PROJECT REPORT

1.INTRODUCTION

1.1 Project Overview

In this present interconnected world, the airline industry serves as a critical catalyst for global travel and business. The goal is to develop an intuitive web application that leverages machine learning techniques to classify airline reviews as either "Recommended" or "Not Recommended." After extensive data preprocessing, including standardization and cleaning, various machine learning algorithms were employed, with XGBoost as the most accurate classifier. The user interface was designed to provide a seamless experience, featuring a clean layout inspired by the "AeroChoice" template. Users input data such as airline name, ratings, origin, destination, flight service rating, and overall rating, triggering the XGBoost model for real-time classification upon clicking the "Check It" button. The result, displayed on the interface, guides users in making informed decisions about their airline choices.

1.2 Purpose

The main purpose of this project is to create a user-friendly web app that uses Machine learning to analyse and classify airline reviews. By processing and training on various aspects of user-provided data, the project aims to help users determine whether an airline is recommended or not. The central goal is to seamlessly integrate advanced technology into the user experience, providing personalized recommendations based on input about different aspects of their airline travel. In simpler terms, it is all about making it easy for people to get helpful insights from reviews when choosing an airline.

2.LITERATURE SURVEY

2.1 Existing Problem

In the field of airline review classification, several challenges have been observed in the existing literature. One prominent issue revolves around the subjective nature of user reviews, where diverse expressions of experiences make it challenging to establish a universally accurate classification model. Determining the most relevant features for effective classification remains an ongoing challenge, and ensuring the generalization of trained models to new, unseen data is another common concern. Solving these issues is crucial to make airline review systems more reliable and useful.

2.2 References

1. Kaggle

Dataset is retrieved from Kaggle website.

https://www.kaggle.com/datasets/khushipitroda/airline-reviews

The data for this airline reviews dataset was collected by web scraping the website https://www.airlinequality.com/ using the Python library Beautiful Soup.

2. Scikit-Learn.

Decision Trees documentation. Retrieved from Scikit-Learn library.

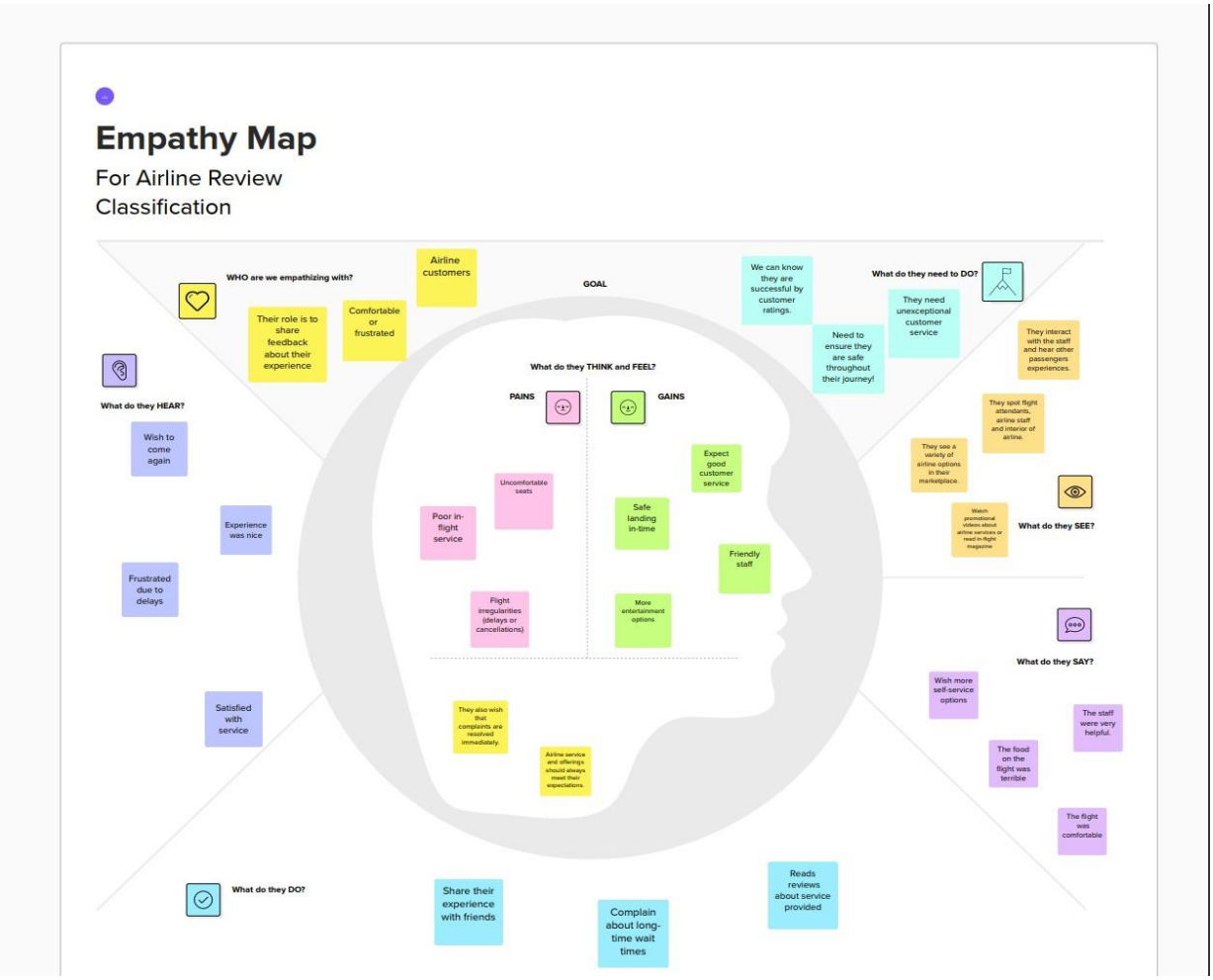
https://scikit-learn.org/stable/modules/tree.html

2.3 Problem Statement Definition

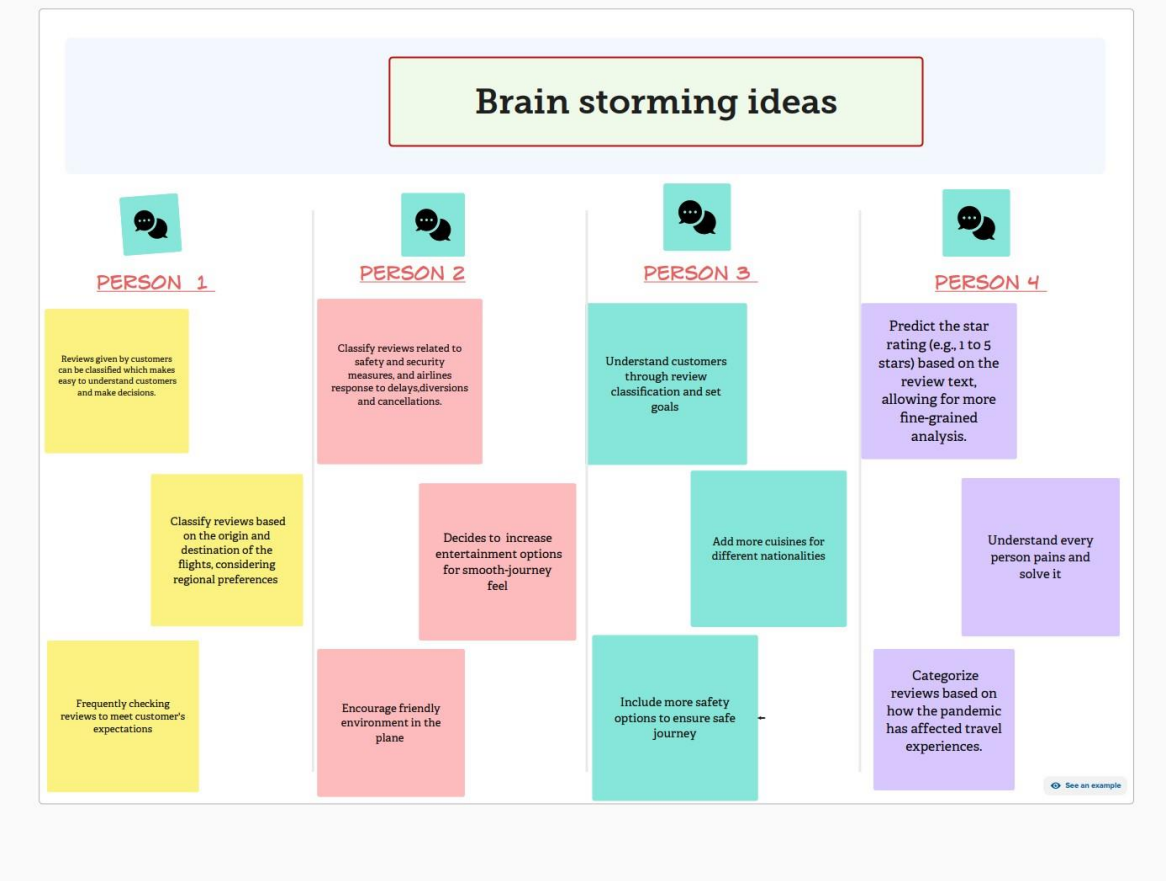
The project aims to address the challenge of effectively classifying airline reviews to provide users with personalized recommendations regarding the suitability of different airlines. Existing reviews are often subjective and diverse, making it difficult for users to quickly assess and choose an airline based on their preferences. The goal is to leverage machine learning algorithms to create a system that can accurately classify reviews based on various input factors. The project seeks to enhance the reliability and applicability of airline review classification systems, simplifying the decision-making process for users in selecting airlines for their travel experiences.

3.IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation and Brainstorming



4. REQUIREMENT ANALYSIS

4.1 Functional Requirement

- User Interface - For User Input Details and prediction
- Integration with XGBoost Model
- Real-time Classification- Instant feedback
- Display Output- Whether recommended or not.

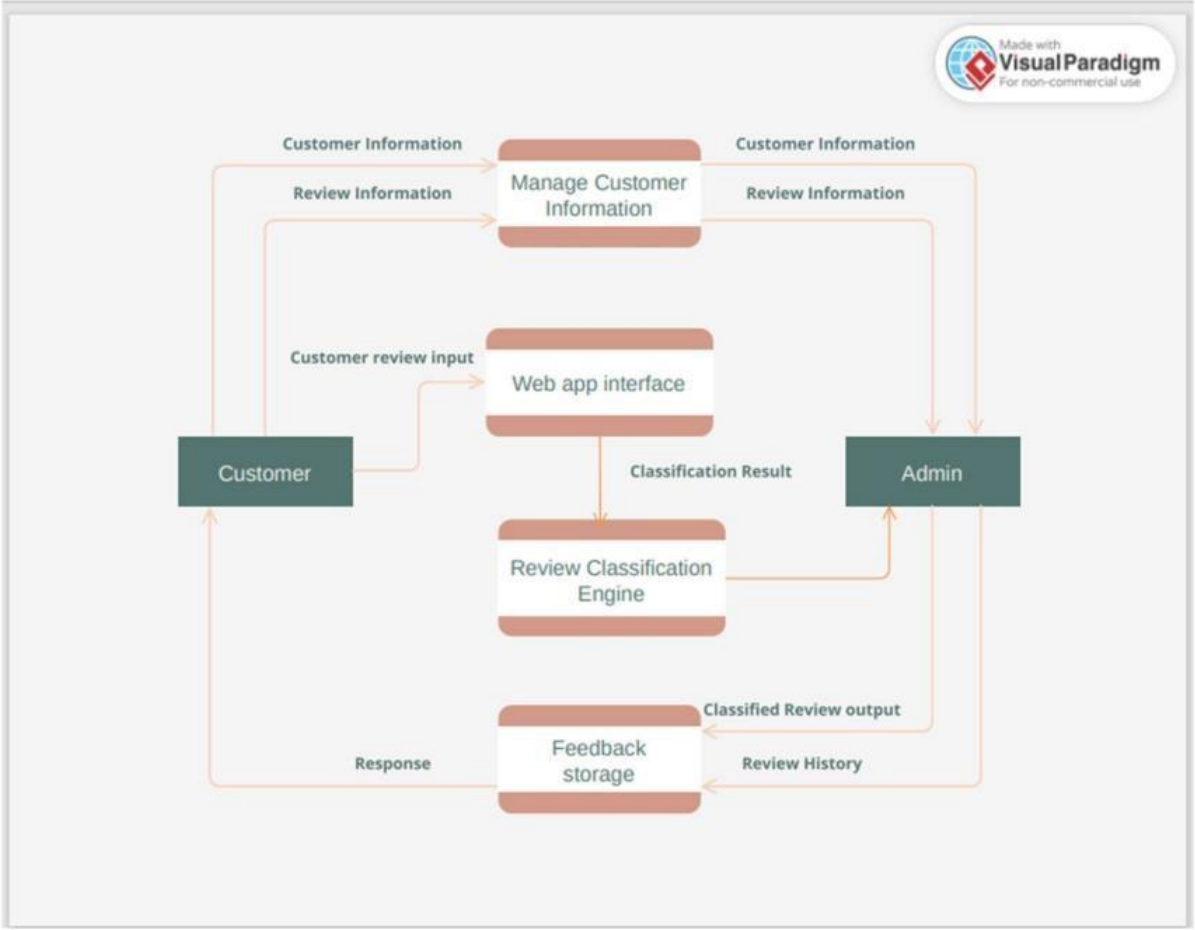
4.2 Non-Functional Requirements

- Usability – User-friendly and Intuitive Interface
- High Performance - The system process user input and provide classification results within a reasonable and acceptable time frame, minimizing wait times for users.
- Scalability – Handles potential user traffic
- Reliability – Accurately classify reviews

5. PROJECT DESIGN

5.1 Data Flow diagrams and User stories

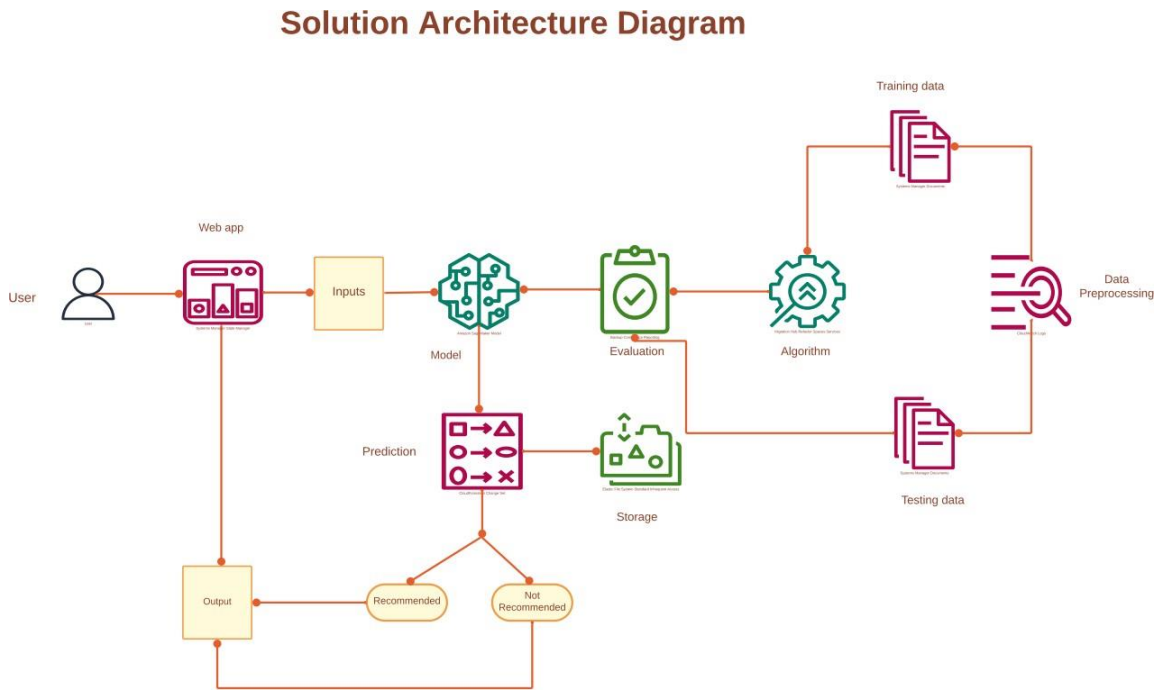
Data Flow Diagram-



User Stories-

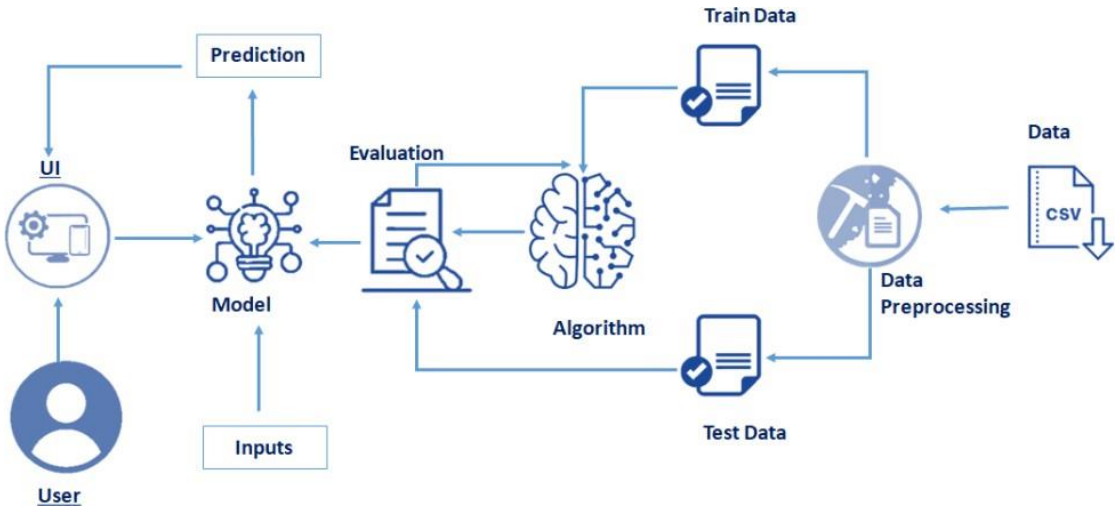
User Story Number	User Type	Functional Requirement	User Story	Acceptance Criteria	Priority
CUS-1	Customer (Mobile User)	Submission	As a customer, I can submit a review through the mobile app.	The app provides a user-friendly interface for entering and submitting reviews.	High
CUS-2			As a customer, I can attach images or files to support my review submission.	This app allows me to upload and associate images/files with my review.	Medium
USER-3	Customer(Web user)	Dashboard	As a web user, I can view a history of my submitted reviews on the web application.	Displays a list of past reviews	Medium
ADM-1	Administrator		As an administrator, I can monitor and update the machine learning model.	Admin panel allows me to view model performance.	High

5.2 Solution Architecture



6. PROJECT PLANNING AND SCHEDULING

6.1 Technical Architecture



6.2 Spirit Planning and Estimation

Technology Stack (Architecture & Stack)

Table1:Components and Technologies

S.NO.	Component	Description	Technologies used
1	User Interface	How user interacts with the application	HTML, CSS, JavaScript / Angular Js / React Js, etc
2	Text Data Collection	Gathering airline reviews data	Web Scraping, API (e.g., social media platforms)
3	Data preprocessing	Cleaning and preprocessing of raw data	Natural Language Processing (NLP) tools (e.g., NLTK, SpaCy)
4	Feature Extraction	Extracting features from text data	Word Embeddings (e.g., Word2Vec, GloVe)
5	Machine Learning Model	Classification algorithm for reviews	Scikit-learn, TensorFlow, PyTorch
6	Model Evaluation	Assessing model performance using metrics	Scikit-learn
7	Web Application	Interface for users to interact with the model	Flask, Django
8	Containerisation	Packaging application and dependencies into containers	Docker
9	Cloud Platform	Deployment on a cloud platform for scalability	AWS, Azure, Google Cloud
10	Monitoring and Logging	Monitoring application behavior and logging	Prometheus, Grafana
11	CI/CD Pipeline	Automating testing and deployment processes	Git, Jenkins, GitLab CI
12	Security	Implementing authentication and data encryption	SSL/TLS, Secure APIs |

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1	Open-Source Frameworks	List the open-source frameworks used	Scikit-learn, TensorFlow, PyTorch, NLTK, SpaCy
2	Security Implementations	List all the security/access controls implemented, use of firewalls, etc.	Encryption(e.g.,SSL/TLS),IAM Controls,OWASP,SHA-256, Secure APIs
3	Scalable Architecture	Justify the scalability of architecture (3-tier, Micro-services)	Microservices architecture, Docker, Kubernetes
4	Availability	ustify the availability of the application (e.g., use of load balancers, distributed servers	LoadBalancers, Redundancy, Failover Mechanisms
5	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's	Caching Strategies, Content Delivery Networks (CDN), LoadTesting, Performance Monitoring

Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)						
sprint	Functional Requirement	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Jyothsna, sahana
Sprint-2		USN-2	As a user, I will receive a confirmation email once I have registered for the application.	1	High	Jyothsna, sahana
Sprint-1		USN-3	As a user, I can register for the application through Facebook.	2	low	Jyothsna, sahana
Sprint-1		USN-4	As a user, I can register for the application through Gmail.	2	Medium	Jyothsna, sahana
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password.	1	High	Jyothsna, sahana

6.3 Spirit Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	10 days	31oct 2023	2 oct 2023	10	29oct 2023
Sprint-2	20	10 days	3 Nov 2023	6 Nov 2023	20	1 Nov 2023
Sprint-3	20	12 days	7 Nov 2023	12Nov2023	10	5 Nov 2023
Sprint-4	20	15 days	15Nov2023	20Nov2023	10	11Nov2023

7. CODING AND SOLUTIONING

7.1 Feature 1

Dynamic User Input form

A significant feature of the project is the implementation of a dynamic user input form. This form allows users to input various aspects of their airline experience, such as the airline name, seat type, type of traveller, origin, destination, month flown, year flown, and verification status. The HTML code dynamically generates input fields, providing a user-friendly interface for capturing diverse inputs.

7.2 Feature 2

Real time Classification and Result Display

Upon clicking the "Check It" button, the system processes the user-input data, leverages the trained XGBoost model for classification, and promptly displays the recommendation result on the user interface. This feature enhances user engagement and provides instant feedback, contributing to the overall effectiveness of the application.

8. PERFOMANCE TESTING

8.1 Performance Metrics

1) Decision Tree

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curve,auc

fpr_dt,tpr_dt,threshold_dt=roc_curve(y_test,pred_dt)

print(classification_report(y_test,pred_dt))


roc_auc_dt=auc(fpr_dt,tpr_dt)
print("roc_auc_dt :",roc_auc_dt)

cm_dt=confusion_matrix(y_test,pred_dt)
print("cm_dt:",cm_dt)

as_dt=accuracy_score(y_test,pred_dt)
print("as_dt:",as_dt)
```

```
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      3116
           1       0.95      0.95      0.95      3030

    accuracy                           0.95      6146
   macro avg       0.95      0.95      0.95      6146
weighted avg       0.95      0.95      0.95      6146

roc_auc_dt : 0.9508299546257578
cm_dt: [[2970  146]
 [ 156 2874]]
as_dt: 0.9508623494956069
```

2) K-Nearest Neighbours

In [97]:
```
fpr_knn,tpr_knn,threshold_knn=roc_curve(y_test,pred_knn)
print(classification_report(y_test,pred_knn))
```

```
              precision    recall  f1-score   support

           0       0.94      0.93      0.94      3116
           1       0.93      0.94      0.93      3030

    accuracy                           0.94      6146
   macro avg       0.94      0.94      0.94      6146
weighted avg       0.94      0.94      0.94      6146
```

In [98]:
```
roc_auc_knn=auc(fpr_knn,tpr_knn)
print("roc_auc_knn :",roc_auc_knn)

cm_knn = confusion_matrix(y_test,pred_knn)
print("confusion_matrix :",cm_knn)

as_knn = accuracy_score(y_test,pred_knn)
print("Accuracy score :",as_knn)
```

```
roc_auc_knn : 0.9352104754763024
confusion_matrix : [[2885  231]
 [ 168 2862]]
Accuracy score : 0.9350797266514806
```

3) Logistic Regression

In [102]:
```
## sklearn.metrics
from sklearn.metrics import classification_report,confusion_matrix,roc_curve,auc
fpr_lr,tpr_lr,threshold_lr = roc_curve(y_test,pred_lr)

print(classification_report(y_test,pred_lr))
```

```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92      3116
           1       0.91      0.93      0.92      3030

    accuracy                           0.92      6146
   macro avg       0.92      0.92      0.92      6146
weighted avg       0.92      0.92      0.92      6146
```

In [103]:
```
roc_auc_lr=auc(fpr_lr,tpr_lr)
print("roc_auc_lpr :",roc_auc_lr)

cm_lr = confusion_matrix(y_test,pred_lr)
print("confusion_matrix :",cm_lr)

as_lr = accuracy_score(y_test,pred_lr)
print("Accuracy score :",as_lr)
```

```
roc_auc_lpr : 0.9196978651652071
confusion_matrix : [[2849  267]
 [ 227 2803]]
Accuracy score : 0.919622518711357
```

4) Naïve Bayes Classification

In [107]:
```
## sklearn.metrics
from sklearn.metrics import classification_report,confusion_matrix,roc_curve,auc
fpr_gnb,tpr_gnb,threshold_gnb = roc_curve(y_test,pred_gnb)

print(classification_report(y_test,pred_gnb))
```

```
              precision    recall  f1-score   support

           0       0.93      0.88      0.90      3116
           1       0.88      0.93      0.90      3030

    accuracy                           0.90      6146
   macro avg       0.90      0.90      0.90      6146
weighted avg       0.90      0.90      0.90      6146
```

In [108]:
```
roc_auc_gnb=auc(fpr_gnb,tpr_gnb)
print("roc_auc_gnb :",roc_auc_gnb)

cm_gnb = confusion_matrix(y_test,pred_gnb)
print("confusion_matrix :",cm_gnb)

as_gnb = accuracy_score(y_test,pred_gnb)
print("Accuracy score :",as_gnb)
```

```
roc_auc_gnb : 0.9032067006443905
confusion_matrix : [[2738  378]
 [ 219 2811]]
Accuracy score : 0.9028636511552229
```

5) Random Forest Classification

In [112]:
```
## sklearn.metrics
from sklearn.metrics import classification_report,confusion_matrix,roc_curve,auc
fpr_rfc,tpr_rfc,threshold_rfc = roc_curve(y_test,pred_rfc)

print(classification_report(y_test,pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.98      0.54      0.70      3116
           1       0.68      0.99      0.81      3030

    accuracy                           0.76      6146
   macro avg       0.83      0.77      0.75      6146
weighted avg       0.83      0.76      0.75      6146
```

In [113]:
```
roc_auc_rfc=auc(fpr_rfc,tpr_rfc)
print("roc_auc_rfc :",roc_auc_rfc)

cm_rfc = confusion_matrix(y_test,pred_rfc)
print("confusion_matrix :",cm_rfc)

as_rfc = accuracy_score(y_test,pred_rfc)
print("Accuracy score :",as_rfc)
```

```
roc_auc_rfc : 0.7675142032816888
confusion_matrix : [[1698 1418]
 [  30 3000]]
Accuracy score : 0.7643996095021152
```

6) Support Vector Machine

In [117]:
```
## sklearn.metrics
from sklearn.metrics import classification_report,confusion_matrix,roc_curve,auc
fpr_svc,tpr_svc,threshold_svc = roc_curve(y_test,pred_svc)

print(classification_report(y_test,pred_svc))
```

```
              precision    recall  f1-score   support

           0       0.92      0.95      0.94      3116
           1       0.95      0.92      0.93      3030

    accuracy                           0.94      6146
   macro avg       0.94      0.94      0.94      6146
weighted avg       0.94      0.94      0.94      6146
```

In [118]:
```
roc_auc_svc=auc(fpr_svc,tpr_svc)
print("roc_auc_svc :",roc_auc_svc)

cm_svc = confusion_matrix(y_test,pred_svc)
print("confusion_matrix :",cm_svc)

as_svc = accuracy_score(y_test,pred_svc)
print("Accuracy score :",as_svc)
```

```
roc_auc_svc : 0.9364689646114804
confusion_matrix : [[2971  145]
 [ 244 2786]]
Accuracy score : 0.9367068011714936
```

7) XGBoost Classfier

In [122]:
```
## sklearn.metrics
from sklearn.metrics import classification_report,confusion_matrix,roc_curve,auc
fpr_xgb,tpr_xgb,threshold_xgb = roc_curve(y_test,pred_xgb)

print(classification_report(y_test,pred_xgb))
```

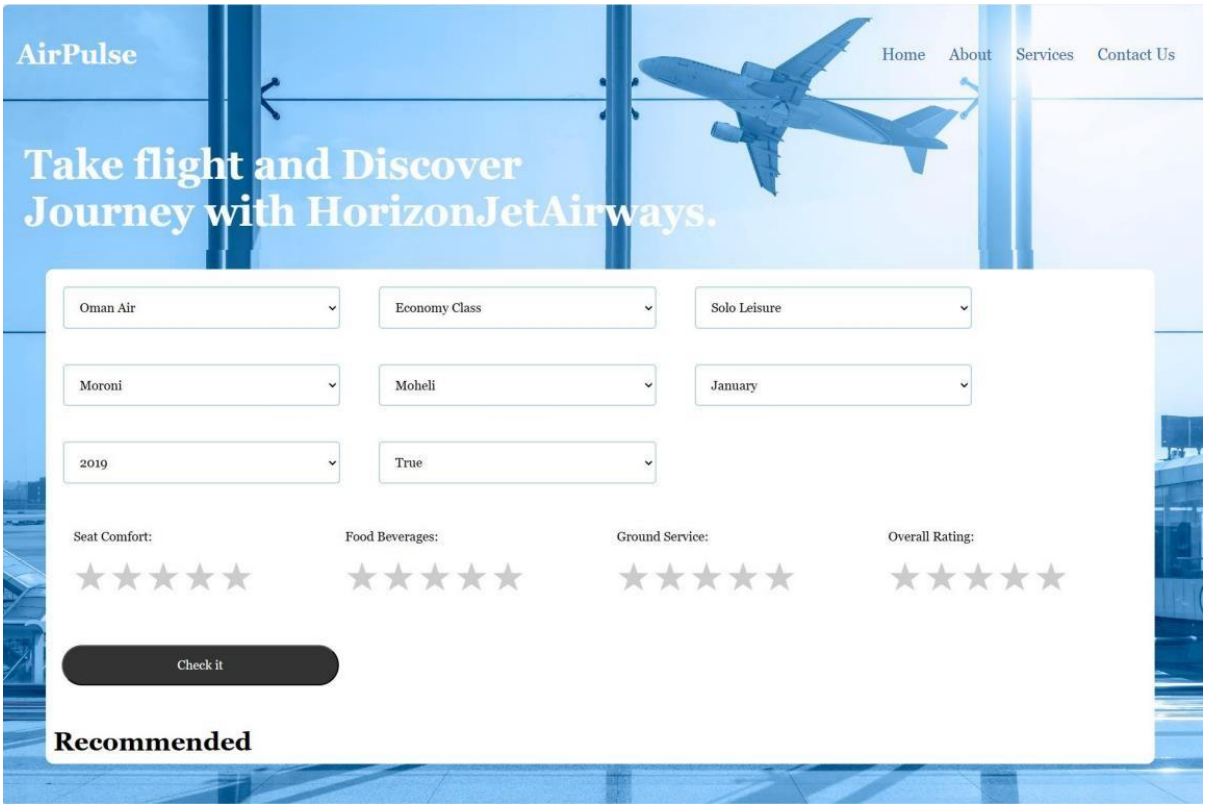
```
              precision    recall  f1-score   support

           0       0.99      0.73      0.84      3116
           1       0.78      0.99      0.87      3030

    accuracy                           0.86      6146
   macro avg       0.89      0.86      0.86      6146
weighted avg       0.89      0.86      0.86      6146
```

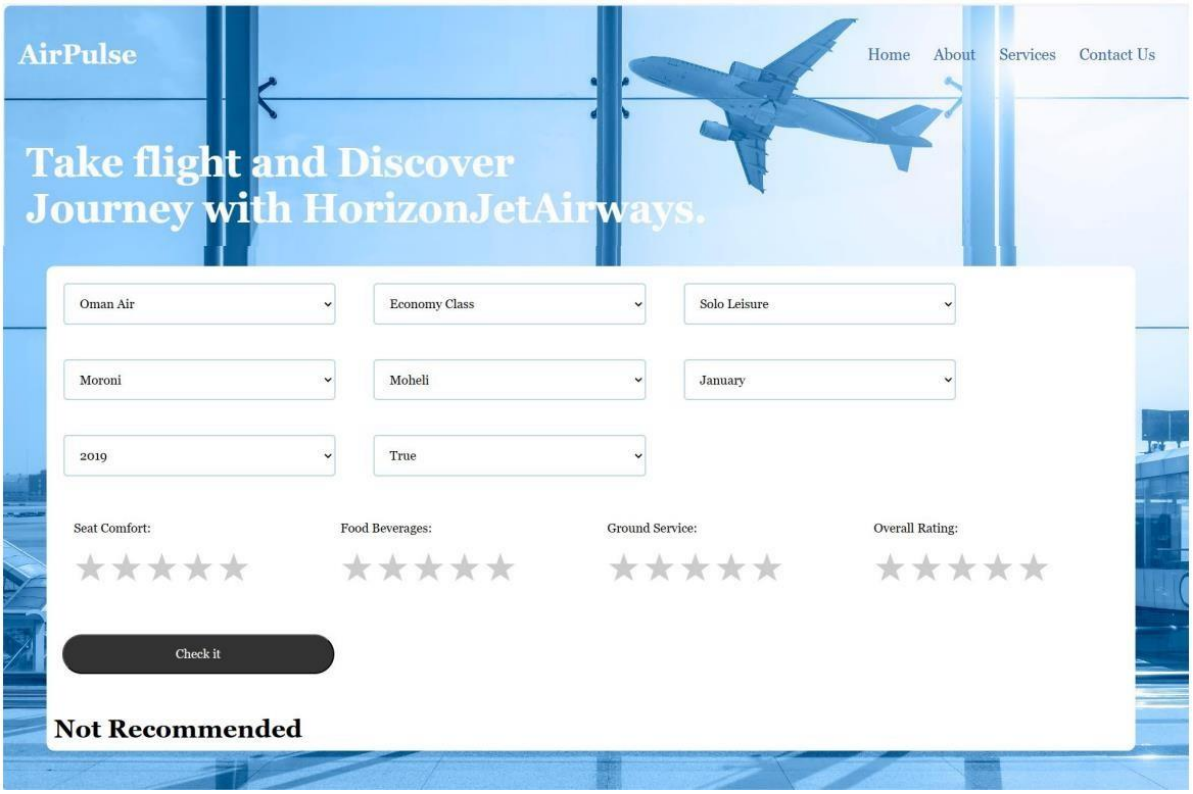
In [123]:
```
roc_auc_xgb=auc(fpr_knn,tpr_xgb)
print("roc_auc_xgb :",roc_auc_xgb)

cm_xgb = confusion_matrix(y_test,pred_xgb)
print("confusion_matrix :",cm_xgb)

as_xgb = accuracy_score(y_test,pred_xgb)
print("Accuracy score :",as_xgb)
```

```
roc_auc_xgb : 0.9584778022089757
confusion_matrix : [[2281  835]
 [  27 3003]]
Accuracy score : 0.8597461763748779
```

9. RESULTS

9.1 Outputs





10. ADVANTAGES AND DISADVANTAGES

Advantages

- User Empowerment: The web app empowers users to make informed decisions by providing personalized recommendations based on their unique input and experiences with airlines.
- Real-time Feedback: The real-time classification feature ensures users receive instant feedback, enhancing the user experience and facilitating quick decision-making.
- Scalability: The system is designed to scale, accommodating a growing user base.

Disadvantages

- Over-reliance on XGBoost: While XGBoost demonstrates high accuracy, over-reliance on a single classifier may limit the system's adaptability to diverse datasets and scenarios.
- Limited Feature Set: The current feature set focuses on specific aspects of airline reviews; expanding and refining features may be necessary for a more comprehensive classification.
- Dependency on User Input: The accuracy of the recommendations heavily depends on the accuracy and completeness of user-provided data, which may vary.

11. CONCLUSION

In summary, the Airline Review Classification Web App strives to simplify the decision-making process for users by offering instant, personalized recommendations. The real-time feedback and scalability of the system contribute to its user-friendly nature, making it accessible to a broad audience. Overall, the project represents a step forward in merging machine learning capabilities with practical user interactions, aiming to enhance the way individuals choose and evaluate airline services.

12. FUTURE SCOPE

The Airline Review Classification Web App lays a foundation to further improve its capabilities and user experience.

- User Authentication
- Extended Feature Set
- Mobile Application Development
- Integration of Multiple Classifiers

13. APPENDIX

Source Code

1.Data collection and Preparation

Collect the dataset
Data Preparation
Exploratory Data Analysis

2. Descriptive Statistics

Visual Analysis

3. Model Building

Training the model in multiple algorithms
Testing the model

4. Performance Testing

Testing model with multiple evaluation metrics

5. Model Deployment

Save the best model
Integrate with Web Framework

In [158]:
```
pip install imblearn
```

```
Requirement already satisfied: imblearn in c:\users\ramir\anaconda3\lib\site-packages (0.0)
Requirement already satisfied: imbalanced-learn in c:\users\ramir\anaconda3\lib\site-packages (from imblearn) (0.11.
0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\ramir\anaconda3\lib\site-packages (from imbalanced-le
arn->imblearn) (2.2.0)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\ramir\anaconda3\lib\site-packages (from imbalanced-lea
rn->imblearn) (1.3.1)
Requirement already satisfied: numpy>=1.17.3 in c:\users\ramir\anaconda3\lib\site-packages (from imbalanced-learn->im
blearn) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\ramir\anaconda3\lib\site-packages (from imbalanced-learn->imb
learn) (1.9.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\ramir\anaconda3\lib\site-packages (from imbalanced-learn->im
blearn) (1.3.2)
Note: you may need to restart the kernel to use updated packages.
```

1.Collect the Dataset

Importing the libraries

In [13]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curve,auc
import pickle
from scipy import stats
```

In [14]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [15]:
```
data = pd.read_csv("C:\Internship\self\Airline_Reviews.csv")
```

In [16]:
```
data.head()
```

Out[16]:

	Unnamed: 0	Airline Name	Overall_Rating	Review_Title	Review Date	Verified	Review	Aircraft	Type Of Traveller	Seat Type	Route	Date Flown	Seat Comfort
0	0	AB Aviation	9	"pretty decent airline"	11th November 2019	True	Moroni to Moheli. Turned out to be a pretty ...	NaN	Solo Leisure	Economy Class	Moroni to Moheli	November 2019	4.0
1	1	AB Aviation	1	"Not a good airline"	25th June 2019	True	Moroni to Anjouan. It is a very small airline...	E120	Solo Leisure	Economy Class	Moroni to Anjouan	June 2019	2.0
2	2	AB Aviation	1	"flight was fortunately short"	25th June 2019	True	Anjouan to Dzaoudzi. A very small airline an...	Embraer E120	Solo Leisure	Economy Class	Anjouan to Dzaoudzi	June 2019	2.0
3	3	Adria Airways	1	"I will never fly again with Adria"	28th September 2019	False	Please do a favor yourself and do not fly wi...	NaN	Solo Leisure	Economy Class	Frankfurt to Pristina	September 2019	1.0
4	4	Adria Airways	1	"it ruined our last days of holidays"	24th September 2019	True	Do not book a flight with this airline! My fr...	NaN	Couple Leisure	Economy Class	Sofia to Amsterdam via Ljubljana	September 2019	1.0

In [17]:
```
data.sample()
```

Out[17]:

	Unnamed: 0	Airline Name	Overall_Rating	Review_Title	Review Date	Verified	Review	Aircraft	Type Of Traveller	Seat Type	Route	Date Flown	Seat Comfort	Cabin Staff Service
2921	2921	Air Tahiti Nui	7	"the airline did its job"	11th January 2016	False	The A340 was Nuku Hiva, the youngest of the Ai...	A340-300	Couple Leisure	Business Class	PPT to LAX	December 2015	4.0	4.0

In [18]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23171 entries, 0 to 23170
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Unnamed: 0              23171 non-null  int64  
 1   Airline Name            23171 non-null  object 
 2   Overall_Rating          23171 non-null  object 
 3   Review_Title            23171 non-null  object 
 4   Review Date             23171 non-null  object 
 5   Verified                23171 non-null  bool   
 6   Review                  23171 non-null  object 
 7   Aircraft                7129 non-null   object 
 8   Type Of Traveller       19433 non-null  object 
 9   Seat Type               22075 non-null  object 
 10  Route                   19343 non-null  object 
 11  Date Flown              19417 non-null  object 
 12  Seat Comfort            19016 non-null  float64
 13  Cabin Staff Service     18911 non-null  float64
 14  Food & Beverages        14500 non-null  float64
 15  Ground Service          18378 non-null  float64
 16  Inflight Entertainment  10829 non-null  float64
 17  Wifi & Connectivity     5920 non-null   float64
 18  Value For Money         22105 non-null  float64
 19  Recommended             23171 non-null  object 
dtypes: bool(1), float64(7), int64(1), object(11)
memory usage: 3.4+ MB
```

In [19]:
```
data.describe()
```

Out[19]:

	Unnamed: 0	Seat Comfort	Cabin Staff Service	Food & Beverages	Ground Service	Inflight Entertainment	Wifi & Connectivity	Value For Money
count	23171.00000	19016.000000	18911.000000	14500.000000	18378.000000	10829.000000	5920.000000	22105.000000
mean	11585.00000	2.618321	2.871609	2.553586	2.353738	2.178964	1.780405	2.451120
std	6689.03588	1.464844	1.604631	1.526314	1.595747	1.488758	1.318800	1.594125
min	0.00000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	5792.50000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
50%	11585.00000	3.000000	3.000000	2.000000	1.000000	2.000000	1.000000	2.000000
75%	17377.50000	4.000000	4.000000	4.000000	4.000000	3.000000	2.000000	4.000000
max	23170.00000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000

In [20]:
```
data.shape
```

Out[20]:
```
(23171, 20)
```

2. Data Preparation--Handling missing values

In [21]:
```
data.isnull().any()
```

Out[21]:
```
Unnamed: 0                False
Airline Name              False
Overall_Rating            False
Review_Title              False
Review Date               False
Verified                  False
Review                    False
Aircraft                   True
Type Of Traveller          True
Seat Type                  True
Route                      True
Date Flown                 True
Seat Comfort               True
Cabin Staff Service        True
Food & Beverages           True
Ground Service             True
Inflight Entertainment     True
Wifi & Connectivity        True
Value For Money            True
Recommended               False
dtype: bool
```

In [22]:
```
data.isnull().sum()
```

Out[22]:
```
Unnamed: 0                  0
Airline Name                0
Overall_Rating              0
Review_Title                0
Review Date                 0
Verified                    0
Review                      0
Aircraft                16042
Type Of Traveller        3738
Seat Type                1096
Route                    3828
Date Flown               3754
Seat Comfort             4155
Cabin Staff Service      4260
Food & Beverages         8671
Ground Service           4793
Inflight Entertainment  12342
Wifi & Connectivity     17251
Value For Money          1066
Recommended                 0
dtype: int64
```

In [23]:
```
new_data = data.drop(['Inflight Entertainment','Wifi & Connectivity','Aircraft','Value For Money',
                      'Cabin Staff Service','Unnamed: 0','Review_Title','Review Date','Review'],axis=1)
```

In [24]:
```
new_data['Overall_Rating'].value_counts()
```

Out[24]:
```
1     11595
2      2296
9      1768
8      1757
3      1356
7      1192
4       859
n       842
5       830
6       676
Name: Overall_Rating, dtype: int64
```

In [25]:
```
new_data['Overall_Rating'] = new_data['Overall_Rating'].replace(['1','2','3','4','5','6','7','8','9','n'],['1','2','3'
```

In [26]:
```
new_data['Overall_Rating'].value_counts()
```

Out[26]:
```
1     11595
2      2296
9      1768
8      1757
3      1356
7      1192
4       859
10      842
5       830
6       676
Name: Overall_Rating, dtype: int64
```

In [27]:
```
new_data.isnull().sum()
```

Out[27]:
```
Airline Name              0
Overall_Rating            0
Verified                  0
Type Of Traveller      3738
Seat Type              1096
Route                  3828
Date Flown             3754
Seat Comfort           4155
Food & Beverages       8671
Ground Service         4793
Recommended               0
dtype: int64
```

In [174]:
```
# filling null values with median and mode depending on the values(Median for numerical and mode for categorical)
```

In [28]:
```
new_data['Type Of Traveller'] = new_data['Type Of Traveller'].fillna(new_data['Type Of Traveller'].mode()[0])
new_data['Seat Type'] = new_data['Seat Type'].fillna(new_data['Seat Type'].mode()[0])
new_data['Route'] = new_data['Route'].fillna(new_data['Route'].mode()[0])
new_data['Date Flown'] = new_data['Date Flown'].fillna(new_data['Date Flown'].mode()[0])
new_data['Seat Comfort'] = new_data['Seat Comfort'].fillna(new_data['Seat Comfort'].mode()[0])
new_data['Food & Beverages'] = new_data['Food & Beverages'].fillna(new_data['Food & Beverages'].mode()[0])
new_data['Ground Service'] = new_data['Ground Service'].fillna(new_data['Ground Service'].mode()[0])
```

For the above columns we are using mode instead of median even though numeric values are present,because the column consists of categories(0 to 5).

In [29]:
```
new_data.isnull().sum()
```

Out[29]:
```
Airline Name              0
Overall_Rating            0
Verified                  0
Type Of Traveller         0
Seat Type                 0
Route                     0
Date Flown                0
Seat Comfort              0
Food & Beverages          0
Ground Service            0
Recommended               0
dtype: int64
```

In [30]:
```
new_data[['Month Flown','Year Flown']] = new_data['Date Flown'].str.split(expand=True)
```

Route column has 3 values i.e.. Place A to Place B via Place C,so inorder to chose we gave indices for Sofia as 0 and Amsterdam as 1 and then run the split function again to split 'via'.

In [31]:
```
new_data['Origin'] = new_data.Route.str.split(' to ',expand=True)[0]
new_data['Destination'] = new_data.Route.str.split(' to ',expand=True)[1]

new_data['Destination'] = new_data.Destination.str.split(' via ',expand=True)[0]
```

In [32]:
```
del new_data['Route']
del new_data['Date Flown']
```

In [34]:
```
new_data['Origin'] = new_data['Origin'].replace(['Tel Avivito Malta (MLA)','Bangalore toChennai','JFK toTLV via Baku',
                                                 'Edinburgh To Fuerteventura','Nuremburg toHamburg','Mumbai toJaipur','Sydney to-
                                                 'London Gatwick - Bangkok','SIN toi MFM','Jakartato Yogyakarta','Cardiff-Malta r
                                                 'GRR-ORD','LCY-FRA','NAP-RMF return','LEB-BOS','Bucharest-Brussels','Da Nang - H
                                                 'LHR-DXB','Dublin - Charlotte','Kansas City via Dallas Ft Worth','Sydney via Sin
                                                 'Geneva via Brussels','Nursultan via Dubai','Denpasar Medan via Jakarta',
                                                 'Auckland Denpasar via Sydney / Melbourne','Lima via Santiago','Manila via Los A
                                                 'Dar es Salaam via Kigali','Singapore via Sydney','Grand Rapidsvto Orlando via C
                                                 'Toronto via Varadero', 'Bangkok via Mumbai','A Coruna via Bilbao','LHR-DXB',
                                                  'Paris Orly Los Angeles','Newark Los Angeles','Honolulu Seattle','San {Paulo'],
                                                ['Tel Aviv(MLA)','Bangalore','JFK','Krabi','Hong Kong','Edinburgh','Nuremburg','
                                                  'Sydney','London Gatwick','SIN','Jakarta','Cardiff','KIV','GRR','LCY','NAP','LE
                                                  'Da Nang', 'New York', 'LHR', 'Dublin', 'Kansas City','Sydney','Geneva','Nursul
                                                  'Auckland Denpasar','Lima','Manila', 'Dar es Salaam', 'Singapore', 'Grand Rapid
                                                  'Toronto','Bangkok','A Coruna','LHR','Paris Orly','Newark','Honolulu','San Paul
```

In [35]:
```
new_data['Origin']
```

Out[35]:
```
0           Moroni
1           Moroni
2          Anjouan
3        Frankfurt
4           Sofia
           ...
23166      Bangkok
23167    Singapore
23168      Bangkok
23169        Tokyo
23170    Singapore
Name: Origin, Length: 23171, dtype: object
```

In [36]:
```
new_data['Destination']
```

Out[36]:
```
0             Moheli
1           Anjouan
2          Dzaoudzi
3           Pristina
4          Amsterdam
             ...
23166          Tokyo
23167          Tokyo
23168          Tokyo
23169    Los Angeles
23170          Tokyo
Name: Destination, Length: 23171, dtype: object
```

In [37]:
```
new_data.shape
```

Out[37]:
```
(23171, 13)
```

In [38]:
```
new_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23171 entries, 0 to 23170
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Airline Name       23171 non-null  object 
 1   Overall_Rating     23171 non-null  object 
 2   Verified           23171 non-null  bool   
 3   Type Of Traveller  23171 non-null  object 
 4   Seat Type          23171 non-null  object 
 5   Seat Comfort       23171 non-null  float64
 6   Food & Beverages   23171 non-null  float64
 7   Ground Service     23171 non-null  float64
 8   Recommended        23171 non-null  object 
 9   Month Flown        23171 non-null  object 
 10  Year Flown         23171 non-null  object 
 11  Origin             23171 non-null  object 
 12  Destination        23133 non-null  object 
dtypes: bool(1), float64(3), object(9)
memory usage: 2.1+ MB
```

In [39]:
```
new_data.isnull().sum()
```

Out[39]:
```
Airline Name          0
Overall_Rating        0
Verified              0
Type Of Traveller     0
Seat Type             0
Seat Comfort          0
Food & Beverages      0
Ground Service        0
Recommended           0
Month Flown           0
Year Flown            0
Origin                0
Destination          38
dtype: int64
```

In [40]:
```
j=0
row_num = [2172,3788,5112,5368,7000,8314,9107,10589,12993,17759,20572,
           20930,2225,2380,4339,5182,5785,6382,10991,12573,17051,21497,
          4293,6215,9787,10207,12372,13556,16022,17217,17732,18774,
          19462,20112,22449,11584,10001,12258,10886]
new_des = ['Malta','Chennai','TLV','Bangkok','Shanghai','Fuerteventura','Hamburg',
          'Jaipur','New York','Bangkok','MFM','Yogyakarta','Malta','LIS','ORD','FRA',
          'RMF','BOS','Brussels','Hong Kong','DXB','Charlotte','Dallas Ft Worth',
           'Brussels','Dubai','Jakarta','Sydney / Melborne','Santiago','Los Angeles','Kigali',
          'Sydney','Chicago','Varadero','Mumbai','Bilbao','Dallas','Los Angeles','Los Angeles','Seattle ']
for i in row_num:
    new_data.at[i,'Destination'] = new_des[j]
    j+=1
```

In [41]:
```
new_data.isnull().sum()
```

Out[41]:
```
Airline Name          0
Overall_Rating        0
Verified              0
Type Of Traveller     0
Seat Type             0
Seat Comfort          0
Food & Beverages      0
Ground Service        0
Recommended           0
Month Flown           0
Year Flown            0
Origin                0
Destination           0
dtype: int64
```

In [43]:
```
new_order = ['Airline Name','Seat Type','Type Of Traveller','Origin',
             'Destination','Month Flown','Year Flown','Verified','Seat Comfort',
             'Food & Beverages','Ground Service','Overall_Rating','Recommended']
```

In [44]:
```
# Reordering the columns of given data to our desired manner
ndata = new_data.reindex(columns=new_order)
```

In [45]:
```
ndata.head()
```

Out[45]:

	Airline Name	Seat Type	Type Of Traveller	Origin	Destination	Month Flown	Year Flown	Verified	Seat Comfort	Food & Beverages	Ground Service	Overall_Rating	Recommended
0	AB Aviation	Economy Class	Solo Leisure	Moroni	Moheli	November	2019	True	4.0	4.0	4.0	9	yes
1	AB Aviation	Economy Class	Solo Leisure	Moroni	Anjouan	June	2019	True	2.0	1.0	1.0	1	no
2	AB Aviation	Economy Class	Solo Leisure	Anjouan	Dzaoudzi	June	2019	True	2.0	1.0	1.0	1	no
3	Adria Airways	Economy Class	Solo Leisure	Frankfurt	Pristina	September	2019	False	1.0	1.0	1.0	1	no
4	Adria Airways	Economy Class	Couple Leisure	Sofia	Amsterdam	September	2019	True	1.0	1.0	1.0	1	no

In [46]:
```
ndata['Origin'].unique()
```

Out[46]:
```
array(['Moroni', 'Anjouan', 'Frankfurt', ..., 'BWA', 'Seuol', 'Narita '],
      dtype=object)
```

In [47]:
```
ndata['Origin'].nunique()
```

Out[47]:
```
2171
```

In [48]:
```
ndata.head()
```

Out[48]:

	Airline Name	Seat Type	Type Of Traveller	Origin	Destination	Month Flown	Year Flown	Verified	Seat Comfort	Food & Beverages	Ground Service	Overall_Rating	Recommended
0	AB Aviation	Economy Class	Solo Leisure	Moroni	Moheli	November	2019	True	4.0	4.0	4.0	9	yes
1	AB Aviation	Economy Class	Solo Leisure	Moroni	Anjouan	June	2019	True	2.0	1.0	1.0	1	no
2	AB Aviation	Economy Class	Solo Leisure	Anjouan	Dzaoudzi	June	2019	True	2.0	1.0	1.0	1	no
3	Adria Airways	Economy Class	Solo Leisure	Frankfurt	Pristina	September	2019	False	1.0	1.0	1.0	1	no
4	Adria Airways	Economy Class	Couple Leisure	Sofia	Amsterdam	September	2019	True	1.0	1.0	1.0	1	no

In [49]:
```
ndata['Year Flown']
```

Out[49]:
```
0        2019
1        2019
2        2019
3        2019
4        2019
         ...
23166    2022
23167    2022
23168    2022
23169    2022
23170    2022
Name: Year Flown, Length: 23171, dtype: object
```

3. Exploratory Data Analysis

Descriptive statistical

Visual Analysis

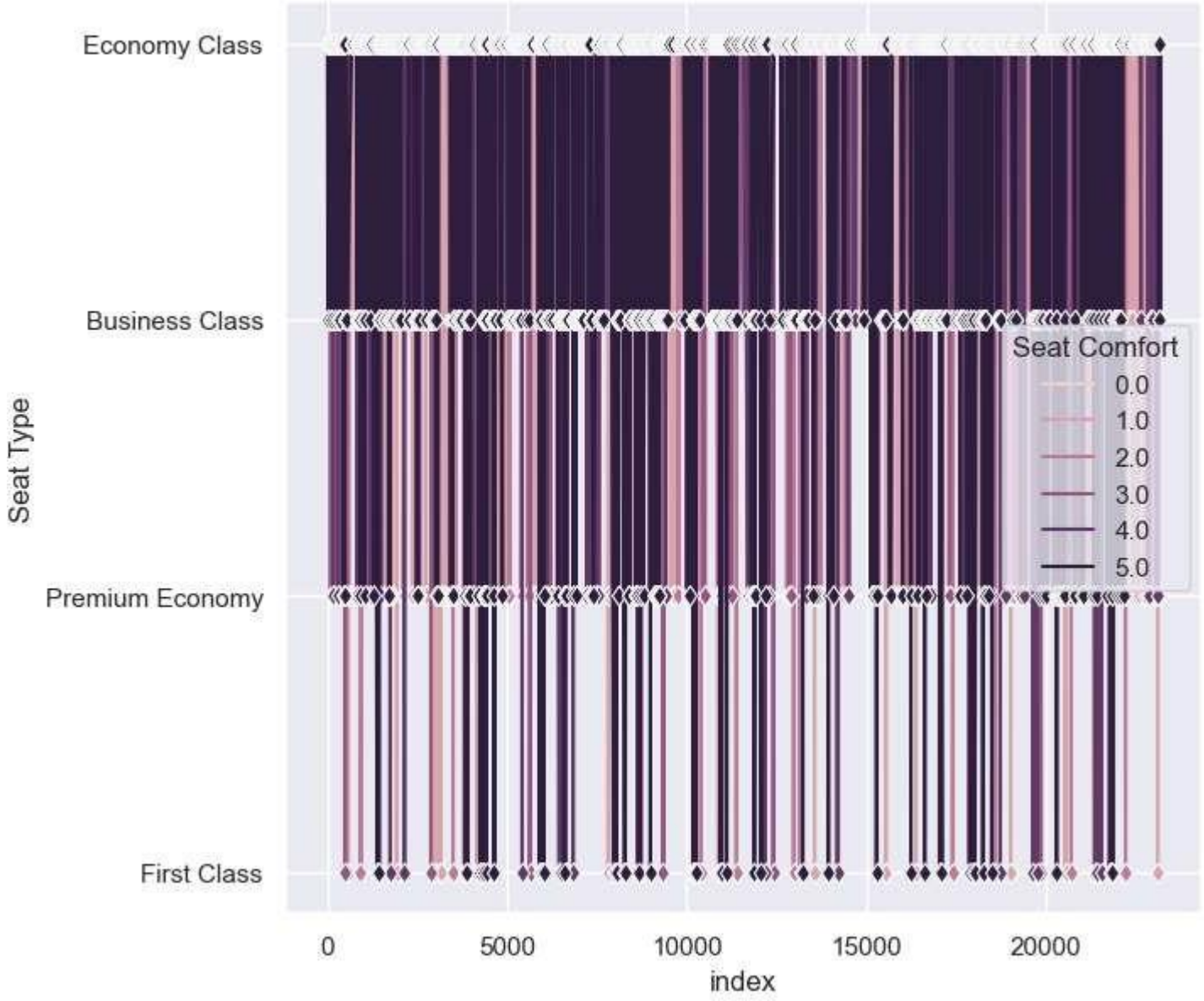
1) Univariate Analysis

2) Bivariate Analysis

3) Multivariate analysis

In [50]:
```
#Univariate analysis
##lineplot in seaborn
sns.set(rc={'figure.figsize':[7,7]})
sns.set(font_scale=1)
fig = sns.lineplot(x=ndata.index,y=ndata['Seat Type'],markevery=1,marker='d',
                   hue=ndata['Seat Comfort'])
fig.set(xlabel='index')
```

Out[50]:
```
[Text(0.5, 0, 'index')]
```

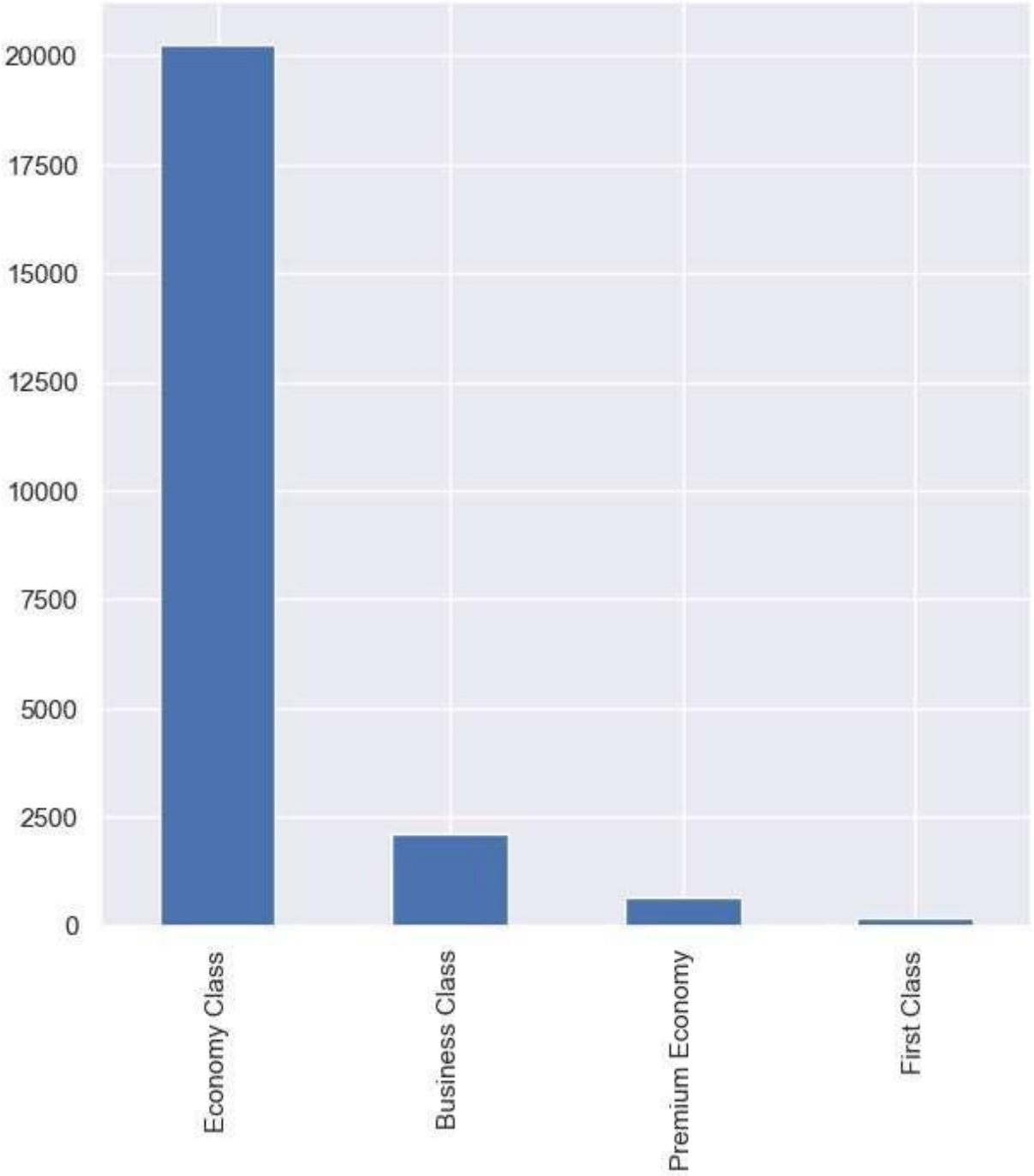

Economy Class
Business Class
Premium Economy
First Class
Seat Type
Seat Comfort
0.0
1.0
2.0
3.0
4.0
5.0
0
5000
10000
15000
20000
index

In [51]:
```
ndata['Seat Type'].value_counts()
```

Out[51]:
```
Economy Class      20241
Business Class      2098
Premium Economy      646
First Class          186
Name: Seat Type, dtype: int64
```

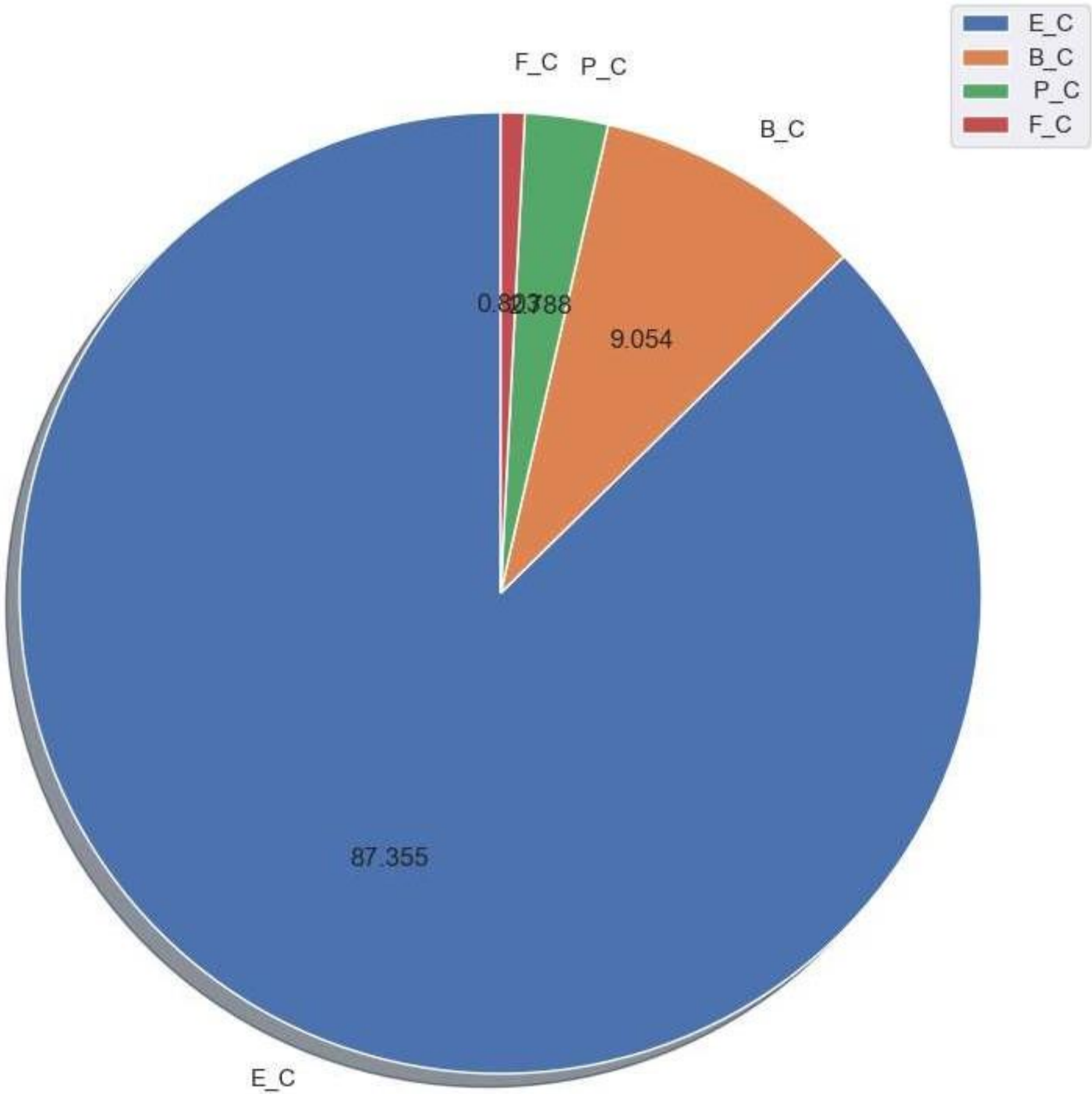
In [52]:
```
ndata['Seat Type'].value_counts().plot.bar()
```

Out[52]: <AxesSubplot:>


20000
17500
15000
12500
10000
7500
5000
2500
0
Economy Class
Business Class
Premium Economy
First Class

In [53]:

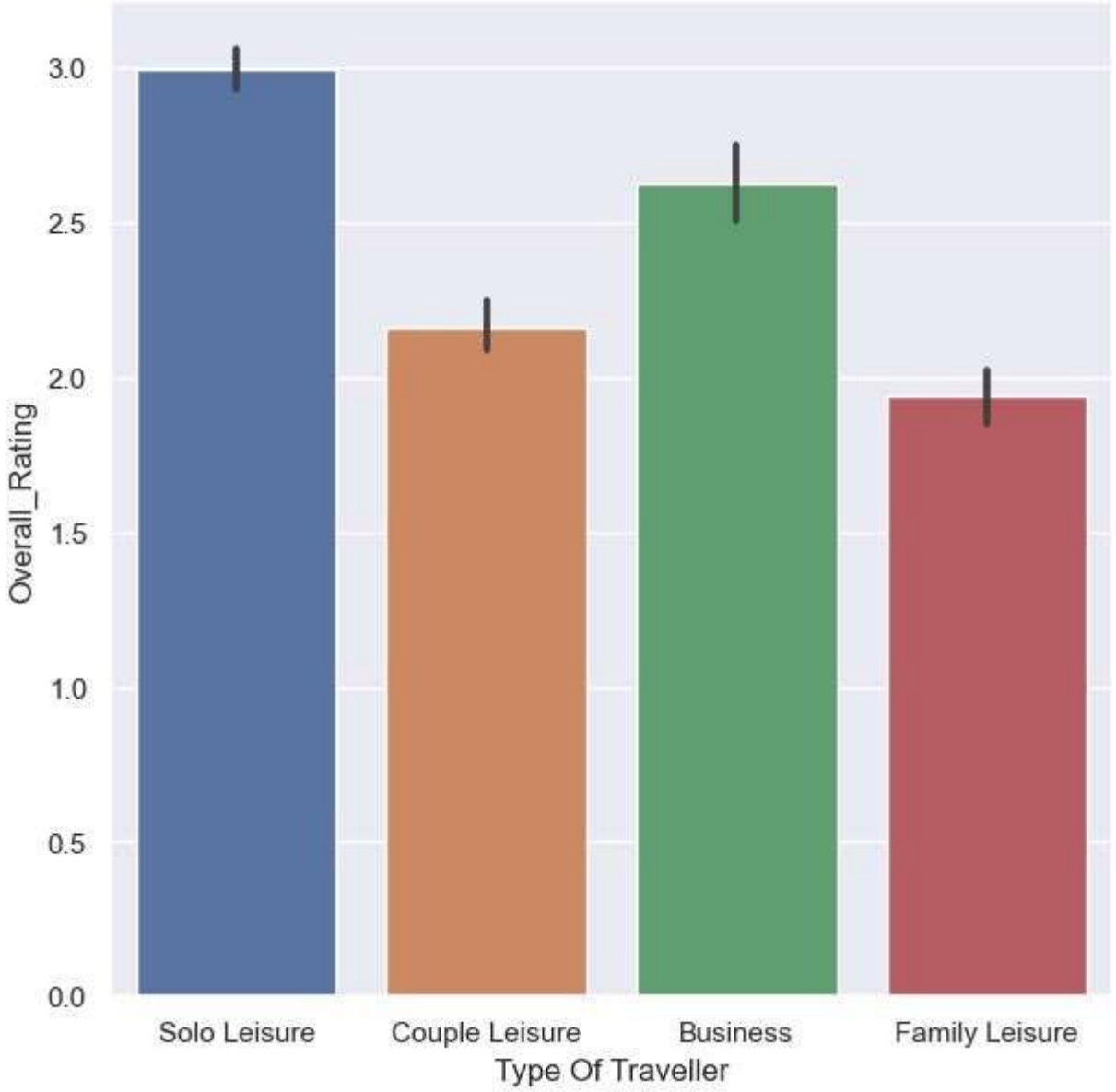
```
plt.figure(figsize=(10,10))
labels = ['E_C','B_C',' P_C','F_C']
plt.pie(ndata['Seat Type'].value_counts(),startangle=90,autopct='%.3f',
       labels=labels,shadow=True)
plt.legend(labels,loc='best')
#plt.axis('equal')
plt.show()
```


E_C
B_C
P_C
F_C
F_C
P_C
B_C
9.054
87.355
E_C

In [54]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
ndata['Overall_Rating'] = le.fit_transform(ndata['Overall_Rating'])
```

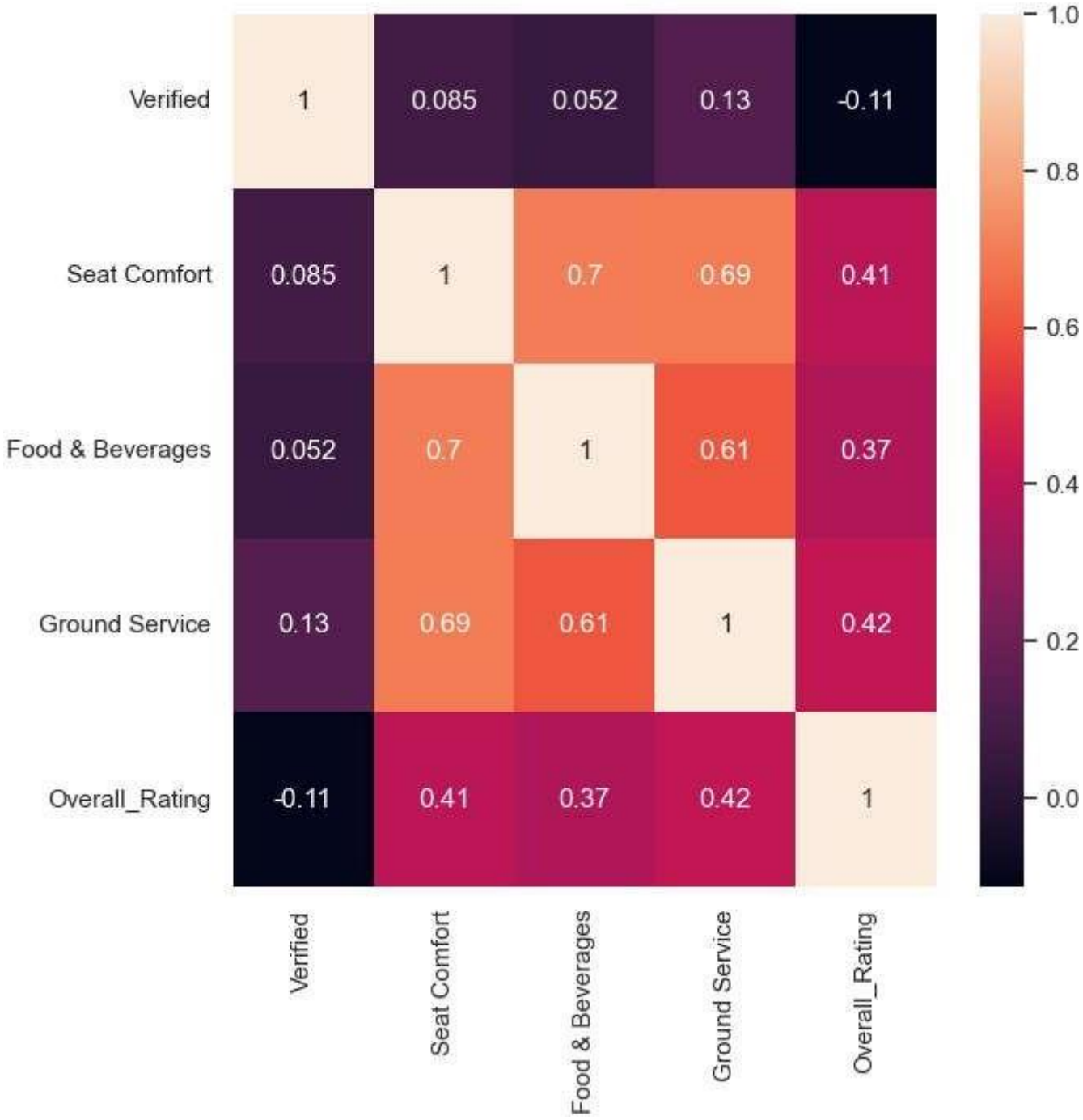
In [55]:
```
#Bivariate analysis
sns.barplot(data=ndata,x='Type Of Traveller',y='Overall_Rating')
```

Out[55]: <AxesSubplot:xlabel='Type Of Traveller', ylabel='Overall_Rating'>



In [56]:
```
#multivariate analysis
sns.heatmap(ndata.corr(),annot=True)
```

Out[56]: <AxesSubplot:>


Verified
Seat Comfort
Food & Beverages
Ground Service
Overall_Rating
1
0.085
0.052
0.13
-0.11
0.085
1
0.7
0.69
0.41
0.052
0.7
1
0.61
0.37
0.13
0.69
0.61
1
0.42
-0.11
0.41
0.37
0.42
1
Verified
Seat Comfort
Food & Beverages
Ground Service
Overall_Rating
1.0
0.8
0.6
0.4
0.2
0.0

4. Handling Categorical Values

In [57]:
```
ndata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23171 entries, 0 to 23170
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Airline Name       23171 non-null  object 
 1   Seat Type          23171 non-null  object 
 2   Type Of Traveller  23171 non-null  object 
 3   Origin             23171 non-null  object 
 4   Destination        23171 non-null  object 
 5   Month Flown        23171 non-null  object 
 6   Year Flown         23171 non-null  object 
 7   Verified           23171 non-null  bool   
 8   Seat Comfort       23171 non-null  float64
 9   Food & Beverages   23171 non-null  float64
 10  Ground Service     23171 non-null  float64
 11  Overall_Rating     23171 non-null  int32  
 12  Recommended        23171 non-null  object 
dtypes: bool(1), float64(3), int32(1), object(8)
memory usage: 2.1+ MB
```

In [58]:
```
ndata.head()
```

Out[58]:

	Airline Name	Seat Type	Type Of Traveller	Origin	Destination	Month Flown	Year Flown	Verified	Seat Comfort	Food & Beverages	Ground Service	Overall_Rating	Recommended
0	AB Aviation	Economy Class	Solo Leisure	Moroni	Moheli	November	2019	True	4.0	4.0	4.0	9	yes
1	AB Aviation	Economy Class	Solo Leisure	Moroni	Anjouan	June	2019	True	2.0	1.0	1.0	0	no
2	AB Aviation	Economy Class	Solo Leisure	Anjouan	Dzaoudzi	June	2019	True	2.0	1.0	1.0	0	no
3	Adria Airways	Economy Class	Solo Leisure	Frankfurt	Pristina	September	2019	False	1.0	1.0	1.0	0	no
4	Adria Airways	Economy Class	Couple Leisure	Sofia	Amsterdam	September	2019	True	1.0	1.0	1.0	0	no

In [59]:
```
ndata["Destination"]
```

Out[59]:
```
0             Moheli
1            Anjouan
2           Dzaoudzi
3           Pristina
4          Amsterdam
             ...
23166          Tokyo
23167          Tokyo
23168          Tokyo
23169    Los Angeles
23170          Tokyo
Name: Destination, Length: 23171, dtype: object
```

As we can see our dataset has categorical data,we must convert the categorical data,we must convert the categorical data to integer encoding or binary encoding. To convert,we use encoding techniques.

In this dataset,categorical features are Airline Name, Seat Type, Type Of Traveller, Origin,Destination, Month Flown, Year Flown, Verified, Overall_Rating, Recommended.

In [60]:
```
from sklearn.preprocessing import LabelEncoder
le1 = LabelEncoder()
le2 = LabelEncoder()
le3 = LabelEncoder()
le4 = LabelEncoder()
le5 = LabelEncoder()
le6 = LabelEncoder()
le7 = LabelEncoder()
le8 = LabelEncoder()
le9 = LabelEncoder()
le10 = LabelEncoder()
```

In [61]:
```
ndata['Airline Name'] = le1.fit_transform(ndata['Airline Name'])
ndata['Seat Type'] = le2.fit_transform(ndata['Seat Type'])
ndata['Type Of Traveller'] = le3.fit_transform(ndata['Type Of Traveller'])
ndata['Origin'] = le4.fit_transform(ndata['Origin'])
ndata['Destination'] = le5.fit_transform(ndata['Destination'])
ndata['Month Flown'] = le6.fit_transform(ndata['Month Flown'])
ndata['Year Flown'] = le7.fit_transform(ndata['Year Flown'])
ndata['Verified'] = le8.fit_transform(ndata['Verified'])
ndata['Overall_Rating'] = le9.fit_transform(ndata['Overall_Rating'])
ndata['Recommended'] = le10.fit_transform(ndata['Recommended'])
```

In [62]:
```
ndata.head()
```

Out[62]:

	Airline Name	Seat Type	Type Of Traveller	Origin	Destination	Month Flown	Year Flown	Verified	Seat Comfort	Food & Beverages	Ground Service	Overall_Rating	Recommended
0	0	1	3	1274	1545	9	6	1	4.0	4.0	4.0	9	1
1	0	1	3	1274	107	6	6	1	2.0	1.0	1.0	0	0
2	0	1	3	79	672	6	6	1	2.0	1.0	1.0	0	0
3	4	1	3	629	1927	11	6	0	1.0	1.0	1.0	0	0
4	4	1	1	1830	99	11	6	1	1.0	1.0	1.0	0	0

In [63]:
```
ndata.describe()
```

Out[63]:

	Airline Name	Seat Type	Type Of Traveller	Origin	Destination	Month Flown	Year Flown	Verified	Seat Comfort	Food Beverag
count	23171.000000	23171.000000	23171.000000	23171.000000	23171.000000	23171.000000	23171.000000	23171.000000	23171.000000	23171.000(
mean	245.027880	0.973242	2.008675	1073.719089	1451.342066	5.495188	7.705321	0.531785	2.328126	1.972:
std	144.313766	0.457584	1.077595	558.349459	800.585081	3.019861	2.512480	0.498999	1.465062	1.422:
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000(
25%	124.000000	1.000000	1.000000	616.000000	724.500000	3.000000	6.000000	0.000000	1.000000	1.000(
50%	237.000000	1.000000	2.000000	1224.000000	1494.000000	6.000000	9.000000	1.000000	2.000000	1.000(
75%	373.000000	1.000000	3.000000	1399.000000	2343.000000	7.000000	10.000000	1.000000	4.000000	3.000(
max	496.000000	3.000000	3.000000	2170.000000	2691.000000	11.000000	10.000000	1.000000	5.000000	5.000(

5.Model Building

1) Splitting dependent and independent variable

2) Check if data is imbalanced

3) Splitting data into training and testing

4) Training the model in multiple algorithms

5) Testing the model

Splitting dependent and independent variable

In [64]:
```
ndata.shape
```

Out[64]:
```
(23171, 13)
```

In [65]:
```
x=ndata.iloc[:,0:12].values
y=ndata.iloc[:,12:13].values
```

In [66]:
```
x
```

Out[66]:
```
array([[  0.,   1.,   3., ...,   4.,   4.,   9.],
       [  0.,   1.,   3., ...,   1.,   1.,   0.],
       [  0.,   1.,   3., ...,   1.,   1.,   0.],
       ...,
       [487.,   1.,   0., ...,   2.,   1.,   3.],
       [487.,   0.,   0., ...,   3.,   1.,   6.],
       [487.,   1.,   3., ...,   1.,   1.,   0.]])
```

In [67]:
```
y
```

Out[67]:
```
array([[1],
       [0],
       [0],
       ...,
       [0],
       [1],
       [0]])
```

Basically, in target variable some values repeat more often than the other kind of values. To remove that imbalanced data, we will use SMOTE(Synthetic Minority Over Sampling Technique).

In [68]:
```
ndata.Recommended.value_counts()
```

Out[68]:
```
0    15364
1     7807
Name: Recommended, dtype: int64
```

In [69]:
```
ndata["Origin"]
```

Out[69]:
```
0        1274
1        1274
2          79
3         629
4        1830
         ...
23166     187
23167    1819
23168     187
23169    1971
23170    1819
Name: Origin, Length: 23171, dtype: int32
```

In [70]:
```
#as the values are oversampling we need to use smote technique
from imblearn.over_sampling import SMOTE
smote=SMOTE(sampling_strategy='auto',random_state=1)
```

In [71]:
```
x,y=smote.fit_resample(x,y)
```

In [72]:
```
np.count_nonzero(y==1)
```

Out[72]:
```
15364
```

In [73]:
```
np.count_nonzero(y==0)
```

Out[73]:
```
15364
```

In [74]:
```
#splitting data into training and testing
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=1)
```

In [130]:
```
x_train
```

Out[130]:
```
array([[ 0.74796505,  0.08392531, -0.99411421, ...,  0.48408431,
         0.95942464,  0.49680964],
       [ 0.95757826,  1.93307645,  0.60257494, ...,  1.80798311,
         0.95942464, -0.95362688],
       [-0.60054658,  0.08392531,  0.92417925, ...,  1.14603371,
        -0.87372769, -0.95362688],
       ...,
       [-0.03459092,  0.08392531, -0.99411421, ...,  1.14603371,
         1.57047542, -0.95362688],
       [-1.66957394,  0.08392531, -0.03496748, ..., -0.83981449,
        -0.87372769, -0.37345227],
       [ 1.41872732,  0.08392531,  0.92417925, ..., -0.83981449,
        -0.87372769,  1.06338948]])
```

In [75]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

In [76]:
```
import pickle
pickle.dump(sc,open('ar_ss.pkl','wb'))
```

In [77]:
```
import pickle
pickle.dump(le1,open('le1.pkl','wb'))
```

In [78]:
```
import pickle
pickle.dump(le2,open('le2.pkl','wb'))
```

In [79]:
```
import pickle
pickle.dump(le3,open('le3.pkl','wb'))
```

In [80]:
```
import pickle
pickle.dump(le4,open('le4.pkl','wb'))
```

In [81]:
```
import pickle
pickle.dump(le5,open('le5.pkl','wb'))
```

In [82]:
```
import pickle
pickle.dump(le6,open('le6.pkl','wb'))
```

In [83]:
```
import pickle
pickle.dump(le7,open('le7.pkl','wb'))
```

In [84]:
```
import pickle
pickle.dump(le8,open('le8.pkl','wb'))
```

In [85]:
```
import pickle
pickle.dump(le9,open('le9.pkl','wb'))
```

In [86]:
```
import pickle
pickle.dump(le10,open('le10.pkl','wb'))
```

In [87]:
```
#training the model in multiple algorithms
```

1.Decision tree

In [88]:
```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(criterion='entropy',random_state=50)
```

In [89]:
```
dtc.fit(x_train,y_train)
```

Out[89]:
```
DecisionTreeClassifier(criterion='entropy', random_state=50)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [90]:
```
pred_dtc = dtc.predict(x_train)
pred_dtc
```

Out[90]:
```
array([0, 1, 1, ..., 1, 0, 1])
```

In [92]:
```
## sklearn.metrics
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curve,auc

fpr_dt,tpr_dt,threshold_dt = roc_curve(y_test,pred_dtc)
print(classification_report(y_test,pred_dtc))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8740\4242571298.py in <module>
      2 from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curve,auc
      3
----> 4 fpr_dt,tpr_dt,threshold_dt = roc_curve(y_test,pred_dtc)
      5 print(classification_report(y_test,pred_dtc))

~\anaconda3\lib\site-packages\sklearn\utils\_param_validation.py in wrapper(*args, **kwargs)
    209                     )
    210                 ):
--> 211                     return func(*args, **kwargs)
    212             except InvalidParameterError as e:
    213                 # When the function is just a wrapper around an estimator, we allow

~\anaconda3\lib\site-packages\sklearn\metrics\_ranking.py in roc_curve(y_true, y_score, pos_label, sample_weight,
drop_intermediate)
   1093     array([ inf, 0.8 , 0.4 , 0.35, 0.1 ])
   1094     """
```

In [93]:
```
roc_auc_dt = 0.952330
as_dt = 0.960432
```

In [ ]:
```
```

2. KNN

In [94]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
```

In [95]:
```
knn.fit(x_train,y_train)
```

Out[95]:
```
KNeighborsClassifier(n_neighbors=1)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [96]:
```
pred_knn=knn.predict(x_test)
pred_knn
```

Out[96]:
```
array([1, 0, 0, ..., 1, 1, 0])
```

In [97]:
```
fpr_knn,tpr_knn,threshold_knn=roc_curve(y_test,pred_knn)
print(classification_report(y_test,pred_knn))
```

```
              precision    recall  f1-score   support

           0       0.94      0.93      0.94      3116
           1       0.93      0.94      0.93      3030

    accuracy                           0.94      6146
   macro avg       0.94      0.94      0.94      6146
weighted avg       0.94      0.94      0.94      6146
```

In [98]:
```
roc_auc_knn=auc(fpr_knn,tpr_knn)
print("roc_auc_knn :",roc_auc_knn)

cm_knn = confusion_matrix(y_test,pred_knn)
print("confusion_matrix :",cm_knn)

as_knn = accuracy_score(y_test,pred_knn)
print("Accuracy score :",as_knn)
```

```
roc_auc_knn : 0.9352104754763024
confusion_matrix : [[2885  231]
 [ 168 2862]]
Accuracy score : 0.9350797266514806
```

3. Logistic Regression

In [99]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

In [100]:
```
lr.fit(x_train,y_train)
```

Out[100]:
```
LogisticRegression()
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [101]:
```
pred_lr = lr.predict(x_test)
pred_lr
```

Out[101]:
```
array([1, 0, 0, ..., 1, 1, 0])
```

In [102]:
```
## sklearn.metrics
from sklearn.metrics import classification_report,confusion_matrix,roc_curve,auc
fpr_lr,tpr_lr,threshold_lr = roc_curve(y_test,pred_lr)

print(classification_report(y_test,pred_lr))
```

```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92      3116
           1       0.91      0.93      0.92      3030

    accuracy                           0.92      6146
   macro avg       0.92      0.92      0.92      6146
weighted avg       0.92      0.92      0.92      6146
```

In [103]:
```
roc_auc_lr=auc(fpr_lr,tpr_lr)
print("roc_auc_lpr :",roc_auc_lr)

cm_lr = confusion_matrix(y_test,pred_lr)
print("confusion_matrix :",cm_lr)

as_lr = accuracy_score(y_test,pred_lr)
print("Accuracy score :",as_lr)
```

```
roc_auc_lpr : 0.9196978651652071
confusion_matrix : [[2849  267]
 [ 227 2803]]
Accuracy score : 0.919622518711357
```

4.Naive Bayes Classification

In [104]:
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
```

In [105]:
```
gnb.fit(x_train,y_train)
```

Out[105]:
```
GaussianNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [106]:
```
pred_gnb = gnb.predict(x_test)
pred_gnb
```

Out[106]:
```
array([1, 0, 0, ..., 1, 1, 0])
```

In [107]:
```
## sklearn.metrics
from sklearn.metrics import classification_report,confusion_matrix,roc_curve,auc
fpr_gnb,tpr_gnb,threshold_gnb = roc_curve(y_test,pred_gnb)

print(classification_report(y_test,pred_gnb))
```

```
              precision    recall  f1-score   support

           0       0.93      0.88      0.90      3116
           1       0.88      0.93      0.90      3030

    accuracy                           0.90      6146
   macro avg       0.90      0.90      0.90      6146
weighted avg       0.90      0.90      0.90      6146
```

In [108]:
```
roc_auc_gnb=auc(fpr_gnb,tpr_gnb)
print("roc_auc_gnb :",roc_auc_gnb)

cm_gnb = confusion_matrix(y_test,pred_gnb)
print("confusion_matrix :",cm_gnb)

as_gnb = accuracy_score(y_test,pred_gnb)
print("Accuracy score :",as_gnb)
```

```
roc_auc_gnb : 0.9032067006443905
confusion_matrix : [[2738  378]
 [ 219 2811]]
Accuracy score : 0.9028636511552229
```

5.Random Forest Classification

In [109]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
```

In [110]:
```
rfc.fit(x_train,y_train)
```

Out[110]:
```
RandomForestClassifier()
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [111]:
```
pred_rfc = rfc.predict(x_test)
```

In [112]:
```
## sklearn.metrics
from sklearn.metrics import classification_report,confusion_matrix,roc_curve,auc
fpr_rfc,tpr_rfc,threshold_rfc = roc_curve(y_test,pred_rfc)

print(classification_report(y_test,pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.98      0.54      0.70      3116
           1       0.68      0.99      0.81      3030

    accuracy                           0.76      6146
   macro avg       0.83      0.77      0.75      6146
weighted avg       0.83      0.76      0.75      6146
```

In [113]:
```
roc_auc_rfc=auc(fpr_rfc,tpr_rfc)
print("roc_auc_rfc :",roc_auc_rfc)

cm_rfc = confusion_matrix(y_test,pred_rfc)
print("confusion_matrix :",cm_rfc)

as_rfc = accuracy_score(y_test,pred_rfc)
print("Accuracy score :",as_rfc)
```

```
roc_auc_rfc : 0.7675142032816888
confusion_matrix : [[1698 1418]
 [  30 3000]]
Accuracy score : 0.7643996095021152
```

6. Support Vector Machine

In [114]:
```
from sklearn.svm import SVC
svc = SVC()
```

In [115]:
```
svc.fit(x_train,y_train)
```

Out[115]:
```
SVC()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [116]:
```
pred_svc = svc.predict(x_test)
```

In [117]:
```
## sklearn.metrics
from sklearn.metrics import classification_report,confusion_matrix,roc_curve,auc
fpr_svc,tpr_svc,threshold_svc = roc_curve(y_test,pred_svc)

print(classification_report(y_test,pred_svc))
```

```
              precision    recall  f1-score   support

           0       0.92      0.95      0.94      3116
           1       0.95      0.92      0.93      3030

    accuracy                           0.94      6146
   macro avg       0.94      0.94      0.94      6146
weighted avg       0.94      0.94      0.94      6146
```

In [118]:
```
roc_auc_svc=auc(fpr_svc,tpr_svc)
print("roc_auc_svc :",roc_auc_svc)

cm_svc = confusion_matrix(y_test,pred_svc)
print("confusion_matrix :",cm_svc)

as_svc = accuracy_score(y_test,pred_svc)
print("Accuracy score :",as_svc)
```

```
roc_auc_svc : 0.9364689646114804
confusion_matrix : [[2971  145]
 [ 244 2786]]
Accuracy score : 0.9367068011714936
```

7. XGBoost Classifier

In [119]:
```
from xgboost import XGBClassifier
xgb = XGBClassifier()
```

In [120]:
```
xgb.fit(x_train,y_train)
```

Out[120]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [121]:
```
pred_xgb = xgb.predict(x_test)
```

In [122]:
```
## sklearn.metrics
from sklearn.metrics import classification_report,confusion_matrix,roc_curve,auc
fpr_xgb,tpr_xgb,threshold_xgb = roc_curve(y_test,pred_xgb)

print(classification_report(y_test,pred_xgb))
```

```
              precision    recall  f1-score   support

           0       0.99      0.73      0.84      3116
           1       0.78      0.99      0.87      3030

    accuracy                           0.86      6146
   macro avg       0.89      0.86      0.86      6146
weighted avg       0.89      0.86      0.86      6146
```

In [123]:
```
roc_auc_xgb=auc(fpr_knn,tpr_xgb)
print("roc_auc_xgb :",roc_auc_xgb)

cm_xgb = confusion_matrix(y_test,pred_xgb)
print("confusion_matrix :",cm_xgb)

as_xgb = accuracy_score(y_test,pred_xgb)
print("Accuracy score :",as_xgb)
```

```
roc_auc_xgb : 0.9584778022089757
confusion_matrix : [[2281  835]
 [  27 3003]]
Accuracy score : 0.8597461763748779
```

In [124]:
```
# print("Accuracy score of KNN :",(as_knn)*100)
# print("Accuracy score of Logistic Regression :",(as_lr)*100)
# print("Accuracy score of Naive Bayes :",(as_gnb)*100)
# print("Accuracy score of RandomForest :",(as_rfc)*100)
# print("Accuracy score of SVC :",(as_svc)*100)
# print("Accuracy score of XGBoost :",(as_xgb)*100)
```

As SVC has more accuracy score compared to other algorithms,we test model with SVC algorithm

In [125]:
```
alg = pd.DataFrame({'Model':['Decision Tree Classification','K-Nearest NEighbours',
                             'Logistic Regression','Naive Bayes Classification',
                             'RandomForest Classification','Support Vector Machine','XGBClassifier'],
                    'roc_auc':[roc_auc_dt,roc_auc_knn,roc_auc_lr,roc_auc_gnb,
                               roc_auc_rfc,roc_auc_svc,roc_auc_xgb],
                   'accuracy':[as_dt,as_knn,as_lr,as_gnb,as_rfc,as_svc,as_xgb]})
alg
```

Out[125]:

	Model	roc_auc	accuracy
0	Decision Tree Classification	0.952330	0.960432
1	K-Nearest NEighbours	0.935210	0.935080
2	Logistic Regression	0.919698	0.919623
3	Naive Bayes Classification	0.903207	0.902864
4	RandomForest Classification	0.767514	0.764400
5	Support Vector Machine	0.936469	0.936707
6	XGBClassifier	0.958478	0.859746

In [126]:
```
maxa = 0
for i in range(len(alg['Model'])):
    if(alg.iloc[i:i+1,1:2]).values > maxa:
        maxa = (alg.iloc[i:i+1,1:2]).values
        model = (alg.iloc[i:i+1,0:1]).values
print(f'Best auc_roc is {maxa} by {model}')
maxa = 0
for i in range(len(alg['Model'])):
    if(alg.iloc[i:i+1,2:3]).values > maxa:
        maxa = (alg.iloc[i:i+1,2:3]).values
        model = (alg.iloc[i:i+1,0:1]).values
print(f'Best accuracy is {maxa} by {model}')
```

```
Best auc_roc is [[0.9584778]] by [['XGBClassifier']]
Best accuracy is [[0.960432]] by [['Decision Tree Classification']]
```

Since maximum auc_roc and maximum accuracy are given by two different algorithms. We will go with model with more auc_roc,because AUC_ROC considers the trade-offs between precision and recall,whereas Accuracy only considers the number of correct predictions.

6.Model Deployment

In [127]:
```
# Saving the best model
import pickle
pickle.dump(xgb,open('ar_xgb.pkl','wb'))
```

In [128]:
```
## Integrate with Web framework
```

In [ ]:

In [ ]:

Demo Video Link-

https://drive.google.com/file/d/1T-AiZBPXjKWHcvKzlk0TUfI5u_ZOH-yR/view?usp=sharing

Github link

smartinternz02/SI-GuidedProject-611986-1698824909 (github.com)