Project Manual

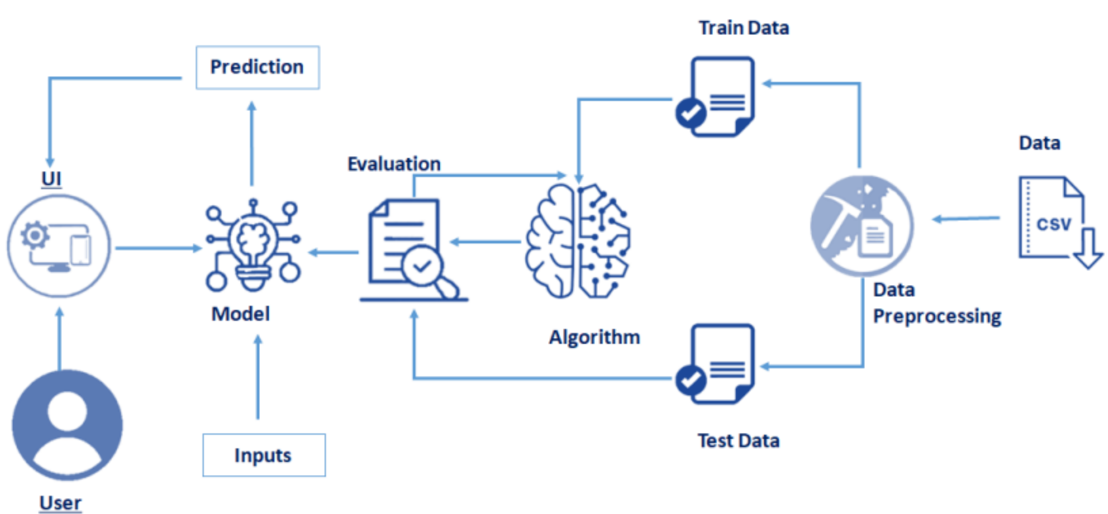
Date	19 September 2022
Team ID	PNT2022TMID592960
Project Name	Project - Airline Review Classification Using Machine Learning

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.

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

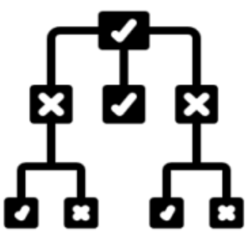
➢ Performance Testing
  - Testing model with multiple evaluation metrics

➢ Model Deployment
  - Save the best model
  - Integrate with Web Framework

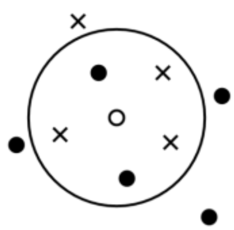
➢ Project Demonstration & Documentation
  - Record explanation Video for project end to end solution

Prior Knowledge:

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

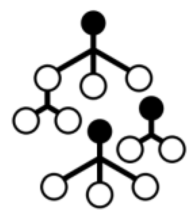


Decision Tree



K-Nearest Neighbors

Logistic Regression



Random Forest

HYPERLINK

XGBoost

Evaluation Metrics

Flask
web development,
one drop at a time

Flask

Airline Reviews Dataset Link:

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

Project Milestones:

1. Import Dataset and Libraries

a. Importing Libraries

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier

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

import pickle
from scipy import stats

import warnings
warnings.filterwarnings('ignore')
```

b. Importing Dataset

In [74]:
```python
reviews=pd.read_csv(r'E:\Internships\AI and ML SmartInternz Externship\AIRLINE REVIEW CLASSIFICATION\Airline_Reviews.csv')
reviews.head()
```

Out[74]:

	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
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
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

2. Data Preprocessing

a. Handling NULL Values

Dropping the Unnecessary Columns

In [7]:
```
reviews.drop(['Inflight Entertainment','Wifi & Connectivity','Aircraft',
              'Cabin Staff Service','Unnamed: 0','Review Date','Review_Title','Review'],axis=1,inplace=True)
```

In [11]:
```
reviews['Overall_Rating']=reviews['Overall_Rating'].replace(['1','2','3','4','5','6','7','8','9','n'],
                                                             ['1','2','3','4','5','6','7','8','9','10'])
```

In [13]:
```
reviews['Type Of Traveller']=reviews['Type Of Traveller'].fillna(reviews['Type Of Traveller'].mode()[0])
reviews['Seat Type']=reviews['Seat Type'].fillna(reviews['Seat Type'].mode()[0])
reviews['Route']=reviews['Route'].fillna(reviews['Route'].mode()[0])
reviews['Date Flown']=reviews['Date Flown'].fillna(reviews['Date Flown'].mode()[0])
reviews['Seat Comfort']=reviews['Seat Comfort'].fillna(reviews['Seat Comfort'].mode()[0])
reviews['Food & Beverages']=reviews['Food & Beverages'].fillna(reviews['Food & Beverages'].mode()[0])
reviews['Ground Service']=reviews['Ground Service'].fillna(reviews['Ground Service'].mode()[0])
reviews['Value For Money']=reviews['Value For Money'].fillna(reviews['Value For Money'].mode()[0])
```

b. Breaking Columns into sub-columns

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

```
reviews['Origin']=reviews['Route'].str.split(' to ',expand=True)[0]
reviews['Destination']=reviews['Route'].str.split(' to ',expand=True)[1]

## Removing the via city
reviews['Destination']=reviews['Destination'].str.split(' via ',expand=True)[0]
```

```
## Reordering the Columns

cols=['Airline Name','Seat Type','Type Of Traveller','Origin','Destination','Month Flown','Year Flown',
      'Verified','Seat Comfort','Food & Beverages','Ground Service','Overall_Rating','Value For Money','Recommended']
reviews=reviews.reindex(columns=cols)
```

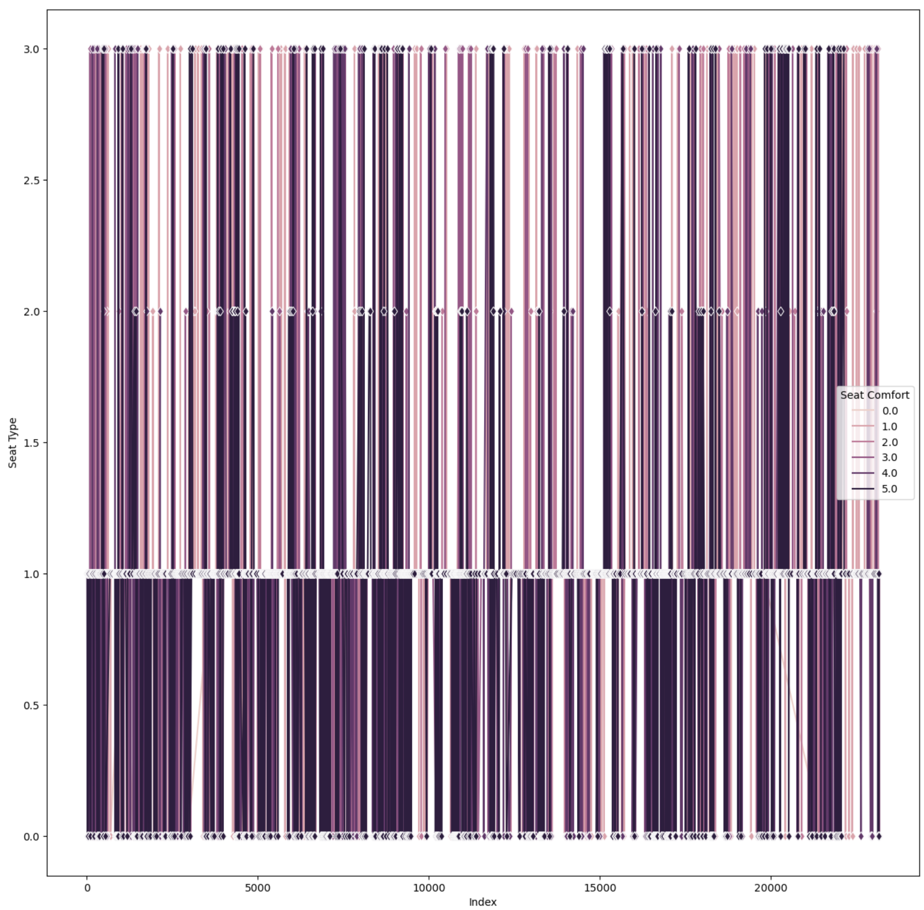
c. Handling Categorical Columns

```
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()
```

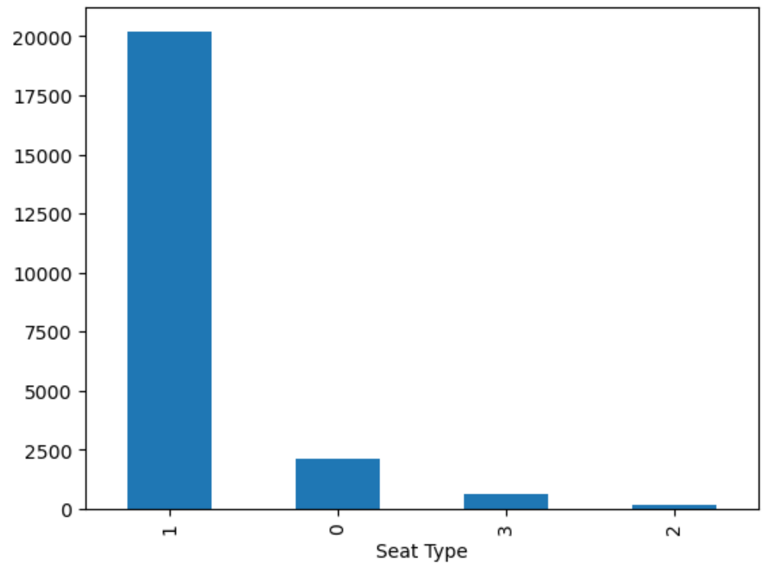
```python
reviews['Airline Name']=le1.fit_transform(reviews['Airline Name'])
reviews['Seat Type']=le2.fit_transform(reviews['Seat Type'])
reviews['Type Of Traveller']=le3.fit_transform(reviews['Type Of Traveller'])
reviews['Origin']=le4.fit_transform(reviews['Origin'])
reviews['Destination']=le5.fit_transform(reviews['Destination'])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

3. Exploratory Data Analysis

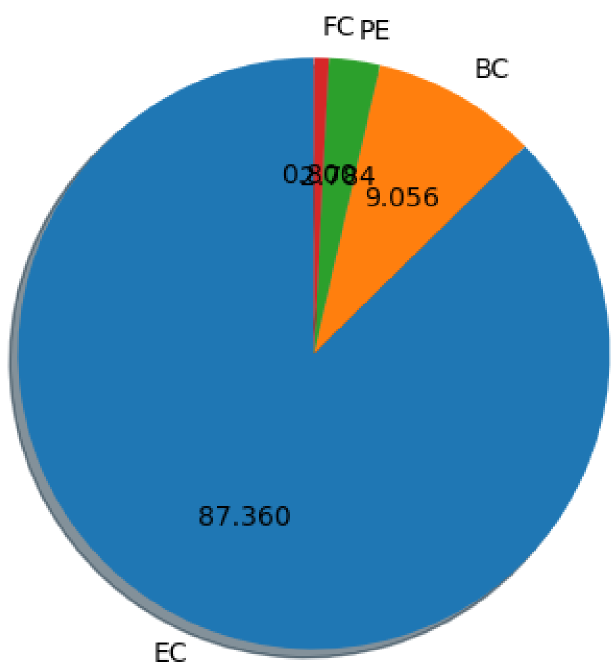
```python
## Line Plot in Seaborn

plt.figure(figsize=(15,15))
fig=sns.lineplot(x=reviews.index,y=reviews['Seat Type'],markevery=1,marker='d',hue=reviews['Seat Comfort'])
fig.set(xlabel='Index')
```



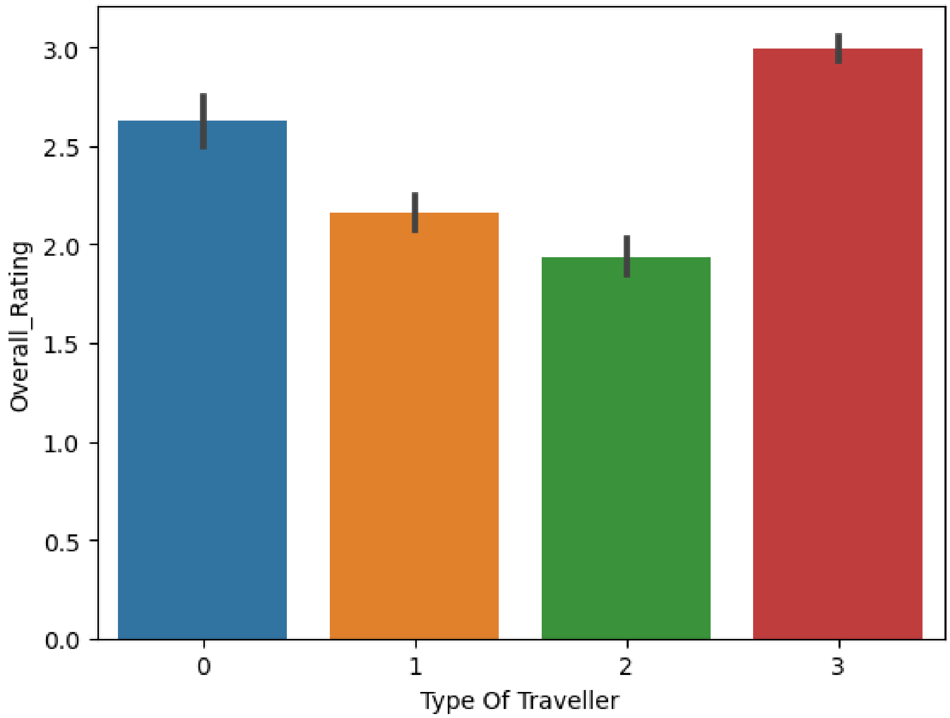

```python
reviews['Seat Type'].value_counts().plot.bar()
```



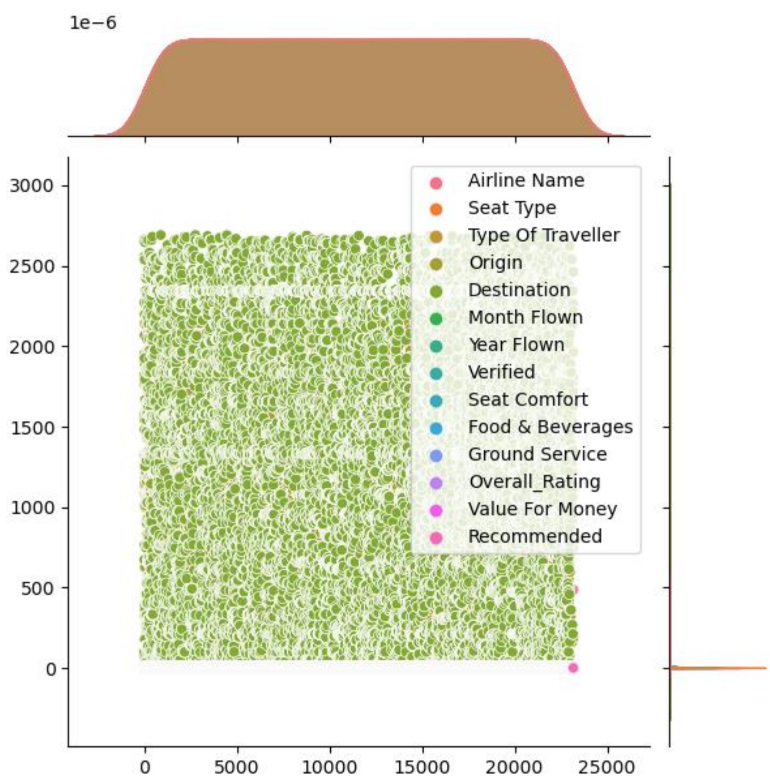

```python
plt.figure(figsize=(5,5))
plt.pie(reviews['Seat Type'].value_counts(),startangle=90,autopct='%.3f',
       labels=['EC','BC','PE','FC'],shadow=True)
```



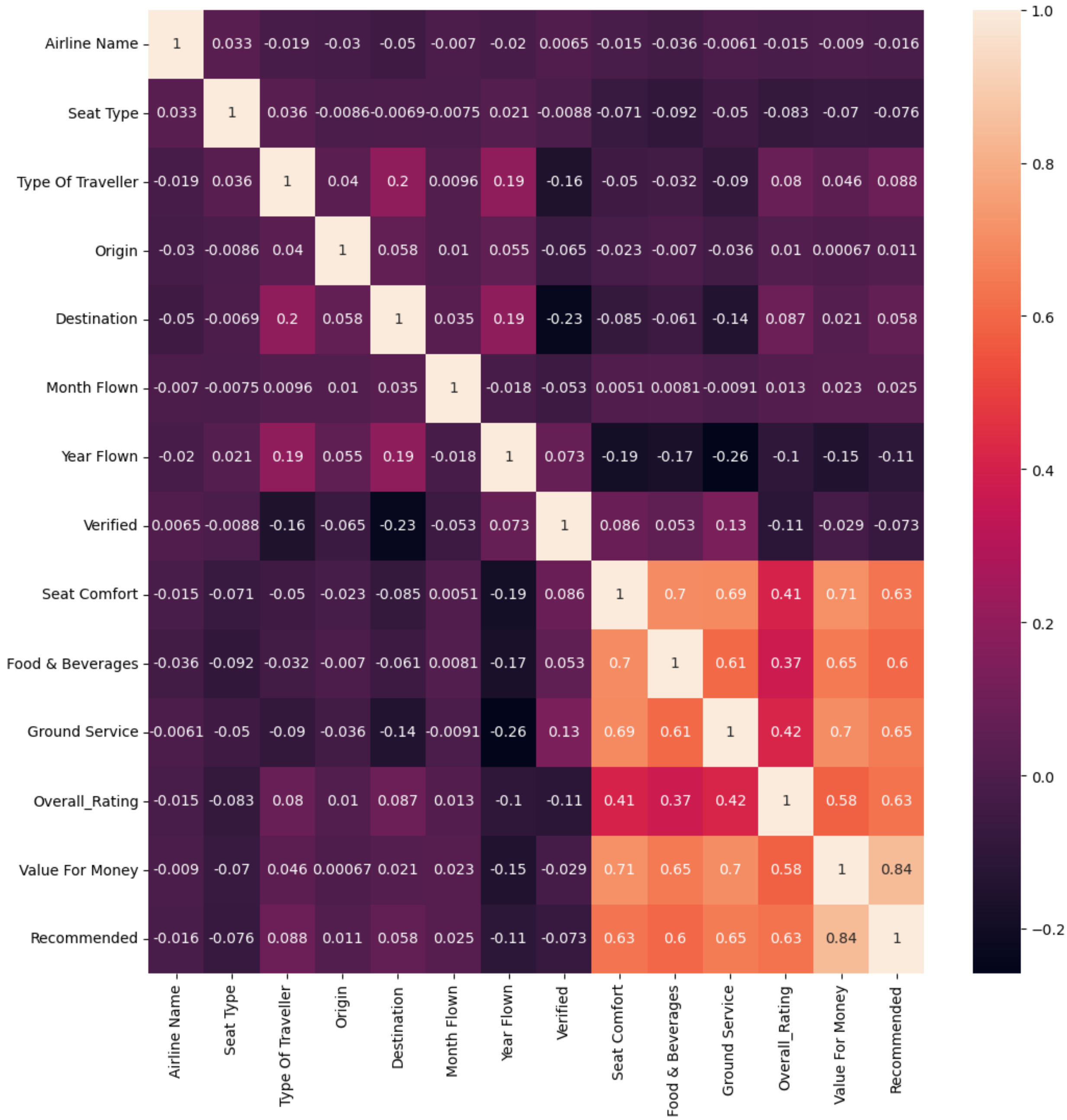

```python
sns.barplot(data=reviews,x='Type Of Traveller',y='Overall_Rating')
```




```
sns.jointplot(reviews)
```



```python
plt.figure(figsize=(12,12))
sns.heatmap(reviews.corr(),annot=True)
```



4. Model Building Preparations

a. Splitting Data into Training and Testing Sets

From the heatmap showing the correlation it is clear that the columns ['Airline Name','Seat Type', 'Type of Traveller','Origin','Destination','Month Flown','Year Flown','Verified'] has very low correlation and thus can be dropped from the training.

```
X=reviews.iloc[:,8:13].values
y=reviews.iloc[:,13:14].values
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1)
```

b. Removing Class Imbalance

```
## As the values are imbalanced and have over sampling, we use SMOTE
smote=SMOTE(sampling_strategy='auto',random_state=50)
```

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

c. Scaling the column values

```
ss=StandardScaler()
X=ss.fit_transform(X)
```

5. Model Training and Testing

a. Decision Tree Classifier

```
dtc=DecisionTreeClassifier(criterion='entropy',random_state=50)
dtc.fit(X_train,y_train)
```

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

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

```
fpr_dt,tpr_dt,thres_dt=roc_curve(y_test,pred_dtc)
roc_auc_dt=auc(fpr_dt,tpr_dt)

print(classification_report(y_test,pred_dtc))

print('ROC AUC DTC= ',roc_auc_dt)

cm_dt=confusion_matrix(y_test,pred_dtc)
print('Confusion Matrix DTC: ')
print(cm_dt)

as_dt=accuracy_score(y_test,pred_dtc)
print('Accuracy DTC: ',as_dt)
```

```
              precision    recall  f1-score   support

           0       0.96      0.97      0.97      3102
           1       0.97      0.96      0.97      3036

    accuracy                           0.97      6138
   macro avg       0.97      0.97      0.97      6138
weighted avg       0.97      0.97      0.97      6138

ROC AUC DTC=  0.9667220306674515
Confusion Matrix DTC: 
[[3011   91]
 [ 113 2923]]
Accuracy DTC:  0.9667644183773216
```

b. K Nearest Neighbors

```
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train,y_train)
```

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

```
fpr_knn,tpr_knn,thres_knn=roc_curve(y_test,pred_knn)
roc_auc_knn=auc(fpr_knn,tpr_knn)

print(classification_report(y_test,pred_knn))

print('ROC AUC KNN= ',roc_auc_knn)

cm_knn=confusion_matrix(y_test,pred_knn)
print('Confusion Matrix KNN: ')
print(cm_knn)

as_knn=accuracy_score(y_test,pred_knn)
print('Accuracy KNN: ',as_knn)
```

```
              precision    recall  f1-score   support

           0       0.97      0.96      0.97      3102
           1       0.96      0.97      0.96      3036

    accuracy                           0.96      6138
   macro avg       0.96      0.96      0.96      6138
weighted avg       0.96      0.96      0.96      6138

ROC AUC KNN=  0.9649735093768397
Confusion Matrix KNN:
[[2993  109]
 [ 106 2930]]
Accuracy KNN:  0.9649723036819811
```

c. Logistic Regression

```python
lr=LogisticRegression()
lr.fit(X_train,y_train)
```

```
]: LogisticRegression()
```

```python
pred_lr=lr.predict(X_test)
pred_lr
```

```python
fpr_lr,tpr_lr,thres_lr=roc_curve(y_test,pred_lr)
roc_auc_lr=auc(fpr_lr,tpr_lr)

print(classification_report(y_test,pred_lr))

print('ROC AUC LR= ',roc_auc_lr)

cm_lr=confusion_matrix(y_test,pred_lr)
print('Confusion Matrix LR: ')
print(cm_lr)

as_lr=accuracy_score(y_test,pred_lr)
print('Accuracy LR: ',as_lr)
```

```
              precision    recall  f1-score   support

           0       0.93      0.92      0.93      3102
           1       0.92      0.93      0.93      3036

    accuracy                           0.93      6138
   macro avg       0.93      0.93      0.93      6138
weighted avg       0.93      0.93      0.93      6138

ROC AUC LR=  0.9257316457825246
Confusion Matrix LR: 
[[2865  237]
 [ 219 2817]]
Accuracy LR:  0.9257086999022482
```

d. Naïve Bayes Classifier

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

```
GaussianNB()
```

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

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

```
fpr_gnb,tpr_gnb,thres_gnb=roc_curve(y_test,pred_gnb)
roc_auc_gnb=auc(fpr_gnb,tpr_gnb)

print(classification_report(y_test,pred_gnb))

print('ROC AUC GNB= ',roc_auc_gnb)

cm_gnb=confusion_matrix(y_test,pred_gnb)
print('Confusion Matrix GNB: ')
print(cm_gnb)

as_gnb=accuracy_score(y_test,pred_gnb)
print('Accuracy GNB: ',as_gnb)
```

```
              precision    recall  f1-score   support

           0       0.94      0.91      0.92      3102
           1       0.91      0.94      0.92      3036

    accuracy                           0.92      6138
   macro avg       0.92      0.92      0.92      6138
weighted avg       0.92      0.92      0.92      6138

ROC AUC GNB=  0.9227637148543716
Confusion Matrix GNB:
[[2819  283]
 [ 192 2844]]
Accuracy GNB:  0.922613229064842
```

e. Random Forest Classifier

```
rf=RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=2)
rf.fit(X_train,y_train)
```

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

```
pred_rf=rf.predict(X_test)
pred_rf
```

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

```
fpr_rf,tpr_rf,thres_rf=roc_curve(y_test,pred_rf)
roc_auc_rf=auc(fpr_rf,tpr_rf)

print(classification_report(y_test,pred_rf))

print('ROC AUC RF= ',roc_auc_rf)

cm_rf=confusion_matrix(y_test,pred_rf)
print('Confusion Matrix RF: ')
print(cm_rf)

as_rf=accuracy_score(y_test,pred_rf)
print('Accuracy RF: ',as_rf)
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      3102
           1       0.97      0.97      0.97      3036

    accuracy                           0.97      6138
   macro avg       0.97      0.97      0.97      6138
weighted avg       0.97      0.97      0.97      6138

ROC AUC RF=  0.967724189162672
Confusion Matrix RF: 
[[3007   95]
 [ 103 2933]]
Accuracy RF:  0.967741935483871
```

f. Support Vector Machine

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

```
SVC()
```

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

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

```
fpr_svc,tpr_svc,thres_svc=roc_curve(y_test,pred_svc)
roc_auc_svc=auc(fpr_svc,tpr_svc)

print(classification_report(y_test,pred_svc))

print('ROC AUC SVC= ',roc_auc_svc)

cm_svc=confusion_matrix(y_test,pred_svc)
print('Confusion Matrix SVC: ')
print(cm_svc)

as_svc=accuracy_score(y_test,pred_svc)
print('Accuracy SVC: ',as_svc)
```

```
              precision    recall  f1-score   support

           0       0.97      0.96      0.96      3102
           1       0.96      0.97      0.96      3036

    accuracy                           0.96      6138
   macro avg       0.96      0.96      0.96      6138
weighted avg       0.96      0.96      0.96      6138

ROC AUC SVC=  0.963203963782132
Confusion Matrix SVC: 
[[2981  121]
 [ 105 2931]]
Accuracy SVC:  0.9631801889866406
```

g. XG Boost

```
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, ...)
```

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

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

```
fpr_xgb,tpr_xgb,thres_xgb=roc_curve(y_test,pred_xgb)
roc_auc_xgb=auc(fpr_xgb,tpr_xgb)

print(classification_report(y_test,pred_xgb))

print('ROC AUC XGB= ',roc_auc_xgb)

cm_xgb=confusion_matrix(y_test,pred_xgb)
print('Confusion Matrix XGB: ')
print(cm_xgb)

as_xgb=accuracy_score(y_test,pred_xgb)
print('Accuracy XGB: ',as_xgb)
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      3102
           1       0.97      0.97      0.97      3036

    accuracy                           0.97      6138
   macro avg       0.97      0.97      0.97      6138
weighted avg       0.97      0.97      0.97      6138

ROC AUC XGB=  0.9708638185742718
Confusion Matrix XGB: 
[[3004   98]
 [  81 2955]]
Accuracy XGB:  0.9708374063212772
```

6. Evaluating best overall model

```
algo=pd.DataFrame({'Model':['Desicion Tree Classifier','K Nearest Neighbors','Logistic Regression',
                            'Naive Bayes Classifier','Random Forest Classifier','Support Vector Machine','XG Boost'],
                   'ROC AUC':[roc_auc_dt, roc_auc_knn, roc_auc_lr, roc_auc_gnb, roc_auc_rf, roc_auc_svc, roc_auc_xgb],
                   'ACCURACY':[as_dt, as_knn, as_lr, as_gnb, as_rf, as_svc, as_xgb]
                  })
```

```
algo
```

	Model	ROC AUC	ACCURACY
0	Desicion Tree Classifier	0.966722	0.966764
1	K Nearest Neighbors	0.964974	0.964972
2	Logistic Regression	0.925732	0.925709
3	Naive Bayes Classifier	0.922764	0.922613
4	Random Forest Classifier	0.967724	0.967742
5	Support Vector Machine	0.963204	0.963180
6	XG Boost	0.970864	0.970837

```
Max1=0
mod1=''
Max2=0
mod2=''
for i in range(len(algo['Model'])):
    if algo.iloc[i:i+1,1:2].values>Max1:
        Max1=algo.iloc[i:i+1,1:2].values
        mod1=algo.iloc[i:i+1,0:1].values
    if algo.iloc[i:i+1,2:3].values>Max2:
        Max2=algo.iloc[i:i+1,2:3].values
        mod2=algo.iloc[i:i+1,0:1].values
print("Best ROC-AUC is ",Max1," by ",mod1)
print("Best ACCURACY is ",Max2," by ",mod2)
```

```
Best ROC-AUC is  [[0.97086382]]  by  [['XG Boost']]
Best ACCURACY is  [[0.97083741]]  by  [['XG Boost']]
```

7. Save the model

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

```
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'))
```

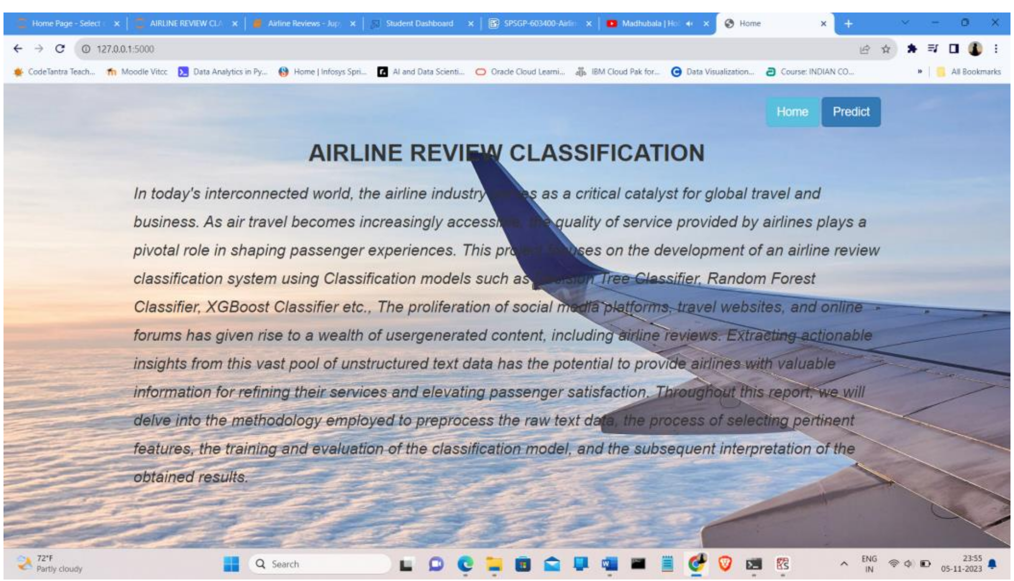
8. Flask Application Development

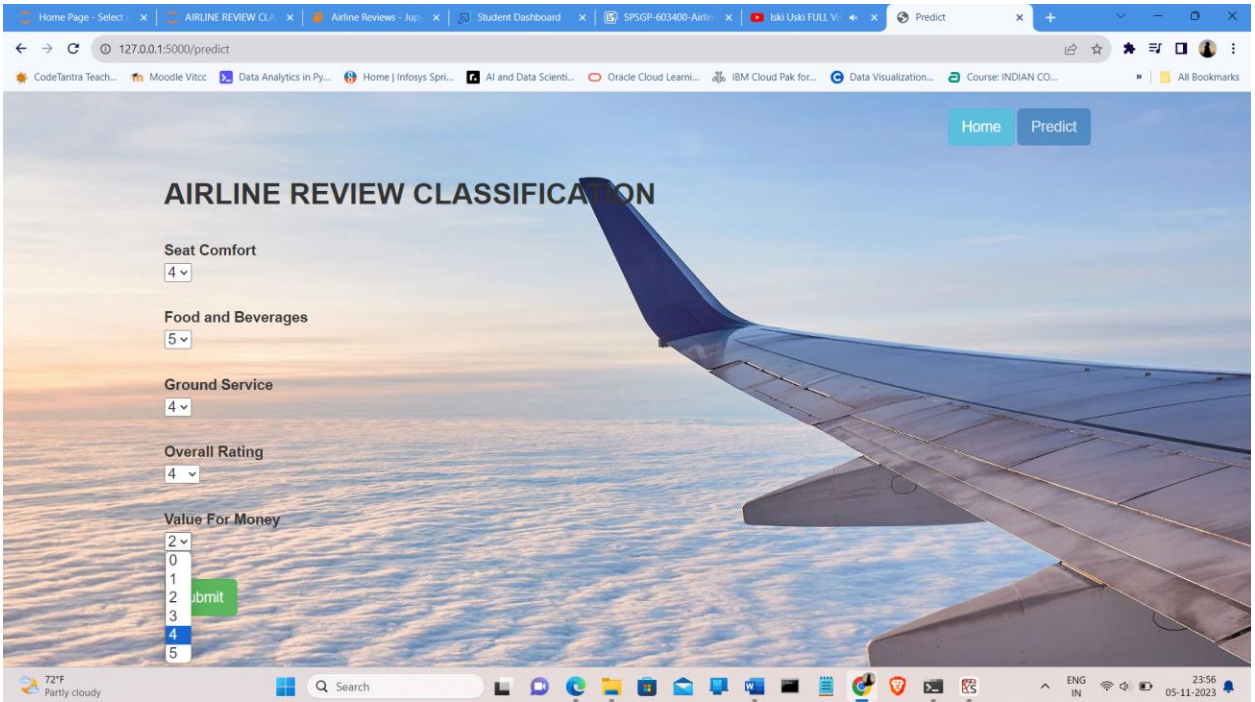
```python
from flask import Flask, render_template, request
import numpy as np
import pickle


app = Flask(__name__)

model = pickle.load(open('ar_xgb.pkl','rb'))
ss1 = pickle.load(open('ar_ss.pkl','rb'))
#le9 = pickle.load(open('le9.pkl','rb'))
#le10 = pickle.load(open('le10.pkl','rb'))


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


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


@app.route("/predict")
def home1():
    return render_template('predict.html')


@app.route("/submit")
def home2():
    return render_template('submit.html')
```

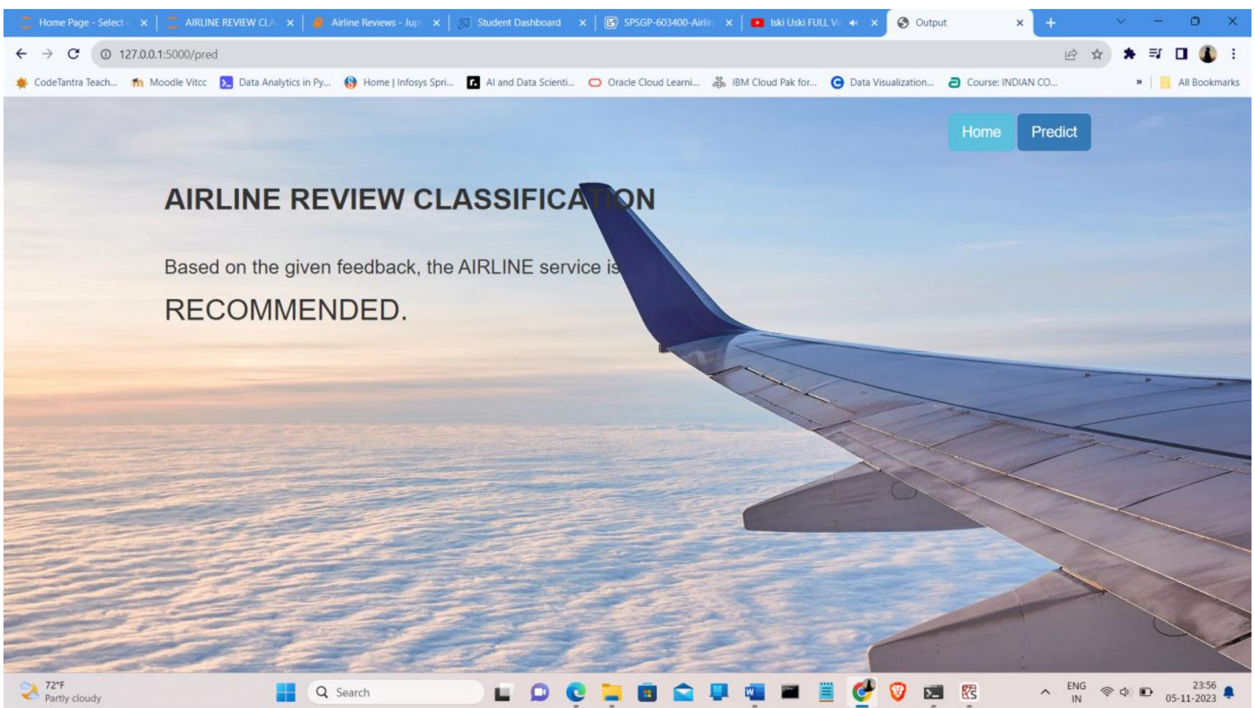
```python
@app.route("/pred", methods=['POST'])
def predict():
    seat = request.form['Seat']
    seat = int(seat)
    food = request.form['Food']
    food = int(food)
    ground = request.form['Ground']
    ground = int(ground)
    value = request.form['Value']
    value = int(value)
    over = request.form['Over']
    over = int(over)

    data = [seat,food,ground,over,value]
    print(data)
    pred = model.predict(ss1.transform([data]))

    if pred==0:
        text = 'NOT RECOMMENDED'
    else:
        text = 'RECOMMENDED'
    print(text)

    return render_template('submit.html', prediction=text)
```

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

Final Project Snapshots:




127.0.0.1:5000/predict
Home
Predict
AIRLINE REVIEW CLASSIFICATION
Seat Comfort
4
Food and Beverages
5
Ground Service
4
Overall Rating
4
Value For Money
2
0
1
2
3
4
5
ubmit


127.0.0.1:5000/pred
Home
Predict
AIRLINE REVIEW CLASSIFICATION
Based on the given feedback, the AIRLINE service is
RECOMMENDED.