To implement machine learning algorithms and identify the factors that play vital role in influencing the customer satisfaction rate and recommend the management ways on how should they go about increasing the customer satisfaction rating?

Data Description

In [1]:
```
#Suppress warnings...
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import pandas as pd
df_data = pd.read_csv("train.csv")
print('...... CSV file read successfully')
df_data.head()
```

```
...... CSV file read successfully
```

Out[2]:

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

5 rows × 24 columns

The data columns are as follows:

Gender: Gender of the passengers (Female, Male) Customer Type: The customer type (Loyal customer, disloyal customer) Age: The actual age of the passengers Type of Travel: Purpose of the flight of the passengers (Personal Travel, Business Travel) Class: Travel class in the plane of the passengers (Business, Eco, Eco Plus) Flight distance: The flight distance of this journey Inflight wifi service: Satisfaction level of the inflight wifi service (0:Not Applicable;1-5) Departure/Arrival time convenient: Satisfaction level of Departure/Arrival time convenient Ease of Online booking: Satisfaction level of online booking Gate location: Satisfaction level of Gate location Food and drink: Satisfaction level of Food and drink Online boarding: Satisfaction level of online boarding Seat comfort: Satisfaction level of Seat comfort Inflight entertainment: Satisfaction level of inflight entertainment On-board service: Satisfaction level of On-board service Leg room service: Satisfaction level of Leg room service Baggage handling: Satisfaction level of baggage handling Check-in service: Satisfaction level of Check-in service Inflight service: Satisfaction level of inflight service Cleanliness: Satisfaction level of Cleanliness Departure Delay in Minutes: Minutes delayed when departure Arrival Delay in Minutes: Minutes delayed when Arrival Satisfaction: Airline satisfaction level(Satisfaction, neutral or dissatisfaction) There are 4 Categorical Variables: ---------- Gender [Female,

Male] Customer Type [Loyal Customer, Disloyal Customer] Type of Travel [Personal Travel, Business Travel] Class [Business, Eco, Eco Plus] Target Variable ---------- Satisfaction [Satisfied, Neutral or Dissatisfied]

In [3]:
```
df_data.columns   #get the name of the columns contained in the data file
```

Out[3]:
```
Index(['id', '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',
       'satisfaction'],
      dtype='object')
```

In [4]:
```
df_data.info()   #get some information about the variables contained in the data file
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103904 entries, 0 to 103903
Data columns (total 24 columns):
 #   Column                            Non-Null Count   Dtype  
---  ------                            --------------   -----  
 0   id                                103904 non-null  int64  
 1   Gender                            103904 non-null  object 
 2   Customer Type                     103904 non-null  object 
 3   Age                               103904 non-null  int64  
 4   Type of Travel                    103904 non-null  object 
 5   Class                             103904 non-null  object 
 6   Flight Distance                   103904 non-null  int64  
 7   Inflight wifi service             103904 non-null  int64  
 8   Departure/Arrival time convenient  103904 non-null  int64  
 9   Ease of Online booking            103904 non-null  int64  
 10  Gate location                     103904 non-null  int64  
 11  Food and drink                    103904 non-null  int64  
 12  Online boarding                   103904 non-null  int64  
 13  Seat comfort                      103904 non-null  int64  
 14  Inflight entertainment            103904 non-null  int64  
 15  On-board service                  103904 non-null  int64  
 16  Leg room service                  103904 non-null  int64  
 17  Baggage handling                  103904 non-null  int64  
 18  Checkin service                   103904 non-null  int64  
 19  Inflight service                  103904 non-null  int64  
 20  Cleanliness                       103904 non-null  int64  
 21  Departure Delay in Minutes        103904 non-null  int64  
 22  Arrival Delay in Minutes          103594 non-null  float64
 23  satisfaction                      103904 non-null  object 
dtypes: float64(1), int64(18), object(5)
memory usage: 19.0+ MB
```

There are 103904 observations or instances in the dataset.The "Arrival Delay in Minutes" column is missing some values. To be precise, it is missing 103904 - 103594 (310) values

In [5]:
```
df_data.describe().T        #Statistics for the numerical columns
```

Out[5]:

	count	mean	std	min	25%	50%	75%	max
id	103904.0	64924.210502	37463.812252	1.0	32533.75	64856.5	97368.25	129880.0
Age	103904.0	39.379706	15.114964	7.0	27.00	40.0	51.00	85.0
Flight Distance	103904.0	1189.448375	997.147281	31.0	414.00	843.0	1743.00	4983.0

	count	mean	std	min	25%	50%	75%	max
Inflight wifi service	103904.0	2.729683	1.327829	0.0	2.00	3.0	4.00	5.0
Departure/Arrival time convenient	103904.0	3.060296	1.525075	0.0	2.00	3.0	4.00	5.0
Ease of Online booking	103904.0	2.756901	1.398929	0.0	2.00	3.0	4.00	5.0
Gate location	103904.0	2.976883	1.277621	0.0	2.00	3.0	4.00	5.0
Food and drink	103904.0	3.202129	1.329533	0.0	2.00	3.0	4.00	5.0
Online boarding	103904.0	3.250375	1.349509	0.0	2.00	3.0	4.00	5.0
Seat comfort	103904.0	3.439396	1.319088	0.0	2.00	4.0	5.00	5.0
Inflight entertainment	103904.0	3.358158	1.332991	0.0	2.00	4.0	4.00	5.0
On-board service	103904.0	3.382363	1.288354	0.0	2.00	4.0	4.00	5.0
Leg room service	103904.0	3.351055	1.315605	0.0	2.00	4.0	4.00	5.0
Baggage handling	103904.0	3.631833	1.180903	1.0	3.00	4.0	5.00	5.0
Checkin service	103904.0	3.304290	1.265396	0.0	3.00	3.0	4.00	5.0
Inflight service	103904.0	3.640428	1.175663	0.0	3.00	4.0	5.00	5.0
Cleanliness	103904.0	3.286351	1.312273	0.0	2.00	3.0	4.00	5.0
Departure Delay in Minutes	103904.0	14.815618	38.230901	0.0	0.00	0.0	12.00	1592.0
Arrival Delay in Minutes	103594.0	15.178678	38.698682	0.0	0.00	0.0	13.00	1584.0

Missing Records

In [6]:
```
df_data.isnull().sum()       #To check if there are any missing values in the columns
```

Out[6]:
```
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           310
satisfaction                         0
dtype: int64
```

From the above statistics we can see that there are 310 Missing records in the "Arrival Delay in Minutes" column. We need to replace these missing records with Non Null values. We are using SimpleImputer to fill out the missing records.

In [7]:
```
# from sklearn.impute import SimpleImputer
# import numpy as np
# imputation_mean = SimpleImputer(strategy = 'mean')
# imputation_mean.fit(df_data[['Arrival Delay in Minutes']])
# imputed_df_data = imputation_mean.transform(df_data[['Arrival Delay in Minutes']])
# df_data['Arrival Delay in Minutes'] = imputed_df_data
```

In [8]:
```
import numpy as np
import pandas as pd
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=2, weights="uniform")
data = imputer.fit_transform(df_data['Arrival Delay in Minutes'].values.reshape(-1,1))
df_data['Arrival Delay in Minutes'] = data
print(df_data['Arrival Delay in Minutes'])
```

```
0         18.0
1          6.0
2          0.0
3          9.0
4          0.0
          ...
103899     0.0
103900     0.0
103901    14.0
103902     0.0
103903     0.0
Name: Arrival Delay in Minutes, Length: 103904, dtype: float64
```

In [9]:
```
df_data.isnull().sum()                #To validate that all the missing values are fill
```

Out[9]:
```
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
```

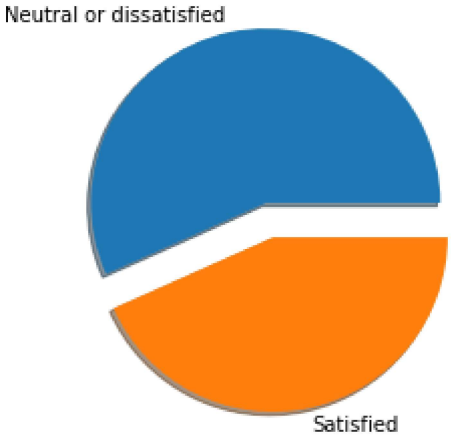
In [10]:
```
df_data['satisfaction'].value_counts()
```

Out[10]:
```
neutral or dissatisfied    58879
satisfied                  45025
Name: satisfaction, dtype: int64
```

From the above statistics we can assume that a random guess will give us around 58% of customer to be not satisfied with the airline. Hence, the predictive model should predict with an accuracy greater than 59%.

Data Visualization

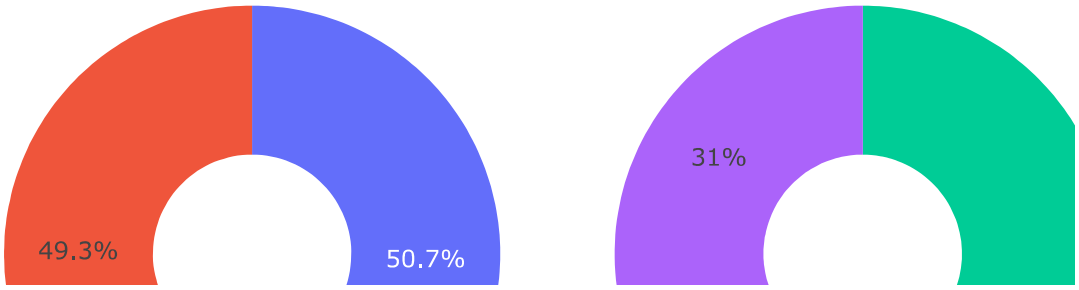
In [11]:
```
import matplotlib.pyplot as plt
myexplode = [0.2, 0]
plt.pie(df_data.satisfaction.value_counts(), labels = ["Neutral or dissatisfied", "Sati
plt.show()
```



As we can see from the above pie chart that the target variable, Satisfaction has approximately equal records containing satisfied and not satisfied customer.

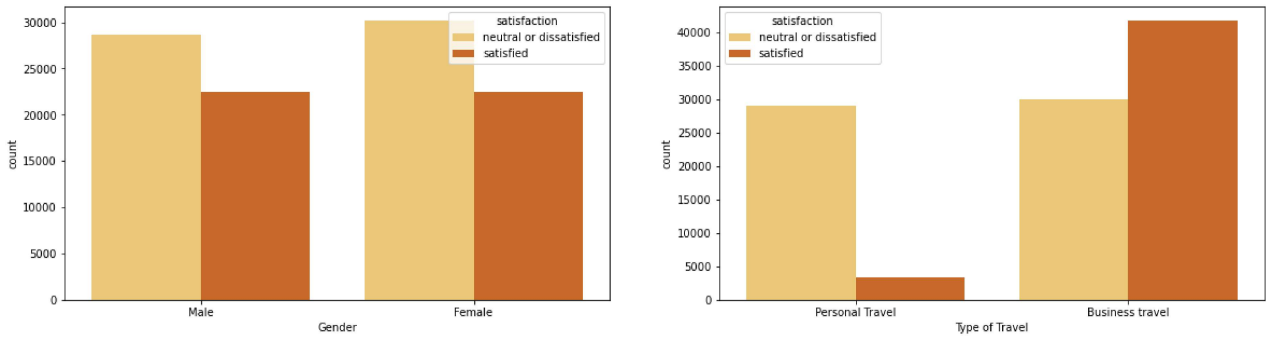
In [12]:
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig = make_subplots(rows=1, cols=3, specs=[[{'type':'domain'},{'type':'domain'},{'type'
fig.add_trace(go.Pie(labels = ["Male","Female"], values=df_data['Gender'].value_counts(
fig.add_trace(go.Pie(labels = ["Personal Travel","Business Travel"], values=df_data['Ty
fig.add_trace(go.Pie(labels = ["Loyal Customer","Disloyal Customer"], values=df_data['C
fig.update_traces(hole=.4, )
fig.update_layout(title_text="Donut chart Subplots",)
fig.show()
print(df_data['Gender'].value_counts())
print('----------------------------------------')
print(df_data['Type of Travel'].value_counts())
print('----------------------------------------')
print(df_data['Customer Type'].value_counts())
```



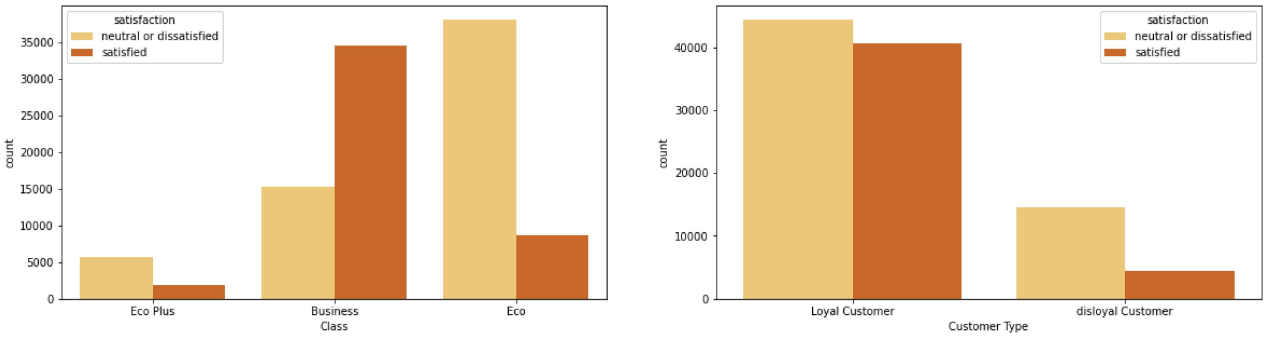

```
Female    52727
Male      51177
Name: Gender, dtype: int64
--------------------------------------
Business travel    71655
Personal Travel    32249
Name: Type of Travel, dtype: int64
--------------------------------------
Loyal Customer       84923
disloyal Customer    18981
Name: Customer Type, dtype: int64
```

In [13]:
```
import seaborn as sns
f, ax = plt.subplots(1, 2, figsize = (20,5))
sns.countplot(x = df_data['Gender'], hue = df_data['satisfaction'], palette = "YlOrBr",
sns.countplot(x = df_data['Type of Travel'], hue = df_data['satisfaction'], palette = "
plt.show()
```



From the above charts, we can conclude that The satisfaction and dissatisfaction of Males as compared with Females is almost similar. We can also see that the customers travelling for Business purpose seem to be more satisfied when compared with the customers travelling for Personal vacation.

In [14]:
```
import seaborn as sns
f, ax = plt.subplots(1, 2, figsize = (20,5))
sns.countplot(x = df_data['Class'], hue = df_data['satisfaction'], palette = "YlOrBr",d
sns.countplot(x = df_data['Customer Type'], hue = df_data['satisfaction'], palette = "Y
plt.show()
```



From the above charts we can conclude that the custoer travelling through Business Class are not satisfied when compared with the customer travelling through Economy or Economy Plus class. In the second chart we can see that the dataset consists more number of records for the Loyal customers than the disloyal customer.

Converting the Categorical variable to numericals

In [15]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
labelmap = {}
for column in ['Gender', 'Customer Type', 'Type of Travel', 'Class', 'satisfaction']:
    df_data[column] = le.fit_transform(df_data[column])
    labelmap[column] = dict(zip(le.classes_,range(len(le.classes_))))
print(f"Label Mapping: {labelmap}")
df_data.head()
```

```
Label Mapping: {'Gender': {'Female': 0, 'Male': 1}, 'Customer Type': {'Loyal Customer':
0, 'disloyal Customer': 1}, 'Type of Travel': {'Business travel': 0, 'Personal Travel':
1}, 'Class': {'Business': 0, 'Eco': 1, 'Eco Plus': 2}, 'satisfaction': {'neutral or diss
atisfied': 0, 'satisfied': 1}}
```

Out[15]:

	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	...	e
0	70172	1	0	13	1	2	460	3	4	3	...	
1	5047	1	1	25	0	0	235	3	2	3	...	
2	110028	0	0	26	0	0	1142	2	2	2	...	
3	24026	0	0	25	0	0	562	2	5	5	...	
4	119299	1	0	61	0	0	214	3	3	3	...	

5 rows × 24 columns

Scaling and Feature Selection Using Boruta

In [16]:
```
X = df_data.drop(['satisfaction'], axis = 1)
y = df_data['satisfaction']
```

Feature selection is a fundamental step in Machine learning as we dispose of a bunch of features that aren't relevant and select only those which are relevant to the model. By removing the features that aren't useful for our model we simplify the problem as the noise is reduced from the data. The reason why many data scientists choose Boruta is because it works extremely well even without any specific input from the user.It was born as a package for R. A version of Boruta for Python called BorutaPy is used by us to implement the algorithm on our dataset.

In [17]:
```
from sklearn.ensemble import RandomForestRegressor
from boruta import BorutaPy

# let's initialize a RF model
model = RandomForestRegressor(n_estimators=100, max_depth=5, random_state=42)

# let's initialize Boruta
feat_selector = BorutaPy(verbose=2,estimator=model,n_estimators='auto',max_iter=10)  #

feat_selector.fit(np.array(X), np.array(y))

# print support and ranking for each feature
print("\n------Support and Ranking for each feature------")
for i in range(len(feat_selector.support_)):
    if feat_selector.support_[i]:
        print("Passes the test: ", X.columns[i],
              " - Ranking: ", feat_selector.ranking_[i])
    else:
       print("Doesn't pass the test: ",
             X.columns[i], " - Ranking: ", feat_selector.ranking_[i])
```

```
Iteration:      1 / 10
Confirmed:      0
Tentative:      23
Rejected:       0
Iteration:      2 / 10
Confirmed:      0
Tentative:      23
Rejected:       0
Iteration:      3 / 10
Confirmed:      0
Tentative:      23
Rejected:       0
Iteration:      4 / 10
Confirmed:      0
Tentative:      23
Rejected:       0
Iteration:      5 / 10
Confirmed:      0
Tentative:      23
Rejected:       0
```

```
Iteration:      6 / 10
Confirmed:      0
Tentative:      23
Rejected:       0
Iteration:      7 / 10
Confirmed:      0
Tentative:      23
Rejected:       0
Iteration:      8 / 10
Confirmed:      13
Tentative:      7
Rejected:       3
Iteration:      9 / 10
Confirmed:      13
Tentative:      7
Rejected:       3


BorutaPy finished running.

Iteration:      10 / 10
Confirmed:      13
Tentative:      5
Rejected:       3

------Support and Ranking for each feature------
Doesn't pass the test:  id  - Ranking:  2
Doesn't pass the test:  Gender  - Ranking:  6
Passes the test:  Customer Type  - Ranking:  1
Doesn't pass the test:  Age  - Ranking:  2
Passes the test:  Type of Travel  - Ranking:  1
Passes the test:  Class  - Ranking:  1
Passes the test:  Flight Distance  - Ranking:  1
Passes the test:  Inflight wifi service  - Ranking:  1
Doesn't pass the test:  Departure/Arrival time convenient  - Ranking:  2
Passes the test:  Ease of Online booking  - Ranking:  1
Passes the test:  Gate location  - Ranking:  1
Doesn't pass the test:  Food and drink  - Ranking:  6
Passes the test:  Online boarding  - Ranking:  1
Doesn't pass the test:  Seat comfort  - Ranking:  2
Passes the test:  Inflight entertainment  - Ranking:  1
Doesn't pass the test:  On-board service  - Ranking:  4
Doesn't pass the test:  Leg room service  - Ranking:  3
Doesn't pass the test:  Baggage handling  - Ranking:  2
Passes the test:  Checkin service  - Ranking:  1
Doesn't pass the test:  Inflight service  - Ranking:  4
Passes the test:  Cleanliness  - Ranking:  1
Passes the test:  Departure Delay in Minutes  - Ranking:  1
Passes the test:  Arrival Delay in Minutes  - Ranking:  1
```

From the above statistics we can see that 13 features have passed the test. As a result we retain these 13 variable to run our regression models. Now we remove all those variable who have ranking higher than 1

In [18]:
```
X  = df_data.drop(['satisfaction','id','Gender','Age','Departure/Arrival time convenien
                   'Seat comfort', 'On-board service', 'Leg room service', 'Baggage hand
```

In [19]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state
x_train.shape, x_test.shape
```

Out[19]:
```
((83123, 13), (20781, 13))
```

In [20]:
```
from sklearn.preprocessing import MinMaxScaler, MaxAbsScaler
scaler = MinMaxScaler()
x_train_sca = scaler.fit_transform(x_train)
x_test_sca = scaler.transform(x_test)
x_train_sca.shape, x_test_sca.shape
```

Out[20]:
```
((83123, 13), (20781, 13))
```

In [21]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import *
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from xgboost.sklearn import XGBClassifier
from sklearn.naive_bayes import GaussianNB

rf_clf = RandomForestClassifier()

svm_clf = SVC()
logisreg_clf = LogisticRegression()
GB_clf = GradientBoostingClassifier()
XGB_clf = XGBClassifier()
GNB_clf = GaussianNB()

clf_list = [rf_clf,svm_clf,logisreg_clf,GB_clf,XGB_clf,GNB_clf]
clf_name_list = ['random_forest','SupportVectorMachine','LogisticRegression','GradientB
                 'XGBoost','GaussianNaiveBayes']
```

In [22]:
```
for clf in clf_list:
    clf.fit(x_train_sca,y_train)

train_acc_list = []
test_acc_list = []

for clf,name in zip(clf_list,clf_name_list):

    y_pred_train = clf.predict(x_train_sca)
    y_pred_test = clf.predict(x_test_sca)

    print('**************************************************************************
    print(name,': \n')


    print(classification_report(y_test, y_pred_test,target_names=['neutral or dissatisf
    train_acc_list.append(accuracy_score(y_train, y_pred_train))
    test_acc_list.append(accuracy_score(y_test, y_pred_test))

    mse = mean_squared_error(y_test, y_pred_test)
    mae = mean_absolute_error(y_test, y_pred_test)
    r2 = clf.score(x_test_sca, y_test)

    print('MSE for',name,'is : ',mse)
    print('MAE for',name,'is : ',mae)
    print('R_Square for',name,'is : ',r2)
```

```
**********************************************************************************
********************
random_forest :
```

```
                           precision    recall  f1-score   support

neutral or dissatisfaction       0.95      0.96      0.96     11713
              satisfaction       0.95      0.93      0.94      9068

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

MSE for random_forest is :  0.050719407150762716
MAE for random_forest is :  0.050719407150762716
R_Square for random_forest is :  0.9492805928492373
********************************************************************************
********************
SupportVectorMachine :

                           precision    recall  f1-score   support

neutral or dissatisfaction       0.94      0.96      0.95     11713
              satisfaction       0.95      0.92      0.93      9068

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

MSE for SupportVectorMachine is :  0.057263846783119195
MAE for SupportVectorMachine is :  0.057263846783119195
R_Square for SupportVectorMachine is :  0.9427361532168808
********************************************************************************
********************
LogisticRegression :

                           precision    recall  f1-score   support

neutral or dissatisfaction       0.87      0.89      0.88     11713
              satisfaction       0.85      0.84      0.84      9068

                  accuracy                           0.87     20781
                 macro avg       0.86      0.86      0.86     20781
              weighted avg       0.87      0.87      0.87     20781

MSE for LogisticRegression is :  0.13454597950050526
MAE for LogisticRegression is :  0.13454597950050526
R_Square for LogisticRegression is :  0.8654540204994947
********************************************************************************
********************
GradientBoosting :

                           precision    recall  f1-score   support

neutral or dissatisfaction       0.94      0.96      0.95     11713
              satisfaction       0.94      0.92      0.93      9068

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

MSE for GradientBoosting is :  0.060391703960348396
MAE for GradientBoosting is :  0.060391703960348396
R_Square for GradientBoosting is :  0.9396082960396516
********************************************************************************
********************
XGBoost :
```

```
                           precision    recall  f1-score   support

neutral or dissatisfaction       0.95      0.97      0.96     11713
              satisfaction       0.96      0.93      0.95      9068

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

MSE for XGBoost is :  0.04696597853808768
MAE for XGBoost is :  0.04696597853808768
R_Square for XGBoost is :  0.9530340214619123
**********************************************************************************
**********
GaussianNaiveBayes :

                           precision    recall  f1-score   support

neutral or dissatisfaction       0.87      0.88      0.88     11713
              satisfaction       0.84      0.84      0.84      9068

                  accuracy                           0.86     20781
                 macro avg       0.86      0.86      0.86     20781
              weighted avg       0.86      0.86      0.86     20781

MSE for GaussianNaiveBayes is :  0.14032048505846687
MAE for GaussianNaiveBayes is :  0.14032048505846687
R_Square for GaussianNaiveBayes is :  0.8596795149415332
```

In [23]:

```
XGB_clf1 = XGBClassifier(booster='gblinear')
XGB_clf1.fit(x_train_sca,y_train)
y_pred_test = XGB_clf1.predict(x_test_sca)
print(classification_report(y_test, y_pred_test,target_names=['neutral or dissatisfacti
mse = mean_squared_error(y_test, y_pred_test)
mae = mean_absolute_error(y_test, y_pred_test)
r2 = XGB_clf1.score(x_test_sca, y_test)

print('MSE for',name,'is : ',mse)
print('MAE for',name,'is : ',mae)
print('R_Square for',name,'is : ',r2)

print('**********************************************************************************
coef_table = pd.DataFrame(list(x_train.columns)).copy()
coef_table.insert(len(coef_table.columns),"Coefs",XGB_clf1.coef_)

print(coef_table)
```

```
                           precision    recall  f1-score   support

neutral or dissatisfaction       0.88      0.88      0.88     11713
              satisfaction       0.85      0.84      0.84      9068

                  accuracy                           0.86     20781
                 macro avg       0.86      0.86      0.86     20781
              weighted avg       0.86      0.86      0.86     20781

MSE for GaussianNaiveBayes is :  0.1366151773254415
MAE for GaussianNaiveBayes is :  0.1366151773254415
R_Square for GaussianNaiveBayes is :  0.8633848226745585
**********************************************************************************
**********
                          0      Coefs
0             Customer Type  -1.755630
1            Type of Travel  -2.593970
```

```
2                       Class  -1.655410
3             Flight Distance  -0.124878
4        Inflight wifi service   2.425730
5       Ease of Online booking  -0.959986
6                 Gate location  -0.482782
7             Online boarding   2.457350
8       Inflight entertainment   2.008540
9             Checkin service   1.856430
10                Cleanliness   0.282083
11  Departure Delay in Minutes   5.125770
12    Arrival Delay in Minutes -12.127200
```

Conclusion

The beta coefficient is the degree of change in the outcome variable for every 1-unit of change in the predictor variable. If the beta coefficient is positive, the interpretation is that for every 1-unit increase in the predictor variable, the outcome variable will increase by the beta coefficient value. If the beta coefficient is negative, the interpretation is that for every 1-unit increase in the predictor variable, the outcome variable will decrease by the beta coefficient value.

From our analysis the factors which majorly affects the satisfaction rate are: Inflight wifi service, Online Boarding, Inflight entertainment, Checkin service, Cleanliness, Departure Delay in Minutes.

In [ ]: