Topic: Solving the problem of **passenger satisfaction** levels in the **airline industry** using machine learning by selecting the best predictive models and analyzing key factors.

```
**Index Of Contents**

  ∘ Problem Statement and Analysis

  ∘ Key Observations

  ∘ Data Collection and Initial Insights

  ∘ Loading the Dataset

  ∘ Exploratory Data Analysis and Visualization

  ∘ Data Preprocessing and Feature Engineering

  ∘ Numerical and categorical Features

  ∘ Train and Validation Split

  ∘ Imputation

  ∘ Scaling the Numeric Features

  ∘ Encode Categorical Feature

  ∘ Mapping

  ∘ Data Modelling

  ∘ Non Tree Models

  ∘ Logistic Regression

  ∘ Tree Based models

  ∘ Decision Tree Classifier

  ∘ Random Forest Classifier

  ∘ XG-boost classifier

  ∘ Hyperparameter Tuning

  ∘ Feature Importance

  ∘ Checking model performance on Test data

  ∘ Conclusion
```

Index Of Contents

- Problem Statement and Analysis
- Key Observations
- Data Collection and Initial Insights
- Loading the Dataset
- Exploratory Data Analysis and Visualization
- Data Preprocessing and Feature Engineering
- Numerical and categorical Features
- Train and Validation Split
- Imputation
- Scaling the Numeric Features
- Encode Categorical Feature
- Mapping
- Data Modelling
- Non Tree Models
- Logistic Regression
- Tree Based models
- Decision Tree Classifier
- Random Forest Classifier
- XG-boost classifier
- Hyperparameter Tuning
- Feature Importance
- Checking model performance on Test data
- Conclusion

Problem Statement

- Following the pandemic, the airline industry suffered a massive setback, with ICAO estimating a 371 billion dollar loss in 2020, and a 329 billion dollar loss with reduced seat capacity. As a result, in order to revitalise the industry, it is absolutely necessary to understand the customer pain points and improve their satisfaction with the services provided.
- This data set contains a survey on air passenger satisfaction survey. Need to predict Airline passenger satisfaction level: 1.Satisfaction 2.Neutral or dissatisfied.
- Select the best predictive models for predicting passengers satisfaction which will help in reducing customer churn.

Key Observations:

This is a binary classification problem,it is necessary to predict which of the two levels of satisfaction with the airline the passenger belongs to: **Satisfaction, Neutral or dissatisfied** Before diving into the data, thinking intuitively and being an avid traveller myself, from my experience, the main factors should be:

Delays in the flight

Staff efficiency to address customer needs

Services provided in the flight

Data Gathering and Initial Insights

Installing and Importing the required packages.

```
## Data Analysis packages
import numpy as np
import pandas as pd

## Data Visualization packages
import matplotlib.pyplot as plt

import seaborn as sns
import matplotlib
%matplotlib inline
from pylab import rcParams
```

```
# sklearn library
import sklearn
```

```
# sklearn preprocessing tools
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import StratifiedKFold,train_test_split
from sklearn.metrics import classification_report,confusion_matrix,roc_curve,auc,accuracy_score,roc_auc_score
from sklearn.preprocessing import StandardScaler, RobustScaler, QuantileTransformer, PowerTransformer,FunctionTransformer,On
```

```
# Error Metrics
from sklearn.metrics import r2_score #r2 square
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import confusion_matrix ,classification_report
from sklearn.metrics import accuracy_score, precision_score,recall_score,f1_score
```

```
! pip install opendatasets
```

```
Collecting opendatasets
  Downloading opendatasets-0.1.22-py3-none-any.whl (15 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from opendatasets) (4.66.1)
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (from opendatasets) (1.5.16)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from opendatasets) (8.1.7)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2023.11.1
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.31.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (8.
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle->opendatasets
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggl
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatase
Installing collected packages: opendatasets
Successfully installed opendatasets-0.1.22
```

```
import opendatasets as od
```

```
## General Tools
import opendatasets as od
import os
import re
import joblib
import json
import warnings
```

```
### Machine learning classification Models
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.linear_model import SGDClassifier #stacstic gradient descent clasifeier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
import lightgbm as lgb
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import AdaBoostClassifier
```

```
#crossvalidation
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import LeaveOneOut
from sklearn.metrics import confusion_matrix
```

```
#hyper parameter tunning
from sklearn.model_selection import GridSearchCV,cross_val_score,RandomizedSearchCV
```

```
### Initial settings
%matplotlib inline
sns.set_style("darkgrid")
matplotlib.rcParams["font.size"] = 10
matplotlib.rcParams["figure.figsize"] = (8,6)
matplotlib.rcParams["figure.facecolor"] = '#00000000'
sns.set(rc={
            "font.size":10,
            "axes.titlesize":10,
            "axes.labelsize":15},
             style="darkgrid",
            )

warnings.filterwarnings('ignore')
pd.set_option('display.max_colwidth', None)
```

Importing the Dataset

Train Dataset

```
train_df= pd.read_csv('train.csv')
train_df.head()
```

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	...	Inflight entertainment	bo serv
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	...	5	
1	1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	...	1	
2	2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	...	5	
3	3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	...	2	
4	4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	3	...	3	

5 rows × 25 columns

```
train_df.shape
```

```
(103904, 25)
```

```
test_df= pd.read_csv('test.csv')
test_df.head()
```

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	...	Inflight entertainment	O boa servi
0	0	19556	Female	Loyal Customer	52	Business travel	Eco	160	5	4	...	5	
1	1	90035	Female	Loyal Customer	36	Business travel	Business	2863	1	1	...	4	
2	2	12360	Male	disloyal Customer	20	Business travel	Eco	192	2	0	...	2	

```
test_df.shape
```

```
(25976, 25)
```

4	4	36875	Female	Loyal	49	Business	Eco	1182	2	3		2

Information about the dataset.

There is the following information about the passengers of some airline:

Gender: male or female

Customer type: regular or non-regular airline customer

Age: the actual age of the passenger

Type of travel: the purpose of the passenger's flight (personal or business travel)

Class: business, economy, economy plus

Flight distance

Inflight wifi service: satisfaction level with Wi-Fi service on board (0: not rated; 1-5)

Departure/Arrival time convenient: departure/arrival time satisfaction level (0: not rated; 1-5)

Ease of Online booking: online booking satisfaction rate (0: not rated; 1-5)

Gate location: level of satisfaction with the gate location (0: not rated; 1-5)

Food and drink: food and drink satisfaction level (0: not rated; 1-5)

Online boarding: satisfaction level with online boarding (0: not rated; 1-5) Seat comfort: seat satisfaction level (0: not rated; 1-5)

Inflight entertainment: satisfaction with inflight entertainment (0: not rated; 1-5)

On-board service: level of satisfaction with on-board service (0: not rated; 1-5)

Leg room service: level of satisfaction with leg room service (0: not rated; 1-5)

Baggage handling: level of satisfaction with baggage handling (0: not rated; 1-5)

Checkin service: level of satisfaction with checkin service (0: not rated; 1-5)

Inflight service: level of satisfaction with inflight service (0: not rated; 1-5)

Cleanliness: level of satisfaction with cleanliness (0: not rated; 1-5)

Departure delay in minutes:

Arrival delay in minutes:

Satisfaction: Airline satisfaction level(Satisfaction, neutral or dissatisfaction).

```
## Initial statistical description
train_df.describe()
```

Observations

- The average delay in flights are 15 minutes, with a deviation of 38 min.
- Median of the delays are 0, which means 50% of the flights from this data, were not delayed

From this we can also conclude that "Unnamed: 0" and 'id' columns are not relavant so we can drop them.

```
train_df.drop(['Unnamed: 0','id'], axis=1, inplace=True)
train_df.head(2)
```

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	Inflight entertainment
0	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3	1	...	5
1	Male	disloyal Customer	25	Business travel	Business	235	3	2	3	3	...	1

2 rows × 23 columns

```
## Shape of the test dataset
train_df.shape
```

```
(103904, 23)
```

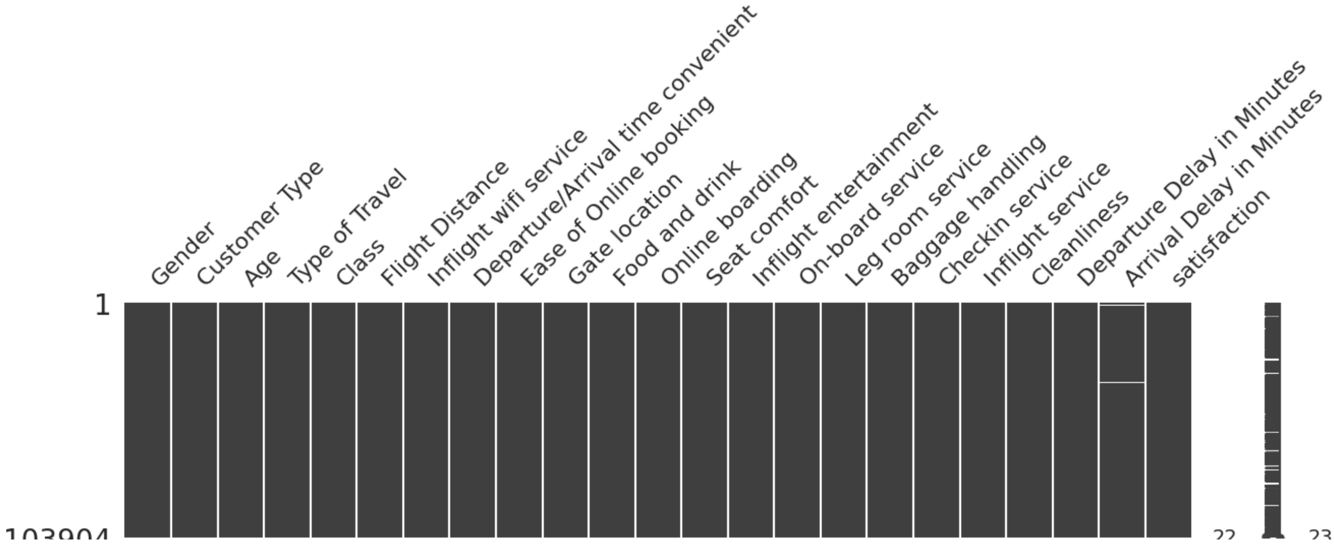
```
## General information about the features in train dataset
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103904 entries, 0 to 103903
Data columns (total 23 columns):
 #   Column                             Non-Null Count   Dtype  
---  ------                             --------------   -----  
 0   Gender                             103904 non-null  object 
 1   Customer Type                      103904 non-null  object 
 2   Age                                103904 non-null  int64  
 3   Type of Travel                     103904 non-null  object 
 4   Class                              103904 non-null  object 
 5   Flight Distance                    103904 non-null  int64  
 6   Inflight wifi service              103904 non-null  int64  
 7   Departure/Arrival time convenient  103904 non-null  int64  
 8   Ease of Online booking             103904 non-null  int64  
 9   Gate location                      103904 non-null  int64  
 10  Food and drink                     103904 non-null  int64  
 11  Online boarding                    103904 non-null  int64  
 12  Seat comfort                       103904 non-null  int64  
 13  Inflight entertainment             103904 non-null  int64  
 14  On-board service                   103904 non-null  int64  
 15  Leg room service                   103904 non-null  int64  
 16  Baggage handling                   103904 non-null  int64  
 17  Checkin service                    103904 non-null  int64  
 18  Inflight service                   103904 non-null  int64  
 19  Cleanliness                        103904 non-null  int64  
 20  Departure Delay in Minutes         103904 non-null  int64  
 21  Arrival Delay in Minutes           103594 non-null  float64
 22  satisfaction                       103904 non-null  object 
dtypes: float64(1), int64(17), object(5)
memory usage: 18.2+ MB
```

- only (arrival dealy in minutes)has null values. Let's visulize this to see any patterns in the missing values

```
import missingno as msno
```

```
##Visualize missing values (NaN) values using Missingno Library
msno.matrix(train_df, figsize=(15,3));
```



Observations:

There are 103904 rows for 23 features in our data. We see in the training data, that all the datatypes belong to a numeric class i.e. int, float and object. Only arrival dealy in minutes have some null values.

```
## Percentage of Null values
null_df = train_df.isnull().sum().sort_values(ascending=False).to_frame()
null_df.columns= ["No of Null values"]
null_df["% of Null values"] = round(null_df["No of Null values"]/len(train_df)*100,2)
null_df[null_df["No of Null values"] > 0]
```

	No of Null values	% of Null values
Arrival Delay in Minutes	310	0.3

```
round(train_df.describe().T,2)
```

	count	mean	std	min	25%	50%	75%	max
Age	103904.0	39.38	15.11	7.0	27.0	40.0	51.0	85.0
Flight Distance	103904.0	1189.45	997.15	31.0	414.0	843.0	1743.0	4983.0
Inflight wifi service	103904.0	2.73	1.33	0.0	2.0	3.0	4.0	5.0
Departure/Arrival time convenient	103904.0	3.06	1.53	0.0	2.0	3.0	4.0	5.0
Ease of Online booking	103904.0	2.76	1.40	0.0	2.0	3.0	4.0	5.0
Gate location	103904.0	2.98	1.28	0.0	2.0	3.0	4.0	5.0
Food and drink	103904.0	3.20	1.33	0.0	2.0	3.0	4.0	5.0
Online boarding	103904.0	3.25	1.35	0.0	2.0	3.0	4.0	5.0
Seat comfort	103904.0	3.44	1.32	0.0	2.0	4.0	5.0	5.0
Inflight entertainment	103904.0	3.36	1.33	0.0	2.0	4.0	4.0	5.0
On-board service	103904.0	3.38	1.29	0.0	2.0	4.0	4.0	5.0
Leg room service	103904.0	3.35	1.32	0.0	2.0	4.0	4.0	5.0
Baggage handling	103904.0	3.63	1.18	1.0	3.0	4.0	5.0	5.0
Checkin service	103904.0	3.30	1.27	0.0	3.0	3.0	4.0	5.0
Inflight service	103904.0	3.64	1.18	0.0	3.0	4.0	5.0	5.0
Cleanliness	103904.0	3.29	1.31	0.0	2.0	3.0	4.0	5.0
Departure Delay in Minutes	103904.0	14.82	38.23	0.0	0.0	0.0	12.0	1592.0
Arrival Delay in Minutes	103594.0	15.18	38.70	0.0	0.0	0.0	13.0	1584.0

```
### Checking for the duplicate values in the dataset
train_df.duplicated().sum()
```

0

Dependent Variable

The Satisfaction is our Target Varible.

```
train_df["satisfaction"].value_counts()
```

```
neutral or dissatisfied    58879
satisfied                  45025
Name: satisfaction, dtype: int64
```

```
round(train_df["satisfaction"].value_counts()[1]/(train_df["satisfaction"].value_counts()[0]+train_df["satisfaction"].value_
```

```
43.33
```

This problem is a binary classification problem of classes 0 or 1 denoting customers satisfaction, The class 1 has 43.33% total values. Hence, this is an balanced learning problem. Hence will not be requiring any resampling techniques to tackle this.

```
#Independent Variables or Features
train_df.columns[:-1]
```

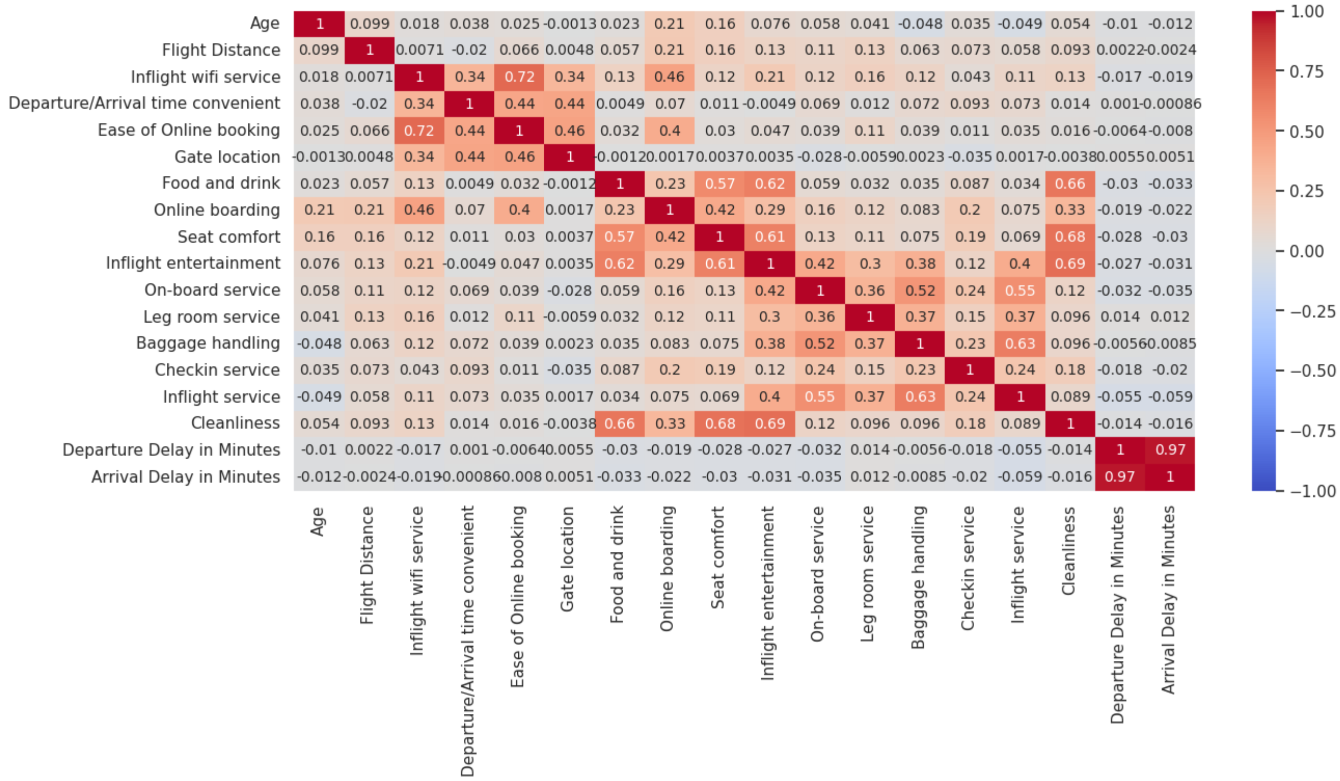
```
Index(['Gender', 'Customer Type', 'Age', 'Type of Travel', 'Class',
       'Flight Distance', 'Inflight wifi service',
       'Departure/Arrival time convenient', 'Ease of Online booking',
       'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort',
       'Inflight entertainment', 'On-board service', 'Leg room service',
       'Baggage handling', 'Checkin service', 'Inflight service',
       'Cleanliness', 'Departure Delay in Minutes',
       'Arrival Delay in Minutes'],
      dtype='object')
```

Exploratory Data Analysis and Visualization

```
train_df.corr()
```

	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Seat comfort	Infli entertainm
Age	1.000000	0.099461	0.017859	0.038125	0.024842	-0.001330	0.023000	0.208939	0.160277	0.076
Flight Distance	0.099461	1.000000	0.007131	-0.020043	0.065717	0.004793	0.056994	0.214869	0.157333	0.128
Inflight wifi service	0.017859	0.007131	1.000000	0.343845	0.715856	0.336248	0.134718	0.456970	0.122658	0.209
Departure/Arrival time convenient	0.038125	-0.020043	0.343845	1.000000	0.436961	0.444757	0.004906	0.070119	0.011344	-0.004
Ease of Online booking	0.024842	0.065717	0.715856	0.436961	1.000000	0.458655	0.031873	0.404074	0.030014	0.047
Gate location	-0.001330	0.004793	0.336248	0.444757	0.458655	1.000000	-0.001159	0.001688	0.003669	0.003
Food and drink	0.023000	0.056994	0.134718	0.004906	0.031873	-0.001159	1.000000	0.234468	0.574556	0.622
Online boarding	0.208939	0.214869	0.456970	0.070119	0.404074	0.001688	0.234468	1.000000	0.420211	0.285
Seat comfort	0.160277	0.157333	0.122658	0.011344	0.030014	0.003669	0.574556	0.420211	1.000000	0.610
Inflight entertainment	0.076444	0.128740	0.209321	-0.004861	0.047032	0.003517	0.622512	0.285066	0.610590	1.000
On-board service	0.057594	0.109526	0.121500	0.068882	0.038833	-0.028373	0.059073	0.155443	0.131971	0.420
Leg room service	0.040583	0.133916	0.160473	0.012441	0.107601	-0.005873	0.032498	0.123950	0.105559	0.299
Baggage handling	-0.047529	0.063184	0.120923	0.072126	0.038762	0.002313	0.034746	0.083280	0.074542	0.378
Checkin service	0.035482	0.073072	0.043193	0.093333	0.011081	-0.035427	0.087299	0.204462	0.191854	0.120
Inflight service	-0.049427	0.057540	0.110441	0.073318	0.035272	0.001681	0.033993	0.074573	0.069218	0.404
Cleanliness	0.053611	0.093149	0.132698	0.014292	0.016179	-0.003830	0.657760	0.331517	0.678534	0.691
Departure Delay in Minutes	-0.010152	0.002158	-0.017402	0.001005	-0.006371	0.005467	-0.029926	-0.018982	-0.027898	-0.027
Arrival Delay in Minutes	-0.012147	-0.002426	-0.019095	-0.000864	-0.007984	0.005143	-0.032524	-0.021949	-0.029900	-0.030

```
plt.figure(figsize=(14, 6))
sns.heatmap(train_df.corr(),annot=True, vmin=-1, vmax=1, cmap="coolwarm")
plt.show()
```

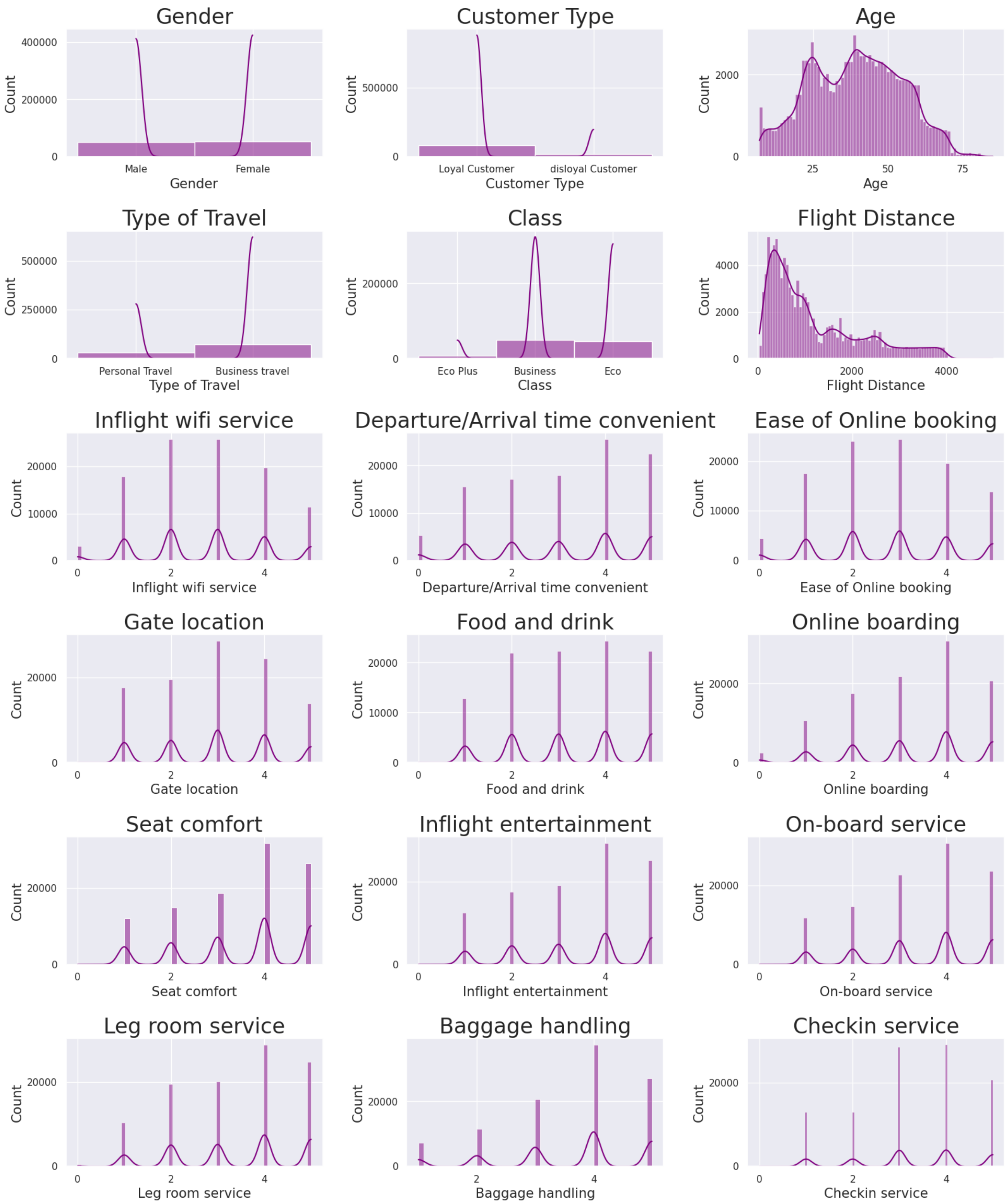


Observations:

Departure delay in minutes and arrival day in minutes are highly co related.

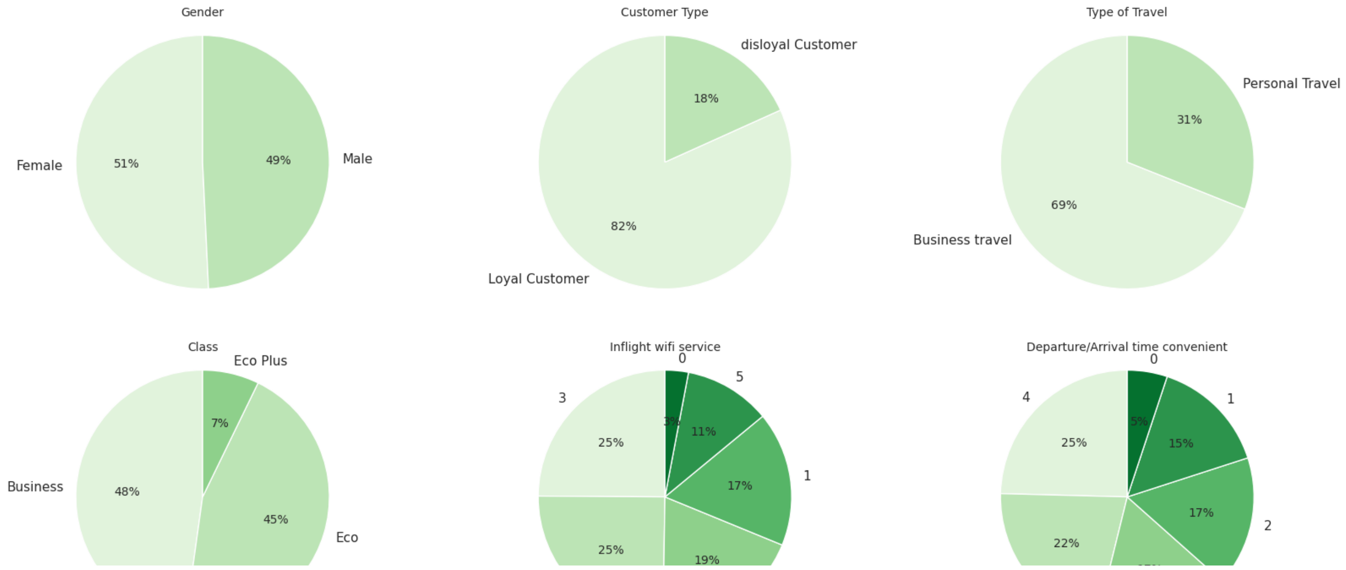
Data distribution graphs

```
sns.set(rc={
            "font.size":15,
            "axes.titlesize":10,
            "axes.labelsize":15},
             style="darkgrid")
fig, axs = plt.subplots(6, 3, figsize=(17,20))
fig.tight_layout(pad=4.0)

for f,ax in zip(train_df,axs.ravel()):
    sns.set(font_scale = 2)
    ax=sns.histplot(ax=ax,data=train_df,x=train_df[f],kde=True,color='purple')
    ax.set_title(f)
```




```
#Piechart percentage distribution features
new_train_df=train_df.copy()
new_train_df.drop(['Age','Flight Distance','Departure Delay in Minutes', 'Arrival Delay in Minutes','satisfaction'], axis=1,
sns.set(rc={
            "font.size":10,
            "axes.titlesize":10,
            "axes.labelsize":13},
             style="whitegrid")
fig, axes = plt.subplots(6, 3, figsize = (20, 30))
for i, col in enumerate(new_train_df):
    column_values = new_train_df[col].value_counts()
    labels = column_values.index
    sizes = column_values.values
    axes[i//3, i%3].pie(sizes,labels = labels, colors = sns.color_palette("Greens"),autopct = '%1.0f%%', startangle = 90)
    axes[i//3, i%3].axis('equal')
```

```
    axes[i//3, i%3].set_title(col)
plt.show()
```



Observations:

The number of men and women in this sample is approximately the same

The vast majority of the airline's customers are repeat customers

Most of our clients flew for business rather than personal reasons

About half of the passengers were in business class

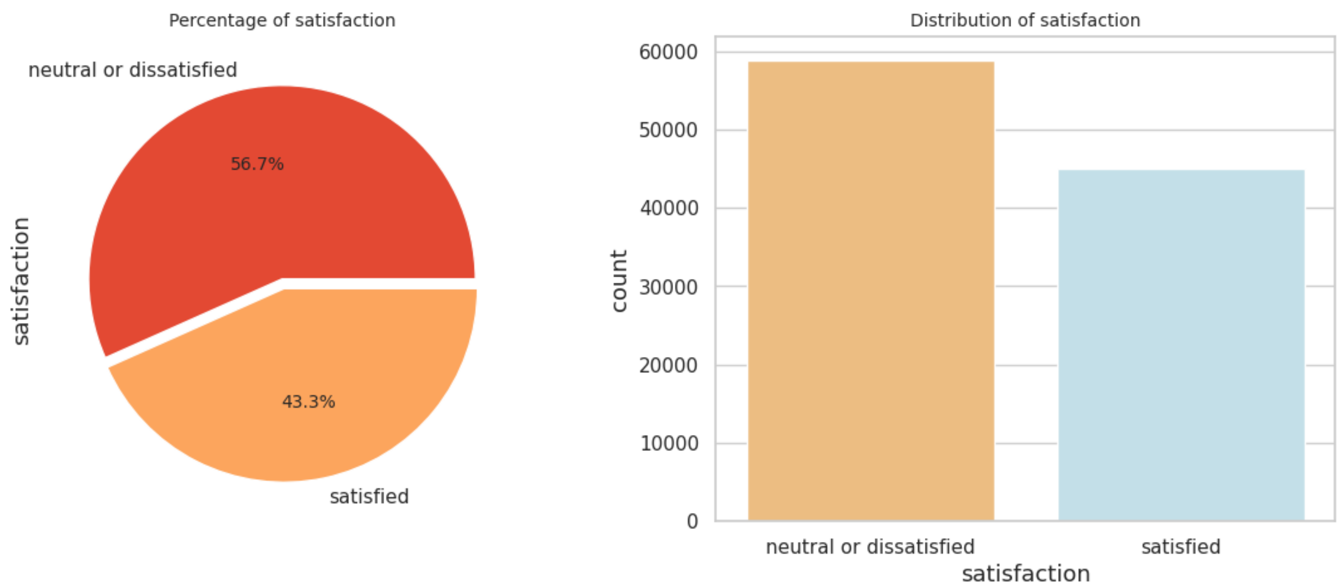
More than 60% of passengers were satisfied with the luggage transportation service (rated 4-5 out of 5)

More than 50% of passengers were comfortable sitting in their seats (rated 4-5 out of 5)

```
# satisfaction
train_df["satisfaction"].value_counts()
```

```
neutral or dissatisfied    58879
satisfied                  45025
Name: satisfaction, dtype: int64
```

```
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(14,5))
train_df['satisfaction'].value_counts().plot.pie(explode=[0,.05],colors = sns.color_palette("RdYlBu"),autopct='%1.1f%%',ax=a
ax1.set_title('Percentage of satisfaction')
sns.countplot(x='satisfaction',data=train_df,ax=ax2,palette='RdYlBu')
ax2.set_title('Distribution of satisfaction')
plt.show();
```



Observations:

As per the given data 56.7% people are dissatisfied or neutral And 43.3% people are satisfied.

To analyse and visualise the data lets divide data columns into categorical and numerical columns.

```
# numerical and categoriacl columns(features)
numeric_cols = train_df.select_dtypes(include=np.number).columns.tolist()
categorical_cols = train_df.select_dtypes('object').columns.tolist()
# numerical features
print("Total numeric columns are:", len(numeric_cols))
print(numeric_cols)
# categorical features
print("Total categorical columns are:", len(categorical_cols))
print(categorical_cols)
```

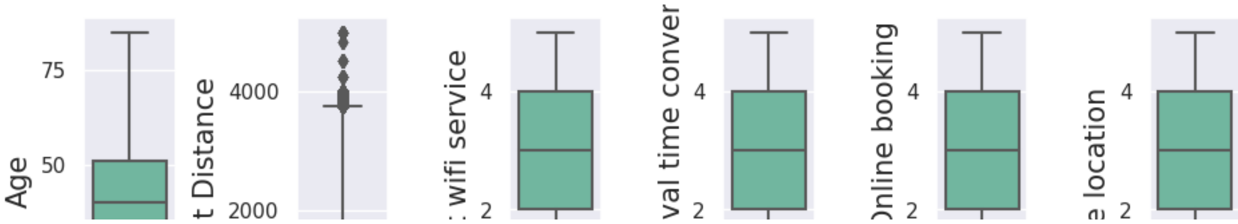
```
Total numeric columns are: 18
['Age', 'Flight Distance', 'Inflight wifi service', 'Departure/Arrival time convenient', 'Ease of Online booking', 'Gate
Total categorical columns are: 5
['Gender', 'Customer Type', 'Type of Travel', 'Class', 'satisfaction']
```

```
categorical_cols.remove("satisfaction")
```

Exploratory Data Analysis and Visualization on Numerical Columns

Boxplot: To check the Outliers in the numerical columns.

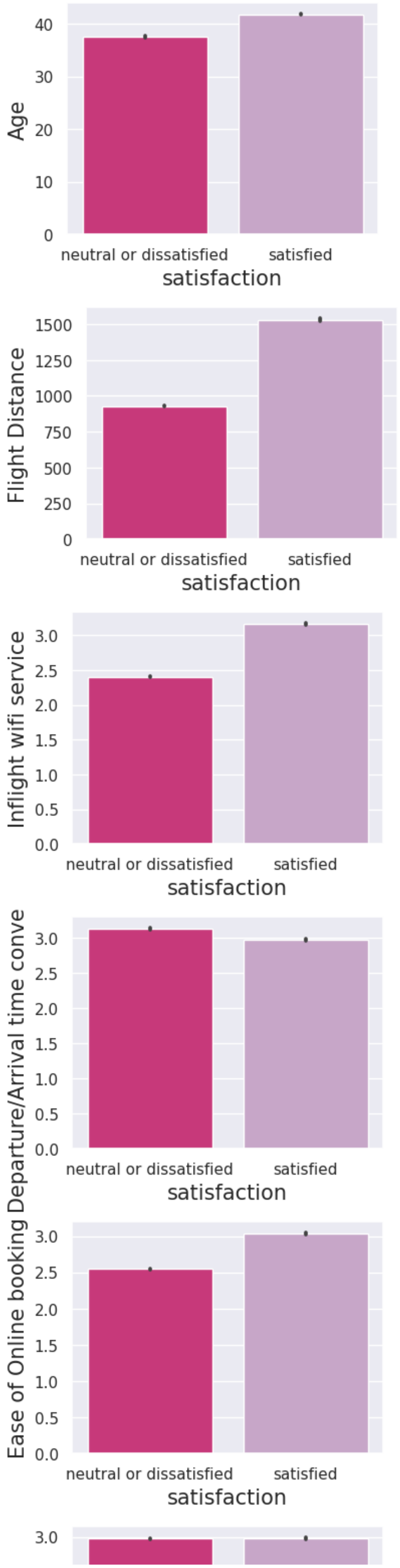
```
sns.set(rc={
            "font.size":10,
            "axes.titlesize":10,
            "axes.labelsize":15},
             style="darkgrid",
            )

fig, axs = plt.subplots(3, 6, figsize=(10,10))
fig.tight_layout(pad=3.0)

for f,ax in zip(numeric_cols,axs.ravel()):
    sns.set(font_scale = 2)
    ax=sns.boxplot(ax=ax,data=train_df,y=train_df[f],palette='BuGn')
```



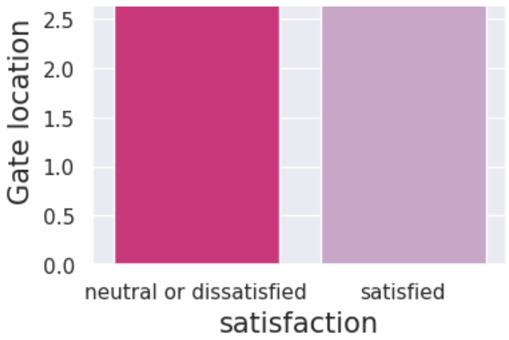
Observations:

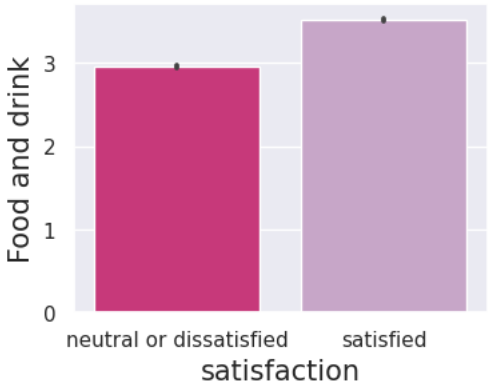
Flight distance, checkin service, Departure Delay in Minutes, Arrival Delay in Minutes has some outliers.

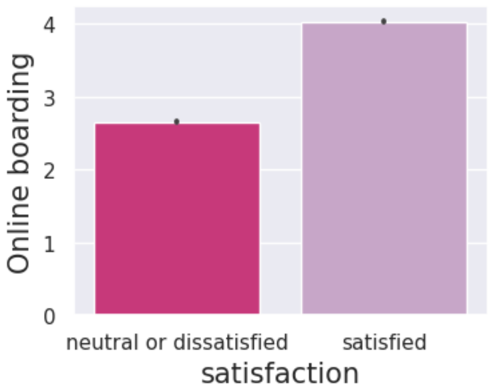
```
#Plotting the barplot of numerical features
sns.set(rc={'figure.figsize':(4,3),
            "font.size":10,
            "axes.titlesize":10,
            "axes.labelsize":15},
             style="darkgrid")

for i in numeric_cols:
    sns.barplot(data=train_df,x='satisfaction',y=i,palette='PuRd_r')
    plt.show()
```

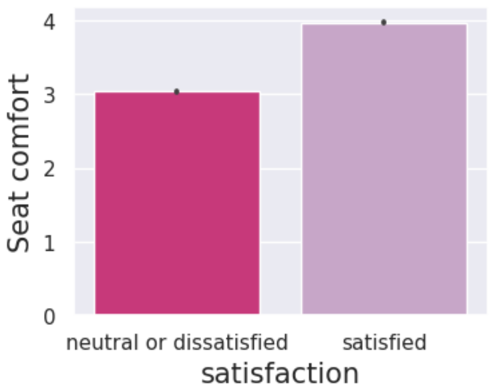


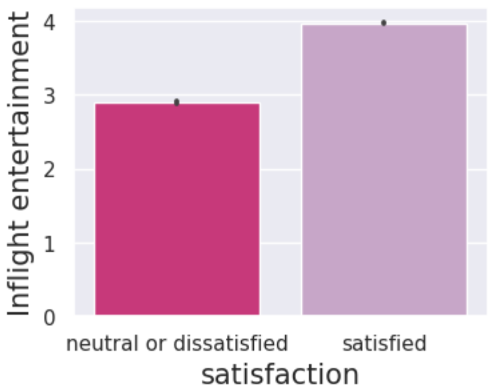
Age
0
10
20
30
40
neutral or dissatisfied
satisfied
satisfaction
Flight Distance
0
250
500
750
1000
1250
1500
neutral or dissatisfied
satisfied
satisfaction
Inflight wifi service
0.0
0.5
1.0
1.5
2.0
2.5
3.0
neutral or dissatisfied
satisfied
satisfaction
Departure/Arrival time conve
0.0
0.5
1.0
1.5
2.0
2.5
3.0
neutral or dissatisfied
satisfied
satisfaction
Ease of Online booking
0.0
0.5
1.0
1.5
2.0
2.5
3.0
neutral or dissatisfied
satisfied
satisfaction
3.0

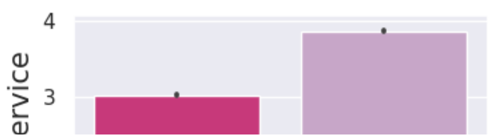

Gate location
2.5
2.0
1.5
1.0
0.5
0.0
neutral or dissatisfied
satisfied
satisfaction

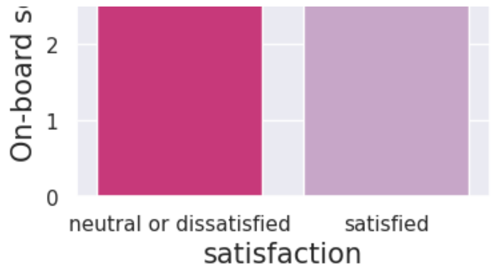

Food and drink
3
2
1
0
neutral or dissatisfied
satisfied
satisfaction

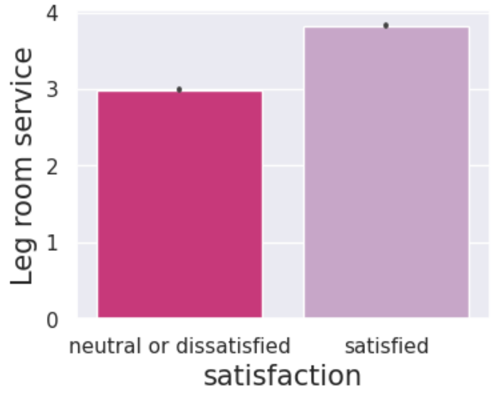

Online boarding
4
3
2
1
0
neutral or dissatisfied
satisfied
satisfaction

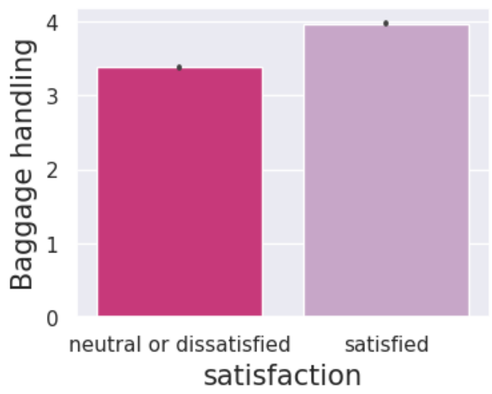

Seat comfort
4
3
2
1
0
neutral or dissatisfied
satisfied
satisfaction

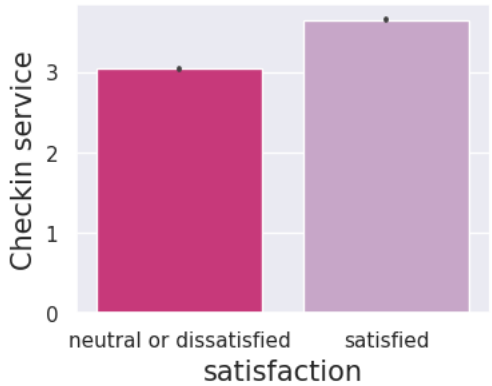

Inflight entertainment
4
3
2
1
0
neutral or dissatisfied
satisfied
satisfaction

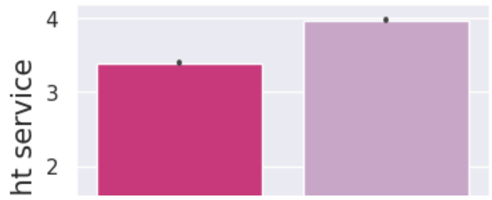

ervice
4
3











Observations:

From above graphs,it is clear that the age and Gate location, does not play a huge role in flight satisfaction and also the gender does not tell us much as seen in the earlier plot. Hence we can drop these values.

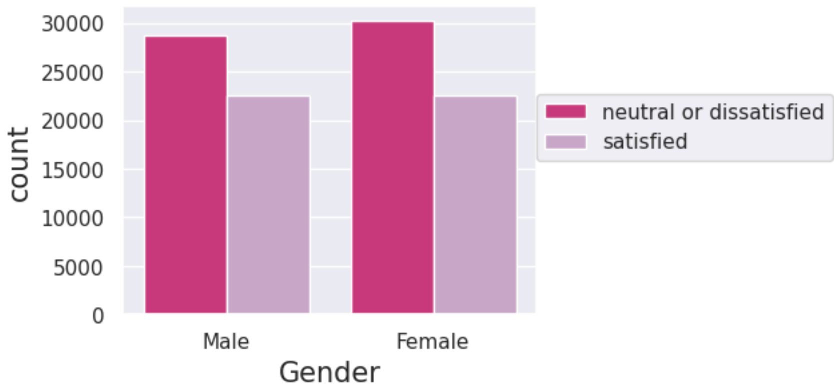
neutral or dissatisfied satisfied

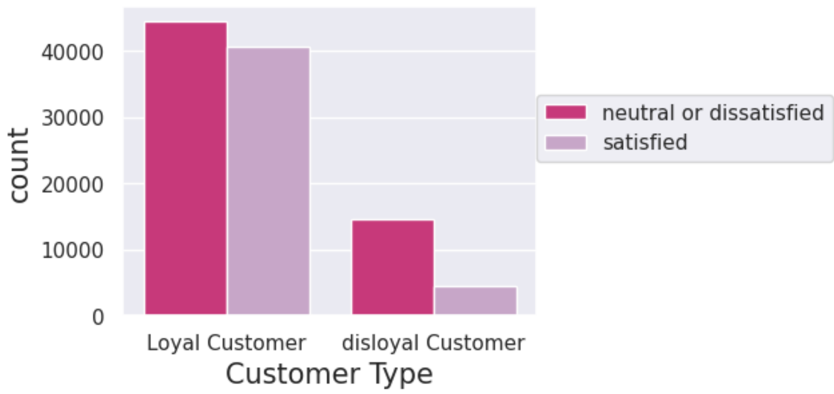
Exploratory Data Analysis and Visualization on categorical column.

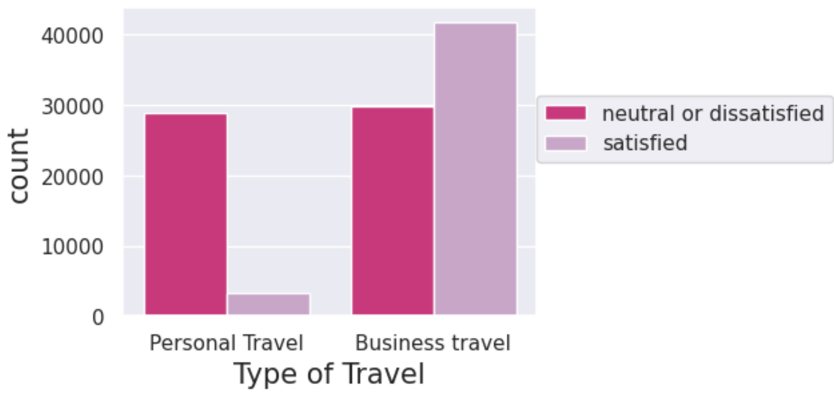
Barplot represntation on categorical features

```
sns.set(rc={'figure.figsize':(11.7,8.27),
            "font.size":10,
```

```
            "axes.titlesize":10,
            "axes.labelsize":15},
             style="darkgrid",
            )

for col in categorical_cols[:-1]:
    plt.figure(figsize=(4,3))
    sns.countplot(data=train_df,x=col,hue ='satisfaction',palette='PuRd_r')
    plt.legend(loc=(1,0.5))
```







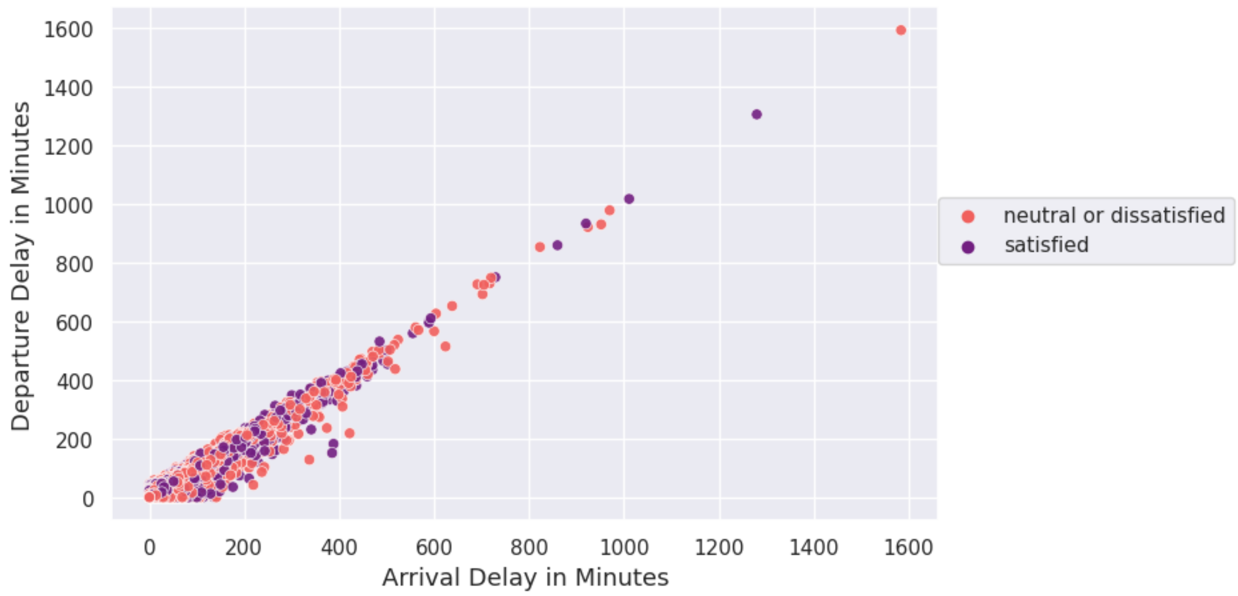
Observations:

- Gender doesn't play an important role in the satisfaction, as men and women seems to equally concerned about the same factors
- Number of loyal customers for this airline is high, however, the dissatisfaction level is high irrespective of the loyalty. Airline will have to work on maintaining the loyal customers
- Business Travellers seems to be more satisfied with the flight than the personal travellers

Arrival Delay in Minutes VS Departure Delay in minutes.

```
sns.set(rc={
            "font.size":10,
            "axes.titlesize":10,
            "axes.labelsize":13},
             style="darkgrid")
plt.figure(figsize=(8,5), dpi=100)
sns.scatterplot(data=train_df,x='Arrival Delay in Minutes',y='Departure Delay in Minutes',hue='satisfaction',palette='magma_
plt.legend(loc=(1,0.5))
```

<matplotlib.legend.Legend at 0x7aaf85c75d50>

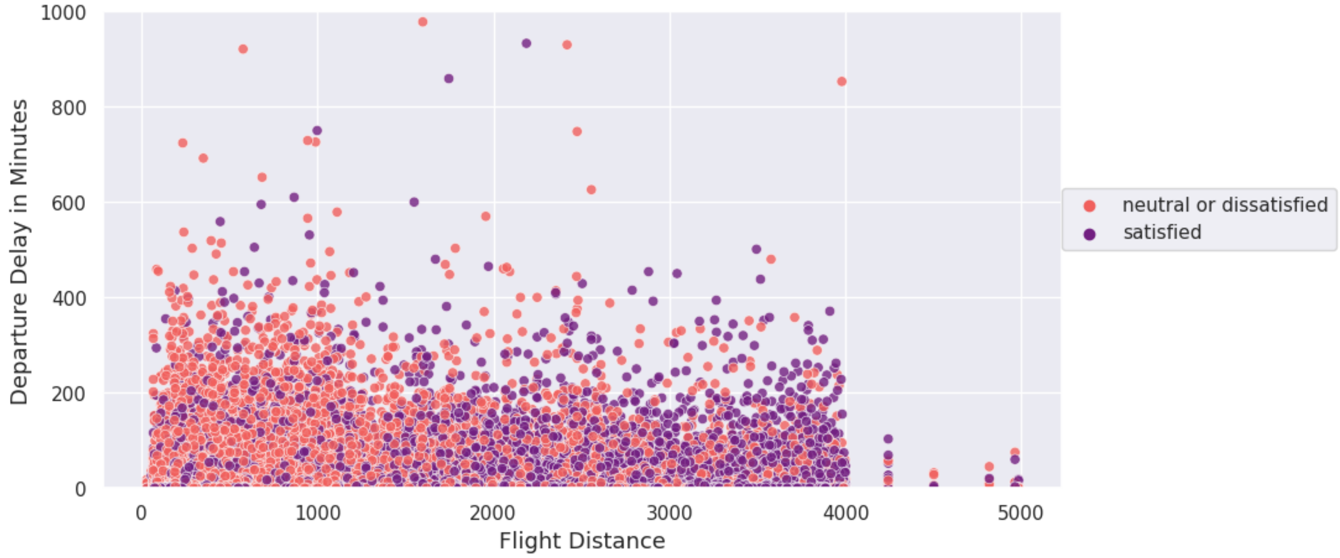


Observations:

- The arrival and departure delay seems to have a linear relationship, which makes complete sense and well, there is 1 customer who was satisfied even after a delay of 1300 minutes.

```
#Flight distance vs Departure Delay in minutes.
sns.set(rc={
            "font.size":10,
            "axes.titlesize":10,
            "axes.labelsize":13},
             style="darkgrid")
plt.figure(figsize=(10,5), dpi=100)
sns.scatterplot(data=train_df,x='Flight Distance',y='Departure Delay in Minutes',hue='satisfaction',palette='magma_r', alpha=
plt.ylim(0,1000)
plt.legend(loc=(1,0.5))
```

<matplotlib.legend.Legend at 0x7aaf85946da0>

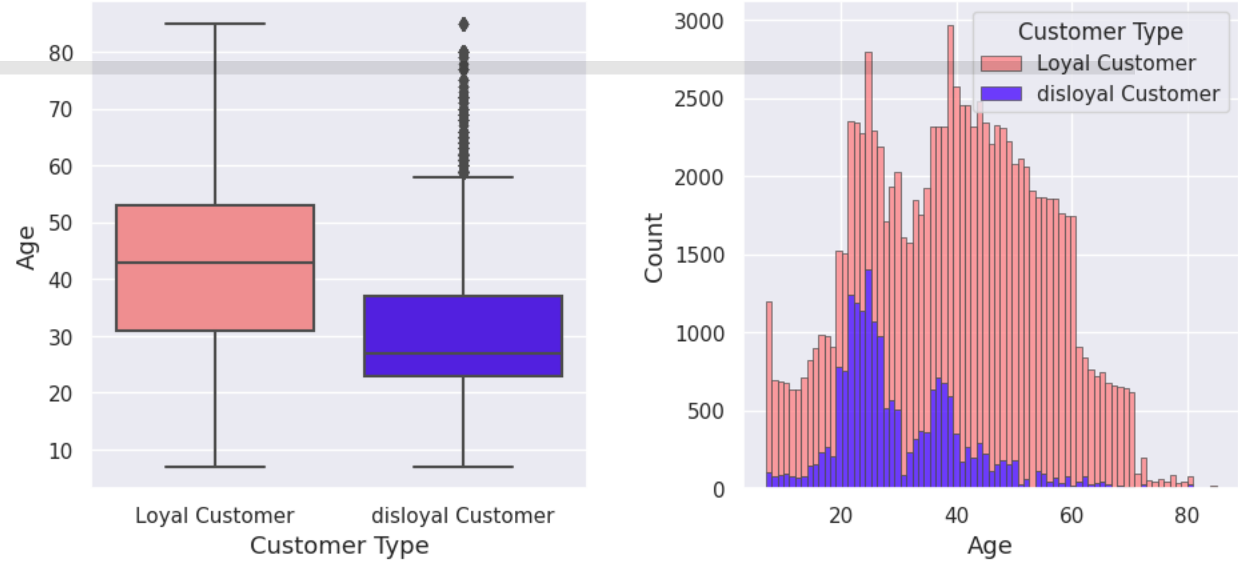


Observations:

- The most important takeaway here is the longer the flight distance, most passengers are okay with a slight delay in departure, which is a strange finding from this plot!
- So departure delay is less of a factor for a long distance flight, comparitively. However short distance travellers does not seem to be excited about the departure delays, which also makes sense.

```
# Age and Customer Type
f, ax = plt.subplots(1, 2, figsize = (10,5))
sns.boxplot(x = "Customer Type", y = "Age", palette = "gnuplot2_r", data = train_df, ax = ax[0])
```

```
sns.histplot(train_df, x = "Age", hue = "Customer Type", multiple = "stack", palette = "gnuplot2_r", edgecolor = ".3", linew:
plt.tight_layout(pad=3.0)
```

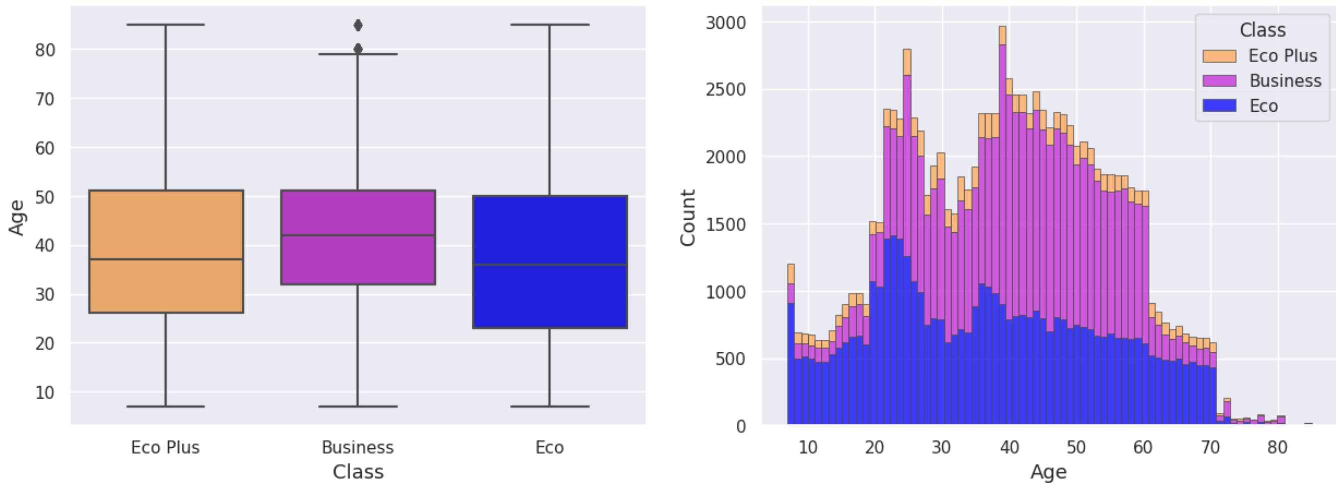


Observations:

- From above we can conclude that most of the airline's regular customers are between the ages of 30 and 50 (their median age is slightly over 40).
- The age range of non-regular customers is slightly smaller (from 25 to 40 years old, on average - a little less than 30).

```
# Age vs Class
f, ax = plt.subplots(1, 2, figsize = (15,5))
sns.boxplot(x = "Class", y = "Age", palette = "gnuplot2_r", data = train_df, ax = ax[0])
sns.histplot(train_df, x = "Age", hue = "Class", multiple = "stack", palette = "gnuplot2_r", edgecolor = ".3", linewidth = .!
```

```
<Axes: xlabel='Age', ylabel='Count'>
```

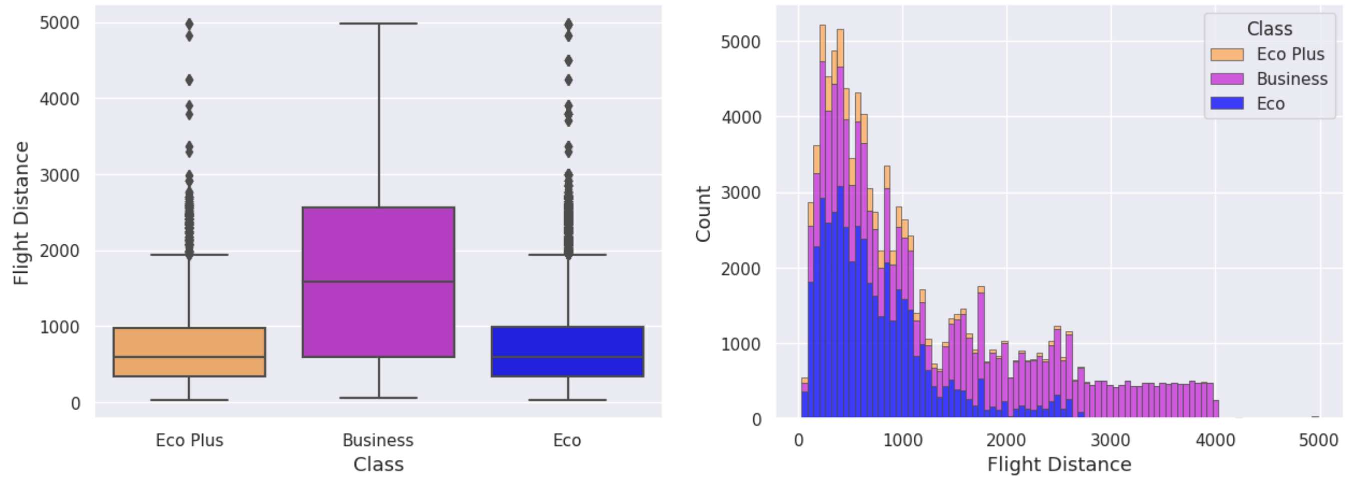


Observations:

- It can be seen that, on average, the age range of those customers who travel in business class is the same (according to the previous box chart) as the age range of regular customers. Based on this observation, it can be assumed that regular customers mainly buy business class for themselves.

```
f, ax = plt.subplots(1, 2, figsize = (15,5))
sns.boxplot(x = "Class", y = "Flight Distance", palette = "gnuplot2_r", data = train_df, ax = ax[0])
sns.histplot(train_df, x = "Flight Distance", hue = "Class", multiple = "stack", palette = "gnuplot2_r", edgecolor = ".3", l:
```

```
<Axes: xlabel='Flight Distance', ylabel='Count'>
```

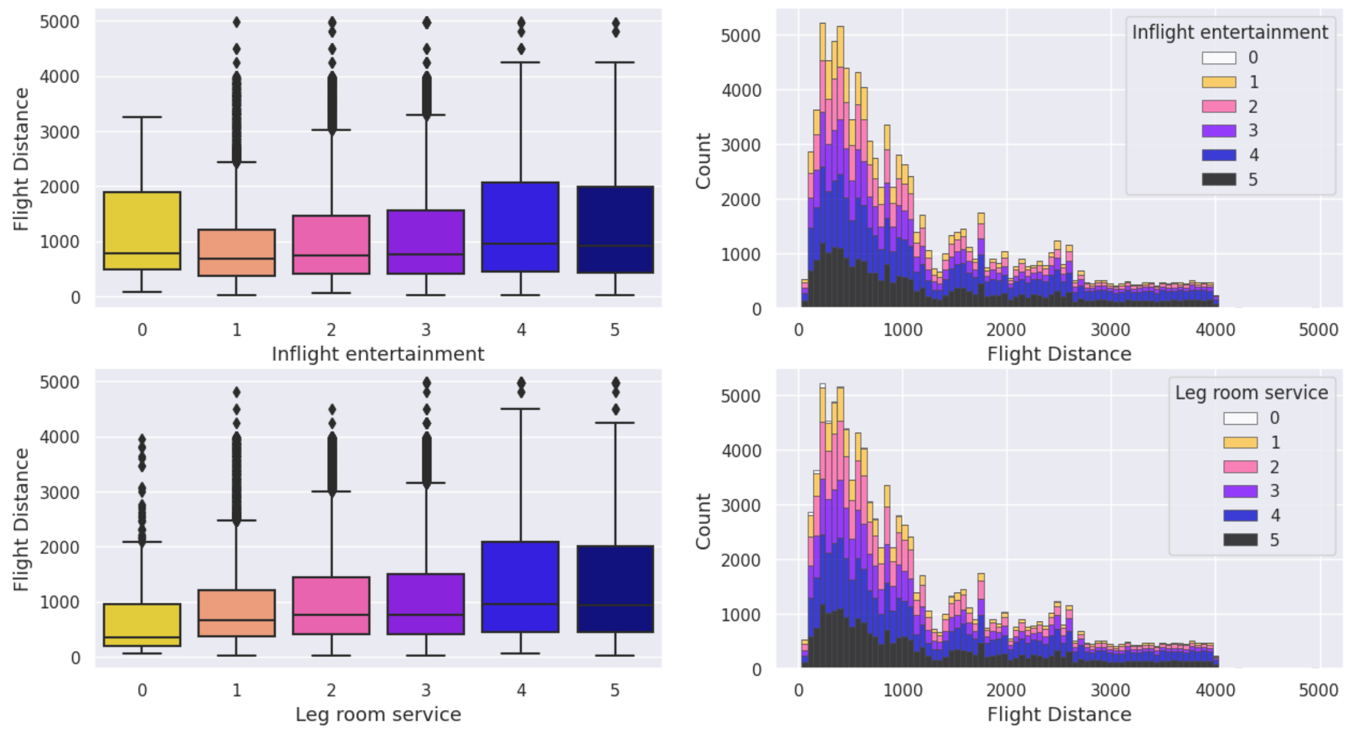


Observations:

customers whose flight distance is long, mostly fly in business class.

```
# Flight Distance
f, ax = plt.subplots(2, 2, figsize = (15,8))
sns.boxplot(x = "Inflight entertainment", y = "Flight Distance", palette = "gnuplot2_r", data = train_df, ax = ax[0, 0])
sns.histplot(train_df, x = "Flight Distance", hue = "Inflight entertainment", multiple = "stack", palette = "gnuplot2_r", ed
sns.boxplot(x = "Leg room service", y = "Flight Distance", palette = "gnuplot2_r", data = train_df, ax = ax[1, 0])
sns.histplot(train_df, x = "Flight Distance", hue = "Leg room service", multiple = "stack", palette = "gnuplot2_r", edgecolo
```

```
<Axes: xlabel='Flight Distance', ylabel='Count'>
```



Observations:

- The more distance an aircraft passenger travels (respectively, the longer they are in flight),The more they are satisfied with the entertainment in flight and the extra legroom (on average).

Data Preprocessing and Feature Engineering

```
# checking for the null_values
train_df.isnull().sum()
```

```
Gender                                  0
Customer Type                           0
Age                                     0
Type of Travel                          0
Class                                   0
Flight Distance                         0
Inflight wifi service                   0
Departure/Arrival time convenient       0
Ease of Online booking                  0
Gate location                           0
Food and drink                          0
Online boarding                         0
Seat comfort                            0
Inflight entertainment                  0
On-board service                        0
Leg room service                        0
Baggage handling                        0
Checkin service                         0
Inflight service                        0
Cleanliness                             0
Departure Delay in Minutes              0
Arrival Delay in Minutes              310
satisfaction                            0
dtype: int64
```

```
# step1: Independent features and dependent features
input_cols = list(train_df.iloc[:,:-1])# independent features
target_col="satisfaction" # dependent feature
```

```
train_val_df, test_df = train_test_split(train_df, test_size=0.2, random_state=42)
train_df, val_df = train_test_split(train_val_df, test_size=0.25, random_state=42)
```

```
print(train_df.shape)
print(val_df.shape)
print(test_df.shape)
```

```
(62342, 23)
(20781, 23)
(20781, 23)
```

created copy of the datasets, so that there will not be further changes in the orignal dataset

```
# copy of training dataset
train_inputs = train_df[input_cols].copy()
train_targets = train_df[target_col].copy()

# copy of valdation dataset

val_inputs = val_df[input_cols].copy()
val_targets = val_df[target_col].copy()

# copy of test dataset

test_inputs = test_df[input_cols].copy()
test_targets = test_df[target_col].copy()
```

imputing the missing numerical values

The missing values are now filled with the imputer of each column.

```
# Impute missing numerical values
imputer = SimpleImputer(strategy = 'median').fit(train_df[numeric_cols])
train_inputs[numeric_cols] = imputer.transform(train_inputs[numeric_cols])
val_inputs[numeric_cols] = imputer.transform(val_inputs[numeric_cols])
test_inputs[numeric_cols] = imputer.transform(test_inputs[numeric_cols])
```

```
print(list(imputer.statistics_))
```

```
[40.0, 842.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 4.0, 4.0, 4.0, 4.0, 4.0, 3.0, 4.0, 3.0, 0.0, 0.0]
```

```
train_inputs[numeric_cols].isna().sum()
```

```
Age                                   0
Flight Distance                       0
Inflight wifi service                 0
```

```
Departure/Arrival time convenient    0
Ease of Online booking               0
Gate location                        0
Food and drink                       0
Online boarding                      0
Seat comfort                         0
Inflight entertainment               0
On-board service                     0
Leg room service                     0
Baggage handling                     0
Checkin service                      0
Inflight service                     0
Cleanliness                          0
Departure Delay in Minutes           0
Arrival Delay in Minutes             0
dtype: int64
```

```
## scaling the numeric features.
scaler = MinMaxScaler().fit(train_df[numeric_cols])
train_inputs[numeric_cols] = scaler.transform(train_inputs[numeric_cols])
val_inputs[numeric_cols] = scaler.transform(val_inputs[numeric_cols])
test_inputs[numeric_cols] = scaler.transform(test_inputs[numeric_cols])
```

```
train_inputs[numeric_cols].describe()
```

	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Se comfo
count	62342.000000	62342.000000	62342.000000	62342.000000	62342.000000	62342.000000	62342.000000	62342.000000	62342.000
mean	0.414377	0.233852	0.545815	0.611302	0.551355	0.494558	0.640380	0.649014	0.6093
std	0.194144	0.201567	0.265617	0.306021	0.280040	0.319194	0.265363	0.270075	0.3299
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	0.256410	0.077141	0.400000	0.400000	0.400000	0.250000	0.400000	0.400000	0.250
50%	0.423077	0.163772	0.600000	0.600000	0.600000	0.500000	0.600000	0.600000	0.750
75%	0.564103	0.344911	0.800000	0.800000	0.800000	0.750000	0.800000	0.800000	1.000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000

One-hot encode categorical feature

```
! pip install --upgrade scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Collecting scikit-learn
  Downloading scikit_learn-1.3.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (10.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10.8/10.8 MB 23.6 MB/s eta 0:00:00
Requirement already satisfied: numpy<2.0,>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0
Installing collected packages: scikit-learn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.2.2
    Uninstalling scikit-learn-1.2.2:
      Successfully uninstalled scikit-learn-1.2.2
Successfully installed scikit-learn-1.3.2
```

```
from sklearn.preprocessing import OneHotEncoder
```

```
# One-hot encode categorical features
encoder = OneHotEncoder(sparse=False, handle_unknown='ignore').fit(train_df[categorical_cols])
encoded_cols = encoder.get_feature_names_out(categorical_cols)
train_inputs[encoded_cols] = encoder.transform(train_inputs[categorical_cols])
val_inputs[encoded_cols] = encoder.transform(val_inputs[categorical_cols])
test_inputs[encoded_cols] = encoder.transform(test_inputs[categorical_cols])
```

```
encoded_cols = list(encoder.get_feature_names_out(categorical_cols))
print(encoded_cols)
```

```
['Gender_Female', 'Gender_Male', 'Customer Type_Loyal Customer', 'Customer Type_disloyal Customer', 'Type of Travel_Busir
```

We can verify that these new columns have been added to our training, test and validation sets.

```
pd.set_option('display.max_columns', None)
test_inputs.head(2)
```

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding
80638	Female	Loyal Customer	0.243590	Personal Travel	Eco	0.167609	0.4	0.8	0.4	1.0	1.0	0.4
43398	Male	Loyal Customer	0.192308	Business travel	Business	0.073102	0.6	1.0	1.0	1.0	0.6	0.6

```
train_df['satisfaction'] = train_df['satisfaction'].map({'neutral or dissatisfied':0 , 'satisfied':1})
val_df['satisfaction'] = val_df['satisfaction'].map({'neutral or dissatisfied':0 , 'satisfied':1})
test_df['satisfaction'] = test_df['satisfaction'].map({'neutral or dissatisfied':0 , 'satisfied':1})
```

```
# Select the columns to be used for training/prediction

# training dataset
X_train = train_inputs[numeric_cols + encoded_cols]
y_train = train_df["satisfaction"]

# validation dataset
X_val = val_inputs[numeric_cols + encoded_cols]
y_val=  val_df["satisfaction"]

# test dataset
X_test = test_inputs[numeric_cols + encoded_cols]
y_test= test_df["satisfaction"]
```

```
y_train.value_counts()
```

```
0    35308
1    27034
Name: satisfaction, dtype: int64
```

```
y_val.value_counts()
```

```
0    11858
1     8923
Name: satisfaction, dtype: int64
```

```
y_test.value_counts()
```

```
0    11713
1     9068
Name: satisfaction, dtype: int64
```

```
### Distribution after transformation

columnList = list(X_train.columns)
columnList

fig = plt.figure(figsize=[15,20])
for col,i in zip(columnList,range(1,13)):
    axes = fig.add_subplot(5,3,i)
    sns.histplot(X_train[col],ax=axes, kde_kws={'bw':1.5}, color='green')
plt.show();
```

Data Modelling

Helper function

```
def plot_roc_curve(y_true,y_prob_preds,ax):
    """
    To plot the ROC curve for the given predictions and model

    """
    fpr,tpr,threshold = roc_curve(y_true,y_prob_preds)
    roc_auc = auc(fpr,tpr)
    ax.plot(fpr,tpr,"b",label="AUC = %0.2f" % roc_auc)
    ax.set_title("Receiver Operating Characteristic")
    ax.legend(loc='lower right')
    ax.plot([0,1],[0,1],'r--')
    ax.set_xlim([0,1])
    ax.set_ylim([0,1])
    ax.set_xlabel("False Positive Rate")
    ax.set_ylabel("True Positive Rate");
    plt.show();
```

```
def plot_confustion_matrix(y_true,y_preds,axes,name=''):
    """
    To plot the Confusion Matrix for the given predictions

    """
    cm = confusion_matrix(y_true, y_preds)
```

```
    group_names = ['TN','FP','FN','TP']
    group_percentages = ["{0:.2%}".format(value) for value in cm.flatten()/np.sum(cm)]
    group_counts = ["{0:0.0f}".format(value) for value in cm.flatten()]
    labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in zip(group_names,group_counts,group_percentages)]
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(cm, annot=labels, fmt='', cmap='Blues',ax=axes)
    axes.set_ylim([2,0])
    axes.set_xlabel('Prediction')
    axes.set_ylabel('Actual')
    axes.set_title(f'{name} Confusion Matrix');
```

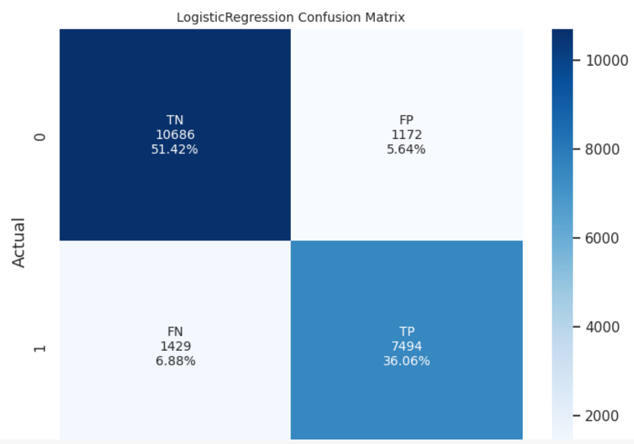
```
def make_classification_report(model,inputs,targets,model_name=None,record=False):
    """
     To Generate the classification report with all the metrics of a given model with confusion matrix as well as ROC AUC cu

    """
    ### Getting the model name from model object
    if model_name is None:
        model_name = str(type(model)).split(".")[-1][0:-2]

    ### Making the predictions for the given model
    preds = model.predict(inputs)
    if model_name in ["LinearSVC"]:
      prob_preds = model.decision_function(inputs)
    else:
      prob_preds = model.predict_proba(inputs)[:,1]

    ### printing the ROC AUC score
    auc_score = roc_auc_score(targets,prob_preds)
    print("ROC AUC Score : {:.2f}%\n".format(auc_score * 100.0))


    ### Plotting the Confusion Matrix and ROC AUC Curve
    fig, axes = plt.subplots(1, 2, figsize=(18,6))
    plot_confustion_matrix(targets,preds,axes[0],model_name)
    plot_roc_curve(targets,prob_preds,axes[1])
```

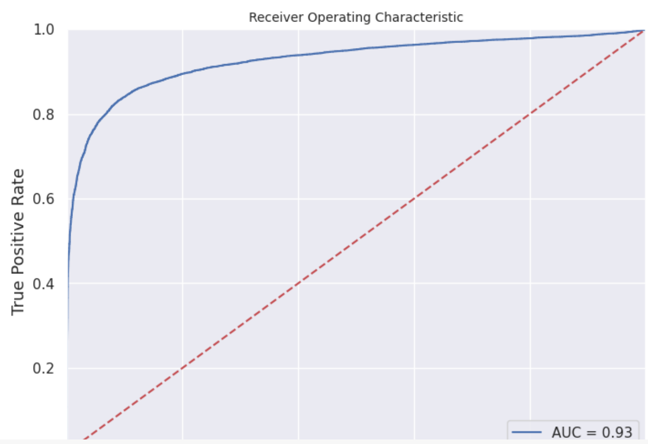
Non Tree Models

Logistic Regression

```
# import the model
from sklearn.linear_model import LogisticRegression

#fit the model
model =LogisticRegression()
model.fit(X_train,y_train)

# prediction
pred_train = model.predict(X_train)
pred_val = model.predict(X_val)

# model name
model_name = str(type(model)).split(".")[-1][0:-2]
print(f"\t\t{model_name.upper()} MODEL\n")

print('Training part:')
print(classification_report(y_train, pred_train,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
print('validation part:')
print(classification_report(y_val, pred_val,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
print("Accuracy score for traing dataset",accuracy_score(y_train, pred_train))
print("Accuracy score for validation dataset",accuracy_score(y_val, pred_val))
print

make_classification_report(model,X_val,y_val)
```

```
                LOGISTICREGRESSION MODEL

Training part:
                              precision    recall  f1-score   support

neutral or dissatisfaction       0.88      0.90      0.89     35308
              satisfaction       0.87      0.84      0.85     27034

                  accuracy                           0.87     62342
                 macro avg       0.87      0.87      0.87     62342
              weighted avg       0.87      0.87      0.87     62342

validation part:
                              precision    recall  f1-score   support

neutral or dissatisfaction       0.88      0.90      0.89     11858
              satisfaction       0.86      0.84      0.85      8923

                  accuracy                           0.87     20781
                 macro avg       0.87      0.87      0.87     20781
              weighted avg       0.87      0.87      0.87     20781

Accuracy score for traing dataset 0.8743383272913926
Accuracy score for validation dataset 0.8748375920311823
ROC AUC Score : 92.68%
```






```
pred_test = model.predict(X_test)

print("Accuracy score for test dataset",accuracy_score(y_test, pred_test))
```

```
Accuracy score for test dataset 0.8764737019392714
```

Observations

- The auc roc score is 92.68 %
- this model is working good with validation data & also with test data so good for generalization.

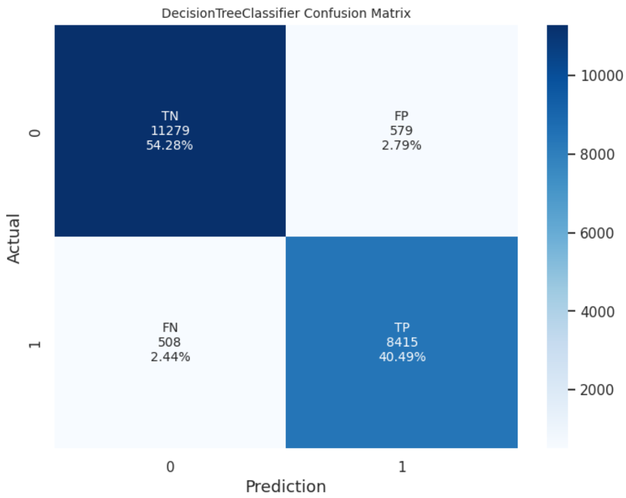
Tree Based models

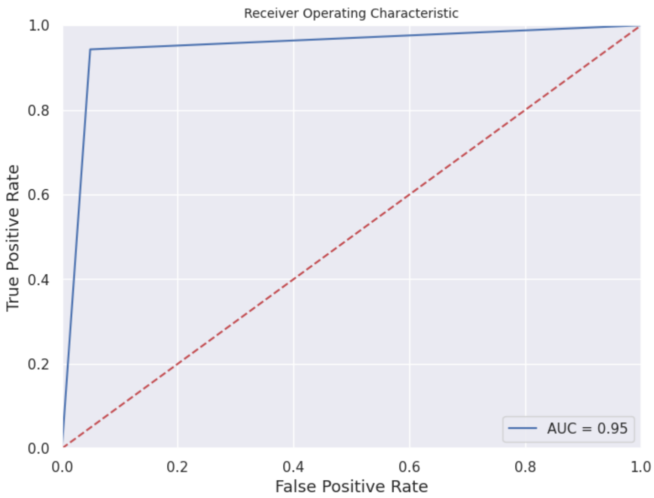
Decision Tree Classifier

```
# import the model
from sklearn.tree import DecisionTreeClassifier

#fit the model
model =DecisionTreeClassifier(random_state=42)
model.fit(X_train,y_train)

# prediction
pred_train = model.predict(X_train)
pred_val = model.predict(X_val)

# model name
model_name = str(type(model)).split(".")[-1][0:-2]
print(f"\t\t{model_name.upper()} MODEL\n")

print('Training part:')
print(classification_report(y_train, pred_train,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
```

```
print('validation part:')
print(classification_report(y_val, pred_val,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
print("Accuracy score for traing dataset",accuracy_score(y_train, pred_train))
print("Accuracy score for validation dataset",accuracy_score(y_val, pred_val))

make_classification_report(model,X_val,y_val)
```

```
                    DECISIONTREECLASSIFIER MODEL

Training part:
                            precision    recall  f1-score   support

neutral or dissatisfaction       1.00      1.00      1.00     35308
              satisfaction       1.00      1.00      1.00     27034

                  accuracy                           1.00     62342
                 macro avg       1.00      1.00      1.00     62342
              weighted avg       1.00      1.00      1.00     62342

validation part:
                            precision    recall  f1-score   support

neutral or dissatisfaction       0.96      0.95      0.95     11858
              satisfaction       0.94      0.94      0.94      8923

                  accuracy                           0.95     20781
                 macro avg       0.95      0.95      0.95     20781
              weighted avg       0.95      0.95      0.95     20781

Accuracy score for traing dataset 1.0
Accuracy score for validation dataset 0.9476926038207979
ROC AUC Score : 94.71%
```





Observations:

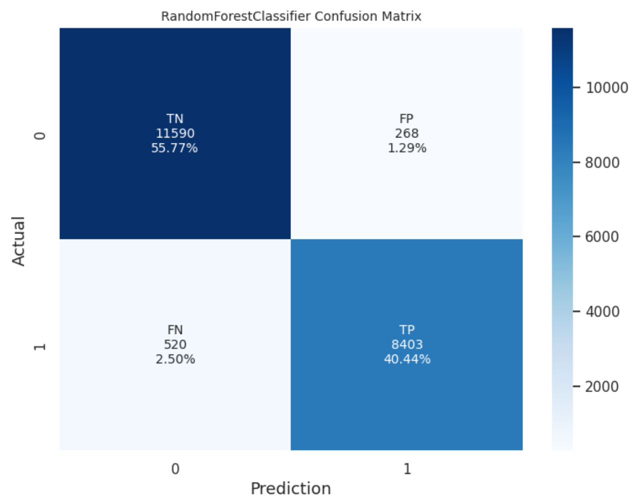
- The ROC AUC score is 94.71%.
- The Recall and F1 scores are good.
- But model will cause overfitting as the accuracy score for training dataset is 1.

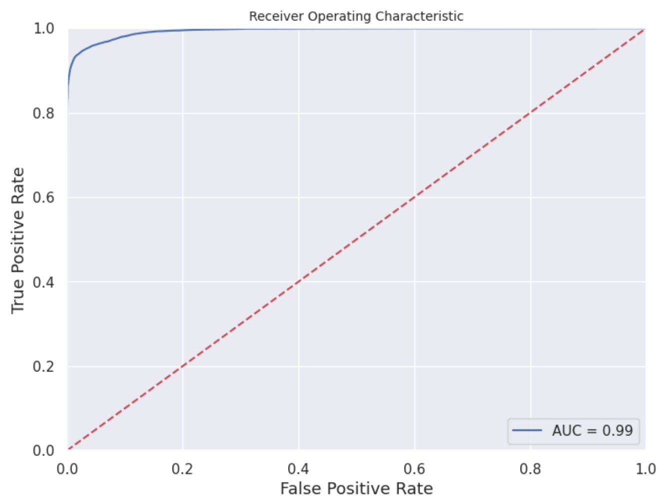
Random Forest Classifier

```
#import the model

from sklearn.ensemble import RandomForestClassifier

#fit the model
model =RandomForestClassifier(random_state=42)
model.fit(X_train,y_train)

# prediction
pred_train = model.predict(X_train)
pred_val = model.predict(X_val)
```

```
# model name
model_name = str(type(model)).split(".")[-1][0:-2]
print(f"\t\t{model_name.upper()} MODEL\n")

print('Training part:')
print(classification_report(y_train, pred_train,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
print('validation part:')
print(classification_report(y_val, pred_val,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
print("Accuracy score for traing dataset",accuracy_score(y_train, pred_train))
print("Accuracy score for validation dataset",accuracy_score(y_val, pred_val))

make_classification_report(model,X_val,y_val)
```

```
                    RANDOMFORESTCLASSIFIER MODEL

Training part:
                            precision    recall  f1-score   support

neutral or dissatisfaction       1.00      1.00      1.00     35308
              satisfaction       1.00      1.00      1.00     27034

                  accuracy                           1.00     62342
                 macro avg       1.00      1.00      1.00     62342
              weighted avg       1.00      1.00      1.00     62342

validation part:
                            precision    recall  f1-score   support

neutral or dissatisfaction       0.96      0.98      0.97     11858
              satisfaction       0.97      0.94      0.96      8923

                  accuracy                           0.96     20781
                 macro avg       0.96      0.96      0.96     20781
              weighted avg       0.96      0.96      0.96     20781

Accuracy score for traing dataset 1.0
Accuracy score for validation dataset 0.9620807468360522
ROC AUC Score : 99.37%
```





Observations:

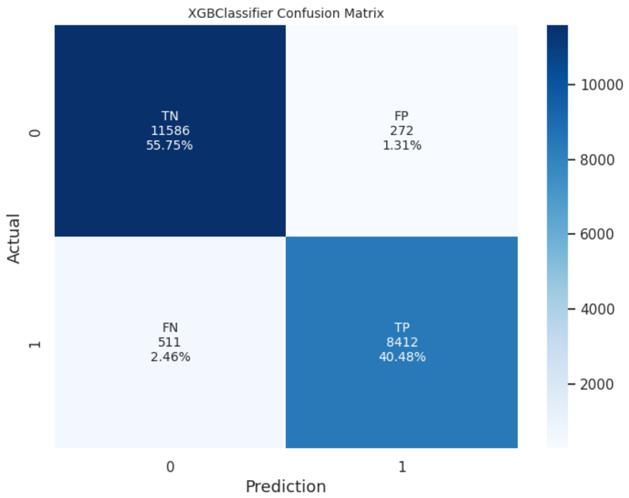
The ROC AUC score is 99.37%.The Recall and F1 scores are good. But model can cause overfitting, as the accuracy score for training dataset is 1

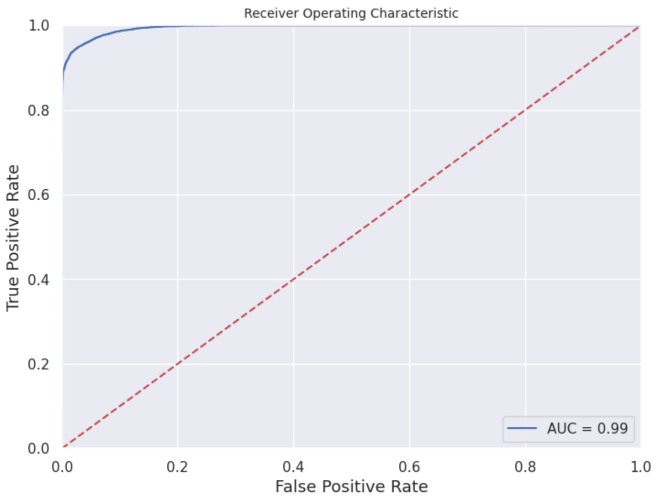
XG-boost model

```
#import the model

from xgboost import XGBClassifier
```

```
#fit the model
model =XGBClassifier()
model.fit(X_train,y_train)

# prediction
pred_train = model.predict(X_train)
pred_val = model.predict(X_val)

# model name
model_name = str(type(model)).split(".")[-1][0:-2]
print(f"\t\t{model_name.upper()} MODEL\n")

print('Training part:')
print(classification_report(y_train, pred_train,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
print('validation part:')
print(classification_report(y_val, pred_val,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
print("Accuracy score for traing dataset",accuracy_score(y_train, pred_train))
print("Accuracy score for validation dataset",accuracy_score(y_val, pred_val))

make_classification_report(model,X_val,y_val)
```

```
                    XGBCLASSIFIER MODEL

Training part:
                            precision    recall  f1-score   support

neutral or dissatisfaction       0.97      0.99      0.98     35308
              satisfaction       0.99      0.96      0.98     27034

                  accuracy                           0.98     62342
                 macro avg       0.98      0.98      0.98     62342
              weighted avg       0.98      0.98      0.98     62342

validation part:
                            precision    recall  f1-score   support

neutral or dissatisfaction       0.96      0.98      0.97     11858
              satisfaction       0.97      0.94      0.96      8923

                  accuracy                           0.96     20781
                 macro avg       0.96      0.96      0.96     20781
              weighted avg       0.96      0.96      0.96     20781

Accuracy score for traing dataset 0.9789387571781464
Accuracy score for validation dataset 0.9623213512343005
ROC AUC Score : 99.50%
```





Observation:

Model is provoding decent results with different parameters such as F-1 score & accuracy though it needs to tune the model for better generalisation

Hyperparameter Tuning

Hyperparameter Tuning of DT

```
dt_classifier = DecisionTreeClassifier()
```

```
# Define the hyperparameter grid
dt_rscv_params = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [2, 5, 7,],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

```
dt_classifier.get_params()
```

```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'best'}
```

```
# Create a RandomizedSearchCV object
dt_rscv = RandomizedSearchCV(estimator=dt_classifier,
                                   param_distributions=dt_rscv_params,
                                   n_iter=10,
                                   cv=5,
                                   verbose=2,
                                   random_state=42,
                                   )
```

```
!pip install scikit-learn==1.0.2
```

```
Collecting scikit-learn==1.0.2
  Downloading scikit_learn-1.0.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (26.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 26.5/26.5 MB 16.5 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.0.2) (1.23.
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.0.2) (1.11.4
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.0.2) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.0.2)
Installing collected packages: scikit-learn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.3.2
    Uninstalling scikit-learn-1.3.2:
      Successfully uninstalled scikit-learn-1.3.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour
bigframes 0.17.0 requires scikit-learn>=1.2.2, but you have scikit-learn 1.0.2 which is incompatible.
Successfully installed scikit-learn-1.0.2
```

```
# Fit the RandomizedSearchCV object to the data
dt_rscv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END criterion=entropy, max_depth=5, min_samples_leaf=1, min_samples_split=10, splitter=random; total time=   0.4s
[CV] END criterion=entropy, max_depth=5, min_samples_leaf=1, min_samples_split=10, splitter=random; total time=   0.5s
[CV] END criterion=entropy, max_depth=5, min_samples_leaf=1, min_samples_split=10, splitter=random; total time=   0.4s
[CV] END criterion=entropy, max_depth=5, min_samples_leaf=1, min_samples_split=10, splitter=random; total time=   0.5s
[CV] END criterion=entropy, max_depth=5, min_samples_leaf=1, min_samples_split=10, splitter=random; total time=   0.4s
[CV] END criterion=gini, max_depth=2, min_samples_leaf=2, min_samples_split=10, splitter=best; total time=   0.3s
[CV] END criterion=gini, max_depth=2, min_samples_leaf=2, min_samples_split=10, splitter=best; total time=   0.2s
[CV] END criterion=gini, max_depth=2, min_samples_leaf=2, min_samples_split=10, splitter=best; total time=   0.2s
[CV] END criterion=gini, max_depth=2, min_samples_leaf=2, min_samples_split=10, splitter=best; total time=   0.1s
[CV] END criterion=gini, max_depth=2, min_samples_leaf=2, min_samples_split=10, splitter=best; total time=   0.1s
[CV] END criterion=gini, max_depth=2, min_samples_leaf=1, min_samples_split=10, splitter=best; total time=   0.1s
[CV] END criterion=gini, max_depth=2, min_samples_leaf=1, min_samples_split=10, splitter=best; total time=   0.1s
[CV] END criterion=gini, max_depth=2, min_samples_leaf=1, min_samples_split=10, splitter=best; total time=   0.1s
[CV] END criterion=gini, max_depth=2, min_samples_leaf=1, min_samples_split=10, splitter=best; total time=   0.1s
[CV] END criterion=gini, max_depth=2, min_samples_leaf=1, min_samples_split=10, splitter=best; total time=   0.1s
[CV] END criterion=entropy, max_depth=5, min_samples_leaf=2, min_samples_split=10, splitter=random; total time=   0.1s
[CV] END criterion=entropy, max_depth=5, min_samples_leaf=2, min_samples_split=10, splitter=random; total time=   0.1s
[CV] END criterion=entropy, max_depth=5, min_samples_leaf=2, min_samples_split=10, splitter=random; total time=   0.1s
[CV] END criterion=entropy, max_depth=5, min_samples_leaf=2, min_samples_split=10, splitter=random; total time=   0.1s
[CV] END criterion=entropy, max_depth=5, min_samples_leaf=2, min_samples_split=10, splitter=random; total time=   0.1s
[CV] END criterion=entropy, max_depth=2, min_samples_leaf=2, min_samples_split=5, splitter=best; total time=   0.1s
[CV] END criterion=entropy, max_depth=2, min_samples_leaf=2, min_samples_split=5, splitter=best; total time=   0.1s
[CV] END criterion=entropy, max_depth=2, min_samples_leaf=2, min_samples_split=5, splitter=best; total time=   0.1s
[CV] END criterion=entropy, max_depth=2, min_samples_leaf=2, min_samples_split=5, splitter=best; total time=   0.1s
[CV] END criterion=entropy, max_depth=2, min_samples_leaf=2, min_samples_split=5, splitter=best; total time=   0.1s
[CV] END criterion=entropy, max_depth=2, min_samples_leaf=4, min_samples_split=2, splitter=random; total time=   0.0s
[CV] END criterion=entropy, max_depth=2, min_samples_leaf=4, min_samples_split=2, splitter=random; total time=   0.0s
[CV] END criterion=entropy, max_depth=2, min_samples_leaf=4, min_samples_split=2, splitter=random; total time=   0.0s
[CV] END criterion=entropy, max_depth=2, min_samples_leaf=4, min_samples_split=2, splitter=random; total time=   0.0s
[CV] END criterion=entropy, max_depth=2, min_samples_leaf=4, min_samples_split=2, splitter=random; total time=   0.0s
[CV] END criterion=gini, max_depth=5, min_samples_leaf=4, min_samples_split=2, splitter=best; total time=   0.2s
[CV] END criterion=gini, max_depth=5, min_samples_leaf=4, min_samples_split=2, splitter=best; total time=   0.2s
[CV] END criterion=gini, max_depth=5, min_samples_leaf=4, min_samples_split=2, splitter=best; total time=   0.2s
[CV] END criterion=gini, max_depth=5, min_samples_leaf=4, min_samples_split=2, splitter=best; total time=   0.2s
[CV] END criterion=gini, max_depth=5, min_samples_leaf=4, min_samples_split=2, splitter=best; total time=   0.2s
[CV] END criterion=gini, max_depth=7, min_samples_leaf=2, min_samples_split=5, splitter=random; total time=   0.1s
[CV] END criterion=gini, max_depth=7, min_samples_leaf=2, min_samples_split=5, splitter=random; total time=   0.1s
```

```python
# Print the best hyperparameters
best_rscv_params= dt_rscv.best_params_
print("Best Hyperparameters:", dt_rscv.best_params_)
```

```
Best Hyperparameters: {'splitter': 'random', 'min_samples_split': 5, 'min_samples_leaf': 2, 'max_depth': 7, 'criterion':
```

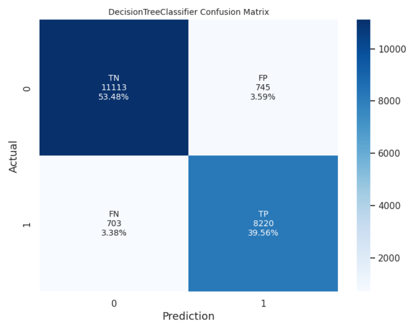
```python
# Get the best model
best_dt_model = dt_rscv.best_estimator_
```

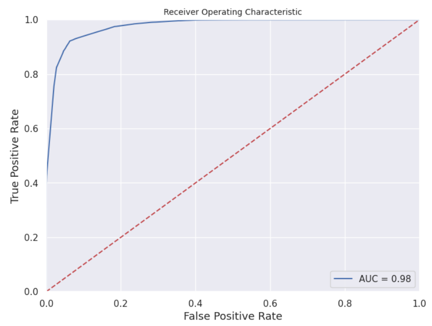
```python
# Evaluate the model on the test set
accuracy = best_dt_model.score(X_test, y_test)
print("Test Accuracy:", accuracy)
```

```
Test Accuracy: 0.9246908233482508
```

```python
# import the model
from sklearn.tree import DecisionTreeClassifier

#fit the model
model =DecisionTreeClassifier(**best_rscv_params )
model.fit(X_train,y_train)

# prediction
pred_train = model.predict(X_train)
pred_val = model.predict(X_val)

# model name
model_name = str(type(model)).split(".")[-1][0:-2]
print(f"\t\t{model_name.upper()} MODEL\n")

print('Training part:')
print(classification_report(y_train, pred_train,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
print('validation part:')
print(classification_report(y_val, pred_val,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
print("Accuracy score for traing dataset",accuracy_score(y_train, pred_train))
print("Accuracy score for validation dataset",accuracy_score(y_val, pred_val))

make_classification_report(model,X_val,y_val)
```

```
                    DECISIONTREECLASSIFIER MODEL

Training part:
                            precision    recall  f1-score   support

neutral or dissatisfaction       0.94      0.94      0.94     35308
              satisfaction       0.92      0.92      0.92     27034

                  accuracy                           0.93     62342
                 macro avg       0.93      0.93      0.93     62342
              weighted avg       0.93      0.93      0.93     62342

validation part:
                            precision    recall  f1-score   support

neutral or dissatisfaction       0.94      0.94      0.94     11858
              satisfaction       0.92      0.92      0.92      8923

                  accuracy                           0.93     20781
                 macro avg       0.93      0.93      0.93     20781
              weighted avg       0.93      0.93      0.93     20781

Accuracy score for traing dataset 0.929806550960829
Accuracy score for validation dataset 0.9303209662672633
ROC AUC Score : 97.70%
```





1. Overfitting is prevented after hyperparameter tuning
2. Roc_auc score is also imporved
3. Model is better suitable for generalisation.

Hyperparameter Tuning of Random Forest

```
# Create a Random Forest classifier
rf_classifier = RandomForestClassifier()
```

```
from scipy.stats import randint
```

```
# Define the parameter grid
rf_rscv_params = {
    'n_estimators': randint(10, 110),  # Number of trees in the forest
    'criterion': ['gini', 'entropy'],
    'max_depth': [5, 10],
    'min_samples_split': randint(2, 10),  # Minimum number of samples required to split an internal node
    'min_samples_leaf': randint(1, 5),  # Minimum number of samples required to be at a leaf node
    'max_features': ['auto', 'sqrt', 'log2'],  # Number of features to consider when looking for the best split
}
```

```
# Create RandomizedSearchCV object
rf_rscv = RandomizedSearchCV(
    rf_classifier,  # Random Forest classifier
    param_distributions=rf_rscv_params,  # Parameter grid
    n_iter=5,  # Number of random combinations to try
    cv=3,  # Number of cross-validation folds
```

```
    scoring='accuracy',  # Evaluation metric
    random_state=42  # Random seed for reproducibility
)
```

```
# Fit the RandomizedSearchCV object to the data
rf_rscv.fit(X_train, y_train)
```

```
▸ RandomizedSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

```
# Print the best parameters and corresponding accuracy
best_rscv_params_1= rf_rscv.best_params_
print("Best Parameters: ", rf_rscv.best_params_)
```

```
Best Parameters:  {'criterion': 'gini', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_spli
```

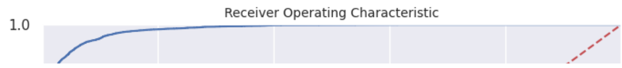
```
# Get the best model
best_rf_model = rf_rscv.best_estimator_
```

```
# Evaluate the model on the test set
accuracy = best_rf_model.score(X_test, y_test)
print("Test Accuracy:", accuracy)
```

```
Test Accuracy: 0.9470189115057023
```

```
# Random Forest with hyperparameter tuning
#import the model
from sklearn.ensemble import RandomForestClassifier
#fit the model
model =RandomForestClassifier(** best_rscv_params_1)
model.fit(X_train,y_train)

# prediction
pred_train = model.predict(X_train)
pred_val = model.predict(X_val)

# model name
model_name = str(type(model)).split(".")[-1][0:-2]
print(f"\t\t{model_name.upper()} MODEL\n")

print('Training part:')
print(classification_report(y_train, pred_train,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
print('validation part:')
print(classification_report(y_val, pred_val,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
print("Accuracy score for traing dataset",accuracy_score(y_train, pred_train))
print("Accuracy score for validation dataset",accuracy_score(y_val, pred_val))

make_classification_report(model,X_val,y_val)
```

```
                RANDOMFORESTCLASSIFIER MODEL

Training part:
                            precision    recall  f1-score   support

neutral or dissatisfaction       0.95      0.96      0.96     35308
              satisfaction       0.95      0.94      0.94     27034

                  accuracy                           0.95     62342
                 macro avg       0.95      0.95      0.95     62342
              weighted avg       0.95      0.95      0.95     62342

validation part:
                            precision    recall  f1-score   support

neutral or dissatisfaction       0.95      0.95      0.95     11858
              satisfaction       0.94      0.93      0.94      8923

                  accuracy                           0.94     20781
                 macro avg       0.94      0.94      0.94     20781
              weighted avg       0.94      0.94      0.94     20781

Accuracy score for traing dataset 0.9497289146963523
Accuracy score for validation dataset 0.9447572301621674
ROC AUC Score : 98.94%
```



Conclusion

1. After Hyperparameter tuning overfitting is prevented
2. Roc_auc score, f1_score have aslo improved which implies that model can be good for generalisation.

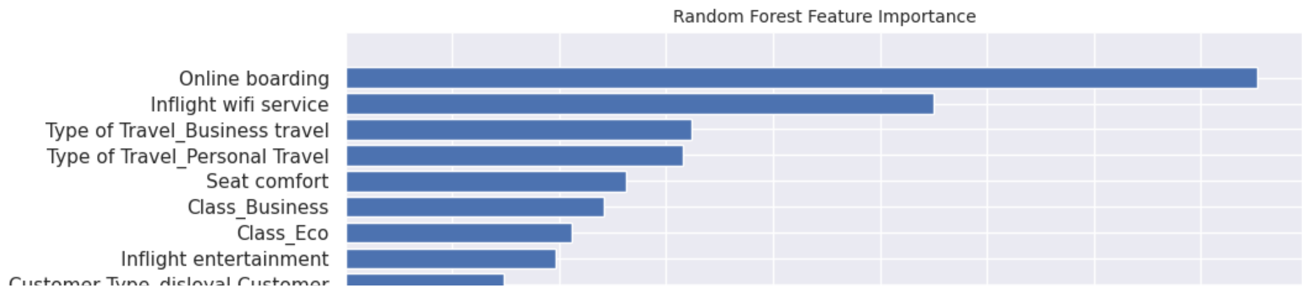
```
# Extract feature importances
feature_importances = best_rf_model.feature_importances_
```

```
feature_names = X_train.columns
```

```
# Creating a DataFrame to visualize feature importance
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importances})
```

```
feature_importance_df = feature_importance_df.sort_values(by='Importance')
```

```
# Visulization of feature importance
plt.figure(figsize=(10, 8))
plt.barh(feature_importance_df['Feature'], feature_importance_df['Importance'])
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Random Forest Feature Importance')
plt.show()
```



Observation :

Features Online boarding & Inflight wifi service are the two most imporatance features in terms of the Random forest model.

Hyperparameter Tuning of XG-boost

```
from scipy.stats import randint, uniform
```

```
xgb_classifier = XGBClassifier()
# Defining the hyperparameter grid
xg_rscv_params = {
    'n_estimators': randint(10,110),
    'learning_rate': uniform(0.01,0.1),
    'max_depth': randint(2, 5),
    'subsample': uniform(0.5, 0.5),
    'colsample_bytree': uniform(0.5, 0.5),
    'gamma': uniform(0, 0.2),
}
```

```
# Creating a RandomizedSearchCV object
xg_rscv = RandomizedSearchCV(
    xgb_classifier,
    param_distributions=xg_rscv_params,
    n_iter=10,
    scoring='accuracy',
    cv=3,
    verbose=1,

)
```

```
# Perform the random search on the training data
xg_rscv.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
▸ RandomizedSearchCV
▸ estimator: XGBClassifier
▸ XGBClassifier
```

```
# Displaying the best parameters
xg_best_param= xg_rscv.best_params_
print("Best Parameters:",xg_best_param )
```

```
Best Parameters: {'colsample_bytree': 0.5443136591027702, 'gamma': 0.05869996717215618, 'learning_rate': 0.1086048672669
```

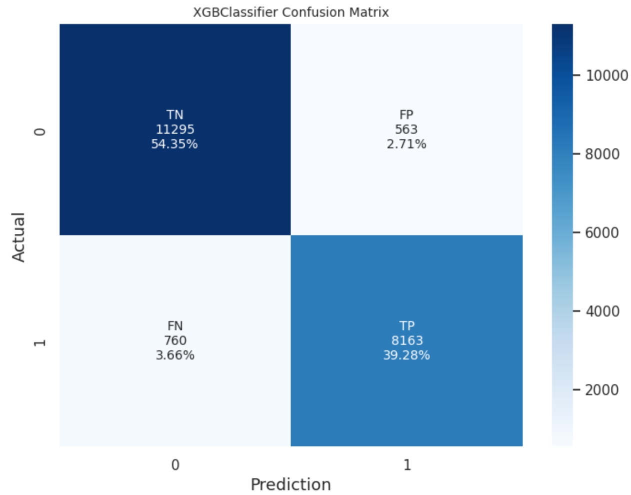
```
# Evaluate the model on the test set
best_model = xg_rscv.best_estimator_
accuracy = best_model.score(X_test, y_test)
print("Test Accuracy:", accuracy)
```

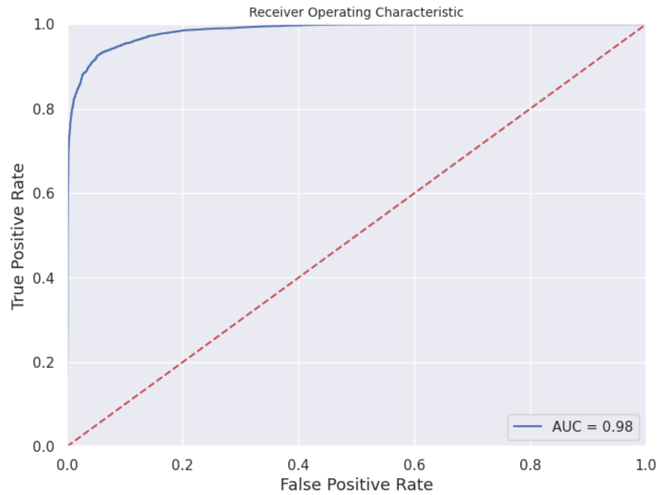
```
Test Accuracy: 0.9369616476589192
```

```
#import the model

from xgboost import XGBClassifier

#fit the model
```

```
model =XGBClassifier(** xg_best_param)
model.fit(X_train,y_train)

# prediction
pred_train = model.predict(X_train)
pred_val = model.predict(X_val)

# model name
model_name = str(type(model)).split(".")[-1][0:-2]
print(f"\t\t{model_name.upper()} MODEL\n")

print('Training part:')
print(classification_report(y_train, pred_train,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
print('validation part:')
print(classification_report(y_val, pred_val,
                                    target_names=['neutral or dissatisfaction', 'satisfaction']))
print("Accuracy score for traing dataset",accuracy_score(y_train, pred_train))
print("Accuracy score for validation dataset",accuracy_score(y_val, pred_val))

make_classification_report(model,X_val,y_val)
```

```
                    XGBCLASSIFIER MODEL

Training part:
                            precision    recall  f1-score   support

neutral or dissatisfaction       0.93      0.96      0.94     35308
              satisfaction       0.94      0.91      0.93     27034

                  accuracy                           0.94     62342
                 macro avg       0.94      0.93      0.94     62342
              weighted avg       0.94      0.94      0.94     62342

validation part:
                            precision    recall  f1-score   support

neutral or dissatisfaction       0.94      0.95      0.94     11858
              satisfaction       0.94      0.91      0.93      8923

                  accuracy                           0.94     20781
                 macro avg       0.94      0.93      0.93     20781
              weighted avg       0.94      0.94      0.94     20781

Accuracy score for traing dataset 0.936768149882904
Accuracy score for validation dataset 0.9363360762234734
ROC AUC Score : 98.42%
```





Double-click (or enter) to edit

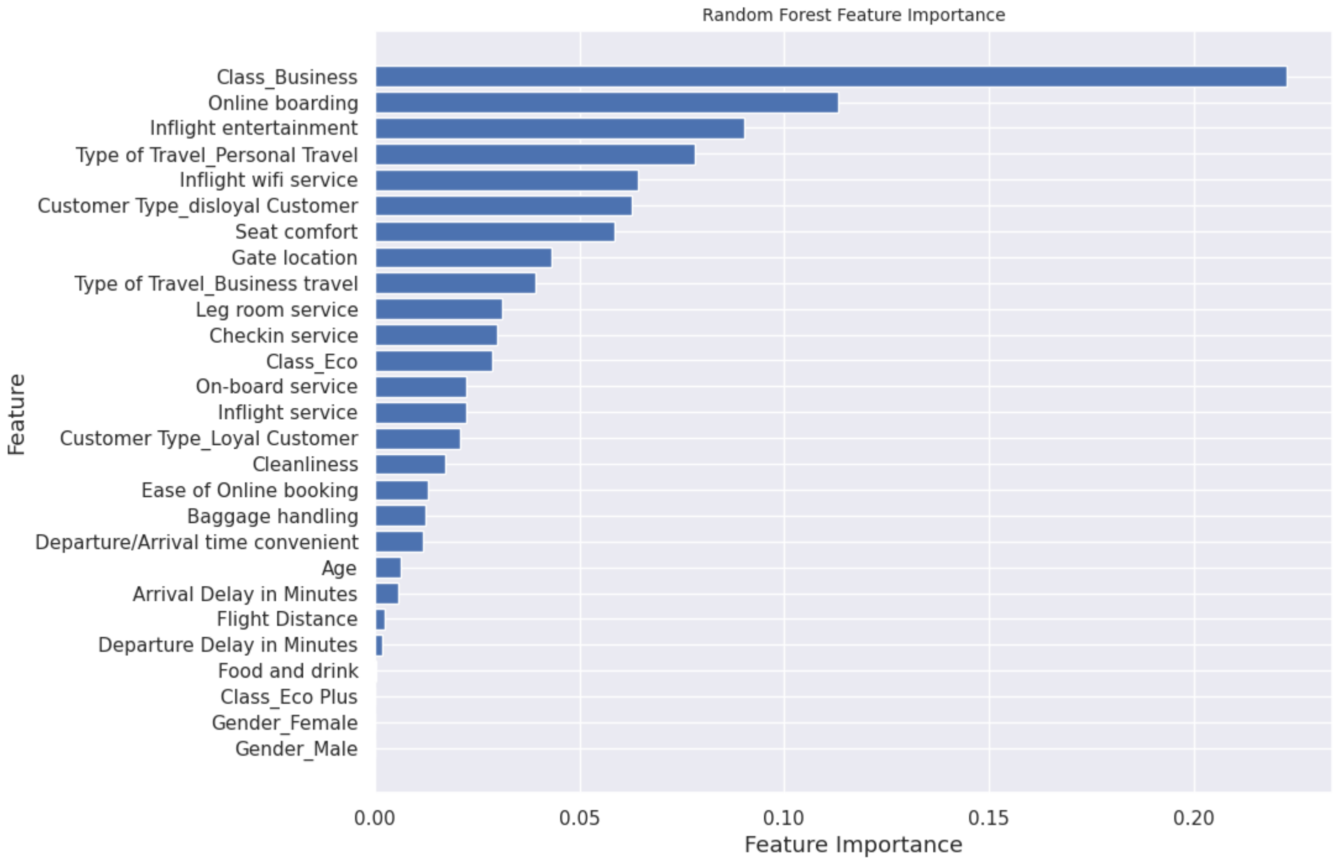
```
# Extracting feature importances
feature_importances_xg= best_model.feature_importances_
```

```
feature_names_xg = X_train.columns
```

```
# Creating a DataFrame to visualize feature importance
feature_importance_df = pd.DataFrame({'Feature': feature_names_xg, 'Importance': feature_importances_xg})
```

```
feature_importance_df = feature_importance_df.sort_values(by='Importance')
```

```
# Visulization of feature importance
plt.figure(figsize=(10, 8))
plt.barh(feature_importance_df['Feature'], feature_importance_df['Importance'])
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Random Forest Feature Importance')
plt.show()
```



Observation:

1. model train , validation & test accuracy and F-1 scores are very near to each other so it favours for model is good fit for genralisation.
2. Business Class ,Online boarding & Inflight entertainment in decrasing orders are the most significant features for the mdoel.

Conclusion

- The goal of the project is accomplished to figure out best model which is XG-boost in this case. By applying this model in deployment production , we can evantually solve the customer chrun problem in aviantion sector.
- I have performed data analysis, data preprocessing, and data modelling with multiple machine learning models to achieve this.