AIRLINE REVIEW CLASSIFICATION

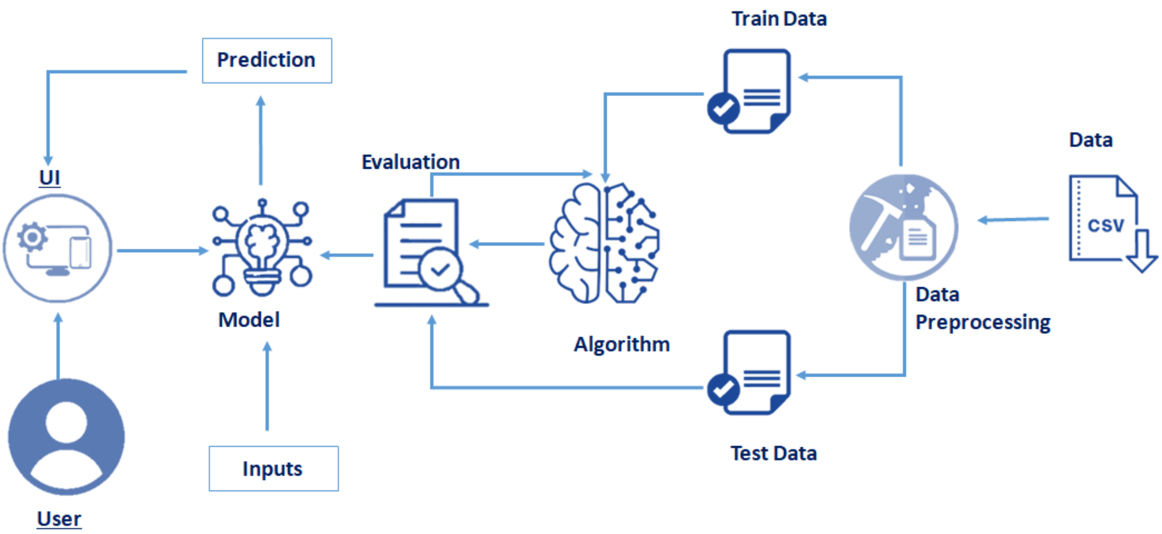
Project Description:

In today's interconnected world, the airline industry serves as a critical catalyst for global travel and business. As air travel becomes increasingly accessible, the quality of service provided by airlines plays a pivotal role in shaping passenger experiences. This project focuses on the development of an airline review classification system using Classification models such as Decision Tree Classifier, Random Forest Classifier, XGBoost Classifier etc.,

The proliferation of social media platforms, travel websites, and online forums has given rise to a wealth of user-generated content, including airline reviews. Extracting actionable insights from this vast pool of unstructured text data has the potential to provide airlines with valuable information for refining their services and elevating passenger satisfaction.

Throughout this report, we will delve into the methodology employed to preprocess the raw text data, the process of selecting pertinent features, the training and evaluation of the classification model, and the subsequent interpretation of the obtained results.

Technical Architecture:

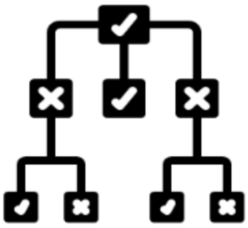


Project Flow:

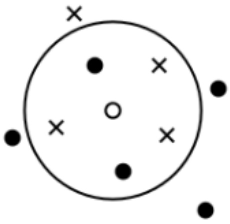
- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI to accomplish this, we have to complete all the activities listed below,
- Data Collection & Preparation
    - Collect the dataset
    - Data Preparation
    - Exploratory Data Analysis
- Descriptive statistical
    - Visual Analysis
- Model Building
    - Training the model in multiple algorithms
    - Testing the model
- Performance Testing
    - Testing model with multiple evaluation metrics
- Model Deployment
    - Save the best model
    - Integrate with Web Framework
- Project Demonstration & Documentation
    - Record explanation Video for project end to end solution

<u>Prior Knowledge</u>:

You must have prior knowledge of following Supervised Learning topics of Machine Learning to complete this project.

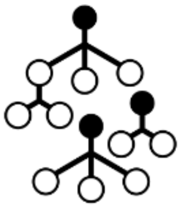


<u>Decision Tree</u>



<u>K-Nearest Neighbors</u>

<u>Logistic Regression</u>



<u>Random Forest</u>

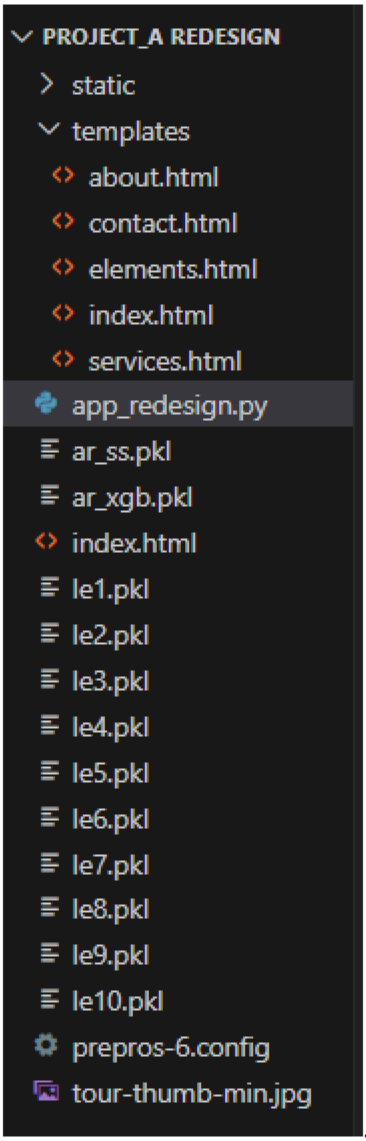
HYPERLINK "https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/"<u>Support Vector Machine</u>

<u>XGBoost</u>

<u>Evaluation Metrics</u>

<u>Flask</u>

Project Structure:



.

Create the project folder which contains files as shown below

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app_redesign.py for scripting.
- ar_xgb.pkl is our saved model. Further we will use this model for flask integration.
- ar_ss.pkl is our pickle file of Standard Scaler Feature.
- [le1.pkl, le2.pkl, le3.pkl, le4.pkl, le5.pkl, le6.pkl, le7.pkl, le8.pkl, le9.pkl, le10.pkl] are the pickle files of Label encoding.

Milestone 1: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project, we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

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

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are several techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 2: Importing the libraries

Import the necessary libraries as shown in the image.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
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.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
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
import warnings
warnings.filterwarnings('ignore')
```

Activity 3: 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.

```
ar=pd.read_csv(r"D:\Project_A\Airline_Reviews.csv")
ar.head(5)
```

	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	Food & Beverages	Ground Service	Inflight Entertainment	Wifi & Connectivity	Value For Money	Recommended
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	4.0	4.0	NaN	NaN	3.0	yes
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	1.0	1.0	NaN	NaN	2.0	no
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	1.0	1.0	NaN	NaN	2.0	no
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	NaN	1.0	NaN	NaN	1.0	no
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	1.0	1.0	1.0	1.0	1.0	1.0	no

Activity 4: 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.

Activity 5: Handling missing values

- Let's find the shape of our dataset first. To find the shape of our data,

```
ar.shape
```

```
(23171, 20)
```

```
ar.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
```

  the ar.shape method is used.
- To find the data type, ar.info() function is used.

- For checking the null values, ar.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 ar.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 ar.drop([ ],axis=1) to remove unnecessary rows

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

In the above dataset check also for any unnecessary values, upon checking the Overall_Rating column we found some irrelevant value, so we replace it with relevant value. And later fill all the null values.

```
nar['Overall_Rating']=nar['Overall_Rating'].replace(['1','2','3','4,','5','6','7','8','9','n'],['1','2','3','4','5','6','7','8','9','10'])
```

i) Filling the null values with median and mode depending on the values(Mode for Categorical and median for Numerical)

```
nar['Type Of Traveller']=nar['Type Of Traveller'].fillna(nar['Type Of Traveller'].mode()[0])
nar['Seat Type']=nar['Seat Type'].fillna(nar['Seat Type'].mode()[0])
nar['Seat Comfort']=nar['Seat Comfort'].fillna(nar['Seat Comfort'].mode()[0])
nar['Route']=nar['Route'].fillna(nar['Route'].mode()[0])
nar['Date Flown']=nar['Date Flown'].fillna(nar['Date Flown'].mode()[0])
nar['Food & Beverages']=nar['Food & Beverages'].fillna(nar['Food & Beverages'].mode()[0])
nar['Ground Service']=nar['Ground Service'].fillna(nar['Ground Service'].mode()[0])
#For the above columns we are using mode instead of median even though numerical values are present
#because 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.

```
nar[['Month Flown','Year Flown']]=nar['Date Flown'].str.split(expand=True)
```

```
nar['Origin']=nar.Route.str.split(' to ',expand=True)[0]
nar['Destination']=nar.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'
nar['Destination']=nar.Destination.str.split(' via ',expand=True)[0]
```

```
del nar['Route']
del nar['Date Flown']
```

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

```
#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:
    nar.at[i,'Destination']=new_des[j]
    j+=1
```

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.

```
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']
```

```
# Reordering the columns of given data to our desired manner
nar=nar.reindex(columns=new_column_order)
```

```
nar.head()
```

	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

Activity 6: Handling Categorical Values

```python
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()
```

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

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

```python
nar.head()
```

	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	1271	1545	9	6	1	4.0	4.0	4.0	9	1
1	0	1	3	1271	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	628	1927	11	6	0	1.0	1.0	1.0	0	0
4	4	1	1	1826	99	11	6	1	1.0	1.0	1.0	0	0

Milestone 2: Exploratory Data Analysis

Activity 1: Descriptive statistics

Descriptive analysis is to study the basic features of data with the statistical

```
nar.describe()
```

	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

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.

Activity 2: Visual analysis

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.

Activity 3: Univariate Analysis

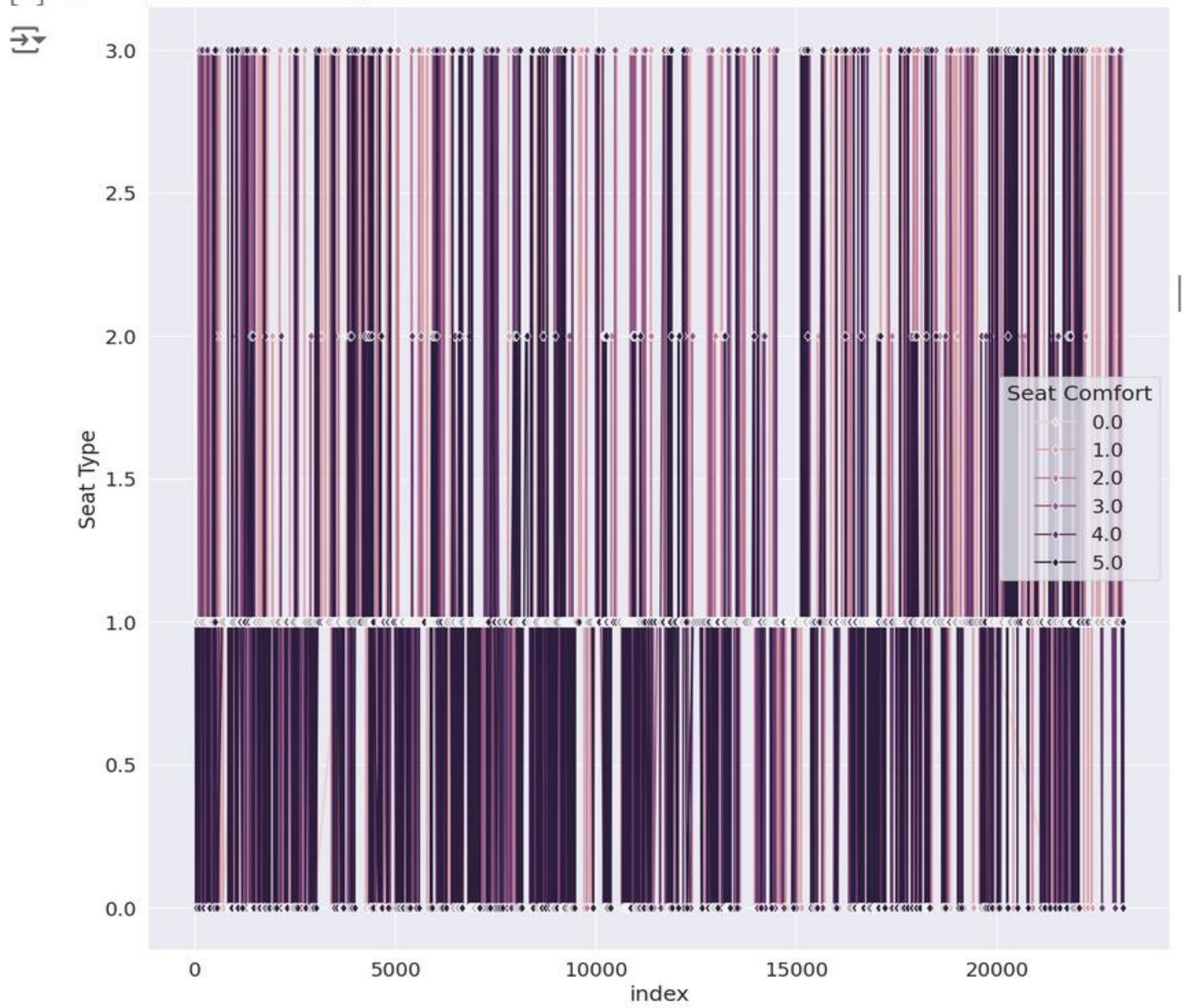
In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and

```
#line plot in seaborn
sns.set(rc={'figure.figsize':[15,15]})
sns.set(font_scale=1.5)
fig=sns.lineplot(x=nar.index,y=nar['Seat Type'],markevery=1,marker='d',
                 hue=nar['Seat Comfort'])
fig.set(xlabel='index')
```

countplot.

Seaborn 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.

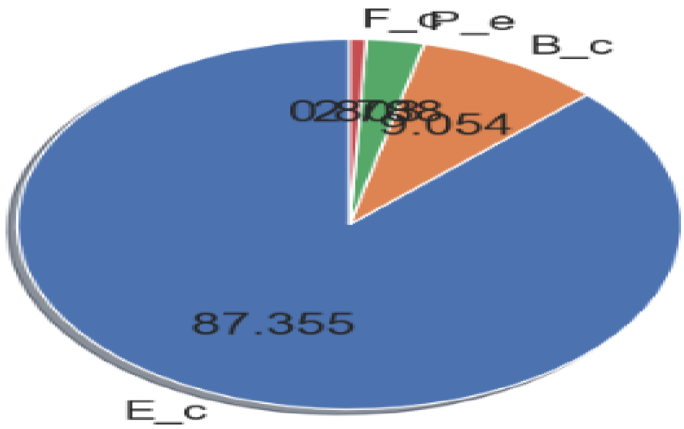
```
[ ]  [Text(0.5, 0, 'index')]
```



Note: In our dataset we used lineplot, bar plot, pieplot and not using distplot and countplot because our dataset is not having continuous values

```
plt.figure(figsize=(5,5))
plt.pie(nar['Seat Type'].value_counts(),startangle=90,autopct='%.3f',
        labels=['E_c','B_c','  P_e','F_c'],shadow=True)
```

```
([<matplotlib.patches.Wedge at 0x1e51ff69590>,
  <matplotlib.patches.Wedge at 0x1e51fe89190>,
  <matplotlib.patches.Wedge at 0x1e51ffa4990>,
  <matplotlib.patches.Wedge at 0x1e51ffa7c50>],
 [Text(-0.4255806105829949, -1.0143377858957072, 'E_c'),
  Text(0.5370556832029272, 0.9599849963095451, 'B_c'),
  Text(0.15134436463791998, 1.0895388397355756, '  P_e'),
  Text(0.02773750481306780 7, 1.099650231131129, 'F_c')],
 [Text(-0.2321348784998153 4, -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')])
```



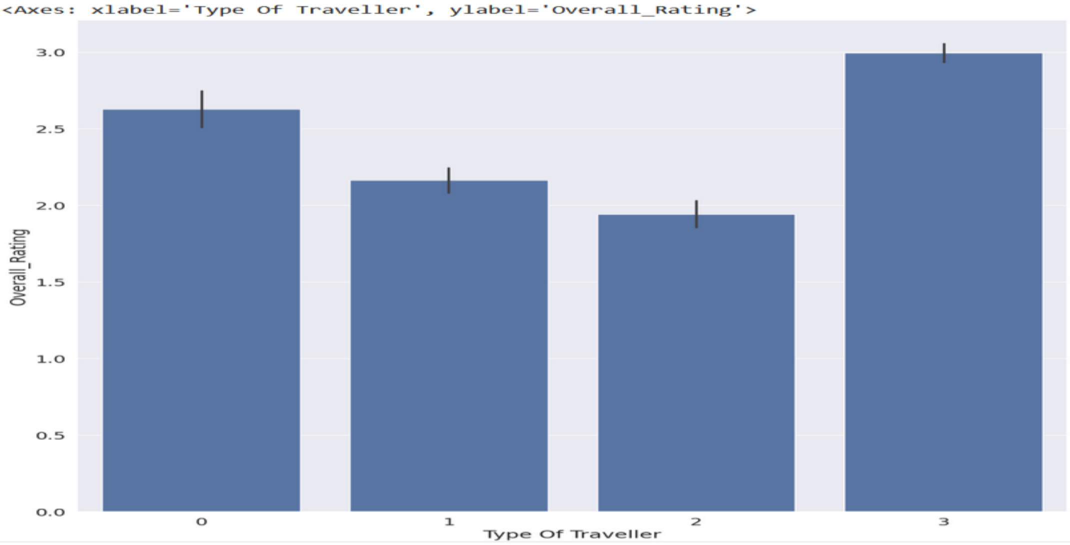

```
nar['Seat Type'].value_counts().plot.bar()
```

Activity 4: 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.

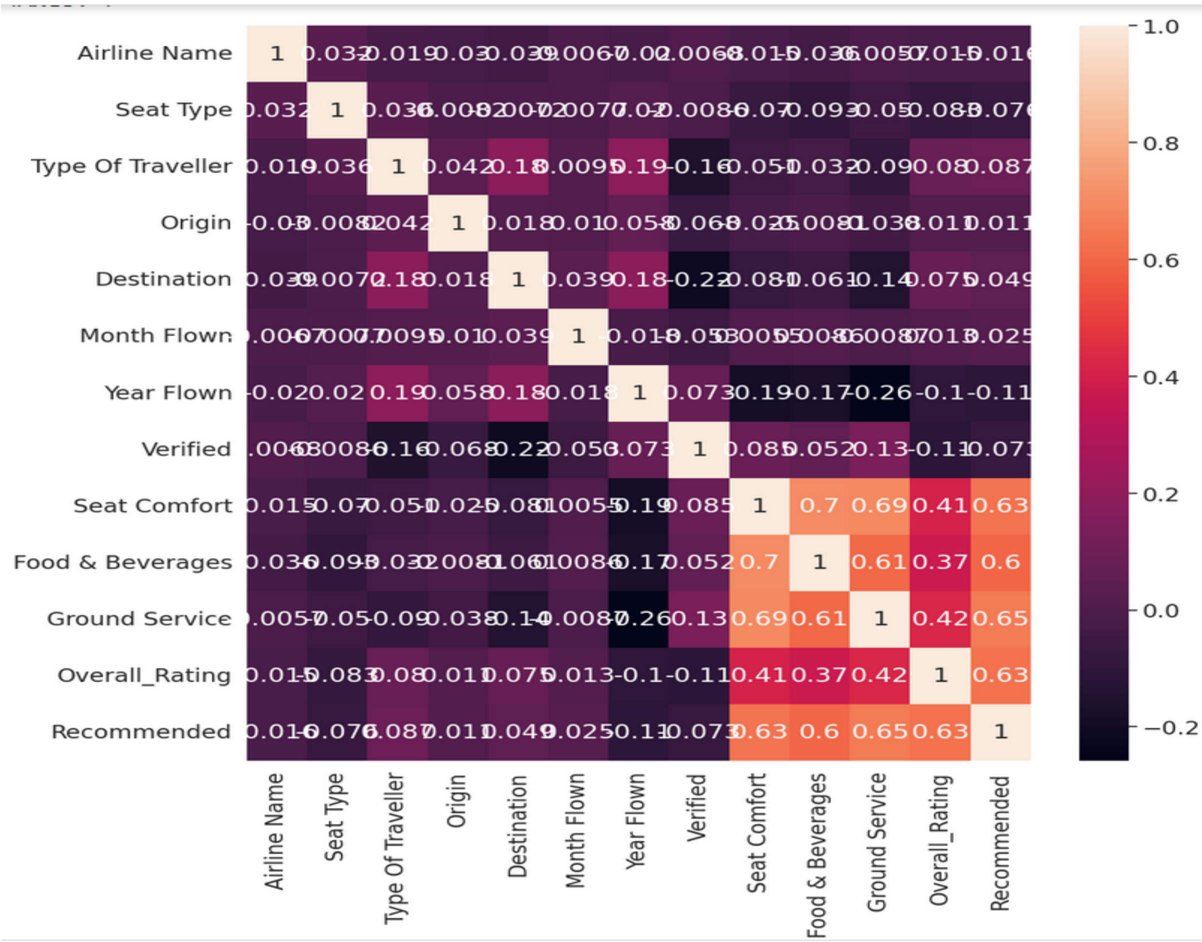
```
sns.barplot(data=nar,x='Type Of Traveller',y='Overall_Rating')
```

```
<Axes: xlabel='Type Of Traveller', ylabel='Overall_Rating'>
```



<u>Activity 5: Multivariate Analysis</u>

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



Milestone 3: Model Building

Activity 1: Splitting Data Into Train And Testing

Now let's split the Dataset into train and test sets. First split the dataset into X and y and then split the dataset. Here X and y variables are

```
X=nar.iloc[:,0:12].values
y=nar.iloc[:,12:13].values
```

```
X
```

```
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.]])
```

```
y
```

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

created.

On X variable, nar is passed with dropping the target variable. And on y target variable is passed.

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).

After checking ,we got to know that there is an imbalance of values in target variable.

```
nar.Recommended.value_counts()
```

```
0    15364
1     7807
Name: Recommended, dtype: int64
```

We can observe that after doing SMOTE the count of each value in target variable became equal.

```
# As the values are over_sampling we need to use smote technique
from imblearn.over_sampling import SMOTE
smote=SMOTE(sampling_strategy='auto',random_state=50)
```

```
X,y=smote.fit_resample(X,y)
```

```
np.count_nonzero(y==1)
```

```
15364
```

```
np.count_nonzero(y==0)
```

```
15364
```

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

```
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)
```

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

Activity 2: 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.

Decision tree model

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

```
dtc.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=dtc.predict(X_test)
pred_dt
```

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

sklearn.metrics section

```
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
```

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.

K-Nearest Neighbors

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

sklearn.metrics section

```
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))


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

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

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

```
              precision    recall  f1-score   support

           0       0.95      0.93      0.94      3116
           1       0.93      0.95      0.94      3030

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

roc_auc_knn : 0.9419214995954025
cm_knn: [[2897  219]
 [ 139 2891]]
as_knn: 0.941750732183534
```

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.

LogisticRegression

```
from sklearn.linear_model import LogisticRegression
Lr=LogisticRegression()
```

```
Lr.fit(X_train,y_train)
```

```
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.

```
pred_Lr=Lr.predict(X_test)
pred_Lr
```

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

sklearn.metrics section

```
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))


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

cm_Lr=confusion_matrix(y_test,pred_Lr)
print("cm_Lr:",cm_Lr)

as_Lr=accuracy_score(y_test,pred_Lr)
print("as_Lr:",as_Lr)
```

```
              precision    recall  f1-score   support

           0       0.93      0.92      0.92      3116
           1       0.92      0.92      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

roc_auc_Lr : 0.9217793184966763
cm_Lr: [[2863  253]
 [ 228 2802]]
as_Lr: 0.9217377155873739
```

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.

Naive Bayes Classification

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.

```
from sklearn.naive_bayes import GaussianNB
gnb=GaussianNB()
```

```
gnb.fit(X_train,y_train)
```

```
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.

```
pred_nb=gnb.predict(X_test)
```

sklearn.metrics section

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curve,auc
fpr_gnb,tpr_gnb,threshold_gnb=roc_curve(y_test,pred_Lr)

print(classification_report(y_test,pred_Lr))


roc_auc_nb=auc(fpr_gnb,tpr_gnb)
print("roc_auc_nb :",roc_auc_nb)

cm_nb=confusion_matrix(y_test,pred_nb)
print("cm_nb:",cm_nb)

as_nb=accuracy_score(y_test,pred_nb)
print("as_nb:",as_nb)
```

```
              precision    recall  f1-score   support

           0       0.93      0.92      0.92      3116
           1       0.92      0.92      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

roc_auc_nb : 0.9217793184966763
cm_nb: [[2754  362]
 [ 203 2827]]
as_nb: 0.9080702896192646
```

Random Forest Classification

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 model, a confusion matrix and classification report is done.

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=2)
```

```
rfc.fit(X_train,y_train)
```

```
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.

```
pred_rfc=rfc.predict(X_test)
pred_rfc
```

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

sklearn.metrics section

```
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))


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

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

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

```
              precision    recall  f1-score   support

           0       0.96      0.97      0.96      3116
           1       0.96      0.96      0.96      3030

    accuracy                           0.96      6146
   macro avg       0.96      0.96      0.96      6146
weighted avg       0.96      0.96      0.96      6146

roc_auc_rfc : 0.9612088359028457
cm_rfc: [[3010  106]
 [ 132 2898]]
as_rfc: 0.9612756264236902
```

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.

```
from sklearn.svm import SVC
svc=SVC()
```

```
svc.fit(X_train,y_train)
```

```
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.

```
pred_svc=svc.predict(X_test)
pred_svc
```

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

sklearn.metrics section

```
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))


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

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

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

```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94      3116
           1       0.95      0.93      0.94      3030

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

roc_auc_svc : 0.9397692946444837
cm_svc: [[2971  145]
 [ 224 2806]]
as_svc: 0.9399609502115197
```

XGBoost Classifier

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 overfitting and at the same time both the testing and training accuracies are best. So will test this model.

```
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, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=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,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=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)
```

sklearn.metrics section

```
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))


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

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

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

```
              precision    recall  f1-score   support

           0       0.96      0.97      0.96      3116
           1       0.97      0.96      0.96      3030

    accuracy                           0.96      6146
   macro avg       0.96      0.96      0.96      6146
weighted avg       0.96      0.96      0.96      6146

roc_auc_xgb : 0.9631753708105085
cm_xgb: [[3013  103]
 [ 123 2907]]
as_xgb: 0.9632281158477058
```

```
pred_xgb1=xgb.predict(X_train)
```

```
print(classification_report(y_train,pred_xgb1))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99     12248
           1       0.99      0.98      0.99     12334

    accuracy                           0.99     24582
   macro avg       0.99      0.99      0.99     24582
weighted avg       0.99      0.99      0.99     24582
```

Activity 3 : Testing The Model

Here we have tested with XGBoost algorithm. You can test with all algorithm. With the help of predict() function.

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

```
array([1])
```

```
# As there is a very less difference between accuracies of training and testing models ,there is no issue of overfitting
```

```
com=pd.DataFrame({'Model':['DecisionTree 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_nb,roc_auc_rfc,roc_auc_svc,roc_auc_xgb],
                  'accuracy':[as_dt,as_knn,as_Lr,as_nb,as_rfc,as_svc,as_xgb]})
```

```
com
```

	Model	roc_auc	accuracy
0	DecisionTree Classification	0.950830	0.950862
1	K-Nearest Neighbours	0.941921	0.941751
2	Logistic Regression	0.921779	0.921738
3	Naive Bayes Classification	0.921779	0.908070
4	RandomForest Classification	0.961209	0.961276
5	Support Vector Machine	0.939769	0.939961
6	XGBClassifier	0.963175	0.963228

```
maxi=0
for i in range(len(com['Model'])):
    if com.iloc[i:i+1,1:2].values>maxi:
        maxi=com.iloc[i:i+1,1:2].values
        model=com.iloc[i:i+1,0:1].values
    else:
        pass
print('Best accuracy score is:',maxi,'by',model)
maxi=0
for i in range(len(com['Model'])):
    if com.iloc[i:i+1,2:3].values>maxi:
        maxi=com.iloc[i:i+1,2:3].values
        model=com.iloc[i:i+1,0:1].values
    else:
        pass
print('Best roc_auc is:',maxi,'by',model)
```

```
Best accuracy score is: [[0.96317537]] by [['XGBClassifier']]
Best roc_auc is: [[0.96322812]] by [['XGBClassifier']]
```

Milestone 4: Model Deployment

Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
import pickle
pickle.dump(xgb,open('ar_xgb.pkl','wb'))
```

Activity 2: Integrate with Web Framework

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

In this section, we will be building a web application that is integrated into 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

Activity 3: Building Html Pages:

For this project create two HTML files namely and save them in the templates folder. Refer this link for templates

. home.html predict.html submit.html

Activity 4: 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.

```python
from flask import Flask, render_template, request
from sklearn.preprocessing import LabelEncoder
import pickle
import os

app = Flask(__name__)

# Load models and encoders
try:
    model_path = os.path.join(os.path.dirname(__file__), 'ar_xgb.pkl')
    ss_path = os.path.join(os.path.dirname(__file__), 'ar_ss.pkl')

    with open(model_path, 'rb') as model_file:
        model = pickle.load(model_file)
    with open(ss_path, 'rb') as ss_file:
        ss1 = pickle.load(ss_file)

    le1 = LabelEncoder()
    le2 = LabelEncoder()
    le3 = LabelEncoder()
    le4 = LabelEncoder()
    le5 = LabelEncoder()
    le6 = LabelEncoder()
    le7 = LabelEncoder()
    le8 = LabelEncoder()
    le9 = LabelEncoder()
    le10 = LabelEncoder()

    # Assuming we need to fit the encoders here
    # Example: le1.fit(['Airline1', 'Airline2', ...])
    # Repeat for other encoders

except Exception as e:
    print(f"Error loading models or encoders: {e}")

@app.route('/')
def home():
    return render_template("home.html")

@app.route('/predict')
```

```python
@app.route('/submit', methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        try:
            # Retrieve form data
            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 = request.form['S_C']
            F_B = request.form['F_B']
            G_S = request.form['G_S']
            O_R = request.form['O_R']

            # Encode categorical data
            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],
                int(S_C),  # Convert to int
                int(F_B),  # Convert to int
                int(G_S),  # Convert to int
                le9.transform([O_R])[0]
            ]

            # Print encoded data for debugging
            print(encoded_data)

            # Make prediction
            prediction = model.predict(ss1.transform([encoded_data]))
```

```python
            # Determine recommendation
            recommendation = "Recommended" if prediction == 1 else "Not Recommended"

        except Exception as e:
            print(f"Error during form processing or prediction: {e}")
            return render_template('submit.html', error=str(e))

    # Render submit page initially or on GET request
    return render_template('submit.html')

if __name__ == '__main__':
    app.run(debug=False, port=5000)
```

Render HTML page:

```python
def hello_world():
    return render_template("index.html")
@app.route('/guest',methods=['POST'])
```

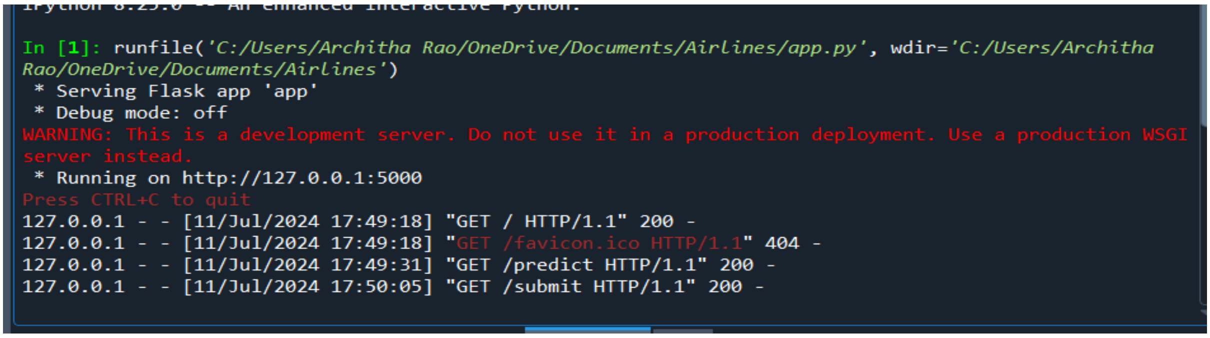
Here we will be using a declared constructor to route to the HTML page which we have created earlier. In the above example, '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method. Retrieves the value from UI:

Here we are routing our app to Guest() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the index.html page. Set app.run(debug=True) so that we can edit.

Activity 5: Run the web application

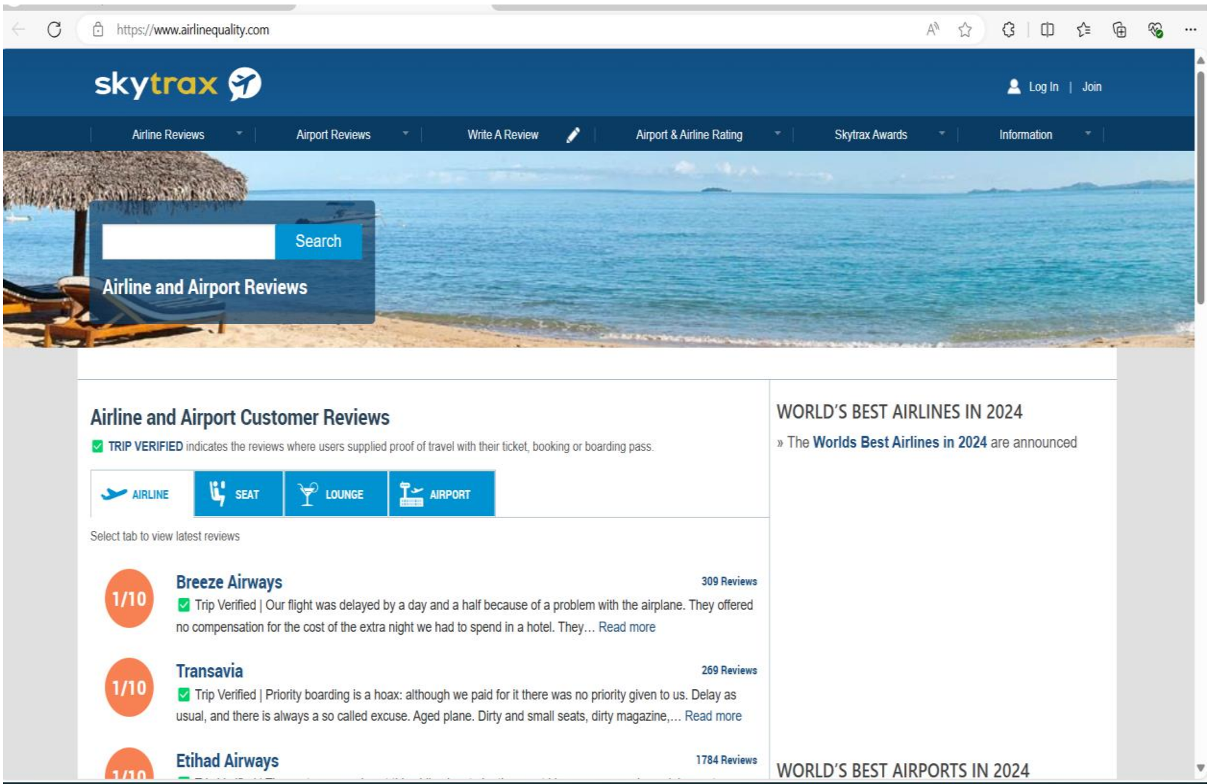
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app.py" command
- Navigate to the localhost where you can view your web page.

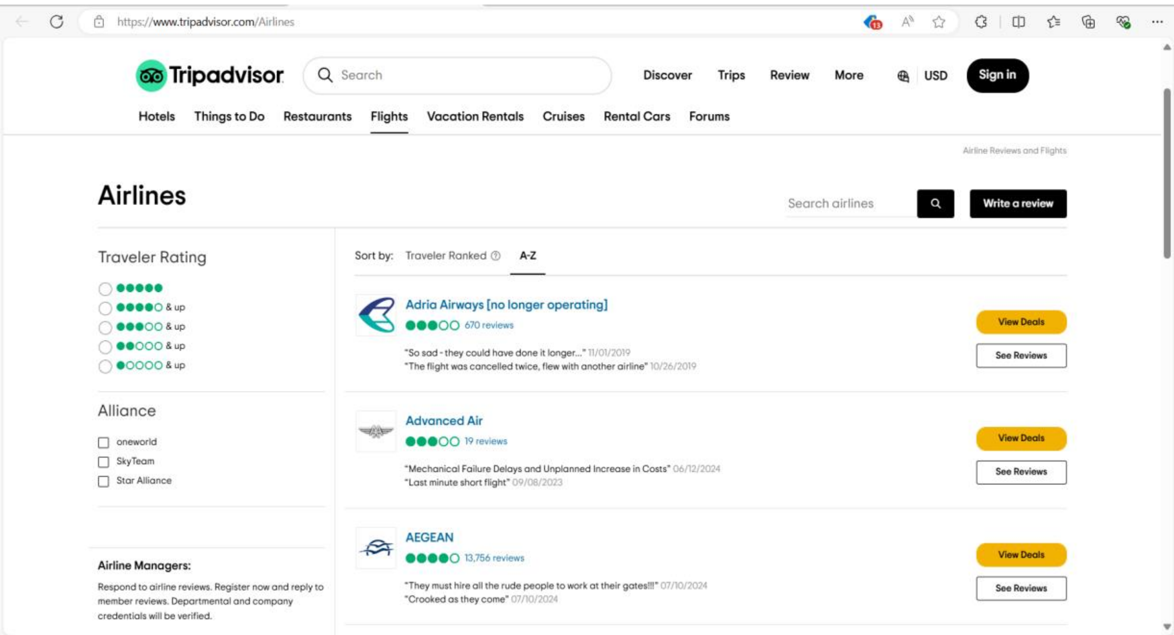
Enter the inputs, click on the Check It button, and see the result/prediction on the web



```
In [1]: runfile('C:/Users/Architha Rao/OneDrive/Documents/Airlines/app.py', wdir='C:/Users/Architha
Rao/OneDrive/Documents/Airlines')
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [11/Jul/2024 17:49:18] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Jul/2024 17:49:18] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [11/Jul/2024 17:49:31] "GET /predict HTTP/1.1" 200 -
127.0.0.1 - - [11/Jul/2024 17:50:05] "GET /submit HTTP/1.1" 200 -
```

Now, Go the web browser and write the localhost URL (http://127.0.0.1:5000) to get the below result




https://www.tripadvisor.com/Airlines
Tripadvisor
Search
Discover
Trips
Review
More
USD
Sign in
Hotels
Things to Do
Restaurants
Flights
Vacation Rentals
Cruises
Rental Cars
Forums
Airline Reviews and Flights
Airlines
Search airlines
Write a review
Traveler Rating
& up
& up
& up
& up
Alliance
oneworld
SkyTeam
Star Alliance
Airline Managers:
Respond to airline reviews. Register now and reply to member reviews. Departmental and company credentials will be verified.
Sort by: Traveler Ranked
A-Z
Adria Airways [no longer operating]
670 reviews
"So sad - they could have done it longer..." 11/01/2019
"The flight was cancelled twice, flew with another airline" 10/26/2019
View Deals
See Reviews
Advanced Air
19 reviews
"Mechanical Failure Delays and Unplanned Increase in Costs" 06/12/2024
"Last minute short flight" 09/08/2023
View Deals
See Reviews
AEGEAN
13,756 reviews
"They must hire all the rude people to work at their gates!!!" 07/10/2024
"Crooked as they come" 07/10/2024
View Deals
See Reviews