```
#reading our dataset
import pandas as pd
df = pd.read_csv('/content/airline customer satisfaction.csv')
```

```
Start coding or generate with AI.
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn import ensemble
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

```
df
```

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	...	Inflight entertainment	s
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	
3	3	77959	Male	Loyal Customer	44	Business travel	Business	3377	0	0	...	1	
4	4	36875	Female	Loyal Customer	49	Business travel	Eco	1182	2	3	...	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	
25971	25971	78463	Male	disloyal Customer	34	Business travel	Business	526	3	3	...	4	
25972	25972	71167	Male	Loyal Customer	23	Business travel	Business	646	4	4	...	4	
25973	25973	37675	Female	Loyal Customer	17	Personal Travel	Eco	828	2	5	...	2	
25974	25974	90086	Male	Loyal Customer	14	Business travel	Business	1127	3	3	...	4	
25975	25975	34799	Female	Loyal Customer	42	Personal Travel	Eco	264	2	5	...	1	

25976 rows × 25 columns

```
df.shape #shows the number of rows and columns in the dataset
```

```
(25976, 25)
```

```
df.shape[0]  #there are 25976 rows in the dataset
```

```
25976
```

```
df.shape[1]   #there are 25 columns in the dataset
```

```
25
```

```
df.columns.tolist() #shows a list of all columns in the dataset
```

```
['Unnamed: 0',
 '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']
```

```
df.size  #total number of cells in the dataset
```

```
649400
```

```
list(df.select_dtypes(include = object).columns)   #list of categorical variables
```

```
['Gender', 'Customer Type', 'Type of Travel', 'Class', 'satisfaction']
```

```
list(df.select_dtypes(exclude = object).columns)    #list of numerical variables
```

```
['Unnamed: 0',
 'id',
 'Age',
 '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']
```

```
target_variable = df['satisfaction']   #defines the target or output variable
```

```
input_variables = list(set(df.columns) - set(df['satisfaction']))   #defines the input variables
```

```
input_variables
```

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

```
df.dtypes  #datatypes of columns
```

```
Unnamed: 0                             int64
id                                     int64
Gender                                object
Customer Type                         object
Age                                    int64
Type of Travel                        object
Class                                 object
Flight Distance                        int64
Inflight wifi service                  int64
Departure/Arrival time convenient      int64
Ease of Online booking                 int64
Gate location                          int64
Food and drink                         int64
Online boarding                        int64
Seat comfort                           int64
Inflight entertainment                 int64
On-board service                       int64
Leg room service                       int64
Baggage handling                       int64
Checkin service                        int64
Inflight service                       int64
Cleanliness                            int64
Departure Delay in Minutes             int64
Arrival Delay in Minutes             float64
satisfaction                          object
dtype: object
```

```
df.info() #datatypes
```

```
<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
```

```
df.isna().sum()    #checking for missing values by column
```

```
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
```

```
for col in df.columns:
  print(col, df[col].isnull().sum())   #another way to check missing values by column
```

```
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
```

You may notice the following:

The column corresponding to the Arrival Delay in Minutes feature has 83 missing values. The id and unnamed: 0 columns are useless and will not affect the classification, in which would be later dropped.

```
#imputing missing values
numerical_variables = list(df.select_dtypes(exclude = object).columns)
df[numerical_variables] = df[numerical_variables].fillna(df[numerical_variables].median(), inplace = False)  #filling in mis

categorical_variables = list(df.select_dtypes(include = object).columns)
df[categorical_variables] = df[categorical_variables].fillna(df[categorical_variables].mode(), inplace = False) #filling in r
```

```
df.isna().sum()   #missing values imputed
```

```
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
```

```
descriptive_statistics = df.describe().T #this shows the descriptive statistics for all numerical variables
descriptive_statistics
```

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	25976.0	12987.500000	7498.769632	0.0	6493.75	12987.5	19481.25	25975.0
id	25976.0	65005.657992	37611.526647	17.0	32170.50	65319.5	97584.25	129877.0
Age	25976.0	39.620958	15.135685	7.0	27.00	40.0	51.00	85.0
Flight Distance	25976.0	1193.788459	998.683999	31.0	414.00	849.0	1744.00	4983.0
Inflight wifi service	25976.0	2.724746	1.335384	0.0	2.00	3.0	4.00	5.0
Departure/Arrival time convenient	25976.0	3.046812	1.533371	0.0	2.00	3.0	4.00	5.0
Ease of Online booking	25976.0	2.756775	1.412951	0.0	2.00	3.0	4.00	5.0
Gate location	25976.0	2.977094	1.282133	1.0	2.00	3.0	4.00	5.0
Food and drink	25976.0	3.215353	1.331506	0.0	2.00	3.0	4.00	5.0
Online boarding	25976.0	3.261665	1.355536	0.0	2.00	4.0	4.00	5.0
Seat comfort	25976.0	3.449222	1.320090	1.0	2.00	4.0	5.00	5.0
Inflight entertainment	25976.0	3.357753	1.338299	0.0	2.00	4.0	4.00	5.0
On-board service	25976.0	3.385664	1.282088	0.0	2.00	4.0	4.00	5.0
Leg room service	25976.0	3.350169	1.318862	0.0	2.00	4.0	4.00	5.0
Baggage handling	25976.0	3.633238	1.176525	1.0	3.00	4.0	5.00	5.0
Checkin service	25976.0	3.314175	1.269332	1.0	3.00	3.0	4.00	5.0
Inflight service	25976.0	3.649253	1.180681	0.0	3.00	4.0	5.00	5.0
Cleanliness	25976.0	3.286226	1.319330	0.0	2.00	3.0	4.00	5.0
Departure Delay in Minutes	25976.0	14.306090	37.423160	0.0	0.00	0.0	12.00	1128.0
Arrival Delay in Minutes	25976.0	14.693756	37.466787	0.0	0.00	0.0	13.00	1115.0

```
mode = df.mode().iloc[0,:]   #this shows the mode for each variable
mode
```

```
Unnamed: 0                                      0
id                                             17
Gender                                     Female
Customer Type                       Loyal Customer
Age                                          39.0
Type of Travel                    Business travel
Class                                    Business
Flight Distance                             337.0
Inflight wifi service                         2.0
Departure/Arrival time convenient             4.0
Ease of Online booking                        2.0
Gate location                                 3.0
Food and drink                                4.0
Online boarding                               4.0
Seat comfort                                  4.0
Inflight entertainment                        4.0
On-board service                              4.0
Leg room service                              4.0
Baggage handling                              4.0
Checkin service                               4.0
Inflight service                              4.0
Cleanliness                                   4.0
Departure Delay in Minutes                    0.0
Arrival Delay in Minutes                      0.0
satisfaction                neutral or dissatisfied
Name: 0, dtype: object
```

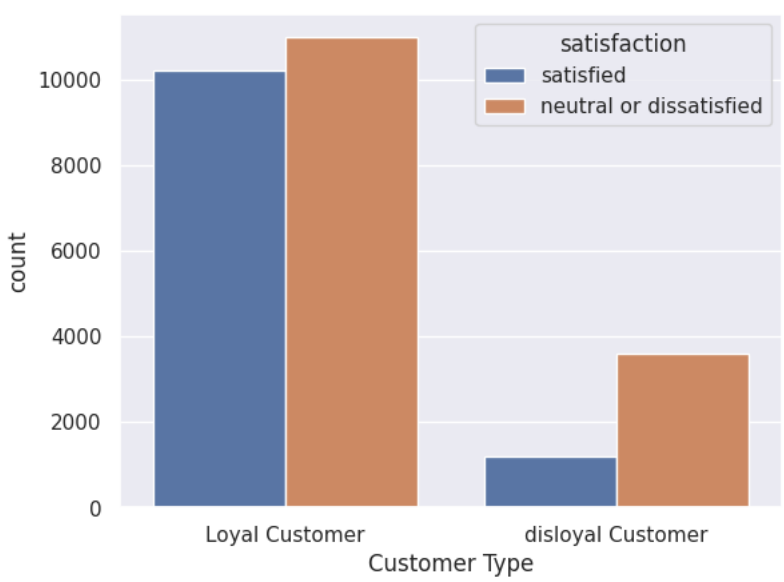
```
Correlation = df.corr()  #shows the correlation between the input variables as well as the correclation between all input va
Correlation
```

	Unnamed: 0	id	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding
Unnamed: 0	1.000000	0.006946	-0.007964	-0.002470	-0.003085	-0.003962	-0.002011	-0.001360	-0.010858	-0.000254
id	0.006946	1.000000	0.010211	0.095335	-0.030303	-0.002502	0.010389	0.001843	-0.006789	0.055767
Age	-0.007964	0.010211	1.000000	0.099409	0.009242	0.032449	0.013565	0.003308	0.023841	0.202058
Flight Distance	-0.002470	0.095335	0.099409	1.000000	0.005007	-0.014401	0.062989	0.008410	0.057322	0.214629
Inflight wifi service	-0.003085	-0.030303	0.009242	0.005007	1.000000	0.349137	0.710684	0.347790	0.122306	0.459366
Departure/Arrival time convenient	-0.003962	-0.002502	0.032449	-0.014401	0.349137	1.000000	0.440230	0.458439	-0.016006	0.080937
Ease of Online booking	-0.002011	0.010389	0.013565	0.062989	0.710684	0.440230	1.000000	0.465514	0.025141	0.408003
Gate location	-0.001360	0.001843	0.003308	0.008410	0.347790	0.458439	0.465514	1.000000	-0.009694	0.006993
Food and drink	-0.010858	-0.006789	0.023841	0.057322	0.122306	-0.016006	0.025141	-0.009694	1.000000	0.229601
Online boarding	-0.000254	0.055767	0.202058	0.214629	0.459366	0.080937	0.408003	0.006993	0.229601	1.000000
Seat comfort	-0.005602	0.049212	0.154507	0.158957	0.116991	-0.001926	0.022811	-0.000721	0.580970	0.415414
Inflight entertainment	-0.012927	-0.001078	0.068998	0.137538	0.201782	-0.022326	0.044715	-0.000340	0.627265	0.279391
On-board service	0.001180	0.056544	0.054977	0.117880	0.113658	0.060982	0.039988	-0.031606	0.050688	0.149430
Leg room service	-0.004194	0.041921	0.033299	0.136995	0.159699	0.003373	0.116754	-0.002428	0.035870	0.120354
Baggage handling	0.000349	0.073085	-0.049863	0.071549	0.118199	0.065684	0.040685	-0.004395	0.037617	0.084583
Checkin service	0.002773	0.079521	0.025388	0.075720	0.046046	0.082461	-0.000108	-0.054954	0.076775	0.203309
Inflight service	-0.008069	0.076587	-0.059083	0.066355	0.108419	0.067804	0.035769	-0.005129	0.039992	0.071972
Cleanliness	-0.005058	0.020411	0.048418	0.105578	0.125768	-0.007670	0.010974	-0.014195	0.659253	0.320912
Departure Delay in Minutes	-0.005948	-0.009884	-0.004334	0.003446	-0.010078	-0.000238	-0.001062	0.008039	-0.025957	-0.021037
Arrival Delay in Minutes	-0.002951	-0.028862	-0.007359	0.000117	-0.012553	-0.001442	-0.003371	0.007559	-0.028392	-0.025841

```
# outliers = [] # Calculate the IQR for the current column
# threshold = 1.5
# i = ()
# for i in i:
 # Q1 = df[i].quantile(0.25)
 # Q3 = df[i].quantile(0.75)
 # IQR = Q3 - Q1
 # outliers.append(((df[i] < (Q1 - threshold * IQR)) | (df[i] > (Q3 + threshold * IQR))))
```

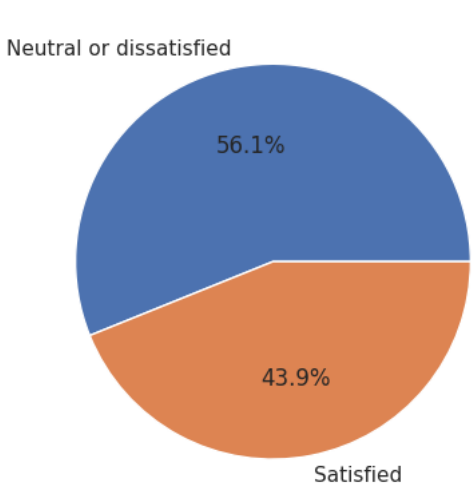
```
sns.set(style = 'darkgrid' )
ax = sns.countplot(x= 'Customer Type', hue = 'satisfaction', data = df)

# data visualisation in the form of countplot, using the customertype and satisfaction variable
```



helps us determine Customer Type based on satisfaction.

```
plt.pie(df.satisfaction.value_counts(), labels = ["Neutral or dissatisfied", "Satisfied"], colors = sns.color_palette(), auto
pass
```



Based on the data presented in the pie chart, 56.7% is not satisfied and 43.3% is satisfied.

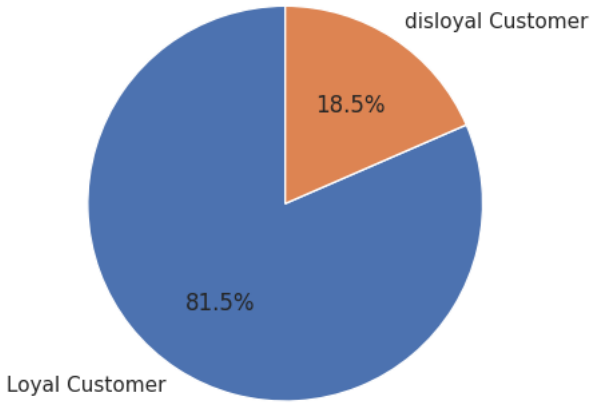
```
df
```

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	...	Inflight entertainment	s
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	
3	3	77959	Male	Loyal Customer	44	Business travel	Business	3377	0	0	...	1	
4	4	36875	Female	Loyal Customer	49	Business travel	Eco	1182	2	3	...	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	
25971	25971	78463	Male	disloyal Customer	34	Business travel	Business	526	3	3	...	4	
25972	25972	71167	Male	Loyal Customer	23	Business travel	Business	646	4	4	...	4	
25973	25973	37675	Female	Loyal Customer	17	Personal Travel	Eco	828	2	5	...	2	
25974	25974	90086	Male	Loyal Customer	14	Business travel	Business	1127	3	3	...	4	
25975	25975	34799	Female	Loyal Customer	42	Personal Travel	Eco	264	2	5	...	1	

25976 rows × 25 columns

```
plt.tight_layout()
fig1, ax1 = plt.subplots()
x = list(df['Customer Type'].value_counts().index)
y = list(df['Customer Type'].value_counts())
ax1.pie(y, labels = x, autopct = '%1.1f%%', startangle = 90)
plt.show
```

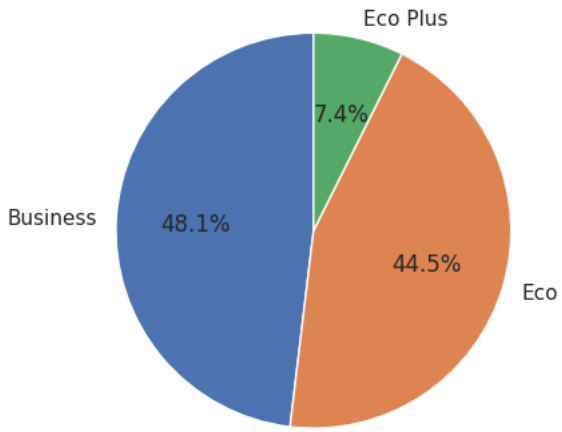
```
<function matplotlib.pyplot.show(close=None, block=None)>
<Figure size 640x480 with 0 Axes>
```



Based on the data presented in the pie chart, 81.5% are loyal customers and 18.5% are not loyal.

```
plt.tight_layout()
fig1, ax1 = plt.subplots()
x = list(df['Class'].value_counts().index)
y = list(df['Class'].value_counts())
ax1.pie(y, labels = x, autopct = '%1.1f%%', startangle = 90)
plt.show
```

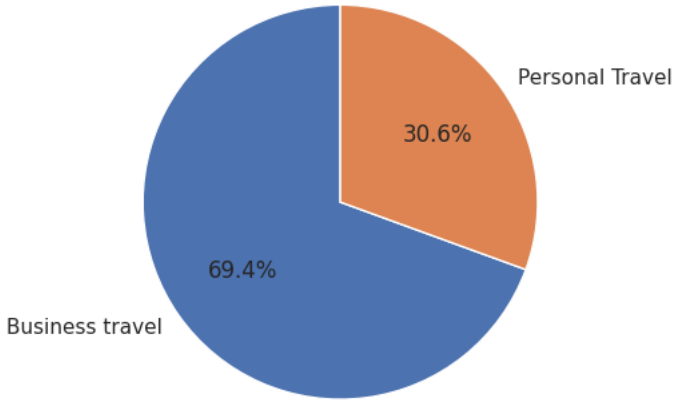
```
<function matplotlib.pyplot.show(close=None, block=None)>
<Figure size 640x480 with 0 Axes>
```



The pie chat illustrates the percentage of customers who prefer a given class

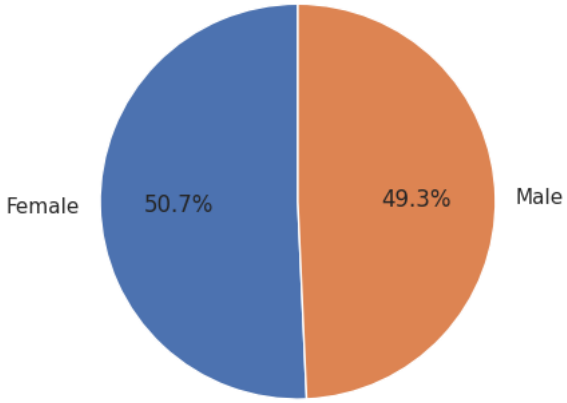
```
plt.tight_layout()
fig1, ax1 = plt.subplots()
x = list(df['Type of Travel'].value_counts().index)
y = list(df['Type of Travel'].value_counts())
ax1.pie(y, labels = x, autopct = '%1.1f%%', startangle = 90)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
<Figure size 640x480 with 0 Axes>
```



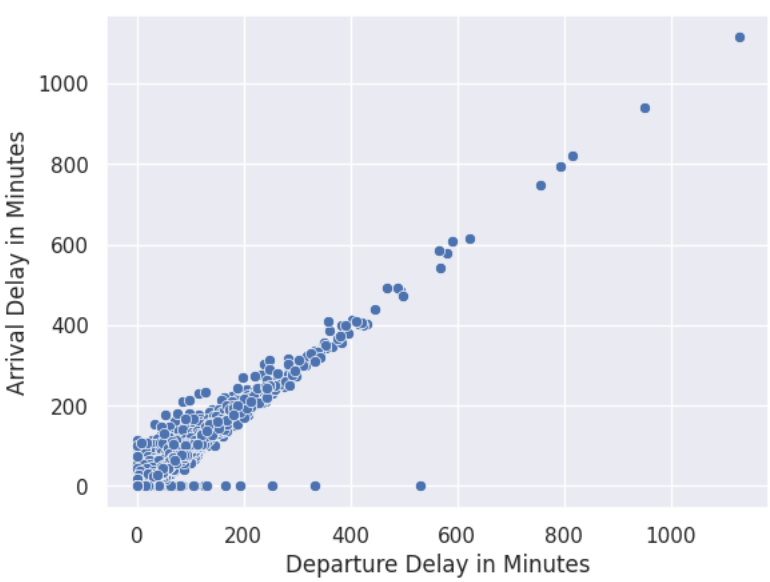

```
plt.tight_layout()
fig1, ax1 = plt.subplots()
x = list(df['Gender'].value_counts().index)
y = list(df['Gender'].value_counts())
ax1.pie(y, labels = x, autopct = '%1.1f%%', startangle = 90)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
<Figure size 640x480 with 0 Axes>
```



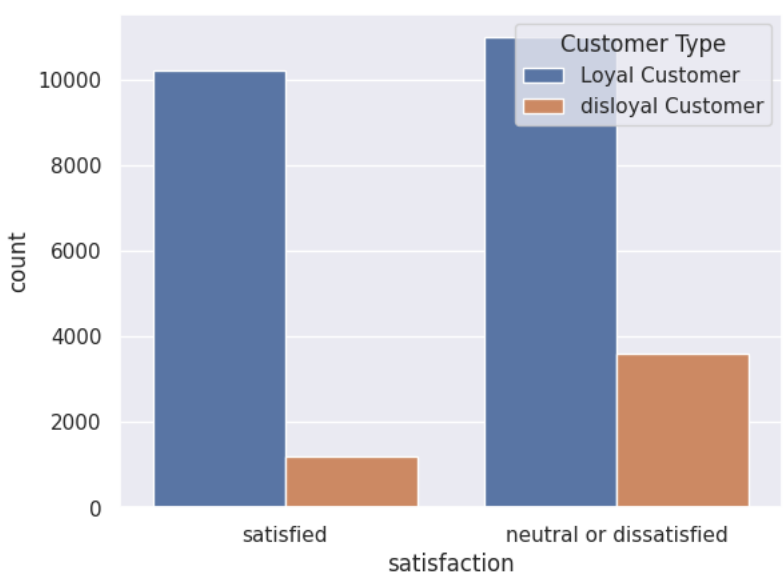
Shows the percentage of males and females

```
ax = sns.scatterplot(x = 'Departure Delay in Minutes', y = 'Arrival Delay in Minutes', data = df)
```



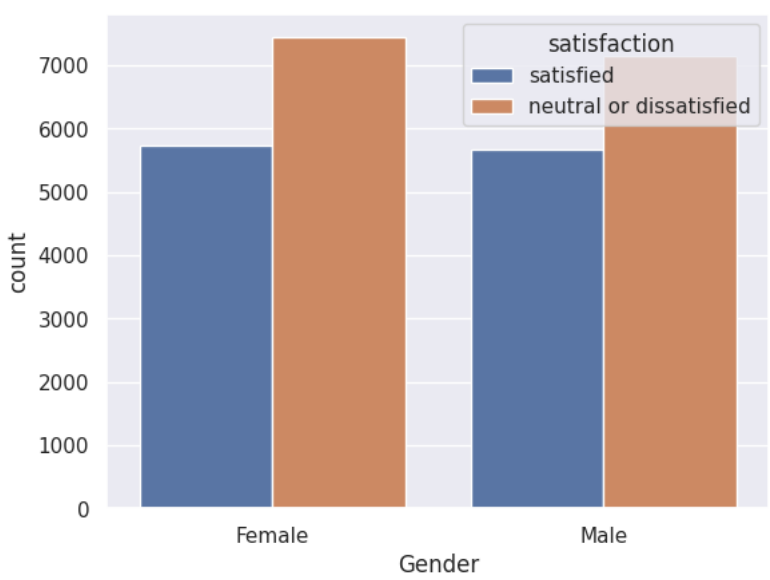
shows the correlation between departure delay and arrival delay. We can say the two variabeles are correlated(an increase in departure delay leads an increase in arrival delay) eventhough there are a few exceptions

```
ax = sns.countplot(x = 'satisfaction', hue = 'Customer Type', data = df)
```



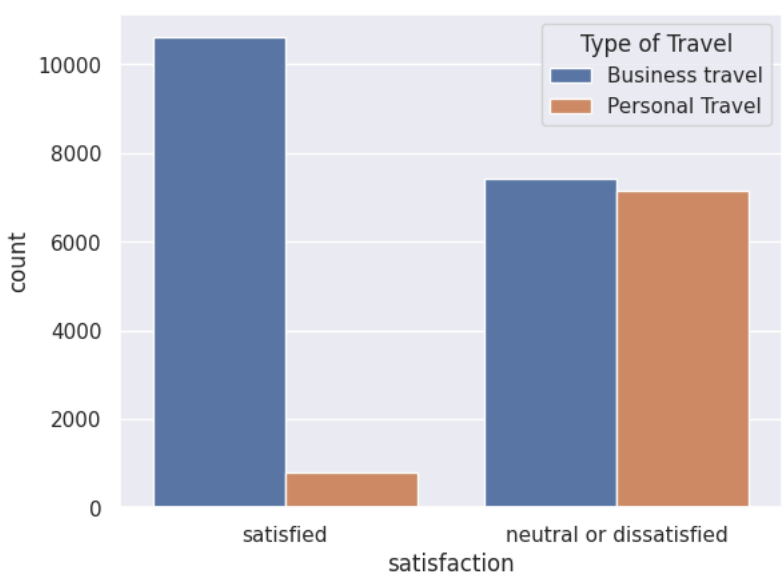
we can see that satisfied customers are more likely to be loyal than neutral or dissatisfied customers

```
ax = sns.countplot(x = 'Gender', hue = 'satisfaction', data = df)
```



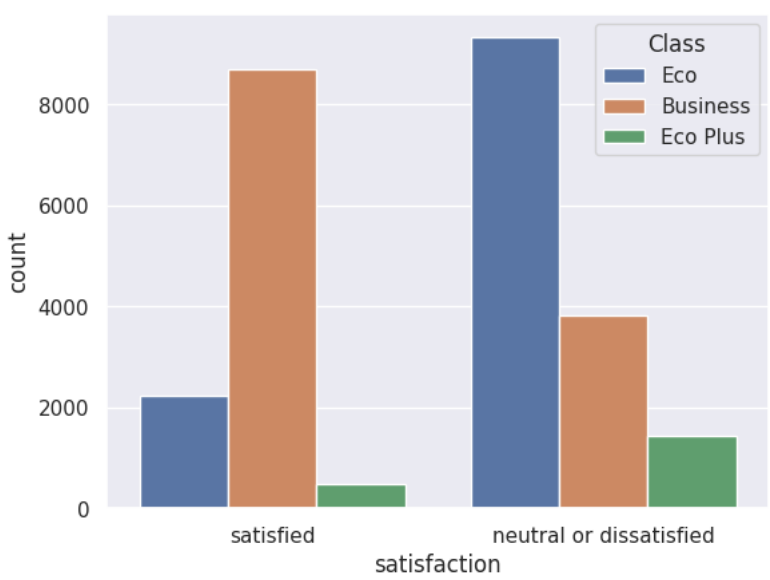
This shows the distribution of satisfaction over the poulation based on gender. The gender category is somewhat equally distributed and therefore has no significant impact on satisfaction.

```
ax = sns.countplot(x = 'satisfaction', hue = 'Type of Travel', data = df)
```



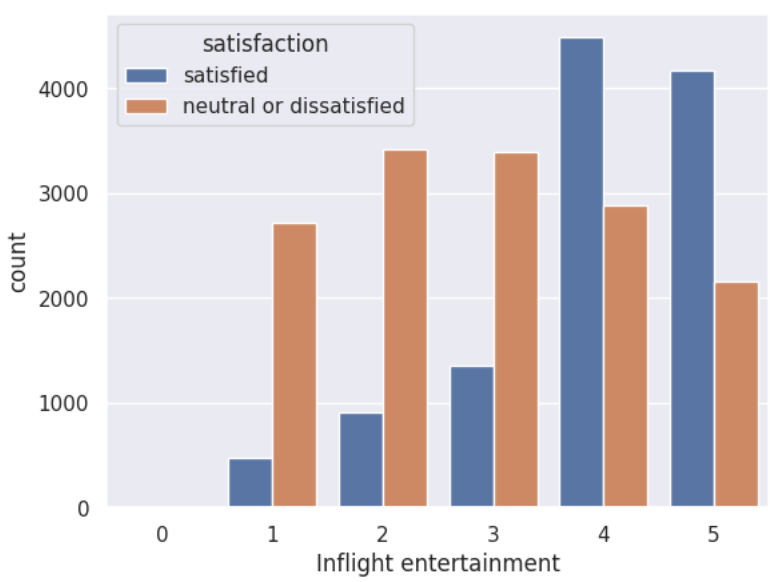
Customers travelling for business purposes tend to be satisfied

```
ax = sns.countplot(x = 'satisfaction', hue = 'Class', data = df)
```

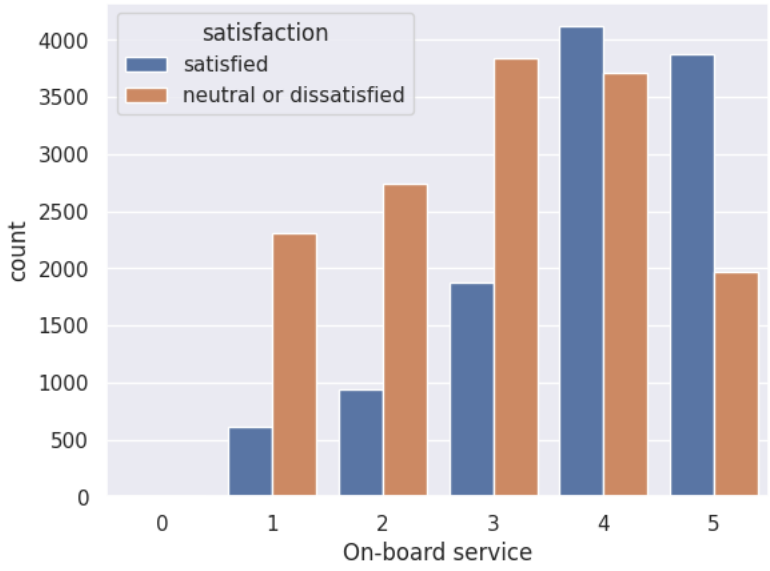


Customers traveling in business class appear to be more satisfied than the customers traveling in economy or economy plus class.

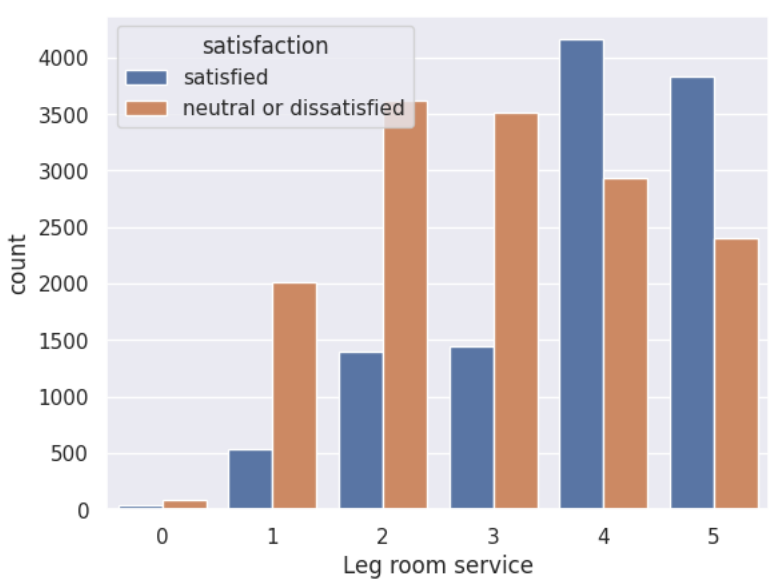
```
ax = sns.countplot(x = 'Inflight entertainment', hue = 'satisfaction', data = df)
```



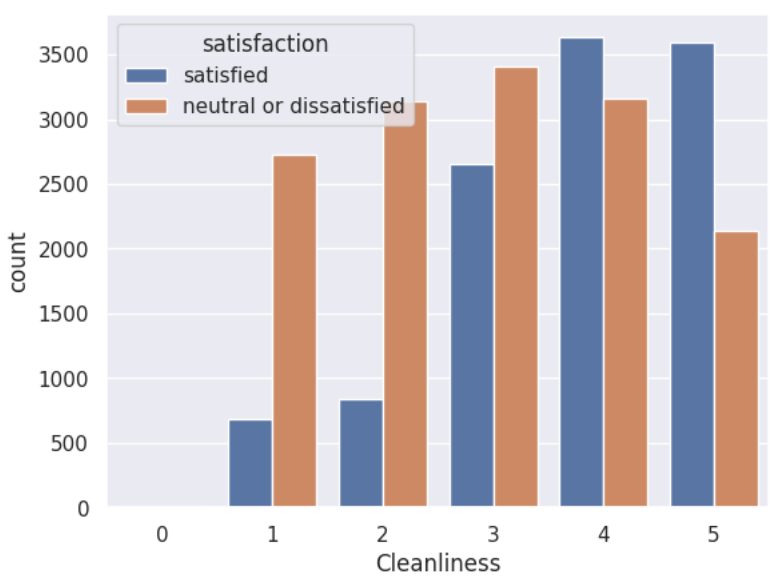

```
ax = sns.countplot(x = 'On-board service', hue = 'satisfaction', data = df)
```



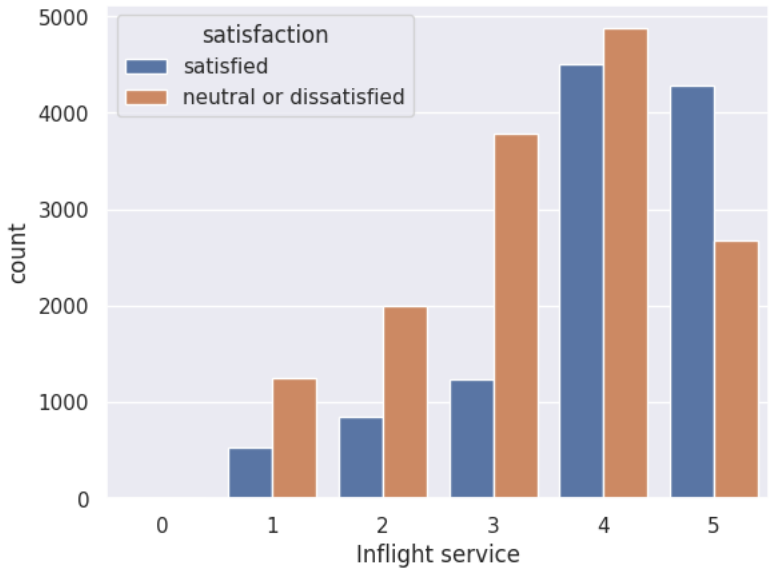

```
ax = sns.countplot(x = 'Leg room service', hue = 'satisfaction', data = df)
```



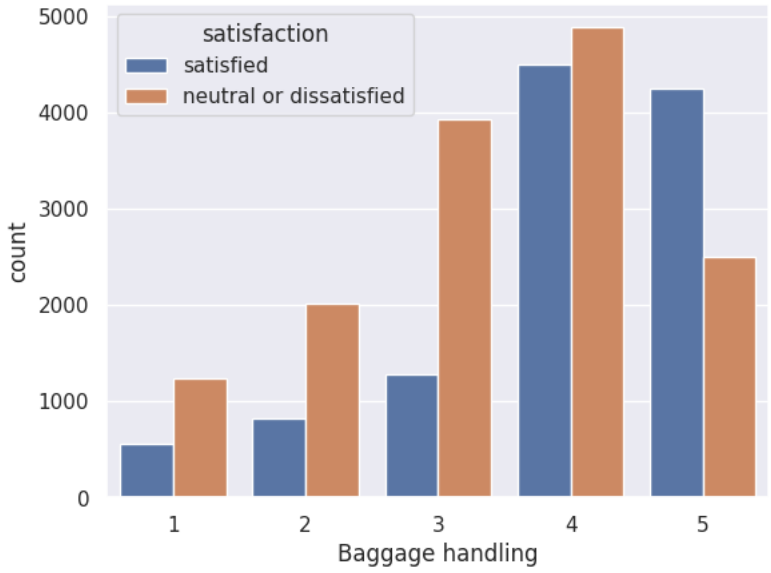

```
ax = sns.countplot(x = 'Cleanliness', hue = 'satisfaction', data = df)
```




```
ax = sns.countplot(x = 'Inflight service', hue = 'satisfaction', data = df)
```




```
ax = sns.countplot(x = 'Baggage handling', hue = 'satisfaction', data = df)
```




```
numerical_variables =  df.select_dtypes(exclude='object').columns.tolist()
categorical_variables =  list(set(df.dtypes[df.dtypes == 'object'].index) - set(['satisfaction']))
```

```
df = df.drop('Unnamed: 0', axis = 1) #dropping the Unnamed: 0 column because it is not needed
```

```
df = df.drop('id', axis = 1) #dropping the id column because it is not needed
```

```
df.std()
```

```
Age                                 15.135685
Flight Distance                    998.683999
Inflight wifi service                1.335384
Departure/Arrival time convenient    1.533371
Ease of Online booking               1.412951
Gate location                        1.282133
Food and drink                       1.331506
Online boarding                      1.355536
Seat comfort                         1.320090
Inflight entertainment               1.338299
On-board service                     1.282088
Leg room service                     1.318862
Baggage handling                     1.176525
Checkin service                      1.269332
Inflight service                     1.180681
Cleanliness                          1.319330
Departure Delay in Minutes          37.423160
Arrival Delay in Minutes            37.466787
dtype: float64
```

```
numerical_variables =  df.select_dtypes(exclude='object').columns.tolist()
zero_variance_numerical_variables = []   #checking for numerical variables with zero variance
for variable in numerical_variables:
  if df[variable].std() == 0:
    zero_variance_numerical_variables.append(variable)
```

```
zero_variance_numerical_variables   #there are no zeo variance numerical variables
```

```
[]
```

```
categorical_variables =  list(set(df.dtypes[df.dtypes == 'object'].index) - set(['satisfaction']))  #checking for categorical
zero_variance_categorical_variables = []
for i in categorical_variables:
  if len(df[i].value_counts().index) == 1:
    zero_variance_categorical_variables.append(i)
zero_variance_categorical_variables                       #there are no zeo variance categorical variables
```

```
[]
```

```
categorical_variables = sorted(list(set(df.select_dtypes(include = object).columns) - set(['satisfaction'])))
high_cardinality_categorical_variables = []  #checking for categorical variables with high cardinality
for item in categorical_variables:
    if len(df[item].value_counts().index) > 200:
        high_cardinality_categorical_variables.append(item)
```

```
high_cardinality_categorical_variables   #there are no categorical variables with high cardinality
```

```
[]
```

```
df.columns
```

```
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',
       'satisfaction'],
      dtype='object')
```

Standardizing all numerical variables in the dataset

```
numerical_variables = list (df.select_dtypes(exclude = object).columns)
array = df[numerical_variables].values
data_scaler = StandardScaler().fit(array)
df[numerical_variables] = pd.DataFrame (data_scaler.transform(array))
```

```
df  #numerical variables has been transformed
```

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	Infl entertai
0	Female	Loyal Customer	0.817887	Business travel	Eco	-1.035171	1.703853	0.621641	0.172143	0.797831	...	1.2
1	Female	Loyal Customer	-0.239238	Business travel	Business	1.671443	-1.291598	-1.334871	0.172143	-1.542065	...	0.4
2	Male	disloyal Customer	-1.296363	Business travel	Eco	-1.003128	-0.542735	-1.987042	-0.535609	0.797831	...	-1.0
3	Male	Loyal Customer	0.289325	Business travel	Business	2.186131	-2.040460	-1.987042	-1.951114	-0.762100	...	-1.7
4	Female	Loyal Customer	0.619676	Business travel	Eco	-0.011804	-0.542735	-0.030530	0.879895	0.017866	...	-1.0
...	...	...	...	...	...	...	...	...	...	...	...	
25971	Male	disloyal Customer	-0.371378	Business travel	Business	-0.668681	0.206128	-0.030530	0.172143	-1.542065	...	0.4
25972	Male	Loyal Customer	-1.098152	Business travel	Business	-0.548521	0.954990	0.621641	0.879895	0.797831	...	0.4
25973	Female	Loyal Customer	-1.494573	Personal Travel	Eco	-0.366278	-0.542735	1.273812	-1.243362	1.577797	...	-1.0
25974	Male	Loyal Customer	-1.692784	Business travel	Business	-0.066878	0.206128	-0.030530	0.172143	0.017866	...	0.4
25975	Female	Loyal Customer	0.157184	Personal Travel	Eco	-0.931032	-0.542735	1.273812	-0.535609	1.577797	...	-1.7

25976 rows × 23 columns

```
categorical_variables = sorted(list(set(df.select_dtypes(include = object).columns) - set(['satisfaction'])))  #encoding cate
dummy_cat = pd.get_dummies(df[categorical_variables], drop_first = True)

df = df.drop(categorical_variables, axis = 1)
df = pd.concat([df, dummy_cat], axis = 1)

df   #categorical input variables encoded
```

	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Seat comfort	Inflight entertainment	...
0	0.817887	-1.035171	1.703853	0.621641	0.172143	0.797831	-0.161739	0.544692	-0.340303	1.227138	...
1	-0.239238	1.671443	-1.291598	-1.334871	0.172143	-1.542065	1.340348	0.544692	1.174774	0.479907	...
2	-1.296363	-1.003128	-0.542735	-1.987042	-0.535609	0.797831	-0.912783	-0.930768	-1.097842	-1.014556	...
3	0.289325	2.186131	-2.040460	-1.987042	-1.951114	-0.762100	-0.161739	0.544692	0.417235	-1.761787	...
4	0.619676	-0.011804	-0.542735	-0.030530	0.879895	0.017866	0.589304	-1.668498	-1.097842	-1.014556	...
...	...	...	...	...	...	...	...	...	...	...	...
25971	-0.371378	-0.668681	0.206128	-0.030530	0.172143	-1.542065	0.589304	-0.193038	0.417235	0.479907	...
25972	-1.098152	-0.548521	0.954990	0.621641	0.879895	0.797831	0.589304	0.544692	0.417235	0.479907	...
25973	-1.494573	-0.366278	-0.542735	1.273812	-1.243362	1.577797	-0.912783	-1.668498	-1.097842	-1.014556	...
25974	-1.692784	-0.066878	0.206128	-0.030530	0.172143	0.017866	0.589304	0.544692	0.417235	0.479907	...
25975	0.157184	-0.931032	-0.542735	1.273812	-0.535609	1.577797	0.589304	-0.930768	-1.097842	-1.761787	...

25976 rows × 24 columns

```
df['satisfaction'] = df['satisfaction'].replace('satisfied',1)  #encoding categorical output variable
df['satisfaction'] = df['satisfaction'].replace('neutral or dissatisfied',0)

df   #categorical output variable encoded
```

	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Seat comfort	Inflight entertainment	...
0	0.817887	-1.035171	1.703853	0.621641	0.172143	0.797831	-0.161739	0.544692	-0.340303	1.227138	...
1	-0.239238	1.671443	-1.291598	-1.334871	0.172143	-1.542065	1.340348	0.544692	1.174774	0.479907	...
2	-1.296363	-1.003128	-0.542735	-1.987042	-0.535609	0.797831	-0.912783	-0.930768	-1.097842	-1.014556	...
3	0.289325	2.186131	-2.040460	-1.987042	-1.951114	-0.762100	-0.161739	0.544692	0.417235	-1.761787	...
4	0.619676	-0.011804	-0.542735	-0.030530	0.879895	0.017866	0.589304	-1.668498	-1.097842	-1.014556	...
...	...	...	...	...	...	...	...	...	...	...	...
25971	-0.371378	-0.668681	0.206128	-0.030530	0.172143	-1.542065	0.589304	-0.193038	0.417235	0.479907	...
25972	-1.098152	-0.548521	0.954990	0.621641	0.879895	0.797831	0.589304	0.544692	0.417235	0.479907	...
25973	-1.494573	-0.366278	-0.542735	1.273812	-1.243362	1.577797	-0.912783	-1.668498	-1.097842	-1.014556	...
25974	-1.692784	-0.066878	0.206128	-0.030530	0.172143	0.017866	0.589304	0.544692	0.417235	0.479907	...
25975	0.157184	-0.931032	-0.542735	1.273812	-0.535609	1.577797	0.589304	-0.930768	-1.097842	-1.761787	...

25976 rows × 24 columns

```
#balancing the dataset
from imblearn.over_sampling import SMOTE
os = SMOTE(random_state = 0)
input_variables = list(set(df.columns) - set(['satisfaction']))
X, y = os.fit_resample(df[df.columns.difference(['satisfaction'])], df['satisfaction'])
X = pd.DataFrame(X, columns = input_variables)     # dataframe containing the input variables after balancing
y = pd.DataFrame(y, columns = ['satisfaction'])       # dataframe containing the output variable Attrition after balancing
df = pd.concat([X, y], axis = 1)
```

```
print(df['satisfaction'].value_counts())  #dataset balanced
```

```
1    14573
0    14573
Name: satisfaction, dtype: int64
```

```
#Splitting the dataset into training and testing
from sklearn.model_selection import train_test_split
input_var = list(set(df.columns) - set(['satisfaction']))
X = df[input_var]
y = df['satisfaction']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 42)
```

```
X_train   #training input variables containing 20402 observations
```

	Cleanliness	Food and drink	Departure/Arrival time convenient	Gate location	Inflight wifi service	Leg room service	Customer Type_disloyal Customer	Gender_Male	Age	Travel_P
10362	-0.216952	-0.161739	1.273812	1.577797	1.703853	-0.265514	0	1	-0.569589	
17941	0.541023	0.589304	-0.030530	0.017866	0.206128	0.492730	0	1	0.355395	
14762	1.298998	1.340348	-1.987042	-1.542065	1.703853	-0.265514	1	1	-1.758855	
20367	1.298998	1.340348	-0.682700	0.797831	-0.542735	-0.265514	1	1	-0.635660	
17646	0.541023	0.589304	-0.030530	0.017866	0.954990	0.492730	0	0	1.412520	
...	...	...	...	...	...	...	...	...	...	
21575	-1.732902	-0.161739	-0.030530	0.017866	0.206128	1.250974	0	0	1.016098	
5390	0.541023	0.589304	-0.682700	-0.762100	0.954990	0.492730	0	1	-0.239238	
860	0.541023	0.589304	1.273812	-0.762100	-0.542735	0.492730	0	0	1.148239	
15795	1.298998	1.340348	0.621641	-0.762100	0.206128	1.250974	0	0	-1.758855	
23654	0.541023	0.589304	-1.334871	0.797831	-0.542735	-1.023758	0	0	1.478590	

20402 rows × 23 columns

```
y_train #training output variable containing 20402 observations
```

```
10362    1
17941    1
14762    1
```

```
20367    0
17646    0
        ..
21575    1
5390     1
860      1
15795    0
23654    0
Name: satisfaction, Length: 20402, dtype: int64
```

```
X_test   #test input variables containing 8744 observations
```

	Cleanliness	Food and drink	Departure/Arrival time convenient	Gate location	Inflight wifi service	Leg room service	Customer Type_disloyal Customer	Gender_Male	Age	Travel_P
17004	0.541023	0.589304	1.273812	-1.542065	1.703853	-1.023758	0	1	-1.890995	
14459	-1.732902	-1.663826	-0.030530	0.797831	-0.542735	-1.782001	0	0	-0.966011	
24108	-0.216952	-0.161739	0.621641	1.577797	0.206128	1.250974	0	1	-0.635660	
10134	1.298998	0.589304	-0.030530	0.017866	0.206128	1.250974	0	0	0.553606	
23658	0.541023	0.589304	1.273812	-1.542065	0.954990	1.250974	0	1	0.619676	
...	...	...	...	...	...	...	...	...	...	
24382	-0.216952	-0.161739	1.273812	0.797831	-1.291598	-0.265514	0	0	0.025043	
3375	-1.732902	-0.912783	-1.334871	0.017866	0.954990	0.492730	0	0	0.817887	
17292	-0.216952	-0.161739	-1.334871	-1.542065	-1.291598	-1.023758	0	0	1.148239	
27078	-0.216952	1.340348	1.273812	1.577797	1.703853	1.250974	0	0	0.928374	
26402	0.134024	-0.565016	-0.030530	0.017866	0.206128	0.085587	0	1	0.187778	

8744 rows × 23 columns

```
y_test   #test output variable containing 8744 observations
```

```
17004    1
14459    0
24108    0
10134    1
23658    0
        ..
24382    0
3375     1
17292    1
27078    1
26402    1
Name: satisfaction, Length: 8744, dtype: int64
```

```
#Building k-NN Model
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)
def get_performance(actual_y, pred_y):
  cm = confusion_matrix(actual_y, pred_y)
  total = sum(sum(cm))
  accuracy = (cm[0, 0] + cm[1, 1]) / total
  sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
  specificity = cm[1, 1] / (cm[1, 0] +cm[1, 1])
  return accuracy, sensitivity, specificity

pred_y_knn = knn.predict(X_test)
accuracy_knn, sensitivity_knn, specificity_knn = get_performance(y_test, pred_y_knn)
```

```
pred_y_knn  #predicted y values based on knn model
```

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

```
confusion_matrix(y_test.values, knn.predict(X_test))   #checking mismatches
```

```
array([[4088,  314],
       [ 373, 3969]])
```

```
accuracy_knn, sensitivity_knn, specificity_knn
```

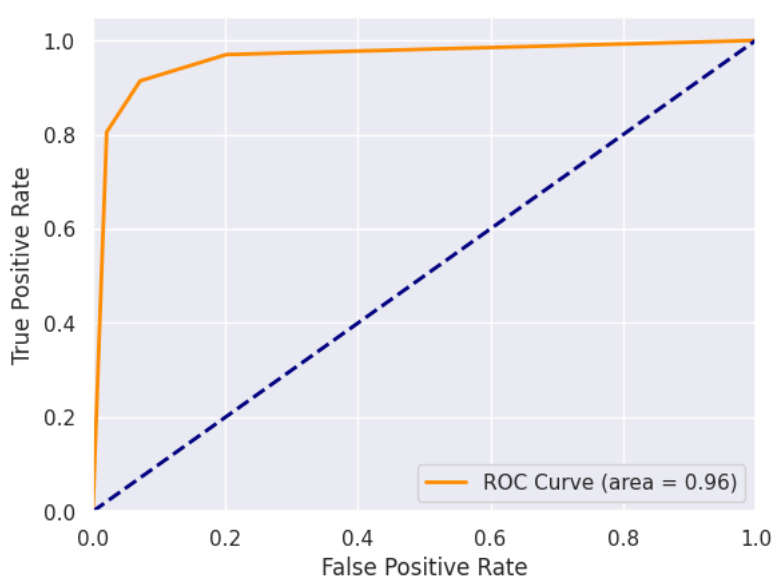
```
(0.9214318389752973, 0.9286687869150386, 0.9140948871487794)
```

```
from sklearn.metrics import roc_curve, auc
knn.predict_proba(X_test)
score_y_knn = knn.predict_proba(X_test)    #gives the probabilities
fpr, tpr, _ = roc_curve(y_test, score_y_knn[:, 1]) #creates false posive and true positive range using actual and predicted v
roc_auc = auc(fpr, tpr)
```

```
roc_auc  #a roc_auc which is closer to 1 is desirable. It means the model is working above random
```

```
0.960958059765556
```

```
#using fpr and tpr to create a roc curve
plt.figure()
lw = 2
plt.plot(fpr, tpr, color = 'darkorange', lw = lw, label = 'ROC Curve (area = %0.2f)'% roc_auc)
plt.plot([0, 1], [0, 1], color = 'navy', lw = lw, linestyle = '--')
plt.xlim([0.0,1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc = 'lower right')
plt.show()
```




```
#Building Random Forest Model
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 100, max_depth = 5, random_state = 0)
rf.fit(X_train, y_train)
def get_performance(actual_y, pred_y):
  cm = confusion_matrix(actual_y, pred_y)
  total = sum(sum(cm))
  accuracy = (cm[0, 0] + cm[1, 1]) / total
  sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
  specificity = cm[1, 1] / (cm[1, 0] +cm[1, 1])
  return accuracy, sensitivity, specificity

pred_y_rf = rf.predict(X_test)
accuracy_rf, sensitivity_rf, specificity_rf = get_performance(y_test, pred_y_rf)
```

```
pred_y_rf   #predicted y values based on rf model
```

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

```
confusion_matrix(y_test.values, rf.predict(X_test))   #checking mismatches
```

```
array([[4104,  298],
       [ 371, 3971]])
```

```
accuracy_rf, sensitivity_rf, specificity_rf
```

```
(0.9234903934126258, 0.9323034984098137, 0.9145555043758636)
```

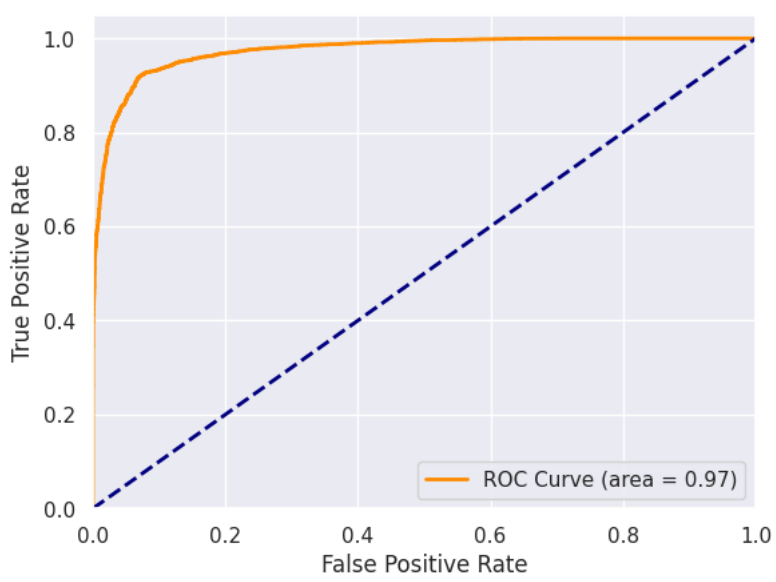
```
rf.predict_proba(X_test)
score_y_rf = rf.predict_proba(X_test)    #gives the probabilities
```

```
fpr, tpr, _ = roc_curve(y_test, score_y_rf[:, 1]) #creates false posive and true positive range using actual and predicted va
roc_auc = auc(fpr, tpr)
```

```
roc_auc
```

```
0.9735446713953353
```

```
#using fpr and tpr to create a roc curve that shows the performance of random forest model at
plt.figure()
lw = 2
plt.plot(fpr, tpr, color = 'darkorange', lw = lw, label = 'ROC Curve (area = %0.2f)'% roc_auc)
plt.plot([0, 1], [0, 1], color = 'navy', lw = lw, linestyle = '--')
plt.xlim([0.0,1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc = 'lower right')
plt.show()
```




```
#Building Decision Tree Model
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion = 'gini', splitter = 'best', max_depth = 15)
dt.fit(X_train, y_train)
def get_performance(actual_y, pred_y):
  cm = confusion_matrix(actual_y, pred_y)
  total = sum(sum(cm))
  accuracy = (cm[0, 0] + cm[1, 1]) / total
  sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
  specificity = cm[1, 1] / (cm[1, 0] +cm[1, 1])
  return accuracy, sensitivity, specificity # compute accuracy here

pred_y_dt = dt.predict(X_test)
accuracy_dt, sensitivity_dt, specificity_dt = get_performance(y_test, pred_y_dt) # compute accuracy here
```

```
pred_y_dt   #predicted y values based on dt model
```

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

```
confusion_matrix(y_test.values, dt.predict(X_test))   #checking mismatches
```

```
array([[4147,  255],
       [ 276, 4066]])
```

```
accuracy_dt, sensitivity_dt, specificity_dt
```

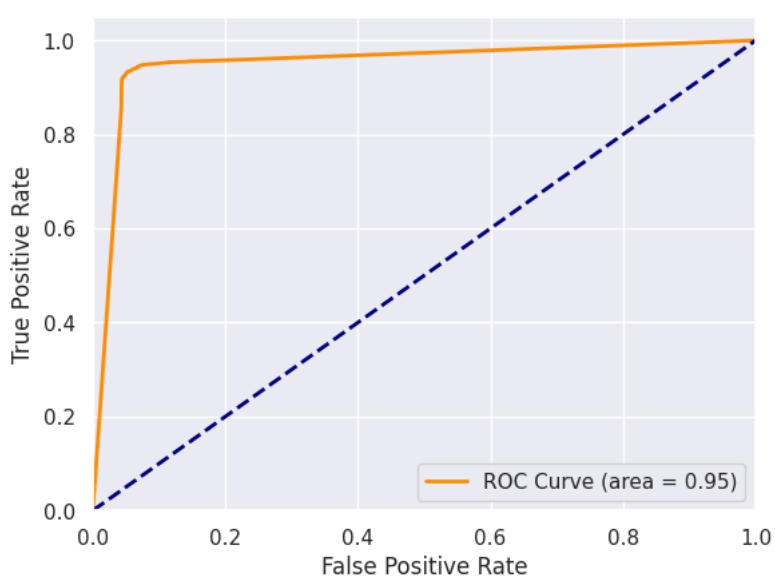
```
(0.9392726440988106, 0.9420717855520218, 0.9364348226623675)
```

```
dt.predict_proba(X_test)
score_y_dt= dt.predict_proba(X_test)    #gives the probabilities
fpr, tpr, _ = roc_curve(y_test, score_y_dt[:, 1]) #creates false posive and true positive range using actual and predicted va
roc_auc = auc(fpr, tpr)
```

```
roc_auc
```

```
0.9507795386754188
```

```
#using fpr and tpr to create a roc curve that shows the performance of random forest model at
plt.figure()
lw = 2
plt.plot(fpr, tpr, color = 'darkorange', lw = lw, label = 'ROC Curve (area = %0.2f)'% roc_auc)
plt.plot([0, 1], [0, 1], color = 'navy', lw = lw, linestyle = '--')
plt.xlim([0.0,1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc = 'lower right')
plt.show()
```




```
#Building a Logistic Regression Model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
lr = LogisticRegression(random_state = 0, solver = 'lbfgs', multi_class = 'ovr')
lr.fit(X_train, y_train)
def get_performance(actual_y, pred_y):
  cm = confusion_matrix(actual_y, pred_y)
  total = sum(sum(cm))
  accuracy = (cm[0, 0] + cm[1, 1]) / total
  sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
  specificity = cm[1, 1] / (cm[1, 0] +cm[1, 1])
  return accuracy, sensitivity, specificity

pred_y_lr = lr.predict(X_test)
accuracy_lr, sensitivity_lr, specificity_lr = get_performance(y_test, pred_y_lr)
```

```
pred_y_lr   #predicted y values based on lr model
```

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

```
confusion_matrix(y_test.values, lr.predict(X_test))   #checking mismatches
```

```
array([[3874,  528],
       [ 626, 3716]])
```

```
accuracy_lr, sensitivity_lr, specificity_lr
```

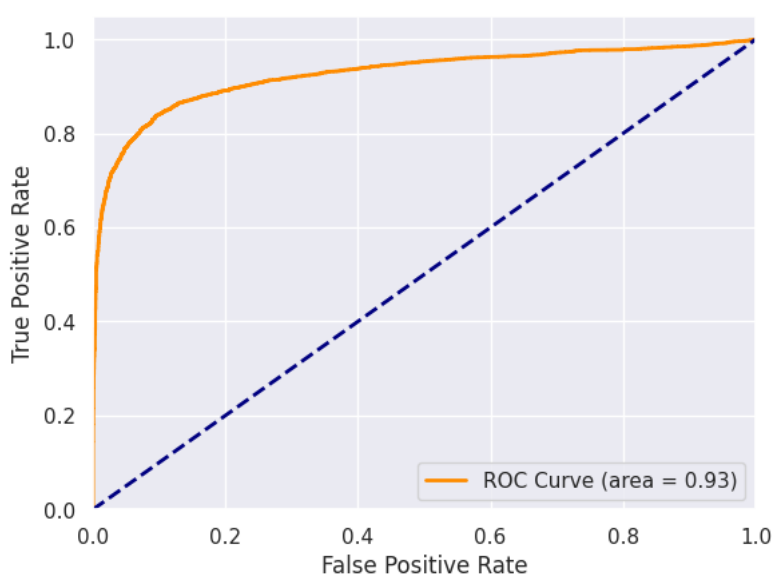
```
(0.8680237877401646, 0.8800545206724216, 0.8558268079226163)
```

```
lr.predict_proba(X_test)
score_y_lr = lr.predict_proba(X_test)    #gives the probabilities
fpr, tpr, _ = roc_curve(y_test, score_y_lr[:, 1]) #creates false posive and true positive range using actual and predicted va
roc_auc = auc(fpr, tpr)
```

```
roc_auc
```

```
0.9257390227757535
```

```
#using fpr and tpr to create a roc curve that shows the performance of logistic regression model
plt.figure()
lw = 2
plt.plot(fpr, tpr, color = 'darkorange', lw = lw, label = 'ROC Curve (area = %0.2f)'% roc_auc)
plt.plot([0, 1], [0, 1], color = 'navy', lw = lw, linestyle = '--')
plt.xlim([0.0,1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc = 'lower right')
plt.show()
```



Creating a dataframe using the various performance indicators so that we can easily compare the different models

```
perf = pd.DataFrame([accuracy_knn, accuracy_rf, accuracy_dt, accuracy_lr], columns = ['accuracy'], index = ['K-NN', 'Random ]
perf['sensitivity'] = np.asarray([sensitivity_knn, sensitivity_rf, sensitivity_dt, sensitivity_lr])
perf['specificity'] = np.asarray([specificity_knn, specificity_rf, specificity_dt, specificity_lr])
perf
```

	accuracy	sensitivity	specificity
K-NN	0.921432	0.928669	0.914095
Random Forest	0.923490	0.932303	0.914556
Decision Trees	0.939273	0.942072	0.936435
Logistic Regression	0.868024	0.880055	0.855827

```
# Evlauate the Model

# Evaluate accuracy
from sklearn.metrics import classification_report
accuracy = accuracy_score(y_test, pred_y_knn)
print(f'Accuracy: {accuracy:.2f}')

# Display classification report and confusion matrix
print('\nClassification Report:')
print(classification_report(y_test, pred_y_knn))

print('\nConfusion Matrix:')
conf_matrix = confusion_matrix(y_test, pred_y_knn)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

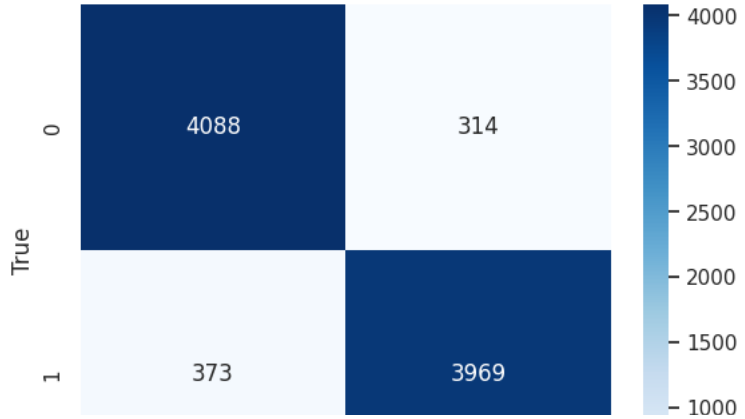
```
Accuracy: 0.92

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.93      0.92      4402
           1       0.93      0.91      0.92      4342

    accuracy                           0.92      8744
   macro avg       0.92      0.92      0.92      8744
weighted avg       0.92      0.92      0.92      8744

Confusion Matrix:
```




```
#our DataFrame is df
correlation_matrix = df.corr(numeric_only=True)
```