Project Report

Airline Review Classification Using Machine Learning

project focuses on the development of an airline review classification system using Classification models

Designed & Developed By:
UMMALETI KUMAR

1. INTRODUCTION

1.1 Project Overview

In the contemporary global landscape, the airline industry serves as a crucial enabler for international travel and business interactions. This project aims to develop a robust airline review classification system utilizing various Classification models such as Decision Tree Classifier, Random Forest Classifier, XGBoost Classifier, etc. The focus is on leveraging the abundance of user-generated content on social media platforms, travel websites, and online forums to extract actionable insights. By categorizing and analysing airline reviews, this system aims to provide valuable information to enhance airline services and elevate overall passenger satisfaction.

1.2 Purpose

The purpose of this project is to leverage the growing volume of user-generated content, particularly airline reviews on social media platforms, travel websites, and online forums, to enhance the airline industry's understanding of passenger sentiments and preferences. With the increasing accessibility of air travel, the quality of service provided by airlines has a profound impact on passenger experiences. This project aims to develop an airline review classification system utilizing advanced Classification models, including Decision Tree Classifier, Random Forest Classifier, XGBoost Classifier, among others.

2. LITERATURE SURVEY

2.1 Existing problem

The airline industry, amid its critical role as a global travel and business facilitator, faces several challenges related to understanding and addressing passenger experiences effectively. The surge in air travel accessibility has led to an unprecedented influx of user-generated content, especially in the form of airline reviews on social media platforms, travel websites, and online forums. However, a significant existing problem lies in the difficulty of extracting meaningful insights from this vast pool of unstructured text data.

2.2 References

- Kumar, S. and Zymbler, M., 2019. A machine learning approach to analyse customer satisfaction from airline tweets. *Journal of Big Data*, *6*(1), pp.1-16.

- Hasib, K. M., Towhid, N. A., & Alam, M. G. R. (2021, November). Online review based sentiment classification on Bangladesh airline service using supervised learning. In *2021 5th International*

Conference on Electrical Engineering and Information Communication Technology (ICEEICT) (pp. 1-6). IEEE.

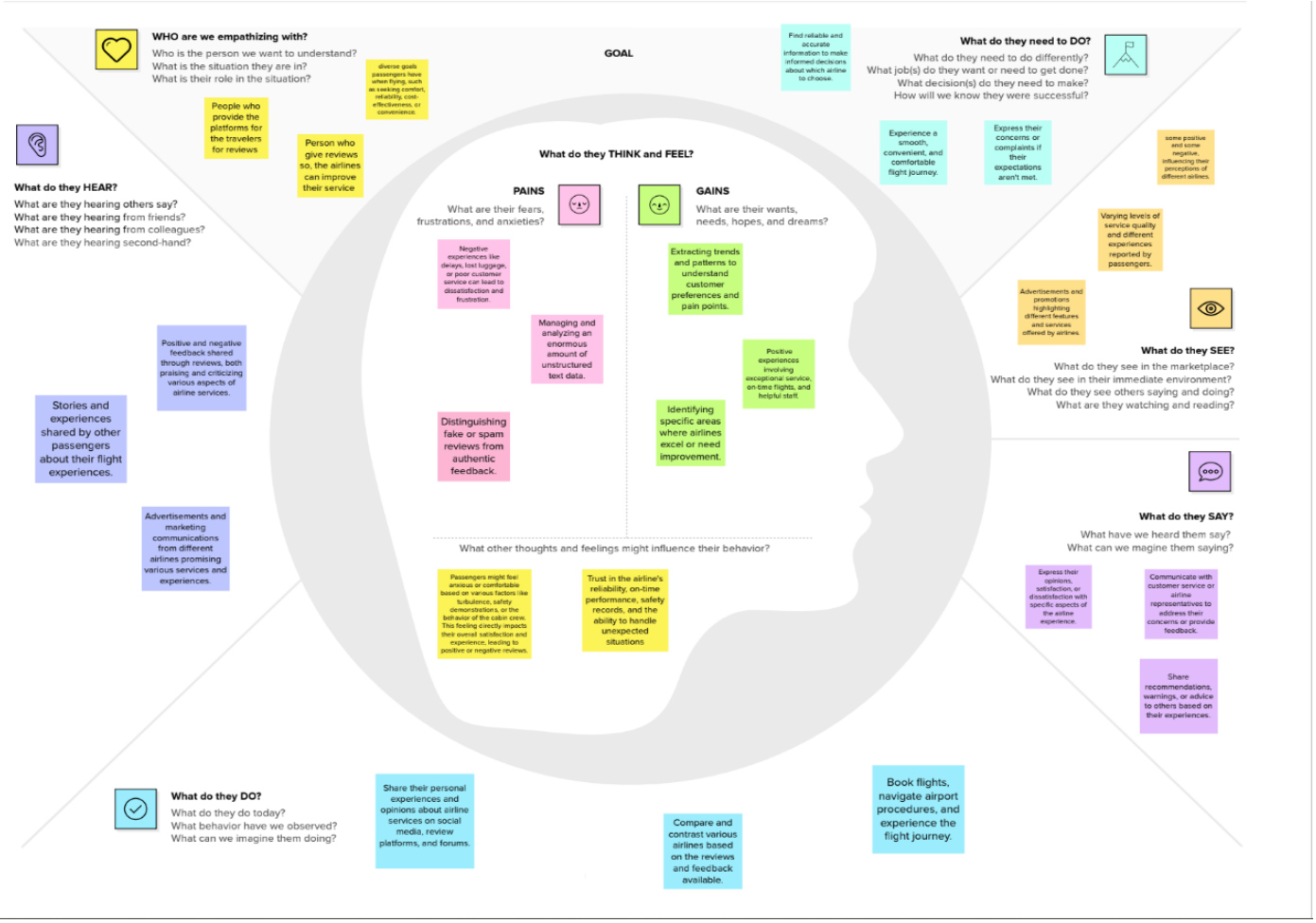
- Jeon, W., Lee, Y., & Geum, Y. (2022). Airline service quality evaluation based on customer review using machine learning approach and sentiment analysis. *Journal of Society for e-Business Studies*, *26*(4).

2.3 Problem Statement Definition

The aviation industry, serving as a pivotal driver for global travel and business, faces a substantial challenge in effectively harnessing the vast pool of unstructured user-generated content, specifically airline reviews from social media platforms, travel websites, and online forums. As air travel becomes more accessible, the quality of service offered by airlines becomes increasingly critical in shaping passenger experiences. The existing problem is the lack of a systematic and efficient method for extracting actionable insights from this rich textual data.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Template

Brainstorm & idea prioritization

Airline Review Classification Using Machine Learning
Our project aims to develop an airline review classification system utilizing machine learning models like Decision Tree, Random Forest, and XGBoost.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Team Members 1

B **Set the goal**
Goal is to train a Machine Learning model to classify the Airline Reviews

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

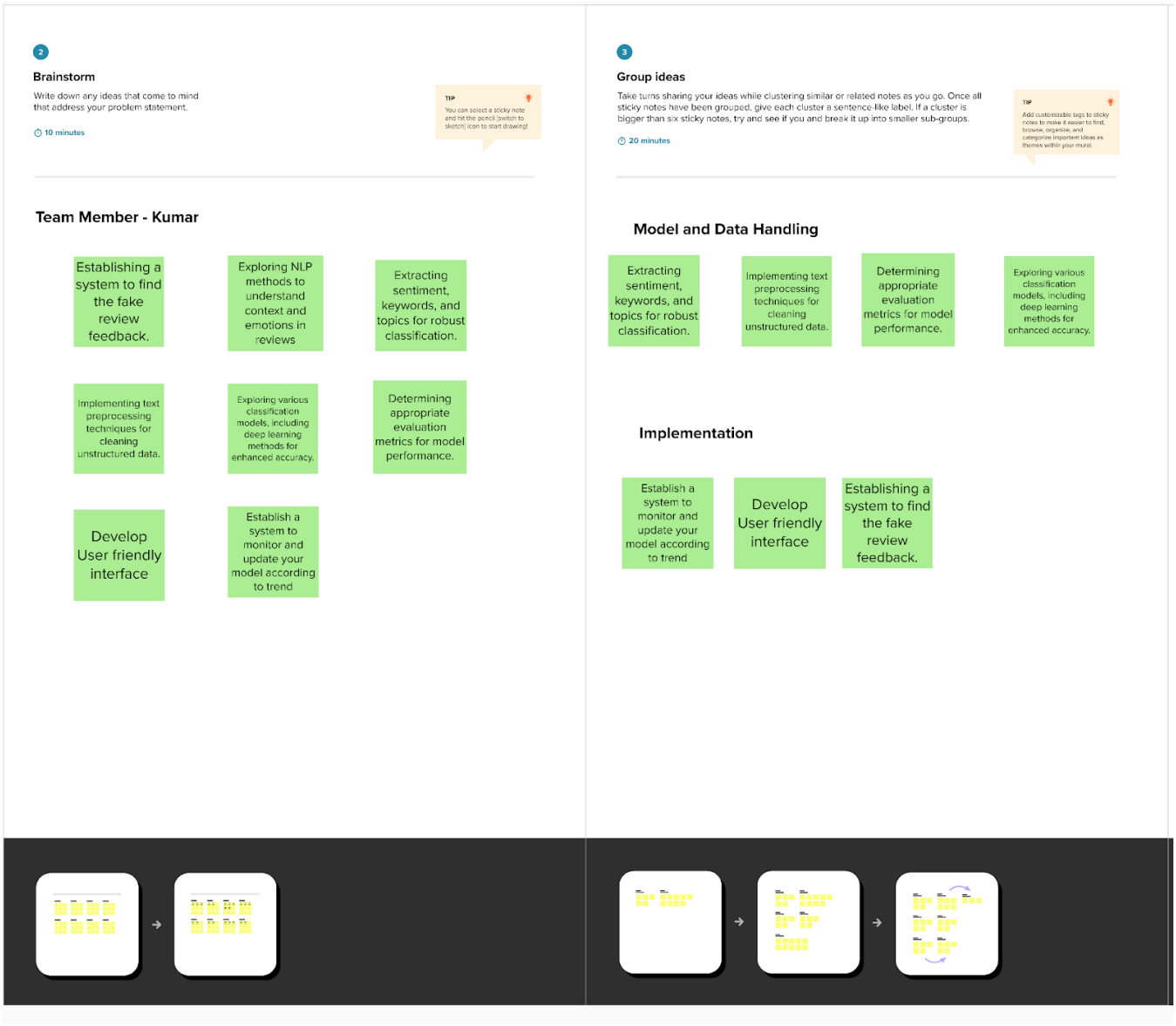
Airline Review Classification Using Machine Learning: With the vast user-generated content on social media and travel websites, our goal is to extract valuable insights from unstructured text data. This system will aid airlines in refining their services to enhance passenger satisfaction.

Key rules of brainstorming

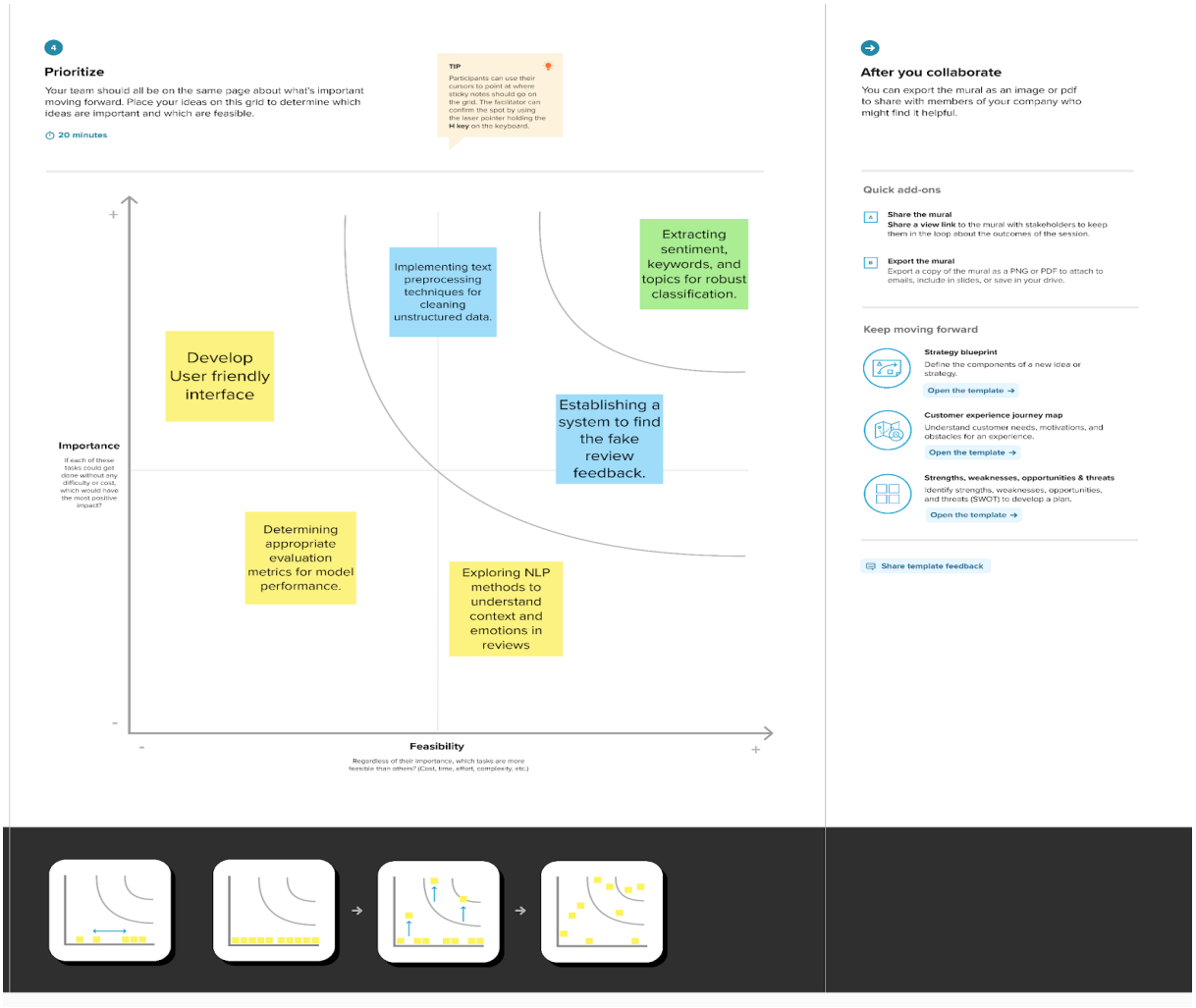
To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Data Collection and Integration:

The system should be capable of collecting airline reviews from various sources, including social media platforms, travel websites, and online forums.

It should integrate data seamlessly into a centralized repository for analysis.

Data Preprocessing:

Implement data preprocessing techniques to clean and standardize raw text data.

Feature Selection:
Identify and select relevant features from the preprocessed text data that contribute significantly to sentiment analysis and categorization of reviews.

Classification Models Integration:
Integrate multiple classification models, such as Decision Tree Classifier, Random Forest Classifier, XGBoost Classifier, etc., to facilitate diverse approaches to sentiment analysis and review categorization.

Model Training:
Train the classification models on a labelled dataset, ensuring they learn and adapt to the specific nuances of airline reviews.

Evaluation Metrics:
Define and implement evaluation metrics (e.g., accuracy, precision, recall) to assess the performance of the classification models effectively.

User Interface:
Develop a user-friendly interface for inputting new data, initiating model analysis, and visualizing the results in an understandable format.

4.2 Non-Functional Requirements:

Scalability:
The system should be scalable to accommodate increasing volumes of airline reviews as the dataset grows over time.

Performance:
Ensure efficient and timely processing of data, model training, and result generation to meet real-time or near-real-time requirements.

Security:
Implement security measures to protect sensitive user data and maintain the confidentiality of airline reviews.

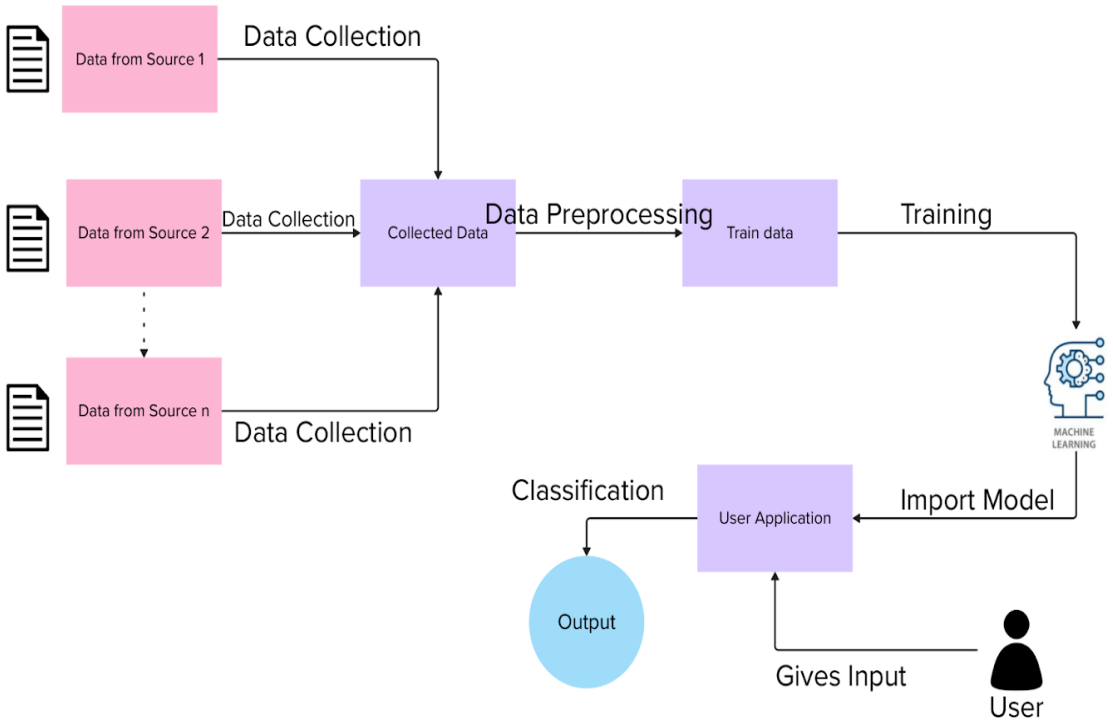
Reliability:
The system should be reliable, with minimal downtime and robust error handling mechanisms.

Compatibility:
Ensure compatibility with different data formats and sources to enhance the system's adaptability to evolving data structures.

User Training and Support:
Provide training resources for users to understand and navigate the system effectively.
Establish a support system for addressing user queries and troubleshooting issues promptly.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories



User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Airlines	Data Collection	USN-1	Collect a diverse dataset of airline reviews from social media, travel websites, and forums.	Successfully gather a dataset containing a variety of reviews from different sources.	High	Sprint-1
Data scientist	Preprocessing	USN-2	Preprocess the collected dataset by cleaning text, tokenizing, and converting it into numerical format.	Dataset is cleaned, tokenized, and prepared in a numerical format for machine learning models.	High	Sprint-1
Web developer	Model Deployment	USN-3	Deploy the best performing model as an API or web service for real-time sentiment analysis.	Model integrated into a user-friendly interface for sentiment analysis.	Low	Sprint-3
ML Engineer	Training	USN-4	Train multiple classification models (e.g., Decision Tree, Random Forest, XGBoost) on the preprocessed dataset.	Successfully train models and evaluate their performance.	Medium	Sprint-2
Tester	Integration	USN-5	Conduct thorough testing of the deployed model and the web interface for any issues or bugs.by	Model and web interface functioning without errors, with appropriate fine-tuning based on user feedback.	Medium	Sprint-3

5.2 Solution Architecture

The proposed solution revolves around utilizing various machine learning classification models such as Decision Tree Classifier, Random Forest Classifier, XGBoost Classifier, among others, to process and categorize the unstructured text data from airline reviews. The process involves a multi-step approach:

1. Data Collection & Preparation
- Collect the dataset
- Data Preparation
- Exploratory Data Analysis

2. Descriptive statistical

- Visual Analysis

3. Model Building

- Training the model in multiple algorithms
- Testing the model

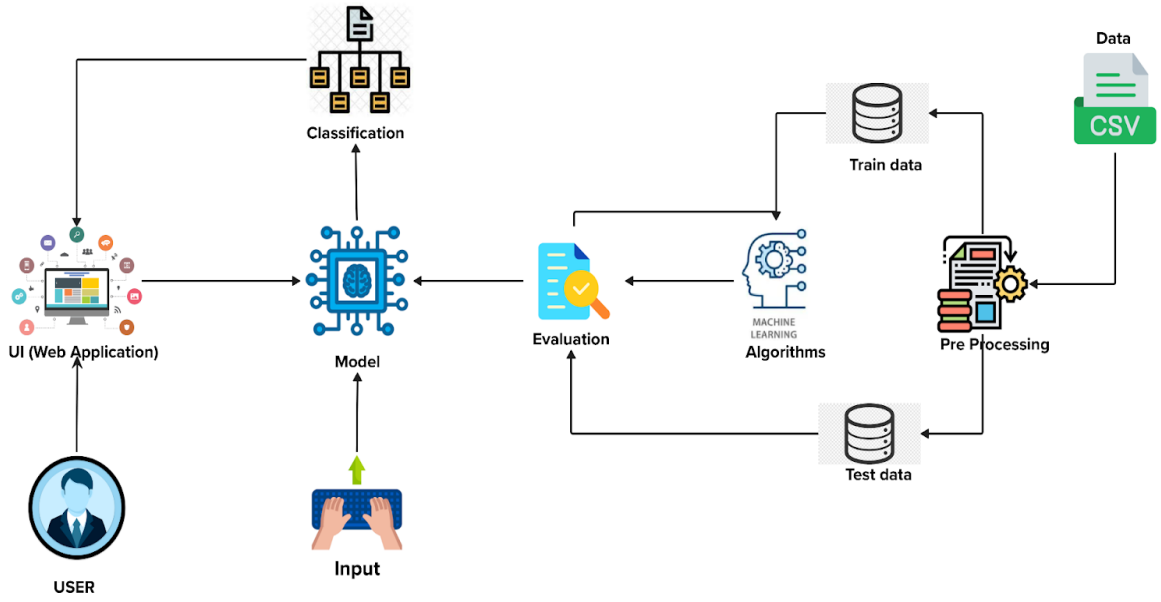
4. Performance Testing

- Testing model with multiple evaluation metrics

5. Model Deployment

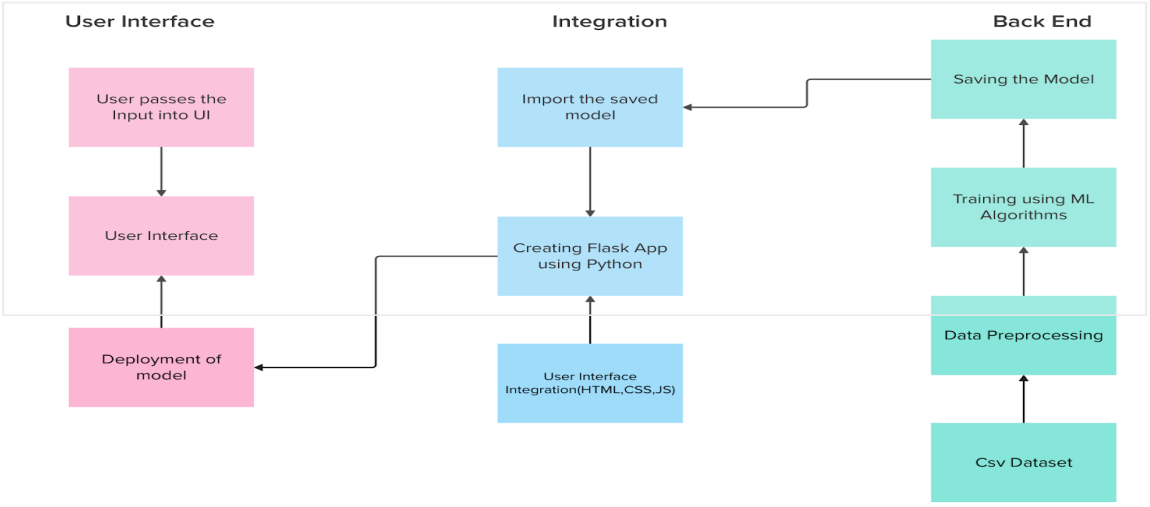
- Save the best model
- Integrate with Web Framework

Solution Architecture Diagram:



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



6.2 Sprint Planning & Estimation

Sprint	Functional Requirement(Epic)	User Story Number	User Story/ Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	Collect a diverse dataset of airline reviews from social media, travel websites, and forums.	5	High	Kumar
Sprint-1	Data Preprocessing	USN-2	Preprocess the collected dataset by cleaning text, tokenizing, and converting it into numerical format.	5	High	Kumar

Sprint-2	Training	USN-3	Train multiple classification models (e.g., Decision Tree, Random Forest, XGBoost) on the preprocessed dataset.	15	Medium	Kumar
Sprint-3	Model Deployment	USN-4	Deploy the best performing model as an API or web service for real-time sentiment analysis.	10	Low	Kumar
Sprint-3	Integration	USN-5	Conduct thorough testing of the deployed model and the web interface for any issues.	5	Medium	Kumar

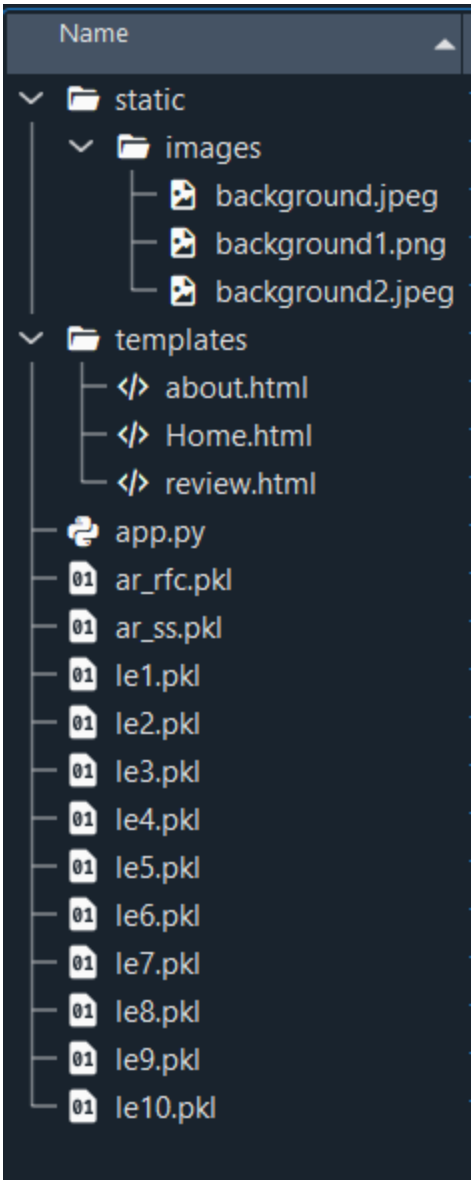
6.3 Sprint Delivery Schedule

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	10	5 Days	30 Oct 2023	4 Nov 2023	10	3 Nov 2023
Sprint-2	15	8 Days	3 Nov 2023	11 Nov 2023	10	11 Nov 2023
Sprint-3	15	7 Days	11 Nov 2023	18 Nov 2023		

7. CODING & SOLUTIONING

Project Structure

- Creating a Project folder which contains files as shown below
- The Dataset folder contains the airline reviews csv file for training our model.
- We are building a Flask Application that needs HTML pages stored in the templates folder and a python script app-py for server-side scripting.
- We need the model which is saved and the saved model in this content is a ar_rfc.pkl
- templates folder contains about.html, Home.html, review.html pages.
- An IPYNB file is a notebook document created by Jupyter Notebook.



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

Import the Libraries:

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')
```

Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas. In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

Data Collection:

Data Collection

```
# Read the Dataset
data=pd.read_csv(r"D:\AI&ML Project\Airline_Reviews.csv")
data.head()
```

	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	F Beve
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	5.0	
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	2.0	
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	1.0	
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	1.0	

Data Preparation

As we have understood how the data is, let's pre-process the collected data. The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the

dataset properly in order to fetch good results. This activity includes the following steps

● Handling missing values ● Handling categorical data

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending
on the condition of your dataset, you may or may not have to go through all these steps.

Checking for Null values

```
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
```

● Let’s find the shape of our dataset first. To find the shape of our data, the ar.shape method is used.
● To find the data type, data.info() function is used.
● For checking the null values, data.isnull() function is used.
● To sum those null values, we append .sum() to the above null function.
● From the image we found that there are null values present in our dataset. But still there is no need to run that command since we can find it out through data.info().
● So now we need to replace those null values based with median or mode values depending on feature type.
● But before that , we need to drop unnecessary columns so that we can create an efficient model. We used data.drop([ ],axis=1) to remove unnecessary rows

```
### drop unnecessary data

data1=data.drop(['Inflight Entertainment','Wifi & Connectivity','Aircraft','Value For Money',
                 'Cabin Staff Service','Unnamed: 0','Review Date','Review_Title','Review'],axis=1)
```

```
data1['Overall_Rating']=data1['Overall_Rating'].replace(['1','2','3','4','5','6','7','8','9','n'],['1','2','3','4','5','6','7','8
```

```
data1['Overall_Rating'].value_counts()
```

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

Fill the null values with mode or median:

Handling Missing Values

```
data1['Type Of Traveller'] = data1['Type Of Traveller'].fillna(data1['Type Of Traveller'].mode()[0])
data1['Seat Type'] = data1['Seat Type'].fillna(data1['Seat Type'].mode()[0])
data1['Seat Comfort'] = data1['Seat Comfort'].fillna(data1['Seat Comfort'].mode()[0])
data1['Route'] = data1['Route'].fillna(data1['Route'].mode()[0])
data1['Date Flown'] = data1['Date Flown'].fillna(data1['Date Flown'].mode()[0])
data1['Food & Beverages'] = data1['Food & Beverages'].fillna(data1['Food & Beverages'].mode()[0])
data1['Ground Service'] = data1['Ground Service'].fillna(data1['Ground Service'].mode()[0])

#For the above columns we are using mode instead of median even though numerical values are present
#becouse the column consists of categories(0 to 5).So its considered as categorical data
```

Later , we will modify the existing columns depending on our requirement. In our dataset I have modified the Date flown and Route columns.

```
In [9]: data1[['Month Flown','Year Flown']] = data1['Date Flown'].str.split(expand=True)
```

```
In [10]: data1['Origin'] = data1.Route.str.split(' to ',expand=True)[0]
         data1['Destination'] = data1.Route.str.split(' to ',expand=True)[1]
         # Route column has 3 values i.e., eg. Place A to Place B via Place C, so inorder to chose
         # we gave indices for Moroni as 0 & Moheli as 1, and then run the split function again to remove "via"
         data1['Destination'] = data1.Destination.str.split(' via ',expand=True)[0]
```

```
In [11]: del data1['Date Flown']
```

```
In [12]: del data1['Route']
```

```
In [13]: data1['Origin'] = data1['Origin'].replace(['Tel Avivito Malta (MLA)','Bangalore toChennai','JFK toTLV via Baku','Krabi toBangkok'
                                                    'Edinburgh To Fuerteventura','Nuremburg toHamburg','Mumbai toJaipur','Sydney to New Yo
                                                    'London Gatwick Bangkok','SIN toi MFM','Jakartato Yogyakarta','Cardiff-Halta return','
                                                    'GRR-ORD','LCY-FRA','NAP-RMF return','LEB-BOS','Bucharest-Brussels','Da Nang Hong Kong
                                                    'LHR-DXB','Dublin Charlotte','Kansas City via Dallas Ft Worth','Sydney via Singapore',
                                                    'Nursultan via Dubai','Denpasar Medan via Jakarta','Auckland Denpasar via Sydney / Mel
                                                    'Lima via Santiago','Manila via Los Angeles','Dar es Salaam via Kigali','Singapore via
                                                    'Grand Rapidsvto Orlando via Chicago','Toronto via Varadero','Bangkok via Humbai','A C
                                                    'LHR-DXB','Paris Orly Los Angeles','Newark Los Angeles','Honolulu Seattle','San Paulo'
                                                   ['Tel Aviv (MLA)','Bangalore','JFK','Krabi','Hong Kong','Edinburgh','Nuremburg','Mumbai
                                                    'Sydney','London Gatwick','SIN','Jakarta','Cardiff','KIV','GRR','LCY','NAP','LEB','Buc
                                                    'Da Nang','New York','LHR','Dublin','Kansas City','Sydney','Geneva','Nursultan','Denpa
                                                    'Auckland Denpasar','Lima','Manila','Dar es Salaam','Singapore','Grand Rapidsvto Orlar
                                                    'Toronto','Bangkok','A Coruna','LHR','Paris Orly','Newark','Honolulu','San Paulo'])
```

In [14]:
```
#Destination recorrections
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 / Melbourne','Santiago','Los Angeles','Kigali',
           'Sydney','Chicago','Varadero','Mumbai','Bilbao','Dallas','Los Angeles','Los Angeles','Seattle']

for i in row_num:
    data1.at[i,'Destination'] = new_des[j]
    j+=1
```

In [15]:
```
new_column_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 [16]:
```
# Reordering the columns of given data to our desired manner
data1 = data1.reindex(columns = new_column_order)
```

After correcting all the values in the respective features as shown in the above images, we shall re order the columns for our convenience. The resultant dataset looks as below.

In [17]:
```
data1.head()
```

Out[17]:

	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

Data Visualization:

Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas have a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, STD, min, max and percentile values of continuous features.

```
In [18]: data1.describe()
```

Out[18]:

	Seat Comfort	Food & Beverages	Ground Service
count	23171.000000	23171.000000	23171.000000
mean	2.328126	1.972207	2.073713
std	1.465062	1.422340	1.523264
min	0.000000	0.000000	1.000000
25%	1.000000	1.000000	1.000000
50%	2.000000	1.000000	1.000000
75%	4.000000	3.000000	3.000000
max	5.000000	5.000000	5.000000

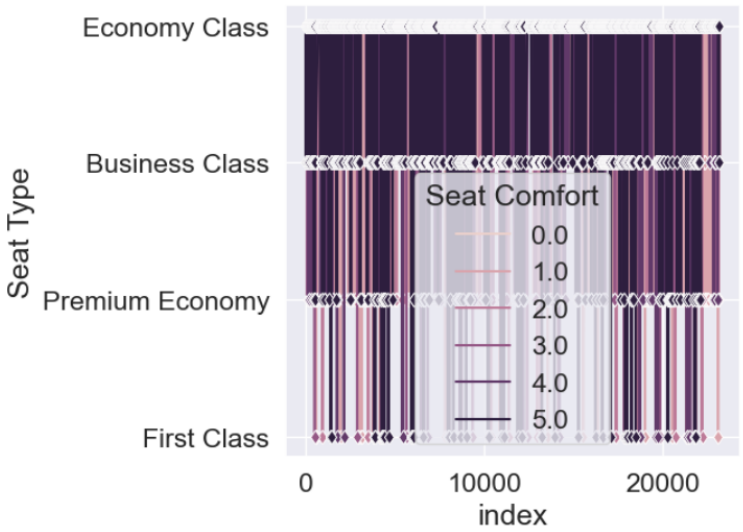
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions

Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and countplot. Sea born package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot. Matplotlib function have many plots. We are using bar plot for our dataset. Note: In our dataset we used line plot, bar plot, pie plot and not using distplot and countplot because our dataset is not having continuous values

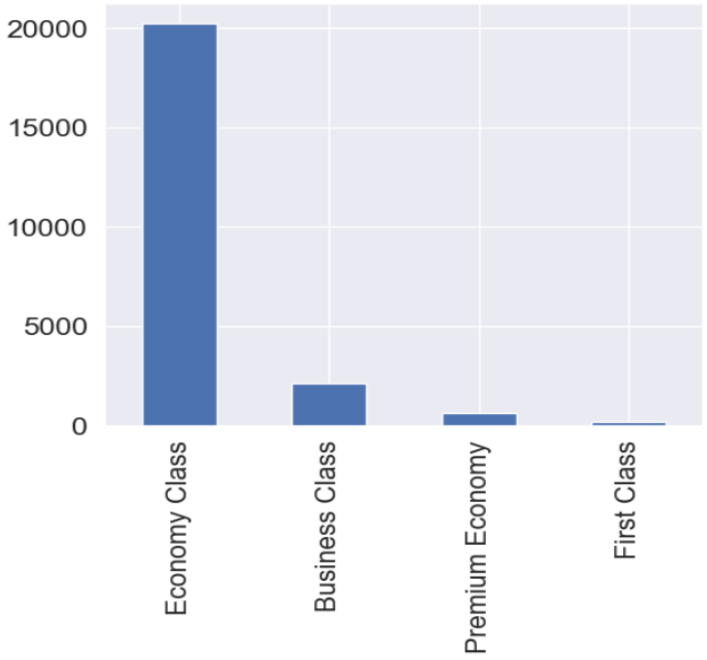
```
In [19]: #line plot in seaborn

         sns.set(rc={'figure.figsize':[5,5]})
         sns.set(font_scale=1.5)
         fig=sns.lineplot(x=data1.index,y=data1['Seat Type'],markevery=1,marker='d',hue=data1['Seat Comfort'])
         fig.set(xlabel='index')
```

```
Out[19]: [Text(0.5, 0, 'index')]
```



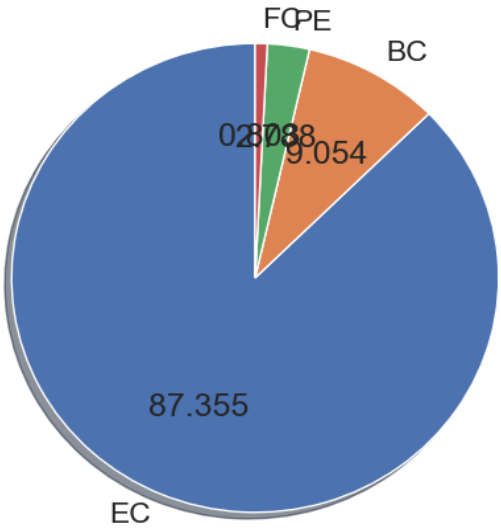
```
20]: plt.figure(figsize=(6,5))
     data1['Seat Type'].value_counts().plot.bar()
```

```
20]: <Axes: xlabel='Seat Type'>
```




```
[21]: plt.figure(figsize=(6,6))
      plt.pie(data1['Seat Type'].value_counts(),startangle=90,autopct='%.3f',
              labels=['EC','BC','PE','FC'],shadow=True)
```

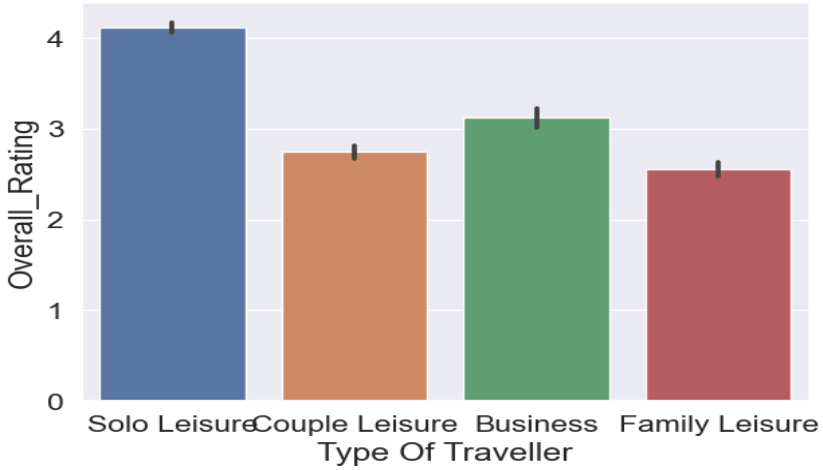
```
:[21]: ([<matplotlib.patches.Wedge at 0x2a3b789a210>,
          <matplotlib.patches.Wedge at 0x2a3b78be090>,
          <matplotlib.patches.Wedge at 0x2a3b78d4290>,
          <matplotlib.patches.Wedge at 0x2a3b78f9250>],
         [Text(-0.4255806105829949, -1.0143377858957072, 'EC'),
          Text(0.5370556832029272, 0.9599849963095451, 'BC'),
          Text(0.15134436463791998, 1.0895388397355756, 'PE'),
          Text(0.027737504813067807, 1.099650231131129, 'FC')],
         [Text(-0.23213487849981534, -0.5532751559431129, '87.355'),
          Text(0.292939463565233, 0.5236281798052064, '9.054'),
          Text(0.08255147162068362, 0.5942939125830412, '2.788'),
          Text(0.015129548079855165, 0.5998092169806156, '0.803')])
```



Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we are visualizing the relationship between Type Of Traveller, Overall_Rating

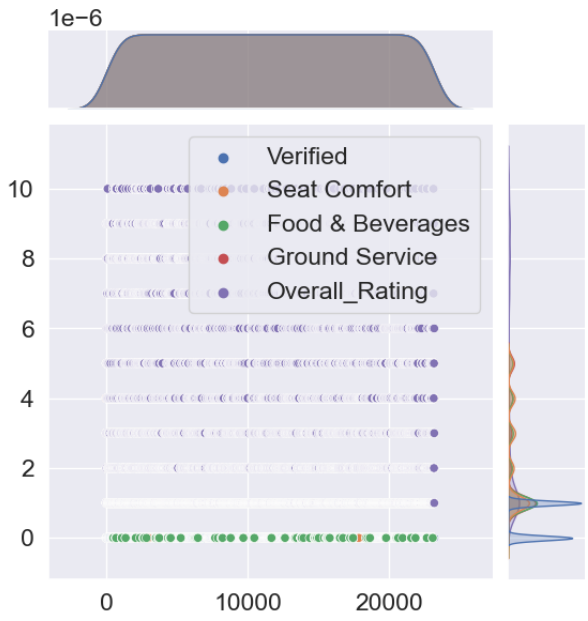
```
In [22]: plt.figure(figsize=(7,5))
         data1['Overall_Rating'] = pd.to_numeric(data1['Overall_Rating'], errors='coerce')
         sns.barplot(data=data1,x='Type Of Traveller',y='Overall_Rating')
Out[22]: <Axes: xlabel='Type Of Traveller', ylabel='Overall_Rating'>
```



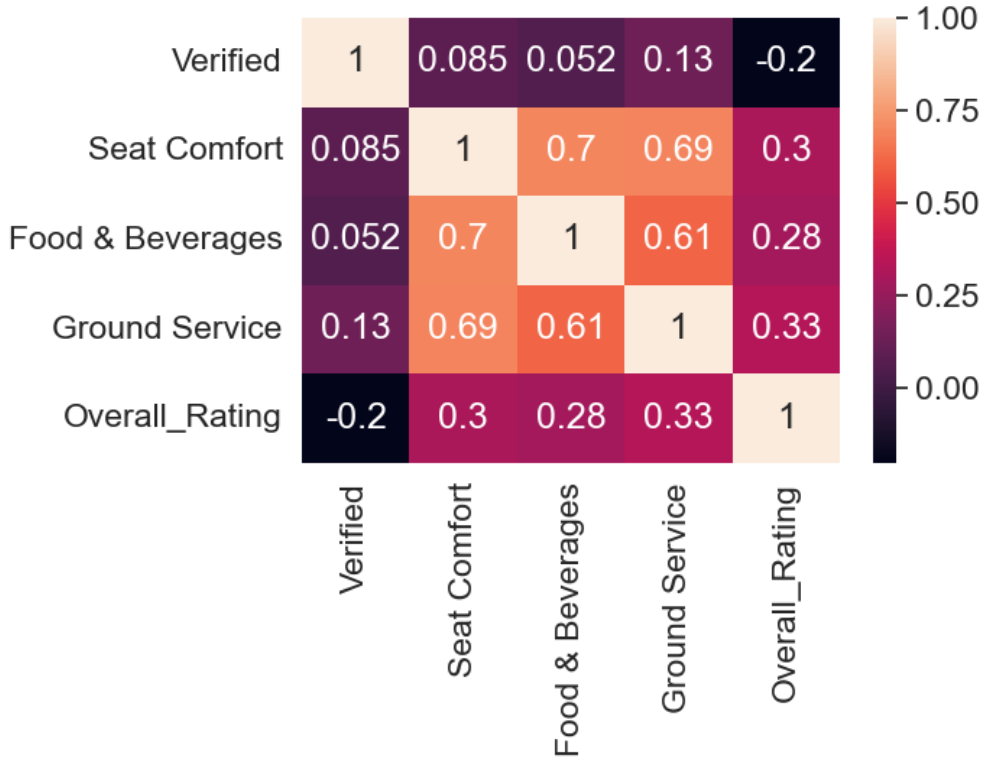
Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used joint plot and heatmap(for finding correlation) from sea born package.

```
[23]: plt.figure(figsize=(5,5))
      sns.jointplot(data1)
t[23]: <seaborn.axisgrid.JointGrid at 0x2a3b7952510>
       <Figure size 500x500 with 0 Axes>
```



```
24]: plt.subplots(figsize=(6,4))
     sns.heatmap(data1.corr(numeric_only=True),annot=True)
24]: <Axes: >
```



Handling Categorical Values:

Label Encoding:

As we can see our dataset has categorical data, we must convert the categorical data to integer encoding or binary encoding. To convert the categorical features into numerical features we use encoding techniques. There are several techniques but, in our project, we are using label encoding. But prior encoding we need to do EDA. ● In our project, categorical features are Airline Name, Seat Type, Type Of Traveller, Origin, Destination, Month Flown, Year Flown, Verified, Overall_Rating, Recommended. Label encoding is done for those columns.

Dataset will be converted as below image

Encoding

```
5]: 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()
```

```
6]: data1['Airline Name'] = le1.fit_transform(data1['Airline Name'])
    data1['Seat Type'] = le2.fit_transform(data1['Seat Type'])
    data1['Type Of Traveller'] = le3.fit_transform(data1['Type Of Traveller'])
    data1['Origin'] = le4.fit_transform(data1['Origin'])
    data1['Destination'] = le5.fit_transform(data1['Destination'])
    data1['Month Flown'] = le6.fit_transform(data1['Month Flown'])
    data1['Year Flown'] = le7.fit_transform(data1['Year Flown'])
    data1['Verified'] = le8.fit_transform(data1['Verified'])
    data1['Overall_Rating'] = le9.fit_transform(data1['Overall_Rating'])
    data1['Recommended'] = le10.fit_transform(data1['Recommended'])
```

```
?7]: data1.head()
```

?7]:

	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	1278	1545	9	6	1	4.0	4.0	4.0	8	1
1	0	1	3	1278	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	632	1927	11	6	0	1.0	1.0	1.0	0	0
4	4	1	1	1834	99	11	6	1	1.0	1.0	1.0	0	0

Splitting data into train and test:

For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state. After splitting the train and test data, we shall use StandardScaler function to remove the outliers of our dataset. And also save that model with the help of pickle function

```
In [32]: X = data1.iloc[:,0:12].values
         Y = data1.iloc[:,12:13].values
```

```
In [33]: X
```

```
Out[33]: array([[  0.,   1.,   3., ...,   4.,   4.,   8.],
                [  0.,   1.,   3., ...,   1.,   1.,   0.],
                [  0.,   1.,   3., ...,   1.,   1.,   0.],
                ...,
                [487.,   1.,   0., ...,   2.,   1.,   2.],
                [487.,   0.,   0., ...,   3.,   1.,   5.],
                [487.,   1.,   3., ...,   1.,   1.,   0.]])
```

```
In [34]: Y
```

```
Out[34]: array([[1],
                [0],
                [0],
                ...,
                [0],
                [1],
                [0]])
```

```
35]: data1.Recommended.value_counts()
```

```
35]: Recommended
     0    15364
     1     7807
     Name: count, dtype: int64
```

```
36]: 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)
```

```
37]: from sklearn.preprocessing import StandardScaler
     ss = StandardScaler()
     X_train = ss.fit_transform(X_train)
     X_test = ss.fit_transform(X_test)
     import pickle
     pickle.dump(ss,open('ar_ss.pkl','wb'))
```

Save the label encoding:

```
[38]: import pickle
      pickle.dump(le1,open('le1.pkl','wb'))
      pickle.dump(le2,open('le2.pkl','wb'))
      pickle.dump(le3,open('le3.pkl','wb'))
      pickle.dump(le4,open('le4.pkl','wb'))
      pickle.dump(le5,open('le5.pkl','wb'))
      pickle.dump(le6,open('le6.pkl','wb'))
      pickle.dump(le7,open('le7.pkl','wb'))
      pickle.dump(le8,open('le8.pkl','wb'))
      pickle.dump(le9,open('le9.pkl','wb'))
      pickle.dump(le10,open('le10.pkl','wb'))
```

Training the model in multiple algorithms :

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.

Desicion Tree:

A function named Decision Tree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialized and training data is passed to the model with the fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done

Decision Tree

```
]: from sklearn.tree import DecisionTreeClassifier
   dt = DecisionTreeClassifier(criterion = 'entropy',random_state=50)
```

```
]: dt.fit(X_train,Y_train)
```

```
]: 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.

```
]: pred_dt = dt.predict(X_test)
   pred_dt
```

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

K-Nearest Neighbors:

A function named KNeighborsClassifier is created and train and test data are passed as the parameters. Inside the function, KNeighborsClassifier algorithm is initialized and training data is passed to the model with the fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

KNN

```
]: from sklearn.neighbors import KNeighborsClassifier
   knn = KNeighborsClassifier(n_neighbors=5)
```

```
]: knn.fit(X_train,Y_train)
```

```
]: KNeighborsClassifier()
```

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.

```
]: pred_knn = knn.predict(X_test)
   pred_knn
```

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

Logistic Regression:

A function named Logistic Regression is created and train and test data are passed as the parameters. Inside the function, LogisticRegression algorithm is initialized and training data is passed to the model with the fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done

Logistic Regression

```
53]: from sklearn.linear_model import LogisticRegression
     Lr = LogisticRegression()
```

```
54]: Lr.fit(X_train,Y_train)
```

```
54]: 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.

```
55]: pred_Lr = Lr.predict(X_test)
     pred_Lr
```

```
55]: array([0, 1, 0, ..., 0, 1, 0])
```

Naive Bayes:

A function named GaussianNB is created and train and test data are passed as the parameters. Inside the function, GaussianNB algorithm is initialized and training data is passed to the model with the fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

Naive Bayes

```
50]: from sklearn.naive_bayes import GaussianNB
     Gnb = GaussianNB()
```

```
51]: Gnb.fit(X_train,Y_train)
```

```
51]: 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.

```
52]: pred_nb=Gnb.predict(X_test)
     pred_nb
```

```
52]: array([0, 1, 0, ..., 0, 1, 0])
```

Random Forest:

A function named RandomForestClassifier is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialized and training data is passed to the model with the fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done

Random Forest

```
57]: from sklearn.ensemble import RandomForestClassifier
     Rfc = RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=2)
```

```
58]: Rfc.fit(X_train,Y_train)
```

```
58]: RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=2)
```

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.

```
59]: pred_rfc=Rfc.predict(X_test)
     pred_rfc
```

```
59]: array([0, 1, 0, ..., 0, 0, 0])
```

Support Vector Machine:

A function named SVC is created and train and test data are passed as the parameters. Inside the function, SVC algorithm is initialized and training data is passed to the model with the fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

Support Vector(SVC)

```
4]: from sklearn.svm import SVC
    svc = SVC()
```

```
5]: svc.fit(X_train,Y_train)
```

```
5]: 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.

```
6]: pred_svc=svc.predict(X_test)
    pred_svc
```

```
6]: array([0, 1, 0, ..., 0, 1, 0])
```

XG Boost:

A function named XGBClassifier is created and train and test data are passed as the parameters. Inside the function, XGBClassifier algorithm is initialized and training data is passed to the model with the fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done. Let us check the training accuracy also for this algorithm. We get to know that there is no issue of over fitting and at the same time both the testing and training accuracies are best. So will test this model.

XGBoost

```
from xgboost import XGBClassifier
xgb=XGBClassifier()
```

```
xgb.fit(X_train,Y_train)
```

```
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.

```
pred_xgb=xgb.predict(X_test)
```

Save the model:

Saving the model

```
import pickle
pickle.dump(Rfc,open('ar_rfc.pkl','wb'))
```

Model Deployment:

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.
This section has the following tasks
- Building HTML Pages
- Building server-side script
- Run the web application

For this project create two HTML files namely and save them in the templates folder.
home.html ,review.html.

Build Python code:

Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (name) as argument.

```
from flask import Flask, render_template, request
import numpy as np
app=Flask(__name__)
import pickle

model=pickle.load(open('ar_rfc.pkl','rb'))
ss1=pickle.load(open('ar_ss.pkl','rb'))
le1=pickle.load(open('le1.pkl','rb'))
le2=pickle.load(open('le2.pkl','rb'))
le3=pickle.load(open('le3.pkl','rb'))
le4=pickle.load(open('le4.pkl','rb'))
le5=pickle.load(open('le5.pkl','rb'))
le6=pickle.load(open('le6.pkl','rb'))
le7=pickle.load(open('le7.pkl','rb'))
le8=pickle.load(open('le8.pkl','rb'))
le9=pickle.load(open('le9.pkl','rb'))
le10=pickle.load(open('le10.pkl','rb'))

@app.route('/')
def home():
      return render_template('Home.html')
@app.route('/review')
def review():
   return render_template('review.html')
@app.route('/about')
def about():
   return render_template('about.html')

@app.route('/pred', methods=['POST'])
def predict():
      Airline_name=request.form['Airline name']
      Seat_Type=request.form['Seat Type']
      Type_Of_Traveller=request.form['Type Of Traveller']
      Origin=request.form['Origin']
      Destination=request.form['Destination']
      Month_Flown=request.form['Month Flown']
      Year_Flown=request.form['Year Flown']
      Verified=request.form['Verified']
      S_C=int(request.form['S_C'])
      F_B=int(request.form['F_B'])
```

```
    G_S=int(request.form['G_S'])
    O_R=int(request.form['O_R'])
    data=[[Airline_name, Seat_Type, Type_Of_Traveller, Origin, Destination,
          Month_Flown,Year_Flown, Verified,S_C,F_B,G_S,O_R]]
    encoded_data = [
       le1.transform([Airline_name])[0],
       le2.transform([Seat_Type])[0],
       le3.transform([Type_Of_Traveller])[0],
       le4.transform([Origin])[0],
       le5.transform([Destination])[0],
       le6.transform([Month_Flown])[0],
       le7.transform([Year_Flown])[0],
       le8.transform([Verified])[0],
       [S_C][0],[F_B][0],[G_S][0],
       le9.transform([O_R])[0]
    ]

    print (encoded_data)
    prediction=model.predict(ss1.transform([encoded_data]))
    if prediction== 1:
      a="Recommended"
      return render_template('review.html',result=a)
    else:
      b="Not Recommended"
      return render_template('review.html',result=b)

if __name__ == "__main__":
   app.run(debug=True)
```

8. PERFORMANCE TESTING

8.1 Perform ace Metrics

Predicting:

```
[89]: Rfc.predict([[4,71,1,3,900,1133,1,6,1,5,5,5]])

t[89]: array([1])
```

Decision Tree:

```
42]: 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))
```

```
                precision    recall  f1-score   support

           0       0.93      0.96      0.94      3058
           1       0.92      0.86      0.89      1577

    accuracy                           0.93      4635
   macro avg       0.92      0.91      0.92      4635
weighted avg       0.93      0.93      0.93      4635
```

```
43]: roc_auc_dt=auc(fpr_dt,tpr_dt)
     print("roc_auc_dt: ",roc_auc_dt)
```

```
roc_auc_dt:  0.9104432255198898
```

```
44]: cm_dt = confusion_matrix(Y_test,pred_dt)
     print("Cm_dt: ",cm_dt)
```

```
Cm_dt:  [[2933  125]
 [ 218 1359]]
```

```
45]: as_dt = accuracy_score(Y_test,pred_dt)
     print("As_dt: ",as_dt)
```

```
As_dt:  0.9259978425026969
```

K-Nearest Neighbors:

```
)]: from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curve,auc
    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       0.95      0.96      0.95      3058
           1       0.92      0.90      0.91      1577

    accuracy                           0.94      4635
   macro avg       0.93      0.93      0.93      4635
weighted avg       0.94      0.94      0.94      4635
```

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

```
roc_auc_knn:  0.9308984241672207
```

```
.]: cm_knn = confusion_matrix(Y_test,pred_knn)
    print("Cm_knn: ",cm_knn)
```

```
Cm_knn:  [[2934  124]
 [ 154 1423]]
```

```
!]: as_knn = accuracy_score(Y_test,pred_knn)
    print("As_knn: ",as_knn)
```

```
As_knn:  0.9400215749730313
```

Logistic Regression:

```
|: from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,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       0.91      0.94      0.92      3058
           1       0.88      0.81      0.84      1577

    accuracy                           0.90      4635
   macro avg       0.89      0.88      0.88      4635
weighted avg       0.90      0.90      0.90      4635
```

```
|: roc_auc_Lr=auc(fpr_Lr,tpr_Lr)
print("roc_auc_Lr: ",roc_auc_Lr)
```

```
roc_auc_Lr:  0.8762592623773812
```

```
|: cm_Lr = confusion_matrix(Y_test,pred_Lr)
print("Cm_Lr: ",cm_Lr)
```

```
Cm_Lr:  [[2881  177]
 [ 299 1278]]
```

```
|: as_Lr = accuracy_score(Y_test,pred_Lr)
print("As_Lr: ",as_Lr)
```

```
As_Lr:  0.8973031283710895
```

Naive Bayes:

```
]: from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curve,auc
fpr_nb,tpr_nb,threshold_nb=roc_curve(Y_test,pred_nb)

print(classification_report(Y_test,pred_nb))
```

```
              precision    recall  f1-score   support

           0       0.91      0.90      0.90      3058
           1       0.80      0.83      0.82      1577

    accuracy                           0.87      4635
   macro avg       0.86      0.86      0.86      4635
weighted avg       0.87      0.87      0.87      4635
```

```
]: roc_auc_nb=auc(fpr_nb,tpr_nb)
print("roc_auc_Nb: ",roc_auc_nb)
```

```
roc_auc_Nb:  0.8625532041076079
```

```
]: cm_nb = confusion_matrix(Y_test,pred_nb)
print("Cm_Nb: ",cm_nb)
```

```
Cm_Nb:  [[2739  319]
 [ 269 1308]]
```

```
]: as_nb = accuracy_score(Y_test,pred_nb)
print("As_Nb: ",as_nb)
```

```
As_Nb:  0.8731391585760517
```

Random Forest :

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,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       0.95      0.97      0.96      3058
           1       0.95      0.90      0.92      1577

    accuracy                           0.95      4635
   macro avg       0.95      0.94      0.94      4635
weighted avg       0.95      0.95      0.95      4635
```

```
roc_auc_rfc=auc(fpr_rfc,tpr_rfc)
print("roc_auc_Rfc: ",roc_auc_rfc)
```

```
roc_auc_Rfc:  0.9364974268351504
```

```
cm_rfc = confusion_matrix(Y_test,pred_rfc)
print("Cm_Rfc: ",cm_rfc)
```

```
Cm_Rfc:  [[2976   82]
 [ 158 1419]]
```

```
as_rfc = accuracy_score(Y_test,pred_rfc)
print("As_Rfc: ",as_rfc)
```

```
As_Rfc:  0.948220064724919
```

Support Vector Machine:

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,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       0.94      0.97      0.95      3058
           1       0.93      0.88      0.90      1577

    accuracy                           0.94      4635
   macro avg       0.93      0.92      0.93      4635
weighted avg       0.94      0.94      0.94      4635
```

```
roc_auc_svc=auc(fpr_svc,tpr_svc)
print("roc_auc_svc: ",roc_auc_svc)
```

```
roc_auc_svc:  0.9213227216117231
```

```
cm_svc = confusion_matrix(Y_test,pred_svc)
print("Cm_svc: ",cm_svc)
```

```
Cm_svc:  [[2953  105]
 [ 194 1383]]
```

```
as_svc = accuracy_score(Y_test,pred_svc)
print("As_svc: ",as_svc)
```

```
As_svc:  0.9354908306364617
```

XG Boost:

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,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       0.92      0.92      0.92      3058
           1       0.85      0.85      0.85      1577

    accuracy                           0.90      4635
   macro avg       0.88      0.89      0.89      4635
weighted avg       0.90      0.90      0.90      4635
```

```
roc_auc_xgb=auc(fpr_xgb,tpr_xgb)
print("roc_auc_xgb: ",roc_auc_xgb)
```

```
roc_auc_xgb:  0.8862301984088636
```

```
cm_xgb = confusion_matrix(Y_test,pred_xgb)
print("Cm_xgb: ",cm_xgb)
```

```
Cm_xgb:  [[2814  244]
 [ 233 1344]]
```

```
as_xgb = accuracy_score(Y_test,pred_xgb)
print("As_xgb: ",as_xgb)
```

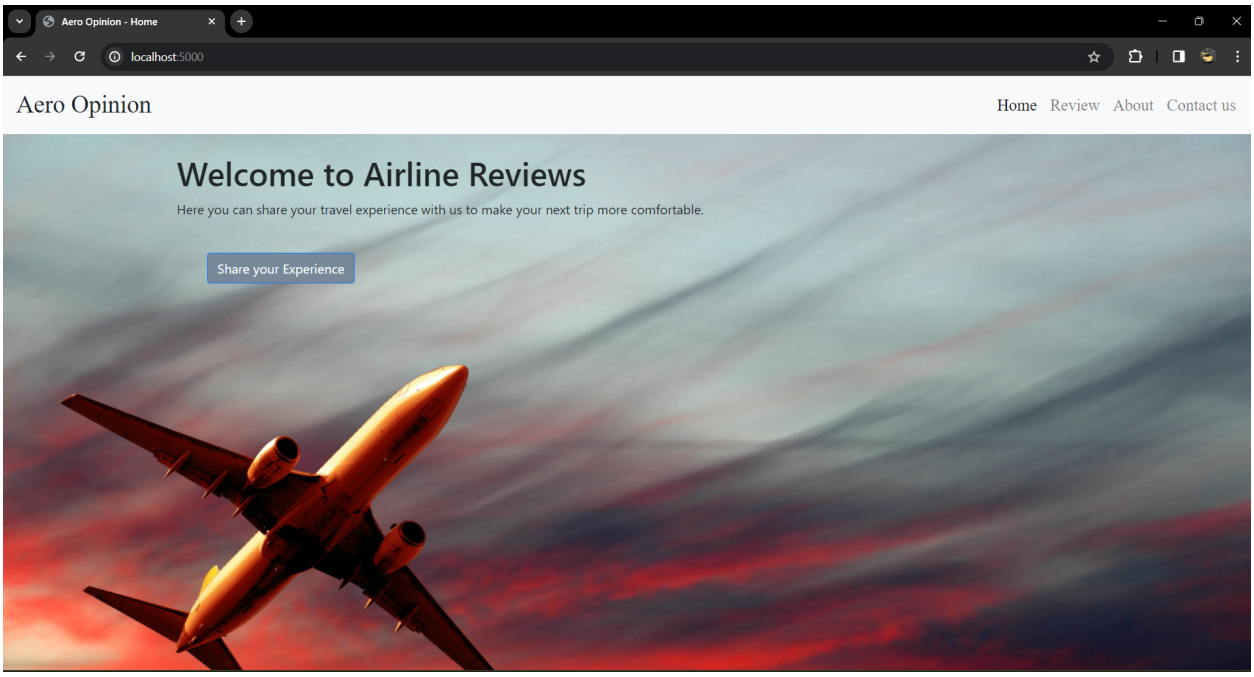
```
As_xgb:  0.8970873786407767
```

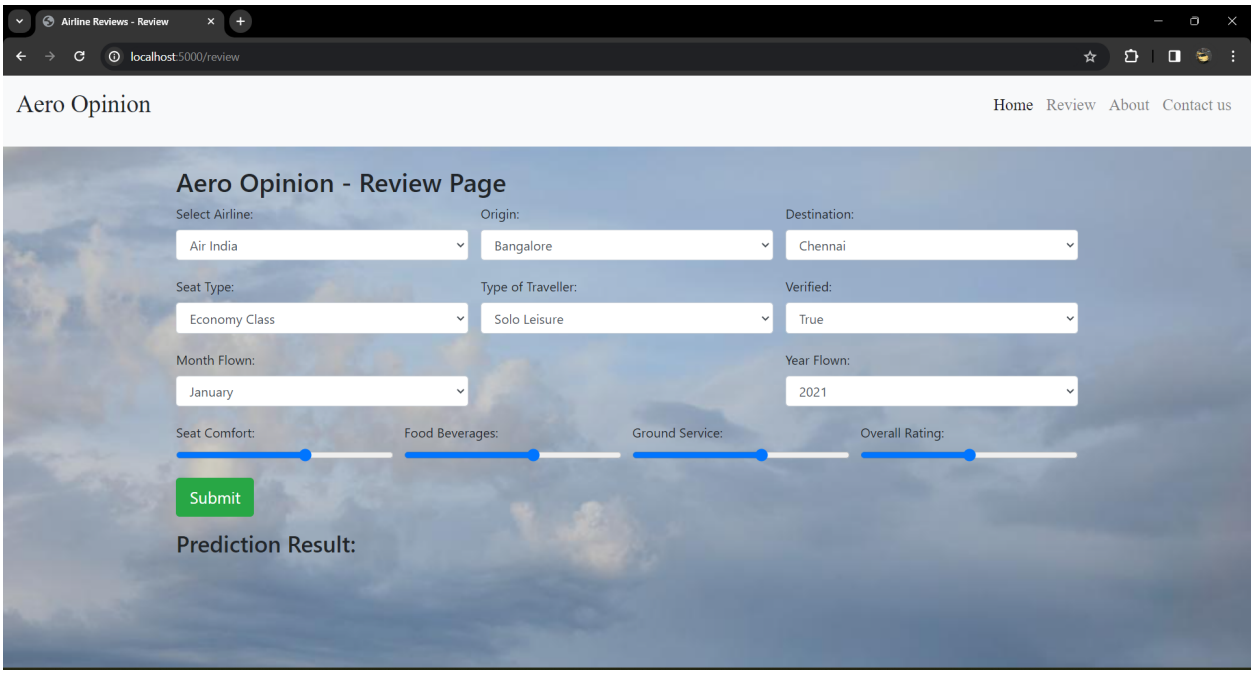
```
compare
```

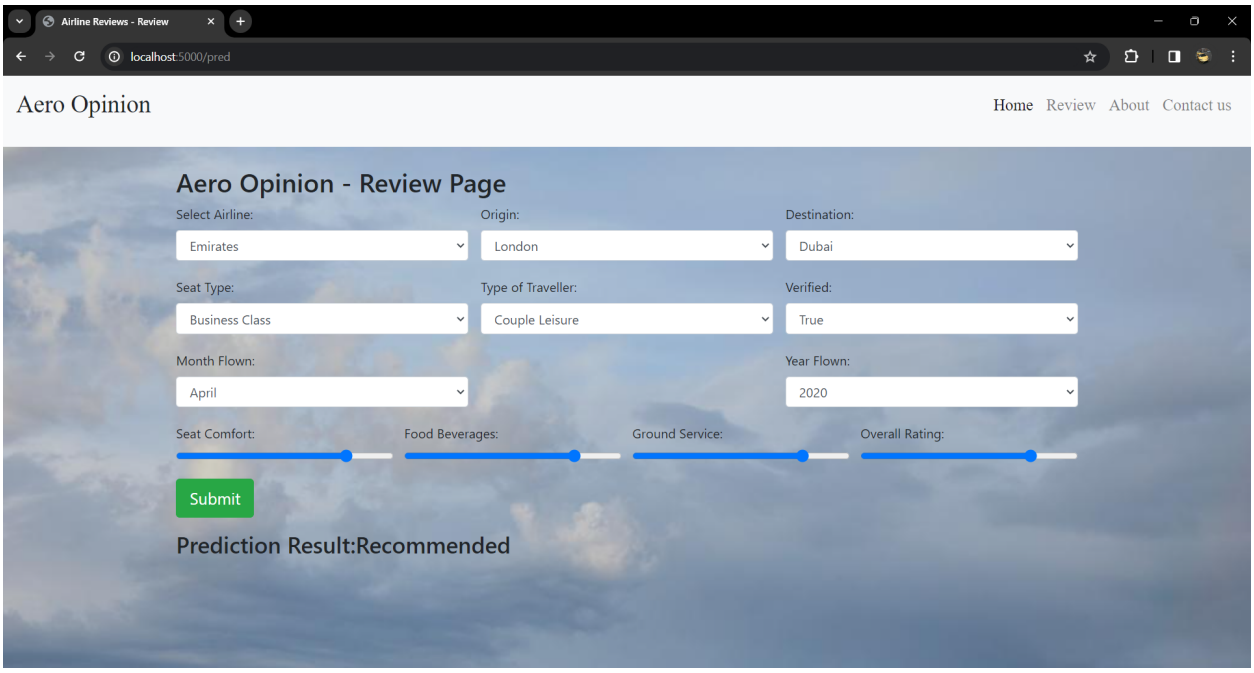
	Model	roc_auc	Accuracy
0	DecisionTree Classification	0.910443	0.925998
1	K-Nearest Neighbours	0.930898	0.940022
2	Logistic Regression	0.876259	0.897303
3	Naive Bayes Classification	0.862553	0.873139
4	RandomForest Classification	0.936497	0.948220
5	Support Vector Machine	0.921323	0.935491
6	XGBClassifier	0.886230	0.897087

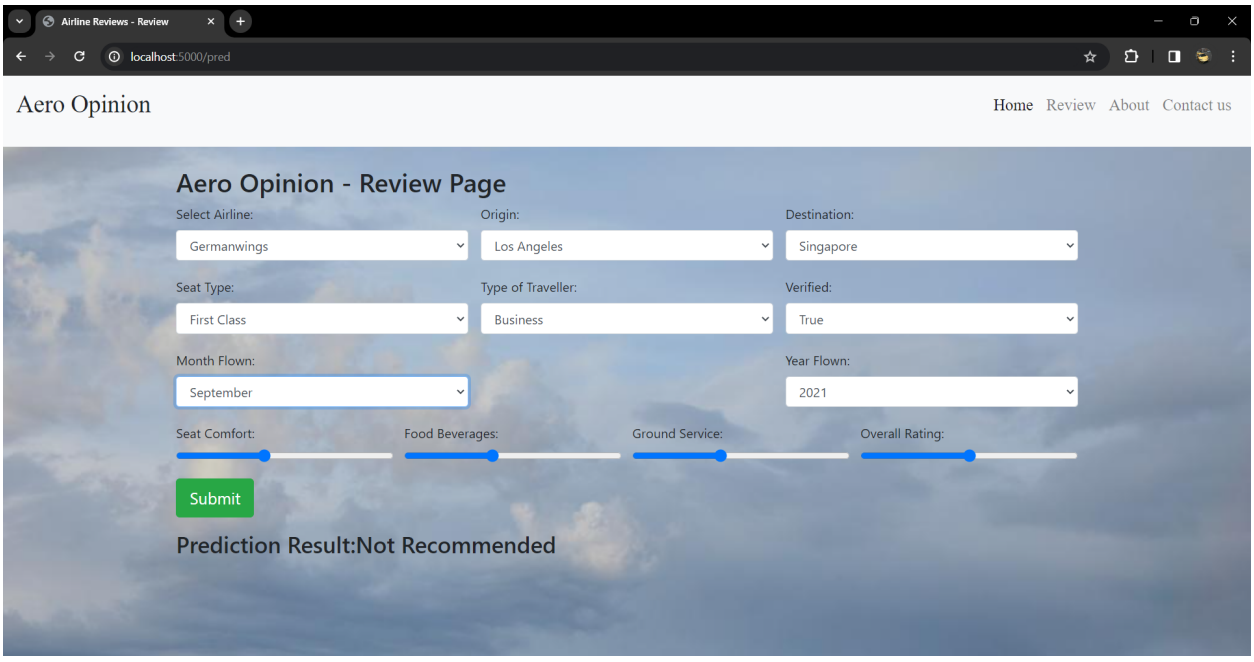
9. RESULTS

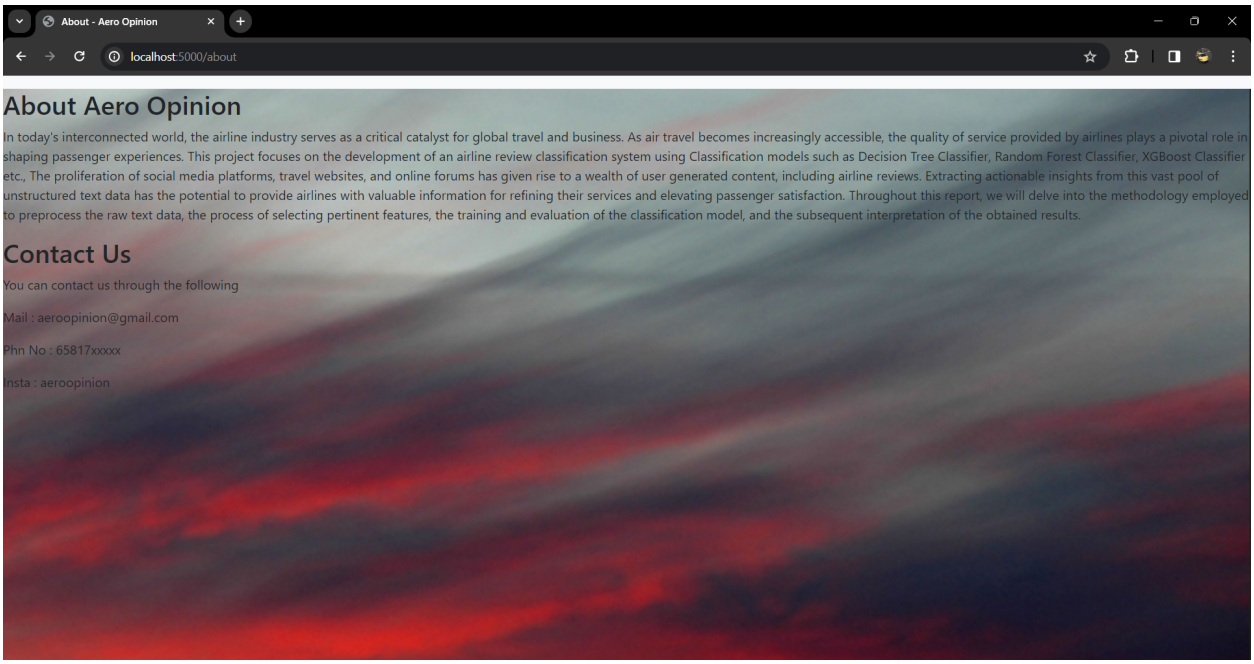
9.1 Output Screenshots






Airline Reviews - Review
localhost:5000/pred
Aero Opinion
Home Review About Contact us
Aero Opinion - Review Page
Select Airline:
Emirates
Origin:
London
Destination:
Dubai
Seat Type:
Business Class
Type of Traveller:
Couple Leisure
Verified:
True
Month Flown:
April
Year Flown:
2020
Seat Comfort:
Food Beverages:
Ground Service:
Overall Rating:
Submit
Prediction Result:Recommended


Airline Reviews - Review
localhost:5000/pred
Aero Opinion
Home Review About Contact us
Aero Opinion - Review Page
Select Airline:
Germanwings
Origin:
Los Angeles
Destination:
Singapore
Seat Type:
First Class
Type of Traveller:
Business
Verified:
True
Month Flown:
September
Year Flown:
2021
Seat Comfort:
Food Beverages:
Ground Service:
Overall Rating:
Submit
Prediction Result:Not Recommended



10. ADVANTAGES & DISADVANTAGES

Advantages:

Improved Service Quality:
By analysing and categorizing airline reviews, the classification system can provide airlines with specific insights into areas where service improvements are needed, leading to an overall enhancement in service quality.

Enhanced Passenger Satisfaction:
Actionable insights derived from the system can contribute to refining services, addressing passenger concerns, and ultimately elevating the overall satisfaction of travellers.

Data-Driven Decision Making:
The use of classification models allows for a systematic and data-driven approach to decision-making, enabling airlines to make informed adjustments based on the analysis of user-generated content.

Efficient Resource Allocation:
The system can help airlines allocate resources more efficiently by identifying priority areas for improvement, optimizing operational processes, and focusing on aspects that matter most to passengers.

Competitive Advantage:
Airlines implementing an effective review classification system can gain a competitive edge by staying attuned to customer sentiments, responding proactively to feedback, and continuously enhancing their services to meet evolving expectations.

Versatility of Classification Models:
The use of various classification models, such as Decision Tree, Random Forest, and XGBoost, provides flexibility in approaching sentiment analysis and review categorization, allowing for a more comprehensive understanding of textual data.

Disadvantages:

Complex Model Selection:
The need to choose and integrate appropriate classification models introduces complexity, and selecting the wrong model for the task may result in suboptimal performance.

Data Preprocessing Challenges:
Preprocessing raw text data involves addressing challenges such as noise, variability in language, and context-dependent sentiments, which can be complex and resource-intensive.

Over fitting Risks:
There's a risk of over fitting, especially when using complex classification models. Over fit models may perform well on the training data but may not generalize effectively to new, unseen data.

Interpretability Issues:
Some advanced classification models, such as Random Forest or XGBoost, might lack interpretability, making it challenging to understand how the model reaches specific decisions. This can be a concern, especially in industries where interpretability is crucial.

Data Bias:
User-generated content might exhibit bias, and the classification system may inadvertently perpetuate or amplify such biases. Addressing this bias requires careful consideration and ongoing monitoring.

Resource Intensiveness:
Training and maintaining sophisticated classification models can be resource-intensive in terms of computational power, time, and expertise. This can pose challenges for smaller airlines or those with limited resources.

11. CONCLUSION

In conclusion, the development of an airline review classification system using advanced Classification models represents a strategic initiative for the aviation industry in adapting to the demands of an interconnected world. The project recognizes the critical role of the airline industry in global travel and business and emphasizes the importance of quality service in shaping positive passenger experiences. The focus on leveraging machine learning models, including Decision Tree Classifier, Random Forest Classifier, XGBoost Classifier, among others, to analyse user-generated content from social media, travel websites, and online forums, holds immense potential for extracting actionable insights.

12. FUTURE SCOPE

Sentiment Analysis Refinement:
Future iterations of the system can explore advanced sentiment analysis techniques, including sentiment intensity and context-based sentiment analysis, to enhance the granularity of insights derived from reviews.

Incorporation of Deep Learning Models:
Integration of deep learning models, such as recurrent neural networks (RNNs) or transformer models, could provide more nuanced understanding of language patterns in reviews, potentially improving the accuracy of sentiment analysis.

Real-time Analysis and Feedback:
Future developments may focus on implementing real-time analysis capabilities, allowing airlines to receive immediate insights into emerging trends and issues, enabling more agile and proactive service adjustments.

13. APPENDIX

GitHub & Project Demo Link

GitHub: https://github.com/smartinternz02/SI-GuidedProject-612319-1698643418

Video Link:
https://drive.google.com/file/d/13DlNiYSZMGlNAb9tzYeB6teCatpdpF0m/view?usp=sharing