```
In [1]: import pandas as pandaX
```

```
In [2]: Ze_Test = pandaX.read_csv(r"C:\Users\20F20753\Downloads\test.csv")
        Ze_Train = pandaX.read_csv(r"C:\Users\20F20753\Downloads\train.csv")
```

```
In [3]: Ze_Train = Ze_Train.drop(Ze_Train.iloc[:,[0, 1]], axis = 1)
```

```
In [4]: Ze_Train.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
```

```
In [5]: Ze_Test = Ze_Test.drop(Ze_Test.iloc[:,[0, 1]], axis = 1)
        Ze_Test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25976 entries, 0 to 25975
Data columns (total 23 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   Gender                             25976 non-null  object 
 1   Customer Type                      25976 non-null  object 
 2   Age                                25976 non-null  int64  
 3   Type of Travel                     25976 non-null  object 
 4   Class                              25976 non-null  object 
 5   Flight Distance                    25976 non-null  int64  
 6   Inflight wifi service              25976 non-null  int64  
 7   Departure/Arrival time convenient  25976 non-null  int64  
 8   Ease of Online booking             25976 non-null  int64  
 9   Gate location                      25976 non-null  int64  
 10  Food and drink                     25976 non-null  int64  
 11  Online boarding                    25976 non-null  int64  
 12  Seat comfort                       25976 non-null  int64  
 13  Inflight entertainment             25976 non-null  int64  
 14  On-board service                   25976 non-null  int64  
 15  Leg room service                   25976 non-null  int64  
 16  Baggage handling                   25976 non-null  int64  
 17  Checkin service                    25976 non-null  int64  
 18  Inflight service                   25976 non-null  int64  
 19  Cleanliness                        25976 non-null  int64  
 20  Departure Delay in Minutes         25976 non-null  int64  
 21  Arrival Delay in Minutes           25893 non-null  float64
 22  satisfaction                       25976 non-null  object 
dtypes: float64(1), int64(17), object(5)
memory usage: 4.6+ MB
```

In [6]:
```
Ze_Train.columns = [c.replace(' ', '_') for c in Ze_Train.columns]
Ze_Test.columns = [c.replace(' ', '_') for c in Ze_Test.columns]
```

In [7]:
```
Ze_Train['satisfaction'].replace({'neutral or dissatisfied': 0, 'satisfied': 1},inplace = True)
Ze_Test['satisfaction'].replace({'neutral or dissatisfied': 0, 'satisfied': 1},inplace = True)
```

In [9]:
```
Ze_Total = Ze_Train.isnull().sum().sort_values(ascending=False)
percentage = (Ze_Train.isnull().sum()/Ze_Train.isnull().count()).sort_values(ascending=False)
inBalance = pandaX.concat([Ze_Total, percentage], axis=1, keys=['Total', 'Percent'])
inBalance.head()
```

Out[9]:

	Total	Percent
Arrival_Delay_in_Minutes	310	0.002984
Gender	0	0.000000
Seat_comfort	0	0.000000
Departure_Delay_in_Minutes	0	0.000000
Cleanliness	0	0.000000

In [10]:
```
Ze_Train['Arrival_Delay_in_Minutes'] = Ze_Train['Arrival_Delay_in_Minutes'].fillna(Ze_Train['Arrival_Delay_in_Minutes'].mean())
Ze_Test['Arrival_Delay_in_Minutes'] = Ze_Test['Arrival_Delay_in_Minutes'].fillna(Ze_Test['Arrival_Delay_in_Minutes'].mean())
```

In [11]:
```
Ze_Train.select_dtypes(include=['object']).columns
```

Out[11]:
```
Index(['Gender', 'Customer_Type', 'Type_of_Travel', 'Class'], dtype='object')
```

In [14]:
```
Ze_Train['Gender'] = Ze_Train['Gender'].fillna(Ze_Train['Gender'].mode()[0])
Ze_Train['Customer_Type'] = Ze_Train['Customer_Type'].fillna(Ze_Train['Customer_Type'].mode()[0])
Ze_Train['Type_of_Travel'] = Ze_Train['Type_of_Travel'].fillna(Ze_Train['Type_of_Travel'].mode()[0])
Ze_Train['Class'] = Ze_Train['Class'].fillna(Ze_Train['Class'].mode()[0])
```

In [15]:
```
Ze_Test['Gender'] = Ze_Test['Gender'].fillna(Ze_Test['Gender'].mode()[0])
Ze_Test['Customer_Type'] = Ze_Test['Customer_Type'].fillna(Ze_Test['Customer_Type'].mode()[0])
Ze_Test['Type_of_Travel'] = Ze_Test['Type_of_Travel'].fillna(Ze_Test['Type_of_Travel'].mode()[0])
Ze_Test['Class'] = Ze_Test['Class'].fillna(Ze_Test['Class'].mode()[0])
```

In [17]:
```
from sklearn.preprocessing import LabelEncoder
lenCode = {}
for column in Ze_Train.select_dtypes(include=['object']).columns:
    lenCode[column] = LabelEncoder()
    Ze_Train[column] = lenCode[column].fit_transform(Ze_Train[column])
```

In [18]:
```
lencoders_t = {}
for col in Ze_Test.select_dtypes(include=['object']).columns:
    lencoders_t[col] = LabelEncoder()
    Ze_Test[col] = lencoders_t[col].fit_transform(Ze_Test[col])
```

In [19]:
```
Q1 = Ze_Train.quantile(0.25)
Q3 = Ze_Train.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

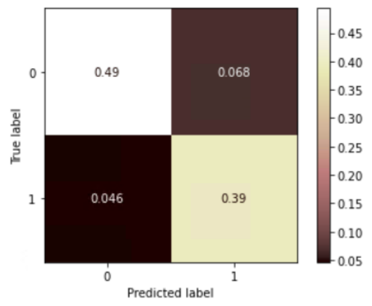
```
Gender                                  1.0
Customer_Type                           0.0
Age                                    24.0
Type_of_Travel                          1.0
Class                                   1.0
Flight_Distance                      1329.0
Inflight_wifi_service                   2.0
Departure/Arrival_time_convenient       2.0
Ease_of_Online_booking                  2.0
Gate_location                           2.0
Food_and_drink                          2.0
Online_boarding                         2.0
Seat_comfort                            3.0
Inflight_entertainment                  2.0
On-board_service                        2.0
Leg_room_service                        2.0
Baggage_handling                        2.0
Checkin_service                         1.0
Inflight_service                        2.0
Cleanliness                             2.0
Departure_Delay_in_Minutes             12.0
Arrival_Delay_in_Minutes               13.0
satisfaction                            1.0
dtype: float64
```

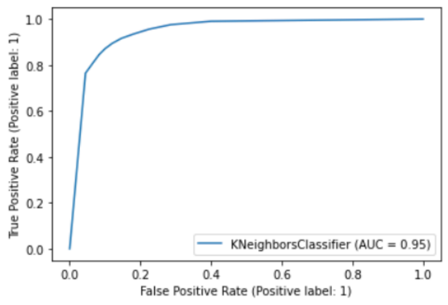
```
In [20]: Ze_Train = Ze_Train[~((Ze_Train < (Q1 - 1.5 * IQR)) |(Ze_Train > (Q3 + 1.5 * IQR))).any(axis=1)]
         Ze_Train.shape

Out[20]: (61197, 23)
```

```
In [21]: features = ['Type_of_Travel','Inflight_wifi_service','Online_boarding','Seat_comfort','Flight_Distance',
                     'Inflight_entertainment','On-board_service','Leg_room_service','Cleanliness','Checkin_service',
                     'Inflight_service', 'Baggage_handling']
         target = ['satisfaction']

         # Split into test and train
         x_train = Ze_Train[features]
         y_train = Ze_Train[target].to_numpy()
         x_test = Ze_Test[features]
         y_test = Ze_Test[target].to_numpy()

         # Normalize Features
         from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()
         x_train = scaler.fit_transform(x_train)
         x_test = scaler.fit_transform(x_test)
```

```
In [24]: import time
         from sklearn.metrics import accuracy_score, roc_auc_score, classification_report, plot_confusion_matrix, plot_roc_curve
         from matplotlib import pyplot as plt
         def run_model(model, x_train, y_train, x_test, y_test, verbose=True):
             t0=time.time()
             if verbose == False:
                 model.fit(x_train,y_train.ravel(), verbose=0)
             else:
                 model.fit(x_train,y_train.ravel())
             y_pred = model.predict(x_test)
             accuracy = accuracy_score(y_test, y_pred)
             roc_auc = roc_auc_score(y_test, y_pred)
             time_taken = time.time()-t0
             print("Accuracy = {}".format(accuracy))
             print("ROC Area under Curve = {}".format(roc_auc))
             print("Time taken = {}".format(time_taken))
             print(classification_report(y_test,y_pred,digits=5))
             plot_confusion_matrix(model, x_test, y_test,cmap=plt.cm.pink, normalize = 'all')
             plot_roc_curve(model, x_test, y_test)

             return model, accuracy, roc_auc, time_taken
```

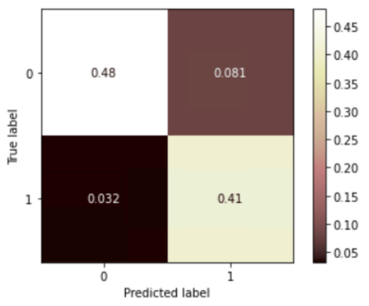
```
In [23]: from sklearn.neighbors import KNeighborsClassifier

         params_kn = {'n_neighbors':10, 'algorithm': 'kd_tree', 'n_jobs':4}

         model_kn = KNeighborsClassifier(**params_kn)
         model_kn, accuracy_kn, roc_auc_kn, tt_kn = run_model(model_kn, x_train, y_train, x_test, y_test)

         Accuracy = 0.8861256544502618
         ROC Area under Curve = 0.8870270908506304
         Time taken = 5.979045391082764
                       precision    recall  f1-score   support

                    0    0.91414   0.87964   0.89656     14573
                    1    0.85326   0.89441   0.87335     11403

             accuracy                        0.88613     25976
            macro avg    0.88370   0.88703   0.88496     25976
         weighted avg    0.88741   0.88613   0.88637     25976
```



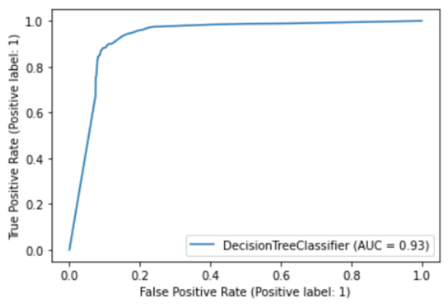


```
In [25]: from sklearn.tree import DecisionTreeClassifier
         params_dt = {'max_depth': 12,
                      'max_features': "sqrt"}

         model_dt = DecisionTreeClassifier(**params_dt)
         model_dt, accuracy_dt, roc_auc_dt, tt_dt = run_model(model_dt, x_train, y_train, x_test, y_test)
```

```
Accuracy = 0.8870110871573761
ROC Area under Curve = 0.8912690081166335
Time taken = 0.06265091896057129
              precision    recall  f1-score   support

           0    0.93680   0.85638   0.89478     14573
           1    0.83460   0.92616   0.87800     11403

    accuracy                        0.88701     25976
   macro avg    0.88570   0.89127   0.88639     25976
weighted avg    0.89193   0.88701   0.88742     25976
```



0.48
0.081
0.032
0.41
True label
Predicted label
0
1
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05



True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
DecisionTreeClassifier (AUC = 0.93)
0.0
0.2
0.4
0.6
0.8
1.0