Airline Satisfaction Analysis

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import warnings
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:

```
df=pd.read_csv('project.csv')
df
```

Out[2]:

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	De ti
0	0	19556	Female	Loyal Customer	52	Business travel	Eco	160	5	
1	1	90035	Female	Loyal Customer	36	Business travel	Business	2863	1	
2	2	12360	Male	disloyal Customer	20	Business travel	Eco	192	2	
3	3	77959	Male	Loyal Customer	44	Business travel	Business	3377	0	
4	4	36875	Female	Loyal Customer	49	Business travel	Eco	1182	2	
...	...	...	...	...	...	...	...	...	...	
25971	25971	78463	Male	disloyal Customer	34	Business travel	Business	526	3	
25972	25972	71167	Male	Loyal Customer	23	Business travel	Business	646	4	
25973	25973	37675	Female	Loyal Customer	17	Personal Travel	Eco	828	2	
25974	25974	90086	Male	Loyal Customer	14	Business travel	Business	1127	3	
25975	25975	34799	Female	Loyal Customer	42	Personal Travel	Eco	264	2	

25976 rows × 25 columns

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25976 entries, 0 to 25975
Data columns (total 25 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Unnamed: 0                        25976 non-null  int64  
 1   id                                25976 non-null  int64  
 2   Gender                            25976 non-null  object 
 3   Customer Type                     25976 non-null  object 
 4   Age                               25976 non-null  int64  
 5   Type of Travel                    25976 non-null  object 
 6   Class                             25976 non-null  object 
 7   Flight Distance                   25976 non-null  int64  
 8   Inflight wifi service             25976 non-null  int64  
 9   Departure/Arrival time convenient  25976 non-null  int64  
 10  Ease of Online booking            25976 non-null  int64  
 11  Gate location                     25976 non-null  int64  
 12  Food and drink                    25976 non-null  int64  
 13  Online boarding                   25976 non-null  int64  
 14  Seat comfort                      25976 non-null  int64  
 15  Inflight entertainment            25976 non-null  int64  
 16  On-board service                  25976 non-null  int64  
 17  Leg room service                  25976 non-null  int64  
 18  Baggage handling                  25976 non-null  int64  
 19  Checkin service                   25976 non-null  int64  
 20  Inflight service                  25976 non-null  int64  
 21  Cleanliness                       25976 non-null  int64  
 22  Departure Delay in Minutes        25976 non-null  int64  
 23  Arrival Delay in Minutes          25893 non-null  float64
 24  satisfaction                      25976 non-null  object 
dtypes: float64(1), int64(19), object(5)
memory usage: 5.0+ MB
```

In [4]:

```
df.describe().T
```

Out[4]:

	count	mean	std	min	25%	50%	75%	
Unnamed: 0	25976.0	12987.500000	7498.769632	0.0	6493.75	12987.5	19481.25	2
id	25976.0	65005.657992	37611.526647	17.0	32170.50	65319.5	97584.25	12
Age	25976.0	39.620958	15.135685	7.0	27.00	40.0	51.00	
Flight Distance	25976.0	1193.788459	998.683999	31.0	414.00	849.0	1744.00	
Inflight wifi service	25976.0	2.724746	1.335384	0.0	2.00	3.0	4.00	
Departure/Arrival time convenient	25976.0	3.046812	1.533371	0.0	2.00	3.0	4.00	
Ease of Online booking	25976.0	2.756775	1.412951	0.0	2.00	3.0	4.00	
Gate location	25976.0	2.977094	1.282133	1.0	2.00	3.0	4.00	
Food and drink	25976.0	3.215353	1.331506	0.0	2.00	3.0	4.00	
Online boarding	25976.0	3.261665	1.355536	0.0	2.00	4.0	4.00	
Seat comfort	25976.0	3.449222	1.320090	1.0	2.00	4.0	5.00	
Inflight entertainment	25976.0	3.357753	1.338299	0.0	2.00	4.0	4.00	
On-board service	25976.0	3.385664	1.282088	0.0	2.00	4.0	4.00	
Leg room service	25976.0	3.350169	1.318862	0.0	2.00	4.0	4.00	
Baggage handling	25976.0	3.633238	1.176525	1.0	3.00	4.0	5.00	
Checkin service	25976.0	3.314175	1.269332	1.0	3.00	3.0	4.00	
Inflight service	25976.0	3.649253	1.180681	0.0	3.00	4.0	5.00	
Cleanliness	25976.0	3.286226	1.319330	0.0	2.00	3.0	4.00	
Departure Delay in Minutes	25976.0	14.306090	37.423160	0.0	0.00	0.0	12.00	
Arrival Delay in Minutes	25893.0	14.740857	37.517539	0.0	0.00	0.0	13.00	

In [5]:

```
df['satisfaction'].unique()
```

Out[5]:

```
array(['satisfied', 'neutral or dissatisfied'], dtype=object)
```

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
Unnamed: 0                            0
id                                    0
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             83
satisfaction                          0
dtype: int64
```

In [7]:

```
df.dropna(inplace=True)
```

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
Unnamed: 0                           0
id                                   0
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             0
satisfaction                         0
dtype: int64
```

In [9]:

```
df.reset_index(drop=True,inplace=True)
```

In [10]:

```
df['Unnamed: 0'].unique()
```

Out[10]:

```
array([    0,     1,     2, ..., 25973, 25974, 25975], dtype=int64)
```

In [11]:

```
df.drop(['Unnamed: 0'],axis=1,inplace=True)
```

In [12]:

```
df.drop(['id'],axis=1,inplace=True)
```

In [13]:

```
df['satisfaction'].value_counts()
```

Out[13]:

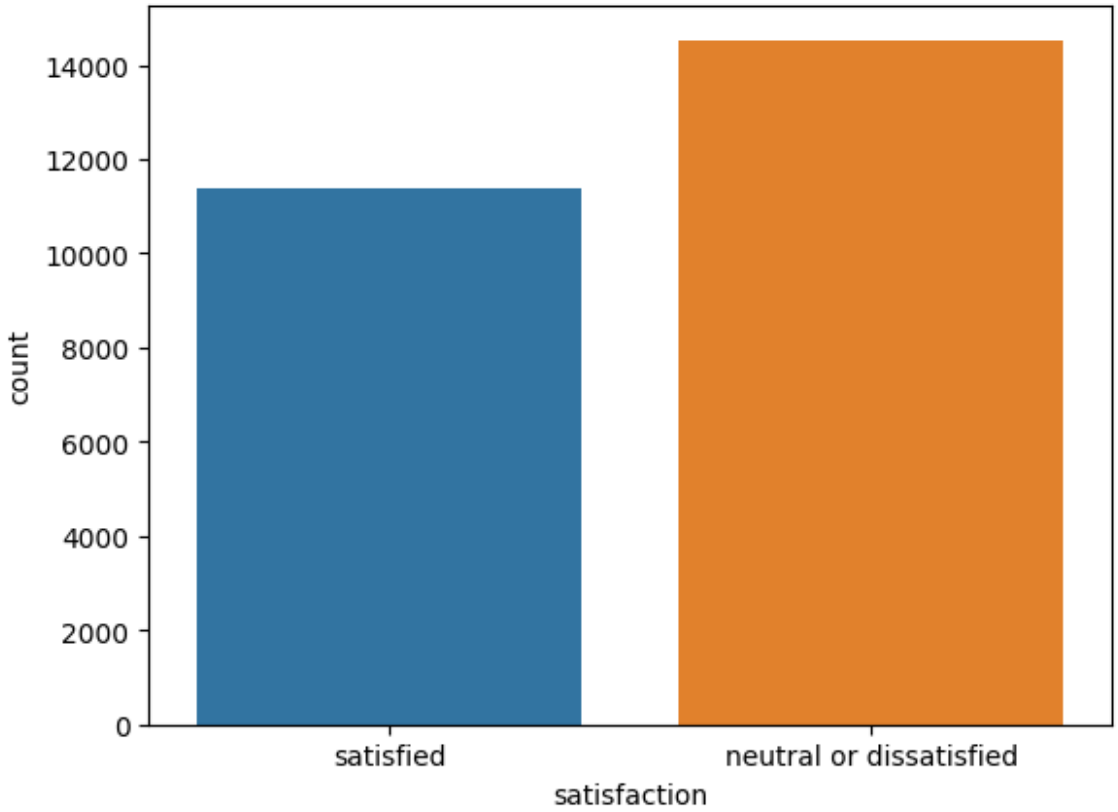
```
neutral or dissatisfied    14528
satisfied                  11365
Name: satisfaction, dtype: int64
```

In [14]:

```
sns.countplot(x=df['satisfaction'])
```

Out[14]:

```
<AxesSubplot: xlabel='satisfaction', ylabel='count'>
```

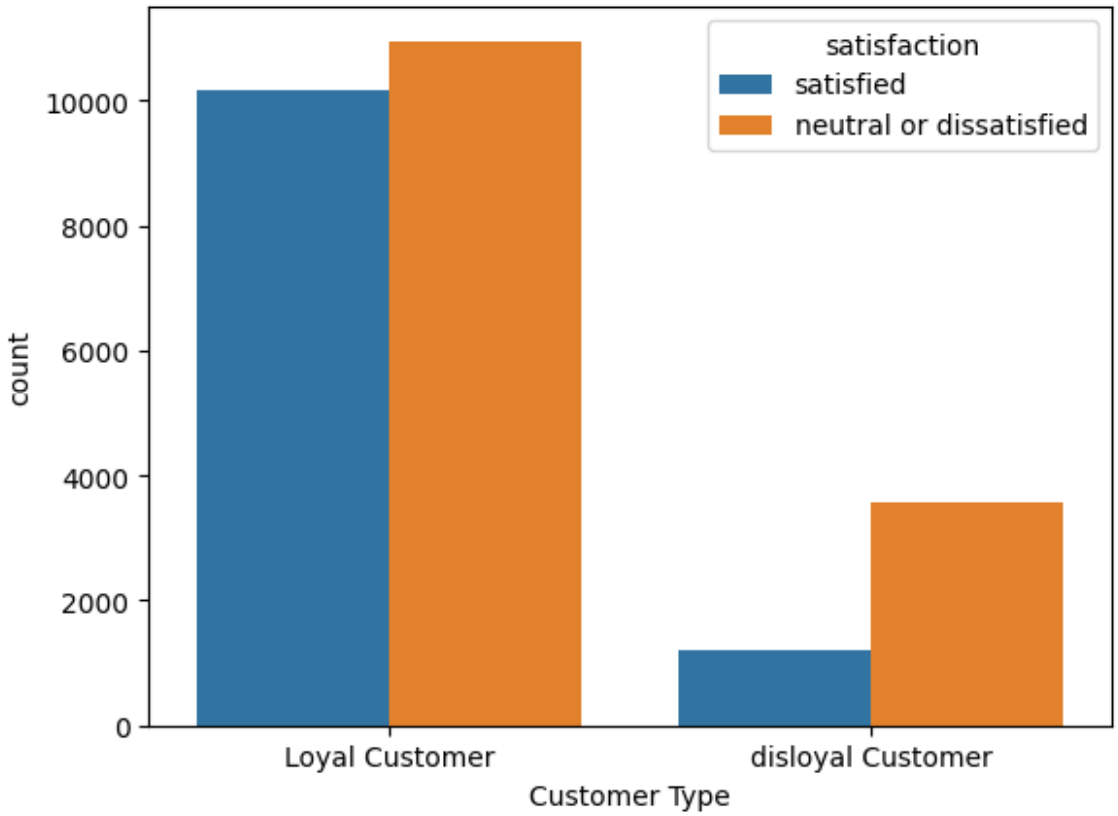


In [15]:

```
sns.countplot(x=df['Customer Type'],hue=df['satisfaction'])
```

Out[15]:

```
<AxesSubplot: xlabel='Customer Type', ylabel='count'>
```

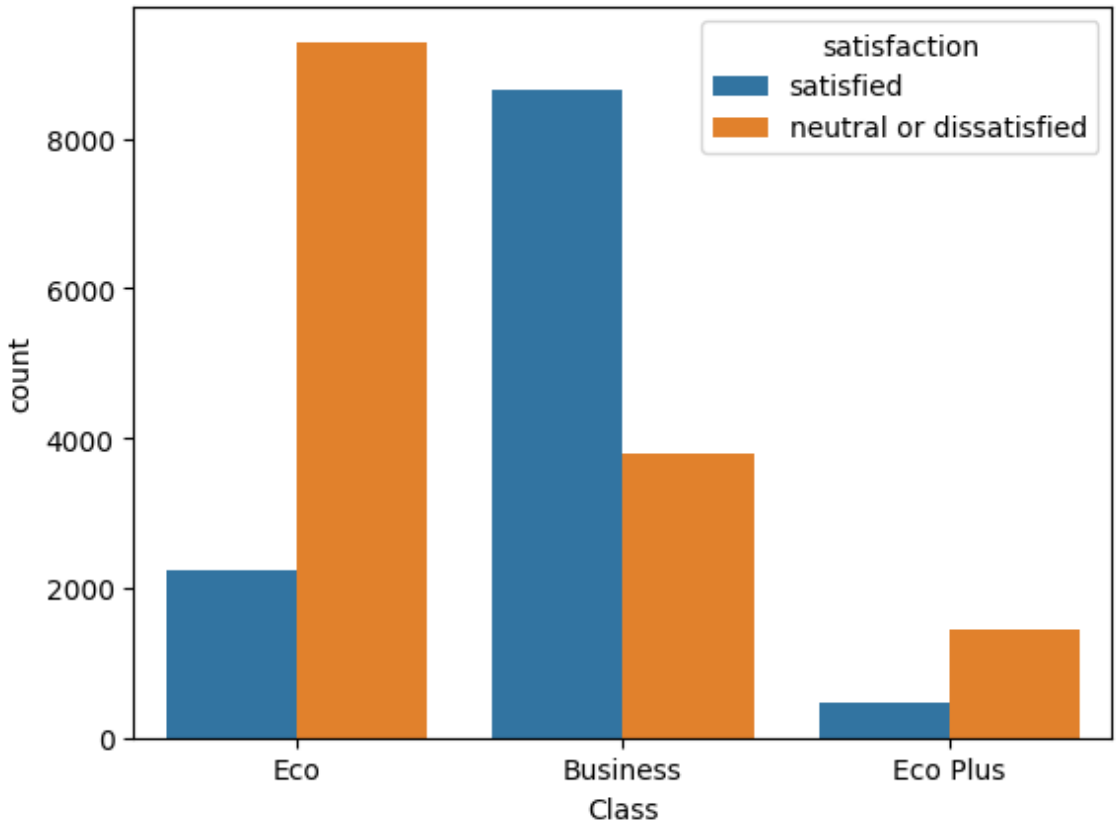


In [16]:

```
sns.countplot(x=df['Class'],hue=df['satisfaction'])
```

Out[16]:

```
<AxesSubplot: xlabel='Class', ylabel='count'>
```

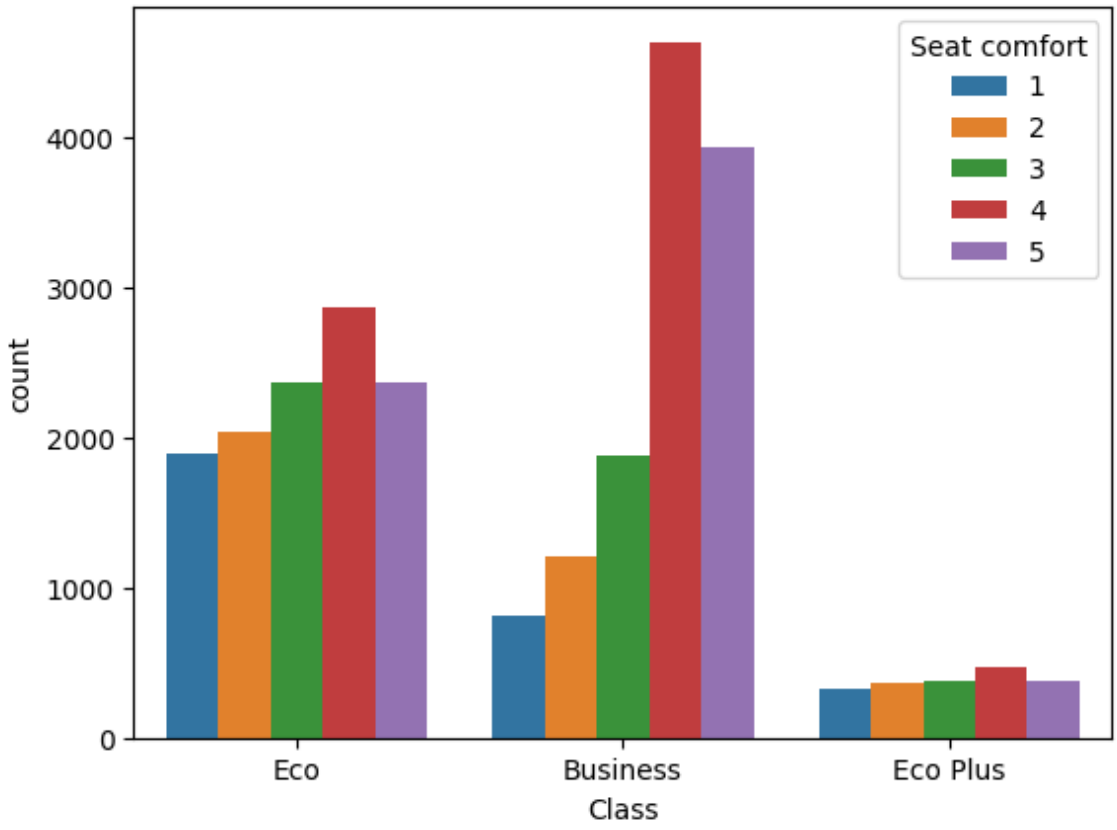


In [17]:

```
sns.countplot(x=df['Class'],hue=df['Seat comfort'])
```

Out[17]:

```
<AxesSubplot: xlabel='Class', ylabel='count'>
```

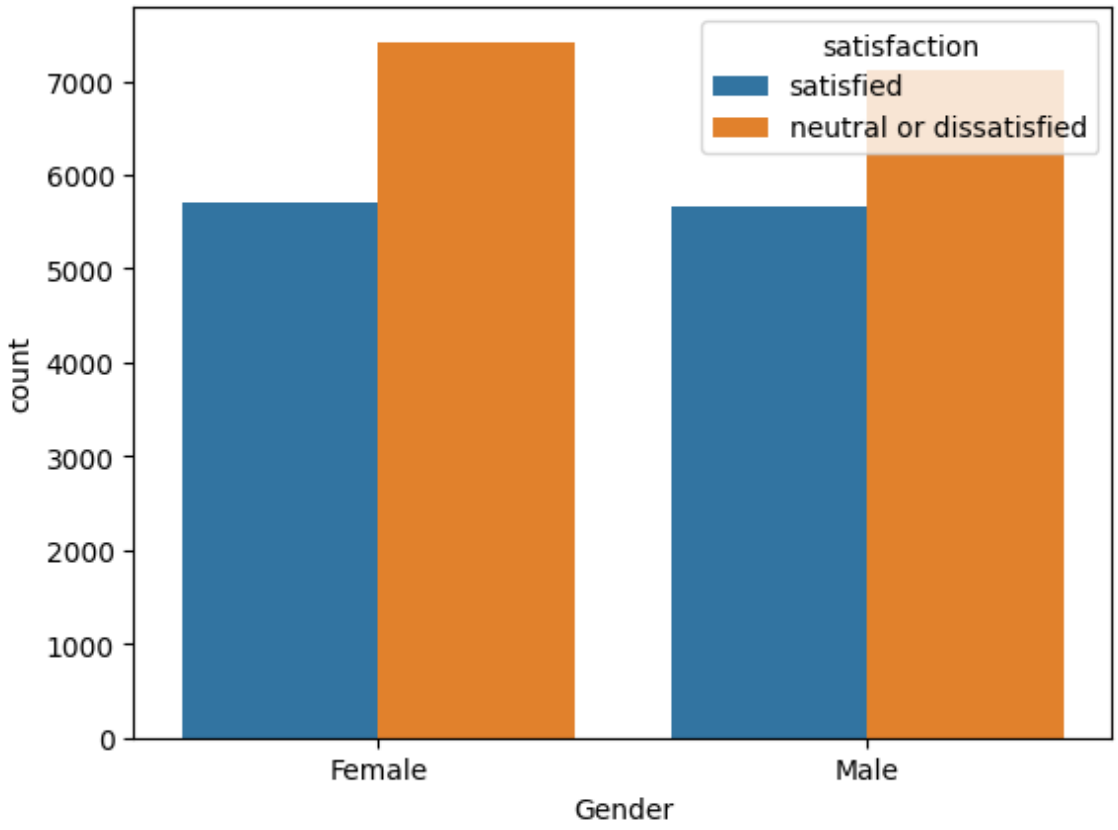


In [18]:

```
sns.countplot(x=df['Gender'],hue=df['satisfaction'])
```

Out[18]:

```
<AxesSubplot: xlabel='Gender', ylabel='count'>
```

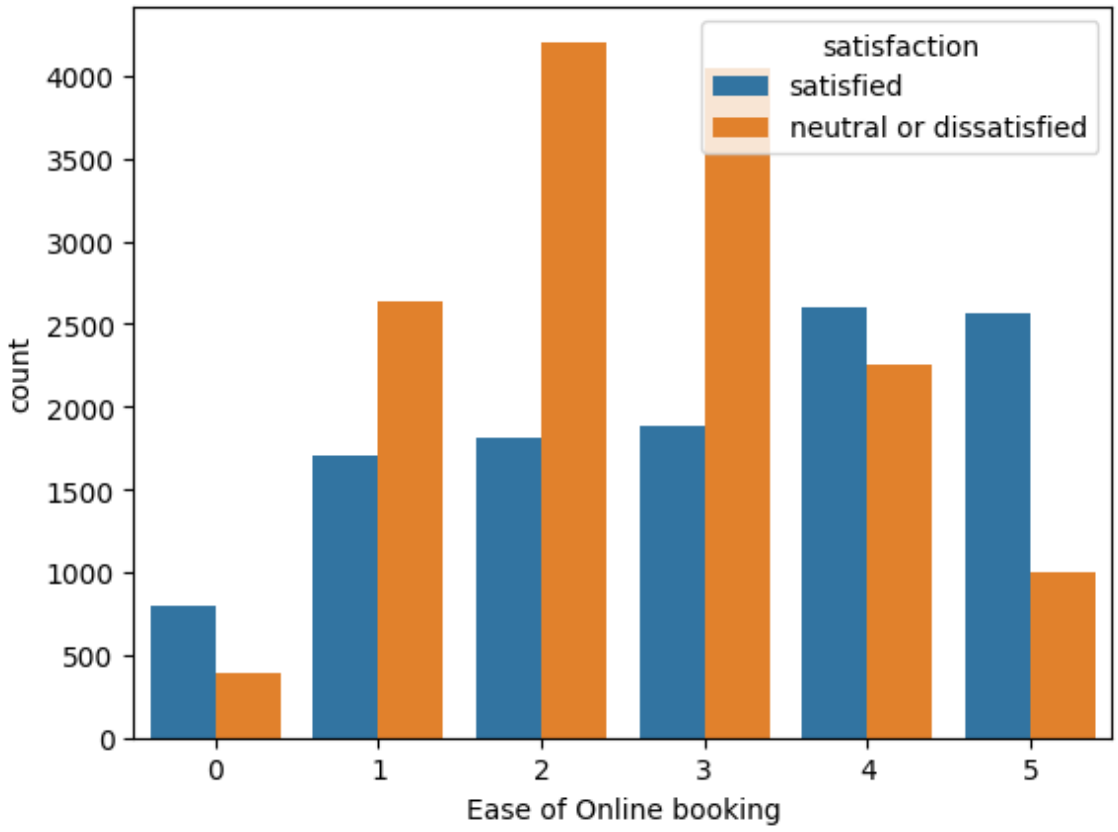


In [19]:

```
sns.countplot(x=df['Ease of Online booking'],hue=df['satisfaction'])
```

Out[19]:

```
<AxesSubplot: xlabel='Ease of Online booking', ylabel='count'>
```

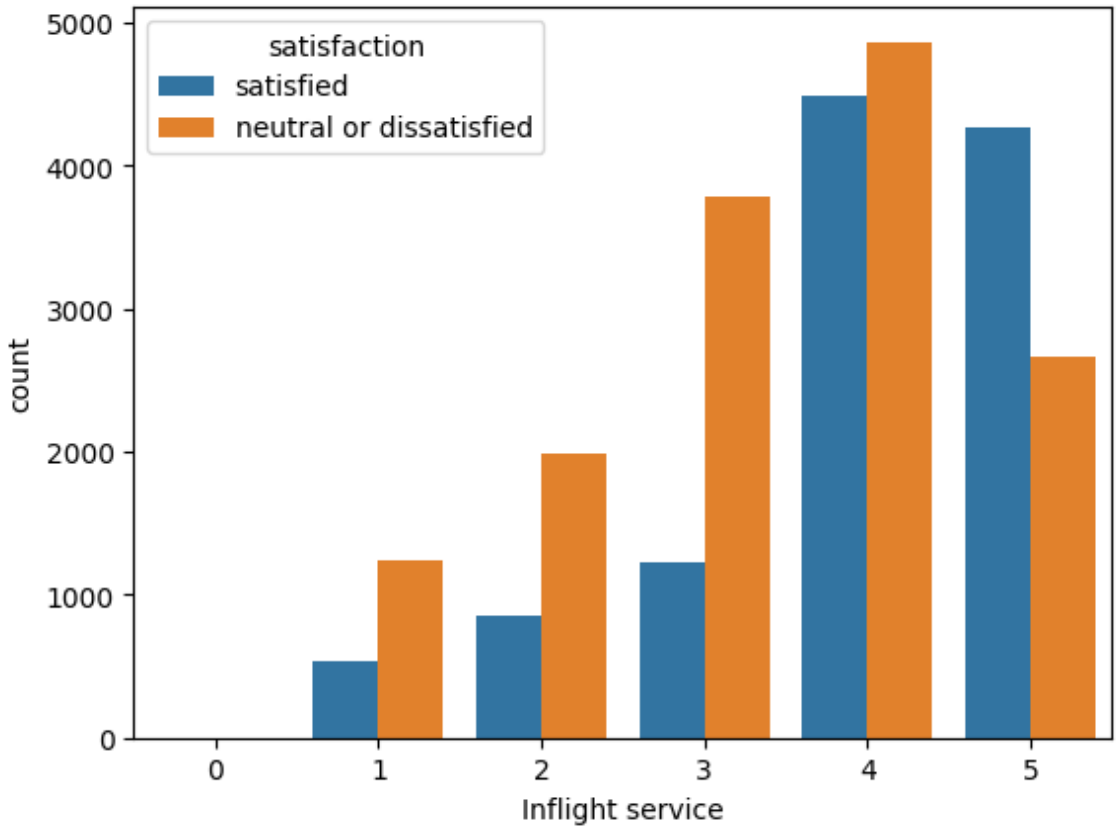


In [20]:

```
sns.countplot(x=df['Inflight service'],hue=df['satisfaction'])
```

Out[20]:

```
<AxesSubplot: xlabel='Inflight service', ylabel='count'>
```

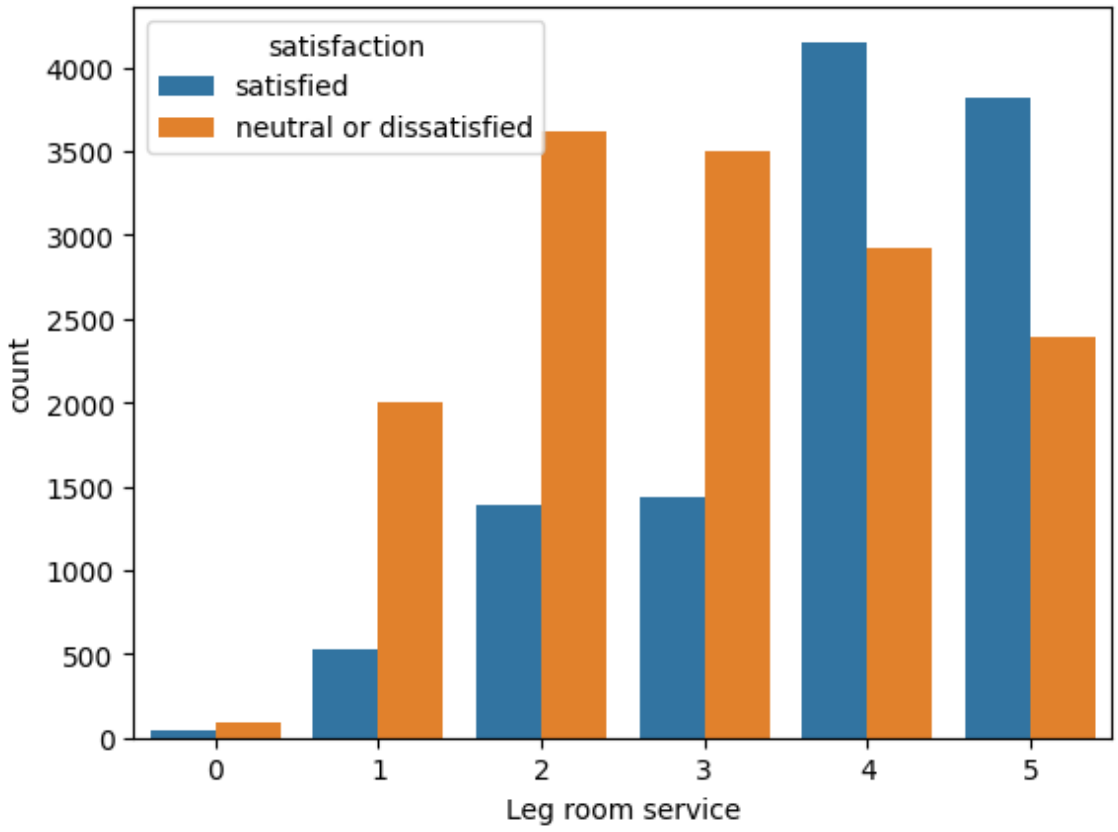


In [21]:

```
sns.countplot(x=df['Leg room service'],hue=df['satisfaction'])
```

Out[21]:

```
<AxesSubplot: xlabel='Leg room service', ylabel='count'>
```

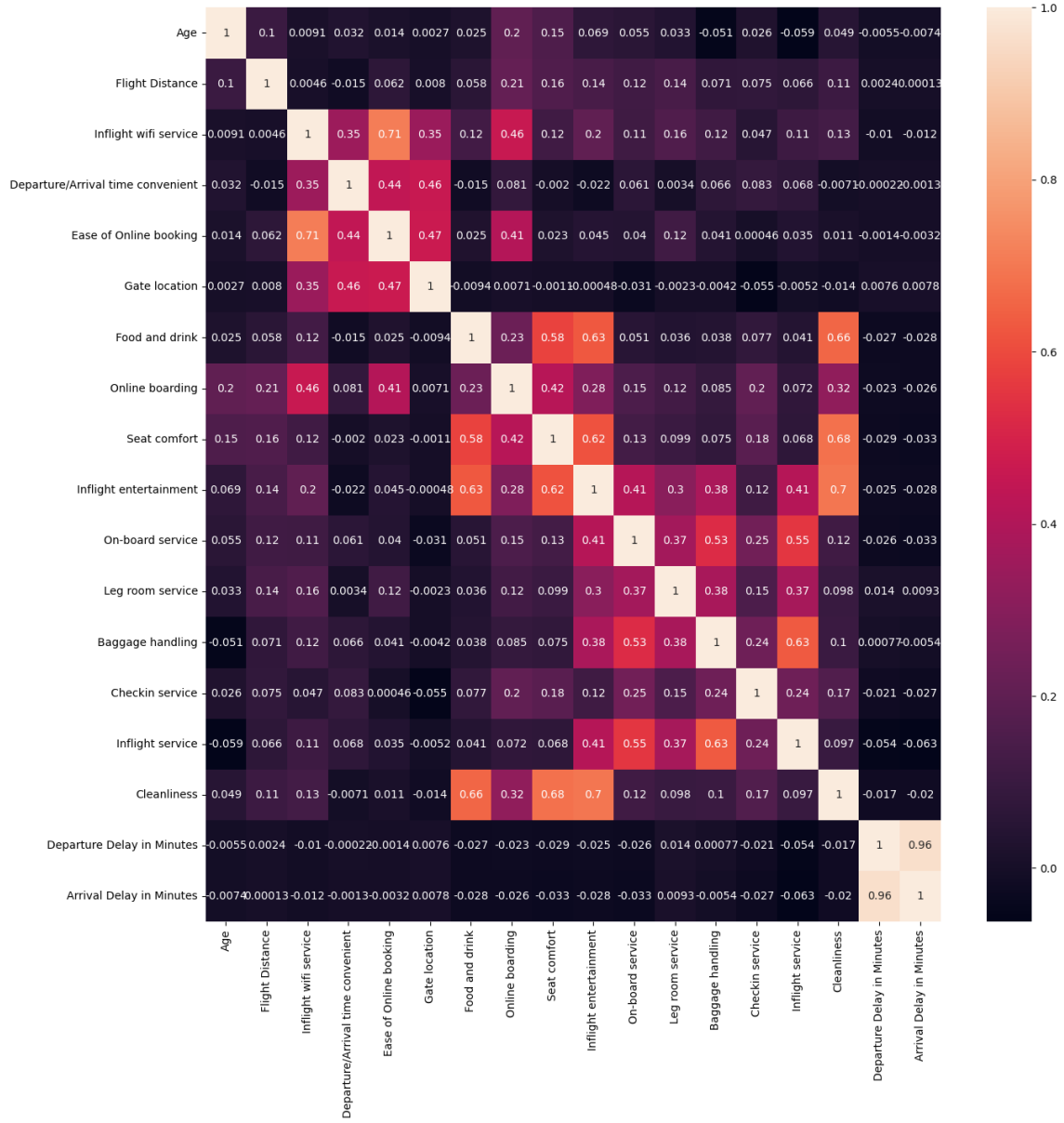


In [22]:

```
plt.figure(figsize=(15,15))
sns.heatmap(df.corr(),annot=True)
```

Out[22]:

```
<AxesSubplot: >
```



Encoding the Object columns

In [23]:

```
from sklearn.preprocessing import LabelEncoder
```

In [24]:

```
le=LabelEncoder()
```

In [25]:

```
cat_col=df.select_dtypes(include=['O']).columns
```

In [26]:

```
cat_col
```

Out[26]:

```
Index(['Gender', 'Customer Type', 'Type of Travel', 'Class', 'satisfactio
n'], dtype='object')
```

In [27]:

```
for i in cat_col:
    df[i]=le.fit_transform(df[i])
```

In [28]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25893 entries, 0 to 25892
Data columns (total 23 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Gender                            25893 non-null  int32
 1   Customer Type                     25893 non-null  int32
 2   Age                               25893 non-null  int64
 3   Type of Travel                    25893 non-null  int32
 4   Class                             25893 non-null  int32
 5   Flight Distance                   25893 non-null  int64
 6   Inflight wifi service             25893 non-null  int64
 7   Departure/Arrival time convenient  25893 non-null  int64
 8   Ease of Online booking            25893 non-null  int64
 9   Gate location                     25893 non-null  int64
 10  Food and drink                    25893 non-null  int64
 11  Online boarding                   25893 non-null  int64
 12  Seat comfort                      25893 non-null  int64
 13  Inflight entertainment            25893 non-null  int64
 14  On-board service                  25893 non-null  int64
 15  Leg room service                  25893 non-null  int64
 16  Baggage handling                  25893 non-null  int64
 17  Checkin service                   25893 non-null  int64
 18  Inflight service                  25893 non-null  int64
 19  Cleanliness                       25893 non-null  int64
 20  Departure Delay in Minutes        25893 non-null  int64
 21  Arrival Delay in Minutes          25893 non-null  float64
 22  satisfaction                      25893 non-null  int32
dtypes: float64(1), int32(5), int64(17)
memory usage: 4.0 MB
```

In [29]:

```
df
```

Out[29]:

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking
0	0	0	52	0	1	160	5	4	3
1	0	0	36	0	0	2863	1	1	3
2	1	1	20	0	1	192	2	0	2
3	1	0	44	0	0	3377	0	0	0
4	0	0	49	0	1	1182	2	3	4
...	...	...	...	...	...	...	...	...	...
25888	1	1	34	0	0	526	3	3	3
25889	1	0	23	0	0	646	4	4	4
25890	0	0	17	1	1	828	2	5	1
25891	1	0	14	0	0	1127	3	3	3
25892	0	0	42	1	1	264	2	5	2

25893 rows × 23 columns

In [30]:

```
df['satisfaction'].unique()
```

Out[30]:

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

Spliting data into train and test data

In [31]:

```
X=df.drop(['satisfaction'],axis=1)
```

In [32]:

```
X
```

Out[32]:

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking
0	0	0	52	0	1	160	5	4	3
1	0	0	36	0	0	2863	1	1	3
2	1	1	20	0	1	192	2	0	2
3	1	0	44	0	0	3377	0	0	0
4	0	0	49	0	1	1182	2	3	4
...	...	...	...	...	...	...	...	...	...
25888	1	1	34	0	0	526	3	3	3
25889	1	0	23	0	0	646	4	4	4
25890	0	0	17	1	1	828	2	5	1
25891	1	0	14	0	0	1127	3	3	3
25892	0	0	42	1	1	264	2	5	2

25893 rows × 22 columns

In [33]:

```
y=df['satisfaction']
```

In [34]:

```
y
```

Out[34]:

```
0        1
1        1
2        0
3        1
4        1
        ..
25888    0
25889    1
25890    0
25891    1
25892    0
Name: satisfaction, Length: 25893, dtype: int32
```

In [35]:

```
from sklearn.model_selection import train_test_split
```

In [36]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.20,random_state=1)
```

Scaling the data

In [37]:

```
from sklearn.preprocessing import StandardScaler
```

In [38]:

```
sc=StandardScaler()
```

In [39]:

```
X_train=sc.fit_transform(X_train)
X_test=sc.fit_transform(X_test)
```

Logistic Regression

In [ ]:

```
from sklearn.linear_model import LogisticRegression
```

In [41]:

```
logi=LogisticRegression()
```

In [42]:

```
from sklearn.metrics import classification_report
```

In [43]:

```
def my_model(model):
    model.fit(X_train,y_train)
    y_pred_train=model.predict(X_train)
    y_pred_test=model.predict(X_test)
    print("Train Data")
    print(classification_report(y_train,y_pred_train))


    print("Test Data")
    print(classification_report(y_test,y_pred_test))
    return model
```

In [44]:

```
my_model(logi)
```

```
Train Data
              precision    recall  f1-score   support

           0       0.87      0.90      0.89     11684
           1       0.87      0.83      0.85      9030

    accuracy                           0.87     20714
   macro avg       0.87      0.87      0.87     20714
weighted avg       0.87      0.87      0.87     20714

Test Data
              precision    recall  f1-score   support

           0       0.87      0.91      0.89      2844
           1       0.89      0.83      0.86      2335

    accuracy                           0.87      5179
   macro avg       0.88      0.87      0.87      5179
weighted avg       0.88      0.87      0.87      5179
```

Out[44]:

```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Random Forest

In [45]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [46]:

```
rf=RandomForestClassifier()
```

In [47]:

```
my_model(rf)
```

```
Train Data
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     11684
           1       1.00      1.00      1.00      9030

    accuracy                           1.00     20714
   macro avg       1.00      1.00      1.00     20714
weighted avg       1.00      1.00      1.00     20714

Test Data
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      2844
           1       0.97      0.93      0.95      2335

    accuracy                           0.95      5179
   macro avg       0.96      0.95      0.95      5179
weighted avg       0.95      0.95      0.95      5179
```

Out[47]:

```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [65]:

```
rf1=RandomForestClassifier(n_estimators=200,max_depth=10,criterion='entropy')
```

In [66]:

```
my_model(rf1)
```

```
Train Data
              precision    recall  f1-score   support

           0       0.95      0.97      0.96     11684
           1       0.96      0.94      0.95      9030

    accuracy                           0.96     20714
   macro avg       0.96      0.95      0.95     20714
weighted avg       0.96      0.96      0.96     20714

Test Data
              precision    recall  f1-score   support

           0       0.94      0.95      0.95      2844
           1       0.94      0.93      0.94      2335

    accuracy                           0.94      5179
   macro avg       0.94      0.94      0.94      5179
weighted avg       0.94      0.94      0.94      5179
```

Out[66]:

```
RandomForestClassifier(criterion='entropy', max_depth=10, n_estimators=20
0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Support Vector Machine

In [48]:

```
from sklearn.svm import SVC
```

In [49]:

```
svc=SVC()
```

In [50]:

```
my_model(svc)
```

```
Train Data
              precision    recall  f1-score   support

           0       0.95      0.97      0.96     11684
           1       0.96      0.94      0.95      9030

    accuracy                           0.96     20714
   macro avg       0.96      0.95      0.95     20714
weighted avg       0.96      0.96      0.96     20714

Test Data
              precision    recall  f1-score   support

           0       0.94      0.96      0.95      2844
           1       0.96      0.93      0.94      2335

    accuracy                           0.95      5179
   macro avg       0.95      0.95      0.95      5179
weighted avg       0.95      0.95      0.95      5179
```

Out[50]:

SVC()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Decision Tree

In [51]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [52]:

```
dt=DecisionTreeClassifier()
```

In [53]:

```
my_model(dt)
```

```
Train Data
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     11684
           1       1.00      1.00      1.00      9030

    accuracy                           1.00     20714
   macro avg       1.00      1.00      1.00     20714
weighted avg       1.00      1.00      1.00     20714

Test Data
              precision    recall  f1-score   support

           0       0.95      0.94      0.94      2844
           1       0.93      0.93      0.93      2335

    accuracy                           0.94      5179
   macro avg       0.94      0.94      0.94      5179
weighted avg       0.94      0.94      0.94      5179
```

Out[53]:

```
DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [54]:

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import KFold,cross_val_score,StratifiedKFold
```

In [55]:

```
kcv=KFold(n_splits=10,shuffle=True,random_state=1)
score=cross_val_score(dt,X,y,scoring='f1',cv=kcv,n_jobs=-1)
print(score)
print('Average score',np.mean(score))
```

```
[0.92635164 0.94132873 0.91906236 0.93000446 0.93327594 0.92405063
 0.92388451 0.92574478 0.92672414 0.92626932]
Average score 0.9276696503458574
```

In [56]:

```
skf=StratifiedKFold(n_splits=10,shuffle=True,random_state=123)
score=cross_val_score(dt,X,y,scoring='f1',cv=skf,n_jobs=-1)
print(score)
print("Average Score",np.mean(score))
```

```
[0.93234395 0.92674316 0.92555507 0.92794376 0.93211488 0.91560997
 0.92850915 0.92822113 0.92151675 0.9211801 ]
Average Score 0.9259737937524862
```

In [57]:

```
param_grid={
    'criterion':['gini','entropy'],
    'max_depth':np.arange(1,50),
    'min_samples_split':np.arange(1,50,2),
    'min_samples_leaf':np.arange(1,50),
    'class_weight':[None,'balanced']
}
```

In [58]:

```
clf=RandomizedSearchCV(dt,param_grid,cv=5,scoring='f1',n_jobs=-1)
clf.fit(X_train,y_train)
```

Out[58]:

```
RandomizedSearchCV(cv=5, estimator=DecisionTreeClassifier(), n_jobs=-1,
                   param_distributions={'class_weight': [None, 'balance
d'],
                                        'criterion': ['gini', 'entropy'],
                                        'max_depth': array([ 1,  2,  3,
4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]),
                                        'min_samples_leaf': array([ 1,  2,
3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]),
                                        'min_samples_split': array([ 1,
3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33,
       35, 37, 39, 41, 43, 45, 47, 49])},
                   scoring='f1')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [59]:

```
clf.best_params_
```

Out[59]:

```
{'min_samples_split': 33,
 'min_samples_leaf': 5,
 'max_depth': 43,
 'criterion': 'gini',
 'class_weight': None}
```

In [60]:

```
dt1=DecisionTreeClassifier(min_samples_split=21,min_samples_leaf=13,max_depth=27,criteri
```

In [61]:

```
my_model(dt1)
```

```
Train Data
              precision    recall  f1-score   support

           0       0.96      0.97      0.96     11684
           1       0.96      0.94      0.95      9030

    accuracy                           0.96     20714
   macro avg       0.96      0.96      0.96     20714
weighted avg       0.96      0.96      0.96     20714

Test Data
              precision    recall  f1-score   support

           0       0.93      0.96      0.95      2844
           1       0.95      0.91      0.93      2335

    accuracy                           0.94      5179
   macro avg       0.94      0.94      0.94      5179
weighted avg       0.94      0.94      0.94      5179
```

Out[61]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=27, min_samples_leaf
=13,
                       min_samples_split=21)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

AdaBoost Classifier

In [62]:

```
from sklearn.ensemble import AdaBoostClassifier
```

In [63]:

```
ada=AdaBoostClassifier(n_estimators=200)
```

In [64]:

```
my_model(ada)
```

```
Train Data
              precision    recall  f1-score   support

           0       0.93      0.94      0.94     11684
           1       0.93      0.91      0.92      9030

    accuracy                           0.93     20714
   macro avg       0.93      0.93      0.93     20714
weighted avg       0.93      0.93      0.93     20714

Test Data
              precision    recall  f1-score   support

           0       0.93      0.94      0.94      2844
           1       0.93      0.92      0.92      2335

    accuracy                           0.93      5179
   macro avg       0.93      0.93      0.93      5179
weighted avg       0.93      0.93      0.93      5179
```

Out[64]:

```
AdaBoostClassifier(n_estimators=200)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [68]:

```
param_grid1={
    'learning_rate':[0.1,0.01,1,2,3],
    'n_estimators':[50,100,150]
}
```

In [69]:

```
ada1=RandomizedSearchCV(ada,param_distributions=param_grid1,n_iter=10,scoring='f1',n_job
```

In [70]:

```
ada1.fit(X_train,y_train)
```

Out[70]:

```
RandomizedSearchCV(estimator=AdaBoostClassifier(n_estimators=200), n_jobs=
-1,
                   param_distributions={'learning_rate': [0.1, 0.01, 1, 2,
3],
                                        'n_estimators': [50, 100, 150]},
                   scoring='f1')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [71]:

```
ada1.best_params_
```

Out[71]:

```
{'n_estimators': 100, 'learning_rate': 1}
```

In [72]:

```
ada2=AdaBoostClassifier(n_estimators=100,learning_rate=1)
```

In [73]:

```
my_model(ada2)
```

```
Train Data
              precision    recall  f1-score   support

           0       0.93      0.94      0.94     11684
           1       0.93      0.91      0.92      9030

    accuracy                           0.93     20714
   macro avg       0.93      0.93      0.93     20714
weighted avg       0.93      0.93      0.93     20714

Test Data
              precision    recall  f1-score   support

           0       0.93      0.94      0.94      2844
           1       0.93      0.92      0.92      2335

    accuracy                           0.93      5179
   macro avg       0.93      0.93      0.93      5179
weighted avg       0.93      0.93      0.93      5179
```

Out[73]:

```
AdaBoostClassifier(learning_rate=1, n_estimators=100)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

All models are giving best prediction besides SVC

In [ ]:

```

```