BRITISH AIRWAYS

PRESENTATION

THE PROBLEM

Context

Predictive insights into booking completion behavior can enable personalized offers, dynamic pricing, and targeted interventions to minimize drop-offs during the booking process.

Problem statement

The objective of this project is to develop a predictive model to analyze and forecast customer booking behavior in the airline industry. Specifically, the model aims to **predict whether a customer will complete a booking** based on various features related to their travel intent, preferences, and contextual booking details.

Challenges deep-dive

Challenge 1

Imbalanced datasets.

Booking_complete was not proportional/balanced As 0 was in maximum number than 1’s

Challenge 2

Model selection:

- Logistic regression
- Random forest regressor

Challenge 3

How to increase :

- Recall score
- F1 score

SOLUTION

- Chose **`stratify`** ,as it helps ensure both training and test sets have the same class distribution.
- Used SMOTE function from python library (imblearn.over_sampling).Because it helps solve a common problem in machine learning called **class imbalance** — where one class is much more frequent than the other (e.g., 90% vs. 10%)
- After that by seeing cross validation score and means of models(logistic and random-forest classifier),i selected random forest as it gave more accuracy.
- since i was using **Random Forest** and dealing with **class imbalance**, here are the best techniques i can apply to improve **recall and F1-score.**
- And to improve more precision i can use scoring in cross validation as precision and set class_weight as “balance” in model.

Implementation

imported libraries and data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn import metrics
from sklearn.linear_model import  LogisticRegression
from sklearn.ensemble  import  RandomForestClassifier
```

```
data = pd.read_csv("customer_booking.csv",encoding ="latin-1")
```

data cleaning

```
# found unwanted data.so,droped those rows
data.isna().sum()
data.loc[data["booking_origin"]== "(not set)"] =np.nan
booking=data.dropna()
booking.reset_index(drop=True)
```

EDA

```
# found unwanted data.so,droped those rows
data.isna().sum()
data.loc[data["booking_origin"]== "(not set)"] =np.nan
booking=data.dropna()
booking.reset_index(drop=True)
```

```
# visulizations
sns.countplot(booking ,x="flight_day",hue="trip_type")
sns.countplot(booking,x="booking_complete",hue="num_passengers")
```

```
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
for columns in booking.select_dtypes(include =["object"]).columns:
    booking[columns]=label.fit_transform(booking[columns])
#DataFrame after encoding
booking.head()
```

training data and model

```
x = booking.drop("booking_complete",axis=1)
y = booking["booking_complete"]
from sklearn.model_selection import train_test_split
stat_col =y
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,  random_state =42,stratify=stat_col)
```

```
# If class imbalance is severe, try oversampling the minority class:
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)

X_train_res, y_train_res = smote.fit_resample(x_train, y_train)
```

```
model= LogisticRegression(max_iter=3000)
model1 = RandomForestClassifier(class_weight='balanced', random_state=42)
model.fit(X_train_res,y_train_res)
#quick prediction check on small sample
sample_x =X_train_res[:5]
sample_y =y_train_res[:5]
sample_ypred=model.predict(sample_x)
sample_accuracy = metrics.accuracy_score(sample_y,sample_ypred)
```

```
#cross validation
from sklearn.model_selection import cross_val_score
scores  =cross_val_score(model,X_train_res,y_train_res,cv=5) #default scoring = accuracy
scores1 =cross_val_score(model1,X_train_res,y_train_res,cv=5) #default scoring = accuracy
```

```
print("model : Linear regression ")
print("cross validation score : ",scores)
print("mean cross validation  : ",scores.mean())
print("model :  RandomForest Classifer")
print("cross validation score : ",scores1)
print("mean cross validation  : ",scores1.mean())
```

model : Linear regression

cross validation score : [0.6139124 0.66058152 0.66092462 0.65444641 0.66342756]

mean cross validation : 0.6506585027395889

model : RandomForest Classifer

cross validation score : [0.61008465 0.97313213 0.97305654 0.97232038 0.97283569]

mean cross validation : 0.9002858772558419

here random forest is performing better

so now choosing model = RandomForestClassifer

```
# saving the model
from joblib import dump,load
dump(model1,"BA.joblib")
```

```
final_pred=model1.predict(x_test)
print(f"accuracy:{metrics.accuracy_score(y_test,final_pred)}")
print(f"precision:{metrics.precision_score(y_test,final_pred)}")
print(f"recall:{metrics.recall_score(y_test,final_pred)}")
print(f"f1 score:{metrics.f1_score(y_test,final_pred)}")


print(confusion_matrix(y_test,final_pred))
print(classification_report(y_test,final_pred))
```

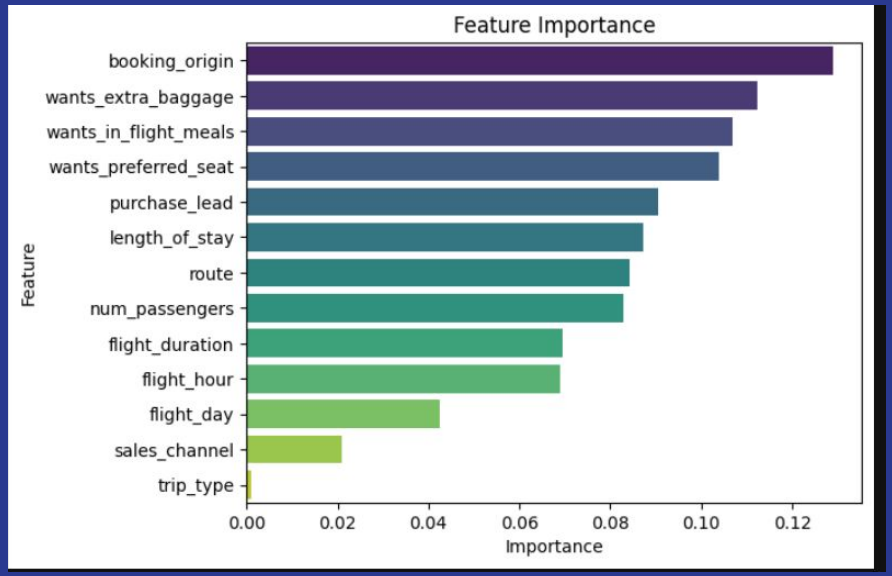
test the model

```
[ ]: # test the model
list(x.iloc[82])
#[1.0, 0.0, 2.0, 15.0, 31.0, 17.0, 1.0, 6.0, 50.0, 0.0, 0.0, 0.0, 8.83]
model=load("BA.joblib")
value= np.array([[1.0, 0.0, 2.0, 15.0, 31.0, 17.0, 1.0, 6.0, 50.0, 0.0, 0.0, 0.0, 8.83]])
predicted_value = model.predict(value)
print("Predicted Value for Given Input:",predicted_value)
```

```
[ ]:
```

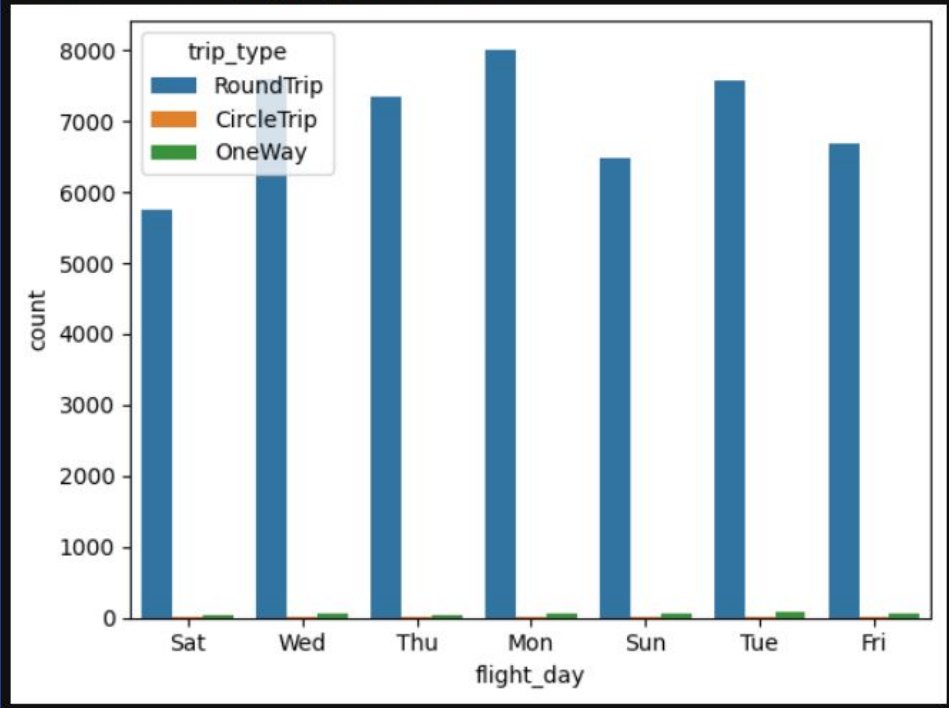
```
[ ]: feature_importance =model1.feature_importances_
feature  =x.columns
impotance_df =pd.DataFrame({"Feature":feature,"Importance": feature_importance})
impotance_df =impotance_df.sort_values(by="Importance",ascending=False)
```

```
[ ]: #model features importance
sns.barplot(impotance_df,x="Importance",y="Feature",palette ="viridis")
plt.figsize=(15,20)
plt.title("Feature Importance")
plt.show()
```



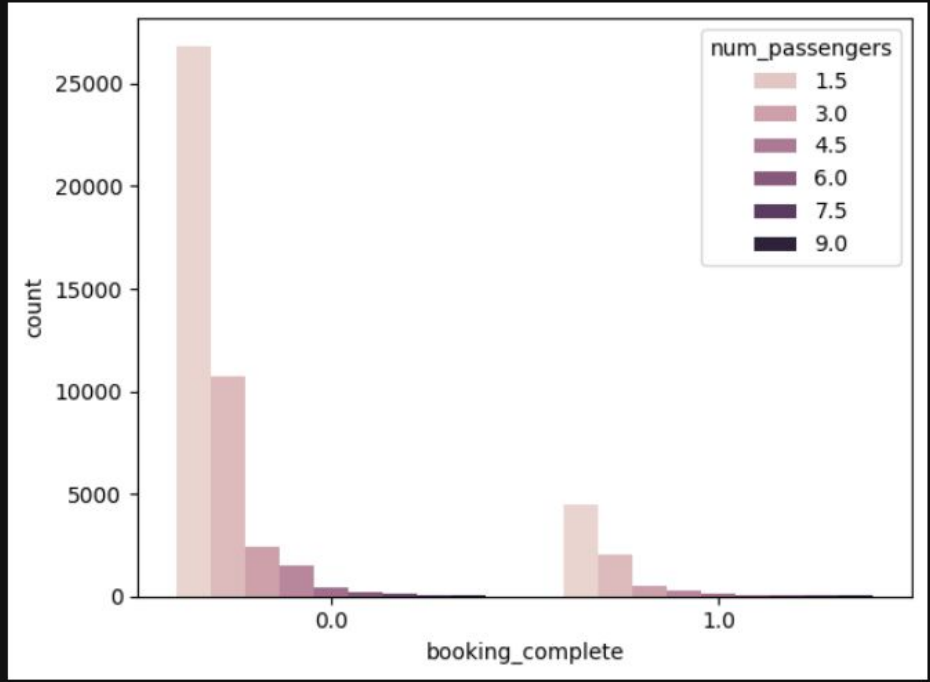

```
sns.countplot(booking ,x="flight_day",hue="trip_type")
```

```
<Axes: xlabel='flight_day', ylabel='count'>
```



```
sns.countplot(booking,x="booking_complete",hue="num_passengers")
```

```
<Axes: xlabel='booking_complete', ylabel='count'>
```



THANK YOU