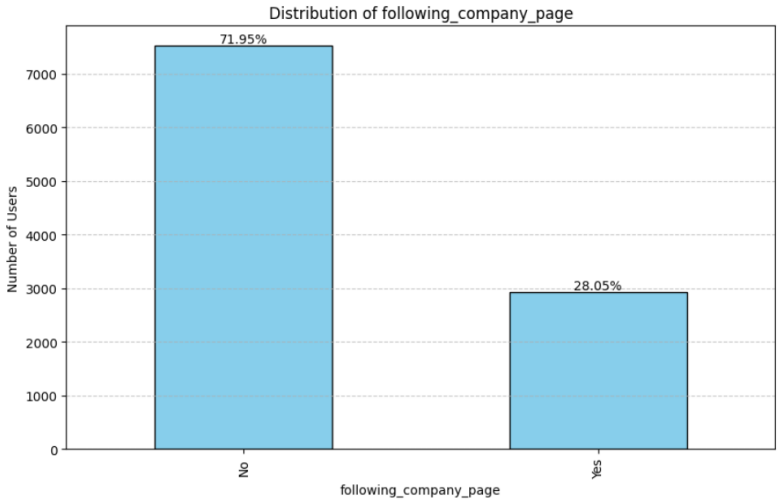

Figure 3.13: Bar graph representing the following_company_page vs Number of Users with outliers handled

The histogram reveals that over 70% of users are not following the company's page. This suggests a significant opportunity for the company to increase user engagement and promote its offerings more effectively. By implementing targeted marketing campaigns and incentives to encourage users to follow the company's page, the company can enhance brand awareness, disseminate promotional offers, and potentially drive higher ticket sales by leveraging the direct communication channel with its user base.

- **montly_avg_comment_on_company_page**

  The histogram illustrates the distribution of monthly average comments on company pages received among users. The x-axis represents the "montly_avg _comment_on_company_page," while the y-axis indicates the number of users, as depicted in Figure 3.14. Each bar represents the range "montly_avg_comment_on_co and the percentage label at the top of each bar signifies the proportion of users that fall in that range.

```
counts = plt.figure(figsize=(10, 6))
hist, bins, _ = plt.hist(df_cleaned['
    montly_avg_comment_on_company_page'], bins=20, color='
    skyblue', edgecolor='black', density=False)
plt.xlabel('montly_avg_comment_on_company_page')
plt.ylabel('Number of Users')
plt.title('Distribution of montly_avg_comment_on_company_page'
    )
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Add count and percentage labels on top of each bar
total_users = len(df_cleaned['
    montly_avg_comment_on_company_page'])
for i in range(len(bins) - 1):
    count = int(hist[i])
    percentage = (count / total_users) * 100
    plt.text(bins[i] + (bins[i+1] - bins[i]) / 2, hist[i], f"{
    percentage:.2f}%", ha='center', va='bottom', rotation=25)

plt.show()
```

Listing 3.16: Code for UniVariable analysis montly_avg_comment_on _company_page

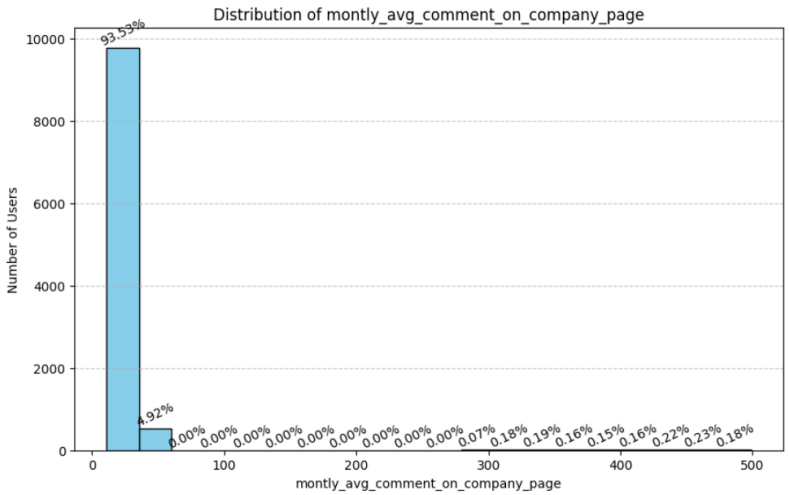

Figure 3.14: Bar graph representing the montly_avg_comment_on_company_page vs Number of Users

Very few users have the number of montly_avg_comment_on_company_page are more than 50. So we are treating a value of more than 50 as 50. The histogram with outliers handled can be seen in Figure 3.15.

```
masked_values = np.where(df_cleaned['
    montly_avg_comment_on_company_page'] > 50, 50,df_cleaned['
    montly_avg_comment_on_company_page'])
plt.figure(figsize=(10, 6))
hist, bins, _ = plt.hist(masked_values, bins=20, color='
    skyblue', edgecolor='black', density=False)
plt.xlabel('montly_avg_comment_on_company_page')
plt.ylabel('Number of Users')
plt.title('Distribution of montly_avg_comment_on_company_page'
    )
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Add count and percentage labels on top of each bar
total_users = len(df_cleaned['
    montly_avg_comment_on_company_page'])
for i in range(len(bins) - 1):
     count = int(hist[i])
     percentage = (count / total_users) * 100
     plt.text(bins[i] + (bins[i+1] - bins[i]) / 2, hist[i], f"{
    percentage:.2f}%", ha='center', va='bottom', rotation=25)

plt.show()
```

Listing 3.17: Code for outlier treatment of UniVariable analysis montly_avg_comment_on_company_page

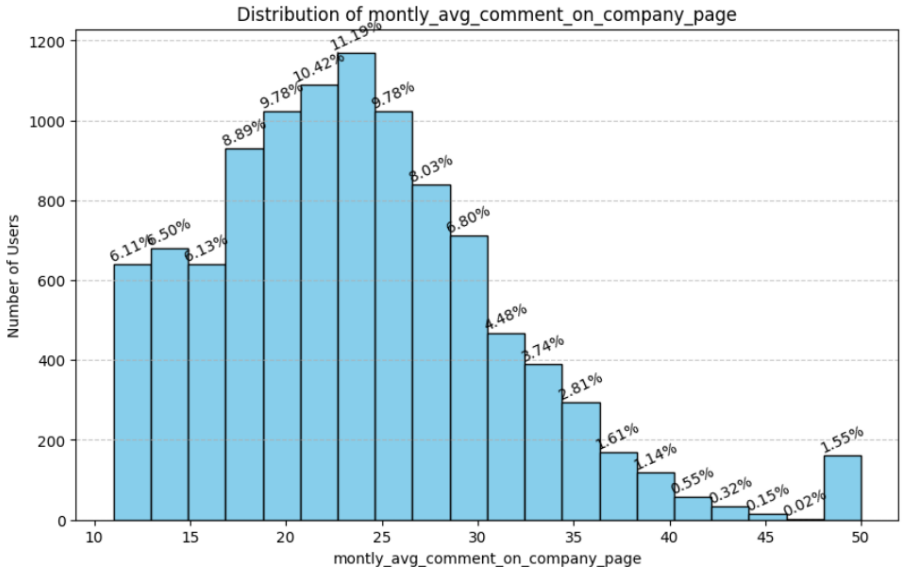

Figure 3.15: Bar graph representing the montly_avg_comment_on_company_page vs Number of Users with outliers handled

- **working_flag**

  The histogram Figure 3.16 illustrates the distribution of users based on whether they are working. The x-axis represents the attribute "working_flag," indicating whether users are working (yes or no). The y-axis represents the number of users. Each bar in the histogram corresponds to either "yes" or "no" for the working_flag. The percentage label at the top of each bar signifies the proportion of users in that category.

```
counts = df_cleaned['working_flag'].value_counts()
total_users = counts.sum()
# Plot the bar plot
plt.figure(figsize=(10, 6))
counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.xlabel('working_flag')
plt.ylabel('Number of Users')
plt.title('Distribution of working_flag')
# plt.xticks(rotation=90)  # Rotate x-axis labels if needed
plt.grid(axis='y', linestyle='--', alpha=0.7)
for i, count in enumerate(counts):
    percentage = '{:.2f}%'.format(100 * count / total_users)
    plt.text(i, count, percentage, ha='center', va='bottom')

plt.show()
```

Listing 3.18: Code for UniVariable analysis following_company_page

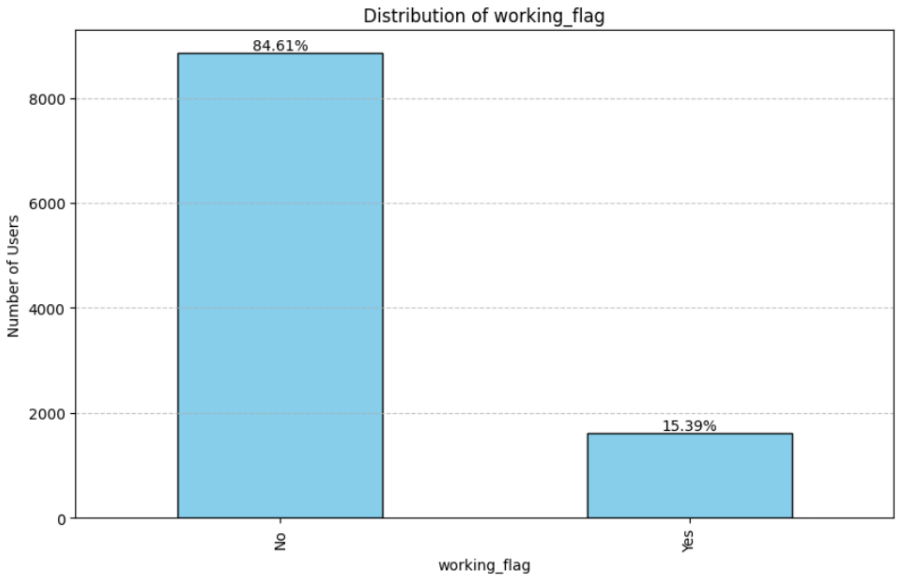

Figure 3.16: Bar graph representing the working_flag vs Number of Users

The histogram reveals that more than 80% of users are not working.

- **travelling_network_rating**

  The histogram Figure 3.17 illustrates the distribution of the travelling network rating among users. The x-axis represents the "travelling_network_rating," while the y-axis indicates the number of users, as depicted in Figure 3.17. Each bar represents the travelling_network_rating number, and the percentage label at the top of each bar signifies the proportion of users given that rating.

```
counts = df_cleaned['week_since_last_outstation_checkin'].
    value_counts().sort_index()
total_users = counts.sum()
# Plot the bar plot
plt.figure(figsize=(10, 6))
counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.xlabel('week_since_last_outstation_checkin')
plt.ylabel('Number of Users')
plt.title('Distribution of week_since_last_outstation_checkin'
    )
plt.xticks(rotation=0)  # Rotate x-axis labels if needed
plt.grid(axis='y', linestyle='--', alpha=0.7)
for i, count in enumerate(counts):
    percentage = '{:.2f}%'.format(100 * count / total_users)
    plt.text(i, count, percentage, ha='center', va='bottom')

plt.show()
```

Listing 3.19: Code for UniVariable analysis week_since_last _outstation_checkin

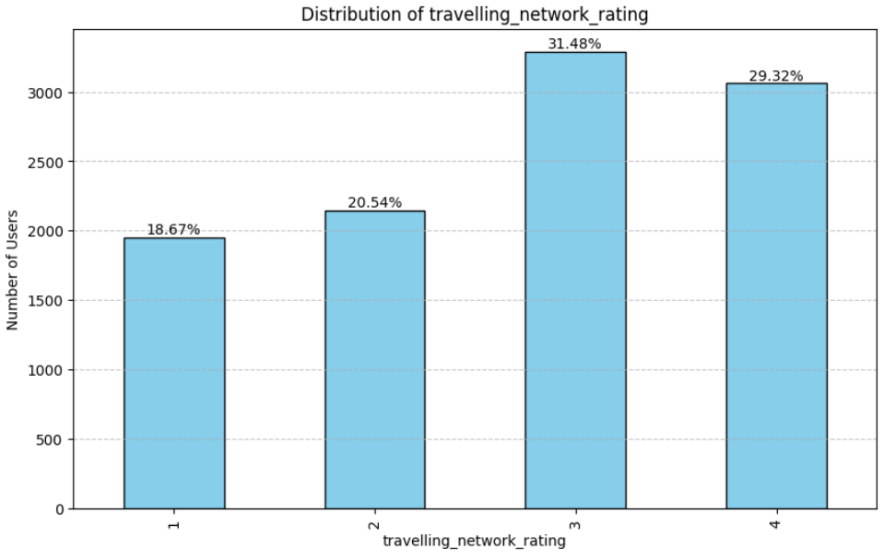

Figure 3.17: Bar graph representing the travelling_network_rating vs Number of Users

The histogram reveals that nearly 60% of users have a travelling network rating of 3 or 4, indicating that a significant portion of their social connections or

friends likes traveling. By leveraging this information, the aviation company can implement features similar to e-commerce applications that request access to users' locations and social networks. This would enable targeted marketing campaigns and personalized travel recommendations based on the preferences and travel patterns of users' social circles, potentially increasing engagement and driving higher ticket sales.

- **Adult_flag**

  According to the attribute name Adult_flag, it should have the yes, and no values or 0 and 1 values but it has values ranging from 0 to 3. These values seem like a number of adults in a family. So the attribute name can be changed to number_of_adults_in_family.

  The histogram Figure 3.18 illustrates the distribution of the number of adults in a family among users. The x-axis represents the "number_of_adults_in_family," while the y-axis indicates the number of users. Each bar represents the number of adults in a family, and the percentage label at the top of each bar signifies the proportion of users who have that specific number of adults.

```
counts = df_cleaned['Adult_flag'].value_counts()
total_users = counts.sum()
# Plot the bar plot
plt.figure(figsize=(10, 6))
counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.xlabel('Adult_flag')
plt.ylabel('Number of Users')
plt.title('Distribution of Adult_flag')
# plt.xticks(rotation=90)  # Rotate x-axis labels if needed
plt.grid(axis='y', linestyle='--', alpha=0.7)
for i, count in enumerate(counts):
    percentage = '{:.2f}%'.format(100 * count / total_users)
    plt.text(i, count, percentage, ha='center', va='bottom')

plt.show()
```

Listing 3.20: Code for UniVariable Adult_flag

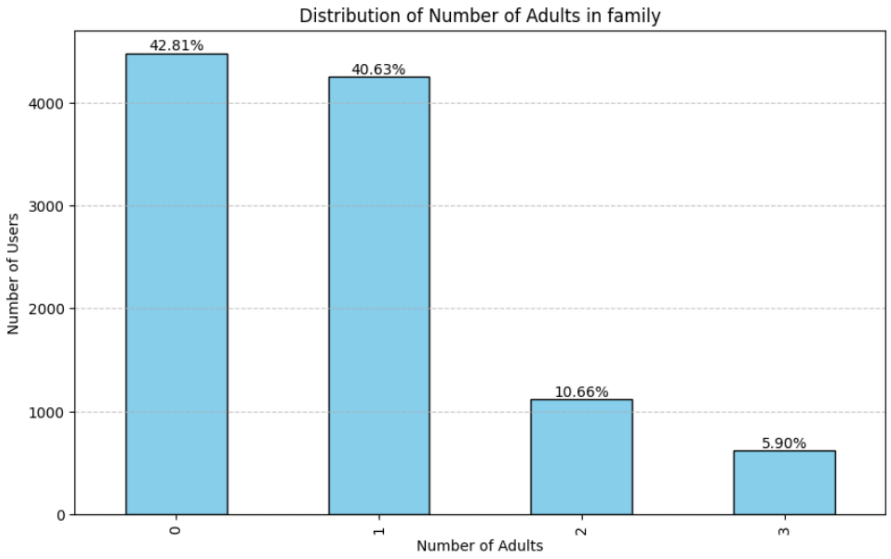

Figure 3.18: Bar graph representing the travelling_network_rating vs Number of Users

- **Daily_Avg_mins_spend_on_traveling_page**

  The histogram illustrates the distribution of Daily average minutes spent on travelling page received among users. The x-axis represents the "Daily_Avg_mins_spend_on_t while the y-axis indicates the number of users, as depicted in Figure 3.19. Each bar represents the range "Daily_Avg_mins_spend_on_traveling_page," and the percentage label at the top of each bar signifies the proportion of users that fall in that range.

```
counts = plt.figure(figsize=(10, 6))
hist, bins, _ = plt.hist(df_cleaned['
    montly_avg_comment_on_company_page'], bins=20, color='
    skyblue', edgecolor='black', density=False)
plt.xlabel('montly_avg_comment_on_company_page')
plt.ylabel('Number of Users')
plt.title('Distribution of montly_avg_comment_on_company_page'
    )
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Add count and percentage labels on top of each bar
total_users = len(df_cleaned['
    montly_avg_comment_on_company_page'])
for i in range(len(bins) - 1):
    count = int(hist[i])
    percentage = (count / total_users) * 100
    plt.text(bins[i] + (bins[i+1] - bins[i]) / 2, hist[i], f"{
    percentage:.2f}%", ha='center', va='bottom', rotation=25)

plt.show()
```

Listing 3.21: Code for UniVariable analysis montly_avg_comment_on _company_page

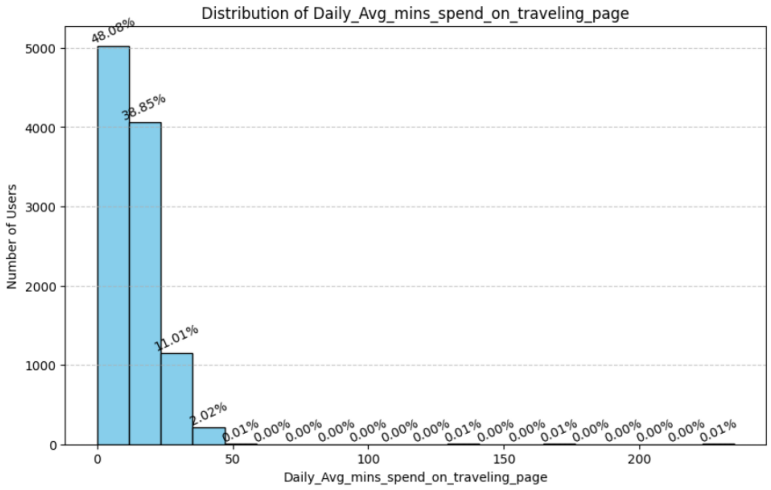

Figure 3.19: Bar graph representing the Daily_Avg_mins_spend_on_traveling_page vs Number of Users

Very few users have the Daily_Avg_mins_spend_on_traveling_page are more than 50. So we are treating a value of more than 50 as 50. The histogram with outliers handled can be seen in Figure 3.20.

```
# Define the condition for handling outliers
condition = df_cleaned['Daily_Avg_mins_spend_on_traveling_page
    '] > 50

# Replace values based on the condition
df_cleaned.loc[condition, '
    Daily_Avg_mins_spend_on_traveling_page'] = 50
plt.figure(figsize=(10, 6))
hist, bins, _ = plt.hist(df_cleaned['
    Daily_Avg_mins_spend_on_traveling_page'], bins=20, color='
    skyblue', edgecolor='black', density=False)
plt.xlabel('Daily_Avg_mins_spend_on_traveling_page')
plt.ylabel('Number of Users')
plt.title('Distribution of
    Daily_Avg_mins_spend_on_traveling_page')
plt.grid(axis='y', linestyle='--', alpha=0.7)
# Add count and percentage labels on top of each bar
total_users = len(df_cleaned['
    Daily_Avg_mins_spend_on_traveling_page'])
for i in range(len(bins) - 1):
     count = int(hist[i])
     percentage = (count / total_users) * 100
     plt.text(bins[i] + (bins[i+1] - bins[i]) / 2, hist[i], f"{
    percentage:.2f}%", ha='center', va='bottom', rotation=25)
plt.show()
```

Listing 3.22: Code for outlier treatment of UniVariable analysis montly_avg_comment_on _company_page

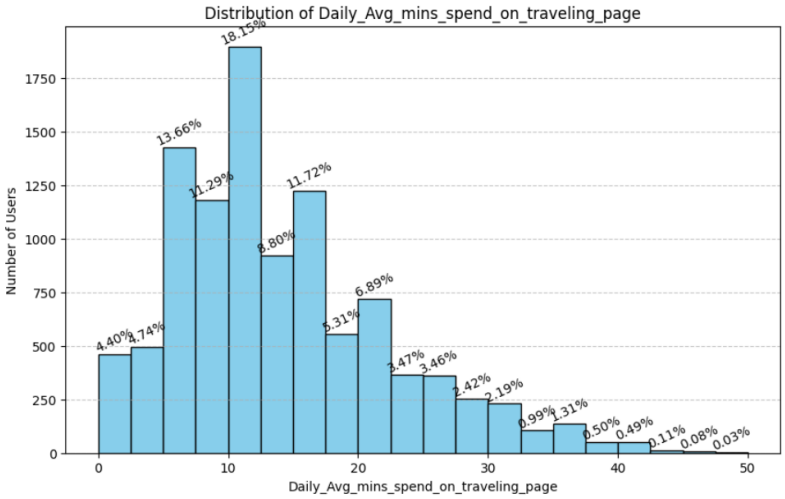

Figure 3.20: Bar graph representing the Daily_Avg_mins_spend_on_traveling_page vs Number of Users With outliers handled

3.2 Bivariate Analysis

Bivariate analysis focuses on exploring relationships between two variables in a dataset. It helps in understanding how one variable (independent variable) is related to another variable (dependent variable) or how two variables are related to each other. By conducting a bi-variate analysis, you gain insights into how changes in advertising spending impact sales revenue, helping optimize marketing strategies and budget allocation.

To perform the bivariate analysis between the numerical attributes we can obtain Pearson correlation coefficient. However, it lacks in the bivariate analysis with respect to the output attribute and some other attributes as they are of categorical type of data.

In the dataset provided, the following attributes are numerical:

- Yearly_avg_view_on_travel_page
- total_likes_on_outstation_checkin_given
- yearly_avg_Outstation_checkins
- member_in_family
- Yearly_avg_comment_on_travel_page
- total_likes_on_outofstation_checkin_received'
- week_since_last_outstation_checkin'
- montly_avg_comment_on_company_page'
- travelling_network_rating'
- number_of_adults_in_family, and
- Daily_Avg_mins_spend_on_traveling_page.

To find the correlation between these numerical attributes, we can use the Pearson correlation coefficient.

The Pearson correlation coefficient measures the linear relationship between two numerical variables. It ranges from -1 to 1, where:

- A value of 1 indicates a perfect positive correlation (as one variable increases, the other variable also increases).
- A value of -1 indicates a perfect negative correlation (as one variable increases, the other variable decreases).
- A value of 0 indicates no linear correlation between the variables.

To calculate the correlation between numerical attributes, we use the .corr method provided by pandas. This method computes pairwise correlation coefficients between columns in a DataFrame. When applied to numerical data, it calculates the Pearson correlation coefficient by default.

Using the .corr method on our dataset, we can generate a correlation matrix that shows the correlation coefficients between all pairs of numerical attributes. This matrix helps identify relationships between variables, allowing us to see which pairs of attributes have strong positive or negative correlations. To understand better refer to Code 3.27.

To visualize the correlation matrix, we can use a heatmap. In a heatmap, the color intensity represents the strength of the correlation, and the values within the cells show the actual correlation coefficients. The heatmap obtained can be seen in Figure 3.21.

```
numerical_cols = ['Yearly_avg_view_on_travel_page', '
    total_likes_on_outstation_checkin_given',
                  'yearly_avg_Outstation_checkins', '
    member_in_family', 'Yearly_avg_comment_on_travel_page', '
    total_likes_on_outofstation_checkin_received',
                  'week_since_last_outstation_checkin', '
    montly_avg_comment_on_company_page',
                  'travelling_network_rating', '
    number_of_adults_in_family',
                  'Daily_Avg_mins_spend_on_traveling_page']

# Compute the correlation matrix
correlation_matrix = df_cleaned[numerical_cols].corr()
plt.figure(figsize=(16, 6))
heatmap = sns.heatmap(correlation_matrix, vmin=-1, vmax=1, annot=
    True)
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12},
    pad=12);
```

Listing 3.23: Code for Bivariate analysis between Numerical data

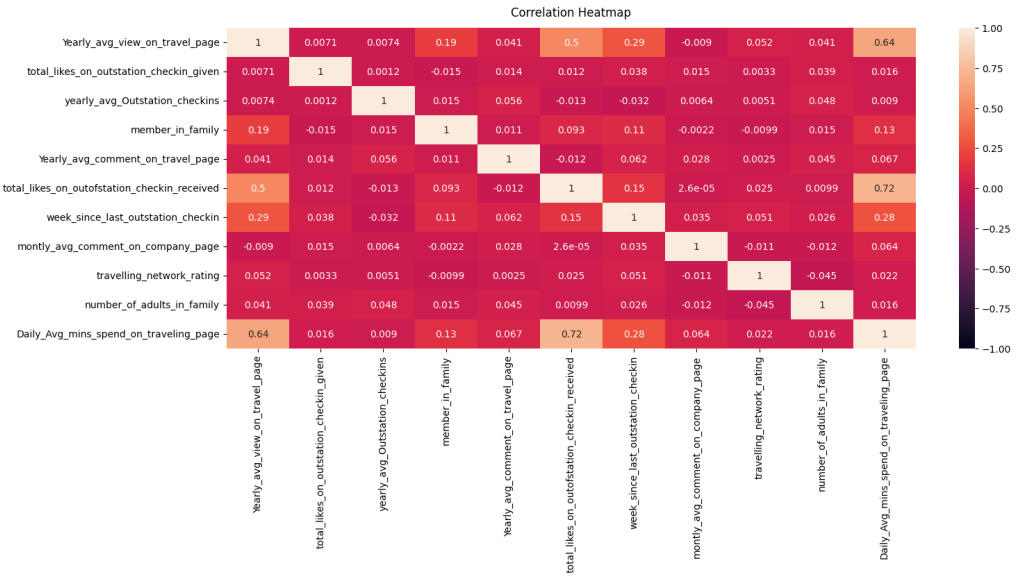

Figure 3.21: Heatmap displaying the Bivariate analysis between Numerical data

By examining the heatmap, we can easily identify highly correlated attributes, which may indicate redundancy or multicollinearity in the dataset.

The attributes with the highest correlation coefficients identified in the heatmap are summarized in Table 3.3.

	Attribute1	Attribute2	Correlation
44	total_likes_on_outofstation_checkin_received	Daily_Avg_mins_spend_on_traveling_page	0.718101
9	Yearly_avg_view_on_travel_page	Daily_Avg_mins_spend_on_traveling_page	0.641726
4	Yearly_avg_view_on_travel_page	total_likes_on_outofstation_checkin_received	0.501331
5	Yearly_avg_view_on_travel_page	week_since_last_outstation_checkin	0.285647
48	week_since_last_outstation_checkin	Daily_Avg_mins_spend_on_traveling_page	0.275839
2	Yearly_avg_view_on_travel_page	member_in_family	0.194894
40	total_likes_on_outofstation_checkin_received	week_since_last_outstation_checkin	0.146637
33	member_in_family	Daily_Avg_mins_spend_on_traveling_page	0.131716
29	member_in_family	week_since_last_outstation_checkin	0.108251
28	member_in_family	total_likes_on_outofstation_checkin_received	0.092650

Table 3.3: Top 10 pairs of attributes with the highest correlation coefficients

The attributes with the lowest correlation coefficients identified in the heatmap are summarized in Table 3.4.

	Attribute1	Attribute2	Correlation
52	travelling_network_rating	number_of_adults_in_family	-0.044515
22	yearly_avg_Outstation_checkins	week_since_last_outstation_checkin	-0.031902
11	total_likes_on_outstation_checkin_given	member_in_family	-0.015296
21	yearly_avg_Outstation_checkins	total_likes_on_outofstation_checkin_received	-0.012635
50	montly_avg_comment_on_company_page	number_of_adults_in_family	-0.012210

Table 3.4: Bottom 5 pairs of attributes with the lowest correlation coefficients

3.3 Multivariate analysis

One of the multivariate analysis techniques is clustering. Clustering is the unsupervised machine learning algorithm and is used on the non labeled data to group the points that have similar attribute characteristics. But the problem we had in our hands was kind of the supervised learning and of type classification. Even though we thought to try clustering to see how many unique categories of the users are present by considering only numerical attributes of data.

To find the unique categories in data the clustering algorithm has some evaluation method called the elbow plot based on that we are getting the elbow between k=2 and k=3 which means data has two or 3 unique categories. The elbow plot can be seen in Figure 3.22. As we know about data which is binary classification data based on using K-means clustering we get appropriate results.

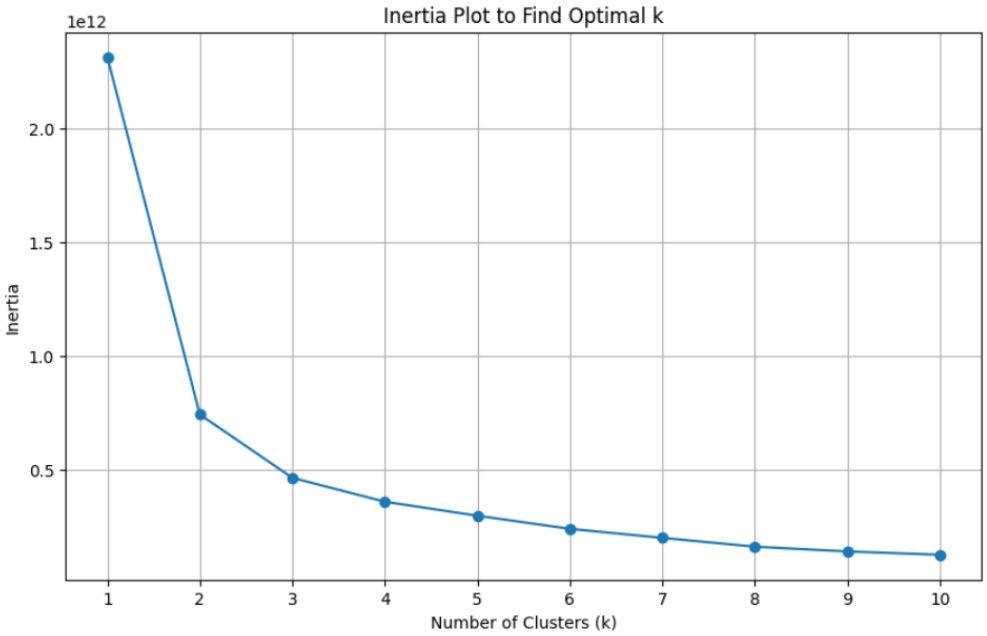

Figure 3.22: Elbow plot showing the inertia values for different numbers of clusters k

3.4 Model building and interpretation

The first step involved before model building is splitting the dataset into training and testing sets for both mobile and laptop data.

```
#Split data into train and test sets.
X_train_mobile, X_test_mobile, y_train_mobile, y_test_mobile =
    train_test_split(X_mobile, y_mobile, test_size=0.3, random_state
    =42)
X_train_laptop, X_test_laptop, y_train_laptop, y_test_laptop =
    train_test_split(X_laptop, y_laptop, test_size=0.3, random_state
    =42)
```

Listing 3.24: Code for splitting the dataset train and test sets for both mobile and laptop

Next, we initialized several classifiers, including Random Forest, Logistic Regression, Decision Tree, Gradient Boosting, AdaBoost, and Support Vector Machines (SVM).

```
# Initialize classifiers
random_forest_clf = RandomForestClassifier(random_state=42)
logistic_regression_clf = LogisticRegression(max_iter=1000,
    random_state=42)
decision_tree_clf = DecisionTreeClassifier(random_state=42)
gradient_boosting_clf = GradientBoostingClassifier(random_state=42)
ada_boosting_clf = AdaBoostClassifier(random_state=42)
svm_clf = SVC(kernel='linear', C=1.0, random_state=42)
classifiers = {
```

```
        "Random Forest": random_forest_clf,
        "Logistic Regression": logistic_regression_clf,
        "Decision Tree": decision_tree_clf,
        "Gradient Boosting": gradient_boosting_clf,
        "AdaBoost": ada_boosting_clf,
        "Support Vector Machines " :  svm_clf

}
```

Listing 3.25: Code for initializing the classifiers

To assess the performance and generalizability of these models, we employed a 5-fold cross-validation technique. This process helps to estimate the model's performance on unseen data and mitigates the risk of overfitting.

```
#Training and cross validation
def training_crossvalidation(X_train,y_train, title):

  # Initialize lists to store results
  classifier_names = []
  cv_scores_list_str = []
  mean_cv_accuracy_list = []

  # Perform cross-validation and evaluate each classifier
  for name, clf in classifiers.items():
      # Perform cross-validation
      cv_scores = cross_val_score(clf, X_train, y_train, cv=5)

      # Calculate mean cross-validation accuracy
      mean_cv_accuracy = cv_scores.mean()

      # Convert cross-validation scores to string representation
      cv_scores_str = [str(scores) for scores in cv_scores]

      # Append results to lists
      classifier_names.append(name)
      cv_scores_list_str.append(cv_scores_str)
      mean_cv_accuracy_list.append(mean_cv_accuracy)
  print(title)
  # Print results as a table using tabulate
  table_data = zip(classifier_names, cv_scores_list_str,
   mean_cv_accuracy_list)
  table_headers = ["Classifier", "Cross-Validation Scores", "Mean
   Cross-Validation Accuracy"]
  print(tabulate(table_data, headers=table_headers, tablefmt="grid"
   ))
training_crossvalidation(X_train_mobile, y_train_mobile, "Cross
   validation output of multiple ML classifiers for mobile data")
training_crossvalidation(X_train_laptop,y_train_laptop,  "Cross
   validation output of multiple ML classifiers for laptop data")
```

Listing 3.26: Code for Training and cross-validation of models

The cross-validation of different machine learning models for both the laptop and

mobile data can be seen in Table 3.5.

Cross validation output of multiple ML classifiers for mobile data

Classifier	Cross-Validation Scores	Mean Cross-Validation Accuracy
Random Forest	['0.9633307868601986', '0.9648586707410237', '0.9701834862385321', '0.9717125382262997', '0.9740061162079511']	0.968818
Logistic Regression	['0.8540870893812077', '0.8624904507257448', '0.8646788990825688', '0.8685015290519877', '0.8639143730886851']	0.862734
Decision Tree	['0.9556913674560733', '0.944996180290298', '0.9686544342507645', '0.9640672782874617', '0.9648318042813455']	0.959648
Gradient Boosting	['0.8991596638655462', '0.8938120702826585', '0.9013761467889908', '0.8990825688073395', '0.9021406727828746']	0.899114
AdaBoost	['0.8594346829640948', '0.8762414056531703', '0.8692660550458715', '0.8669724770642202', '0.8715596330275229']	0.868695
Support Vector Machines	['0.8449197860962567', '0.8449197860962567', '0.845565749235474', '0.845565749235474', '0.8448012232415902']	0.845154

Cross validation output of multiple ML classifiers for laptop data

Classifier	Cross-Validation Scores	Mean Cross-Validation Accuracy
Random Forest	['0.9612903225806452', '0.9290322580645162', '0.9290322580645162', '0.9290322580645162', '0.9870967741935484']	0.947097
Logistic Regression	['0.8258064516129032', '0.832258064516129', '0.7612903225806451', '0.8387096774193549', '0.8193548387096774']	0.815484
Decision Tree	['0.9483870967741935', '0.8903225806451613', '0.9161290322580645', '0.9032258064516129', '0.9483870967741935']	0.92129
Gradient Boosting	['0.9419354838709677', '0.9161290322580645', '0.9096774193548387', '0.9096774193548387', '0.9419354838709677']	0.923871
AdaBoost	['0.8709677419354839', '0.8387096774193549', '0.8451612903225807', '0.832258064516129', '0.8516129032258064']	0.847742
Support Vector Machines	['0.8580645161290322', '0.8193548387096774', '0.7483870967741936', '0.8193548387096774', '0.8']	0.809032

Table 3.5: Cross-validation results of different machine learning models for predicting product purchases based on social media activities

Based on the cross-validation scores, we observed that the Decision Tree, Gradient Boosting, and Random Forest models achieved high accuracies. Specifically, the accuracies for the mobile data were approximately 0.9594, 0.899, and 0.9688 respectively, while for the laptop data, the accuracies were 0.921, 0.923, and 0.947 respectively. Given the similarities in these values, there is a possibility that the model with the slightly lower accuracy may perform better on the test data.

So we have processed to test the decision tree, Gradient Boosting, and random forest model using test data.

3.5 Model Evaluation

In this section, we discuss the performance of different machine learning models on the test dataset for both mobile and laptop data. We evaluate the models based on various metrics such as accuracy, precision, recall, F1-score, and the Receiver Operating Characteristic (ROC) curve. The results obtained using the mentioned metrics can be seen in Figures 3.23, 3.24, 3.25, 3.26, and 3.27.

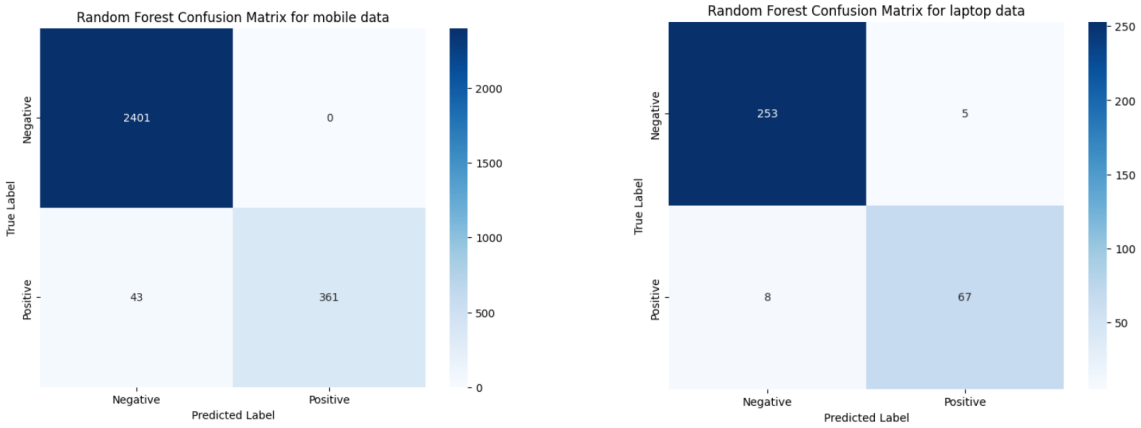

Figure 3.23: Confusion Matrices evaluating the Random Forest ML model

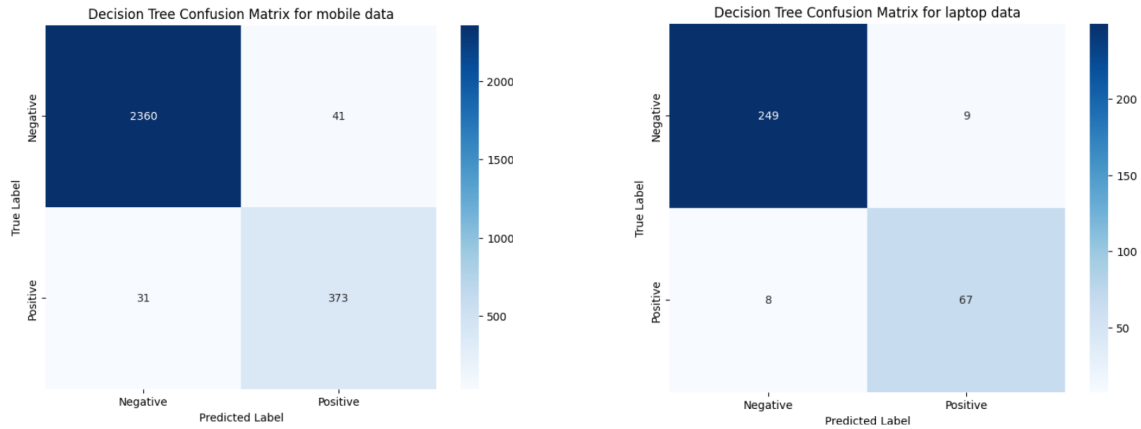

Figure 3.24: Confusion Matrices evaluating the Decision Tree ML model

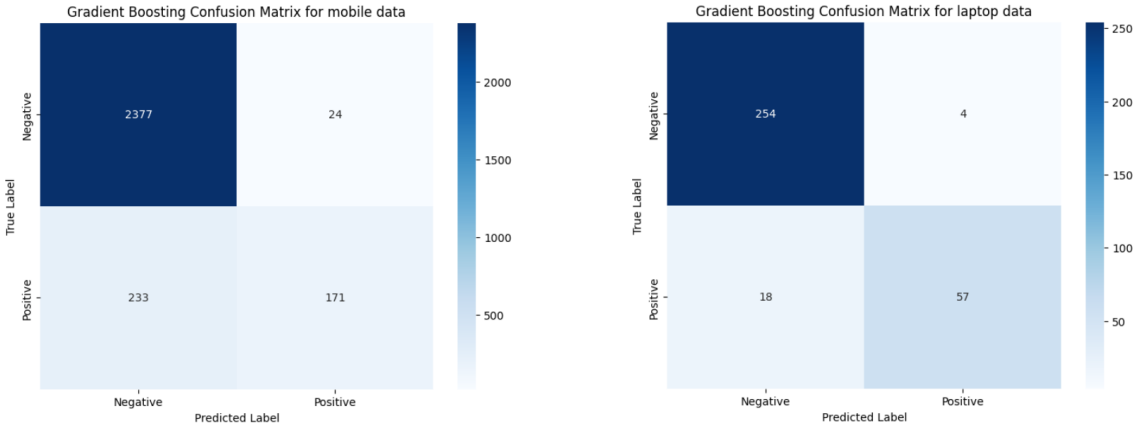

Figure 3.25: Confusion Matrices evaluating the Gradient Boosting ML model

Random Forest Classification Report for mobile data:

	precision	recall	f1-score	support
Negative	0.98	1.00	0.99	2401
Positive	1.00	0.89	0.94	404
accuracy			0.98	2805
macro avg	0.99	0.95	0.97	2805
weighted avg	0.98	0.98	0.98	2805

Random Forest Classification Report for laptop data:

	precision	recall	f1-score	support
Negative	0.97	0.98	0.97	258
Positive	0.93	0.89	0.91	75
accuracy			0.96	333
macro avg	0.95	0.94	0.94	333
weighted avg	0.96	0.96	0.96	333

Decision Tree Classification Report for mobile data:

	precision	recall	f1-score	support
Negative	0.99	0.98	0.98	2401
Positive	0.90	0.92	0.91	404
accuracy			0.97	2805
macro avg	0.94	0.95	0.95	2805
weighted avg	0.97	0.97	0.97	2805

Decision Tree Classification Report for laptop data:

	precision	recall	f1-score	support
Negative	0.97	0.97	0.97	258
Positive	0.88	0.89	0.89	75
accuracy			0.95	333
macro avg	0.93	0.93	0.93	333
weighted avg	0.95	0.95	0.95	333

Gradient Boosting Classification Report for mobile data:

	precision	recall	f1-score	support
Negative	0.91	0.99	0.95	2401
Positive	0.88	0.42	0.57	404
accuracy			0.91	2805
macro avg	0.89	0.71	0.76	2805
weighted avg	0.91	0.91	0.89	2805

Gradient Boosting Classification Report for laptop data:

	precision	recall	f1-score	support
Negative	0.93	0.98	0.96	258
Positive	0.93	0.76	0.84	75
accuracy			0.93	333
macro avg	0.93	0.87	0.90	333
weighted avg	0.93	0.93	0.93	333

Figure 3.26: Classification Report of Evaluating the Random Forest, Decision Tree, and Gradient Boosting ML classifier

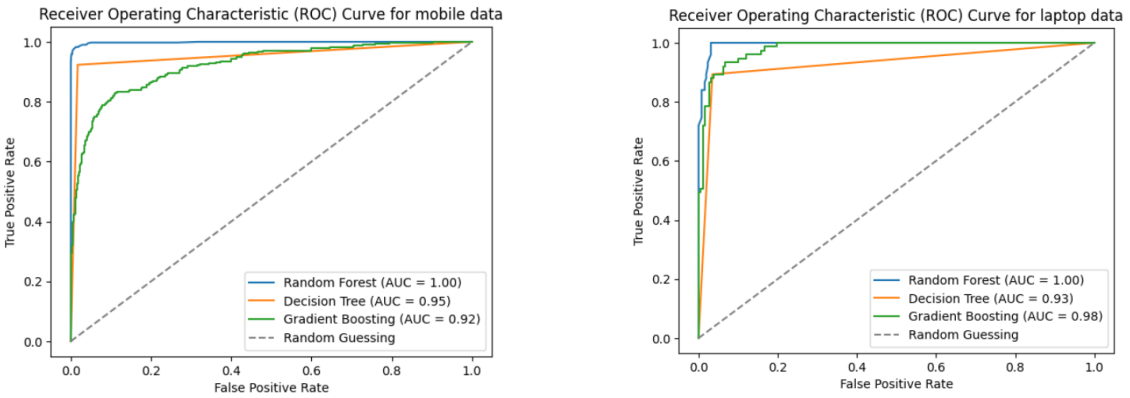

Figure 3.27: ROC curve for Evaluating the Random Forest, Decision Tree, and Gradient Boosting ML classifier

3.5.1 Mobile Data

For the mobile data, we trained and evaluated three different models: Random Forest, Decision Tree, and Gradient Boosting. The performance metrics for each model are as follows:

3.5.1.1 Random Forest

The Random Forest model achieved the highest accuracy of 0.9846 on the test set for mobile data. The confusion matrix shows that it correctly classified 2401 negative instances and 361 positive instances, with only 43 false negatives. The classification report indicates a precision of 0.98 for the negative class and 1.00 for the positive class, with a recall of 1.00 and 0.89, respectively. The F1-score, which is the harmonic mean of precision and recall, is 0.99 for the negative class and 0.94 for the positive class. The ROC score for the Random Forest model on mobile data is 1.00, indicating excellent performance in distinguishing between the two classes.

3.5.1.2 Decision Tree

The Decision Tree model achieved an accuracy of 0.9743 on the test set for mobile data. The confusion matrix shows that it correctly classified 2360 negative instances and 373 positive instances, with 41 false positives and 31 false negatives. The classification report indicates a precision of 0.99 for the negative class and 0.90 for the positive class, with a recall of 0.98 and 0.92, respectively. The F1-score is 0.98 for the negative class and 0.91 for the positive class. The ROC score for the Decision Tree model on mobile data is 0.93, which is lower than the Random Forest model but still indicates good performance.

3.5.1.3 Gradient Boosting

The Gradient Boosting model achieved an accuracy of 0.9083 on the test set for mobile data, which is lower than the Random Forest and Decision Tree models. The

confusion matrix shows that it correctly classified 2377 negative instances and 171 positive instances, with 24 false positives and 233 false negatives. The classification report indicates a precision of 0.91 for the negative class and 0.88 for the positive class, with a recall of 0.99 and 0.42, respectively. The F1-score is 0.95 for the negative class and 0.57 for the positive class. The ROC score for the Gradient Boosting model on mobile data is 0.98, which is higher than the Decision Tree model but lower than the Random Forest model.

3.5.2 Laptop Data

For the laptop data, we also trained and evaluated the same three models: Random Forest, Decision Tree, and Gradient Boosting. The performance metrics for each model are as follows:

3.5.2.1 Random Forest

The Random Forest model achieved an accuracy of 0.9609 on the test set for laptop data. The confusion matrix shows that it correctly classified 253 negative instances and 67 positive instances, with 5 false positives and 8 false negatives. The classification report indicates a precision of 0.97 for the negative class and 0.93 for the positive class, with a recall of 0.98 and 0.89, respectively. The F1-score is 0.97 for the negative class and 0.91 for the positive class. The ROC score for the Random Forest model on laptop data is 1.00, indicating excellent performance in distinguishing between the two classes.

3.5.2.2 Decision Tree

The Decision Tree model achieved an accuracy of 0.9489 on the test set for laptop data. The confusion matrix shows that it correctly classified 249 negative instances and 67 positive instances, with 9 false positives and 8 false negatives. The classification report indicates a precision of 0.97 for the negative class and 0.88 for the positive class, with a recall of 0.97 and 0.89, respectively. The F1-score is 0.97 for the negative class and 0.89 for the positive class. The ROC score for the Decision Tree model on laptop data is 0.95, which is lower than the Random Forest model but still indicates good performance.

3.5.2.3 Gradient Boosting

The Gradient Boosting model achieved an accuracy of 0.9339 on the test set for laptop data, which is lower than the Random Forest and Decision Tree models. The confusion matrix shows that it correctly classified 254 negative instances and 57 positive instances, with 4 false positives and 18 false negatives. The classification report indicates a precision of 0.93 for both the negative and positive classes, with a recall of 0.98 and 0.76, respectively. The F1-score is 0.96 for the negative class and 0.84 for the positive class. The ROC score for the Gradient Boosting model on laptop data is 0.92, which is lower than both the Random Forest and Decision Tree models.

3.5.3 Model Selection

Based on the evaluation metrics and performance on both mobile and laptop data, the Random Forest model emerges as the most suitable choice for this dataset. It consistently outperforms the Decision Tree and Gradient Boosting models in terms of accuracy, precision, recall, F1-score, and ROC score across both mobile and laptop data.

The Random Forest model's high accuracy, balanced precision and recall scores, and excellent ROC scores indicate its ability to effectively classify both positive and negative instances. Additionally, its ensemble nature and the use of multiple decision trees make it more robust to overfitting and noise in the data.

Therefore, we considered selecting the Random Forest model for further deployment and production use, as it demonstrates superior performance and generalization capabilities compared to the other models evaluated.

3.6 Model Tuning

In order to improve the performance of the Random Forest model, we performed hyperparameter tuning using a grid search approach. The goal was to find the optimal combination of hyperparameters that would maximize the model's accuracy and generalization capabilities on both mobile and laptop data.

The hyperparameters considered for tuning were:

- **n_estimators**: The number of trees in the forest. We explored values of 100, 200, and 290.
- **max_depth**: The maximum depth of the trees. We tried values of 10 and 20.
- **criterion**: The function used to measure the quality of a split. We evaluated both 'gini' and 'entropy' criteria.
- **min_samples_split**: The minimum number of samples required to split an internal node. We tested values of 2, 5, and 10.
- **min_samples_leaf**: The minimum number of samples required to be at a leaf node. We tried values of 1 and 2.
- **max_features**: The number of features to consider when looking for the best split. We explored using 50% of the features, 90% of the features, and the square root of the total number of features ('log2').
- **bootstrap**: A boolean indicating whether bootstrap samples are used when building trees. We evaluated both True and False values.
- **n_jobs**: The number of jobs to run in parallel. We set this to 4 to utilize multiple cores and speed up the computation.

We conducted an exhaustive grid search by generating all possible combinations of these hyperparameters, yielding a total of 432 distinct parameter configurations (3 $\times$ 2 $\times$ 2 $\times$ 3 $\times$ 2 $\times$ 3 $\times$ 2 $\times$ 1). Each configuration underwent evaluation through cross-validation on both the mobile and laptop datasets.

```
def tunning(X_train, y_train, X_test, y_test, title):
  param_grid = {
    'n_estimators': [100, 200, 290],
    'max_depth': [10, 20],
    'criterion':['gini','entropy'],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1,2],
    'max_features' : [0.5, 0.9,'log2'],
    'bootstrap':[True, False],
    'n_jobs': [4]
  }


  # Initialize the Random Forest classifier
  random_forest_clf = RandomForestClassifier(random_state=42)

  # Initialize Grid Search with 5-fold cross-validation
  grid_search = GridSearchCV(random_forest_clf, param_grid, cv=5,
   scoring='accuracy')

  # Perform grid search on the training data
  grid_search.fit(X_train, y_train)

  # Get the best parameters
  best_params = grid_search.best_params_
  print(f'Best Parameters for {title}:', best_params)

  # Get the best model
  best_rf_model = grid_search.best_estimator_

  # Evaluate the best model on the test set
  test_accuracy = best_rf_model.score(X_test, y_test)
  print(f'Test Set Accuracy on {title}:', test_accuracy)

tunning(X_train_mobile, y_train_mobile, X_test_mobile,
   y_test_mobile, "mobile data")
tunning(X_train_laptop,y_train_laptop, X_test_laptop, y_test_laptop
   , "laptop data")
```

Listing 3.27: Code for Model Tuning

The best hyperparameter configuration for the mobile data was found to be:

- n_estimators = 290
- max_depth = 20
- criterion = 'entropy'
- min_samples_split = 2

- min_samples_leaf = 1
- max_features = 'log2'
- bootstrap = False
- n_jobs = 4

This configuration achieved an accuracy of 0.9942 on the mobile data test set, representing a significant improvement over the default Random Forest model's accuracy of 0.9846.

For the laptop data, the best hyperparameter configuration was:

- n_estimators = 290
- max_depth = 20
- criterion = 'entropy'
- min_samples_split = 2
- min_samples_leaf = 1
- max_features = 0.5
- bootstrap = False
- n_jobs = 4

This configuration achieved an accuracy of 0.9819 on the laptop data test set, an improvement over the default Random Forest model's accuracy of 0.9609.

In general, the optimal hyperparameter configurations differ for different datasets. However, in our case, we found the optimal hyperparameter that improved the accuracy of the Random Forest Model built on mobile and laptop data are the same parameters.

By performing hyperparameter tuning and identifying the optimal configurations, we were able to enhance the Random Forest model's performance and achieve higher accuracy scores on both the mobile and laptop data. This demonstrates the importance of model tuning in maximizing the predictive power and generalization capabilities of machine learning models.

The final evaluation using the confusion matrix, and classification report on the fine-tuned model can be seen in Figures 3.29, 3.28.

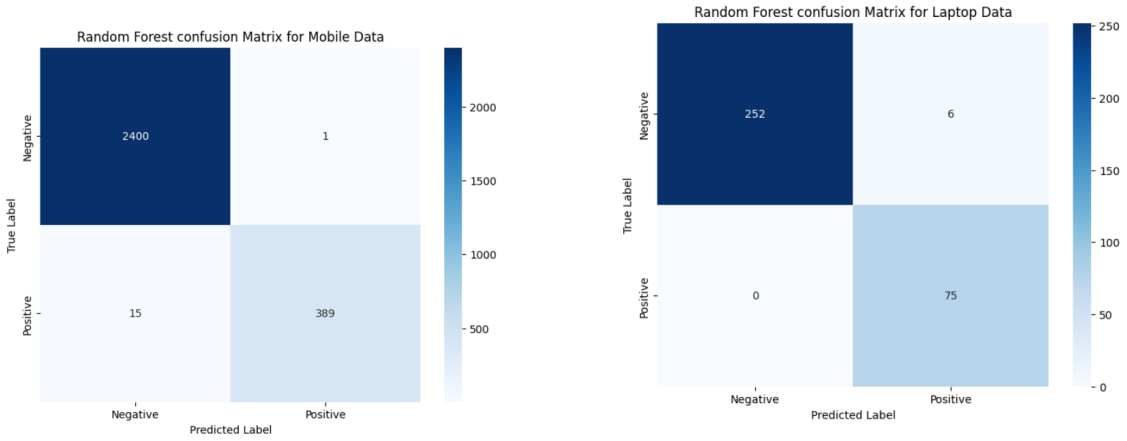

Figure 3.28: Confusion Matrices evaluating the fine-tuned Random Forest ML models

Random Forest classification Report for Mobile Data:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	2401
1	1.00	0.96	0.98	404
accuracy			0.99	2805
macro avg	1.00	0.98	0.99	2805
weighted avg	0.99	0.99	0.99	2805

Random Forest classification Report for Laptop Data:

	precision	recall	f1-score	support
0	1.00	0.98	0.99	258
1	0.93	1.00	0.96	75
accuracy			0.98	333
macro avg	0.96	0.99	0.97	333
weighted avg	0.98	0.98	0.98	333

Figure 3.29: Classification reports evaluating the fine-tuned Random Forest ML models

4

FINDINGS, RECOMMENDATIONS AND CONCLUSION

4.1 Findings Based on Observations

- The dataset contains user engagement metrics related to travel and tourism, such as likes on outstation check-ins, comments on travel pages, and time spent on the company's travel page.
- The data has a longitudinal structure, with variables like yearly averages, monthly averages, and counts, indicating different frequencies of data collection.
- Attributes like 'preferred_device' and 'following_company_page' suggest data collection from platform settings or user preferences.
- Metrics like 'total_likes_on_outstation_checkin_given' and 'total_likes_on_outofstation_ch were likely derived from social media interactions.

4.2 Findings Based on analysis of Data

- Total likes on outstation check-ins given and yearly average views on the travel page have the highest correlation with daily average time spent on the page.
- The majority of customers over 70% are not following the company's page.
- 26% of users made an outstation check-in 1 week ago, suggesting a lower propensity to travel for these users.
- Many customers prefer beach and financial hub locations, followed by historical sites.
- Approximately 90% of customers use mobile devices, with tablets being the preferred device.
- The majority of people (84%) are not working.
- The majority of customers (85%) have not taken the product.

- The Random Forest model achieves the highest prediction accuracy of 96% for both mobile and laptop devices

4.3 General findings

- Machine learning models can effectively predict user behavior and preferences based on social media activity.
- Users display distinct preferences for certain types of locations and devices, which can be leveraged for targeted advertising.
- Personalized marketing strategies can significantly enhance customer engagement and sales.

4.4 Recommendation based on findings

- Implement tailored marketing campaigns and travel suggestions based on user preferences and behavior, including preferred location categories and travel network ratings.
- Develop strategies to increase user engagement on the company's travel page, such as interactive features, personalized content, and gamification elements.
- Leverage social media and influencer marketing to drive more likes, comments, and check-ins related to travel experiences.
- Offer incentives or loyalty programs to encourage users to follow the company's page and engage with its content.
- Analyze the characteristics of highly engaged users and replicate successful strategies to improve overall engagement levels.
- Develop mobile-friendly features and campaigns, given the higher usage of mobile devices.
- Target customers who have not checked in during the last few weeks, as outstation check-in frequency is a key feature.
- Focus marketing efforts on adults, as they play a critical role in buying decisions.
- Digital advertisements should be targeted at specific groups of customers who have a high propensity to buy tickets to receive more return on investment.
- Use the Random Forest model to predict future product purchases and optimize marketing strategies.

4.5 Suggestions for areas of improvement

- Improve data collecting methods to gather more detailed user behavior data, such as travel preferences, booking patterns, and feedback.
- Implement advanced analytics approaches, such as sentiment analysis, to acquire a better understanding of user preferences and sentiment.
- Investigate collaborations or integrations with other travel-related platforms or services to broaden your user base and data sources.
- Continuously monitor and evaluate user interaction numbers to spot new trends and alter marketing strategy as needed.
- Invest in user experience (UX) research and design to make the company's digital platforms more usable and appealing.
- Machine learning models should be updated on a regular basis to reflect changing user preferences and behaviors.

4.6 Scope for future research

The current study examined user engagement metrics and preferences in the travel and tourism industries. Future research should look into other elements that influence travel behavior, such as demographics, psychographic profiles, and external factors like economic conditions or travel trends. Incorporating qualitative data from customer feedback, surveys, or social media sentiment analysis may also provide useful insights into travelers' motivations and pain areas. Furthermore, using advanced machine learning techniques like deep learning and natural language processing may allow for more accurate prediction of user preferences and personalized suggestions.

4.7 Conclusion

The examination of the provided dataset revealed useful information about user interaction and travel and tourist preferences. The information shows skewed distributions and differing degrees of interaction across parameters, but it also shows where there are chances for more individualized suggestions, focused marketing campaigns, and enhanced user experience tactics.

Implementing the recommendations mentioned in this study will help the organization increase user engagement, drive higher ticket sales, and nurture a more loyal customer base. Continuous monitoring, data-driven decision-making, and a dedication to innovation will be critical for staying ahead of changing consumer trends and retaining a competitive advantage in the fast-paced travel and tourism business.

Bibliography

[1] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.

[2] L. A. Yates, Z. Aandahl, S. A. Richards, and B. W. Brook, "Cross validation for model selection: A review with examples from ecology," *Ecological Monographs*, vol. 93, no. 1, e1557, 2023.

[3] K. Stpor, "Evaluating and comparing classifiers: Review, some recommendations and limitations," in *Proceedings of the 10th International Conference on Computer Recognition Systems CORES 2017 10*, Springer, 2018, pp. 12–21.

A

Appendix 1