Amrita Vishwa Vidyapeetham
School of Computing, Coimbatore
ARM7 Assembly Language-Assignment
Fifth Semester
Department of Computer Science and Engineering
19CSE305 Machine Learning

AIRLINE SATISFACTION (GROUP 9)

Contents :

Team Members:

SL.no	Name	Roll Number
1	Amidela Anil Kumar	CB.EN.U4CSE20206
2	Chintapalli Jithendra Durga Kumar	CB.EN.U4CSE20213
3	Shaik Abdul Basith	CB.EN.U4CSE20257
4	Sugrivu Rohit	CB.EN.U4CSE20264
5	Veeraboina Nagoor	CB.EN.U4CSE20269

Introduction

Airline customers are individuals who travel on an airline for leisure or business purposes. They may fly on a range of various types of aircraft, from small regional jets to big long-haul planes, and they may be domestic or international passengers. Customers of an airline may have a variety of wants and preferences, and it is the obligation of the airline to make every effort to accommodate these needs and offer a satisfying travel experience. This may entail giving a range of seating choices, offering meals and entertainment during the journey, and providing a number of customer service alternatives, including online check-in and booking as well as assistance at the airport.

Customer happiness is becoming more widely recognised as a factor in business performance and a tactical instrument for gaining an advantage over competitors. Understanding consumer wants and comfort levels, or customer pleasure during the journey, is crucial.

The degree of passenger pleasure with an airline is referred to as airline satisfaction. Several things might affect it, such as the calibre of the planes and the seating, the punctuality of the flights, the crew's friendliness and effectiveness, the availability and calibre of the in-flight amenities, and the effectiveness of the airline's customer service. For airlines, increasing customer satisfaction is crucial since it can result in more loyalty, favourable word-of-mouth, and higher customer retention rates. Airlines can gauge and monitor consumer satisfaction in a number of methods, including surveys and systems for receiving client feedback.

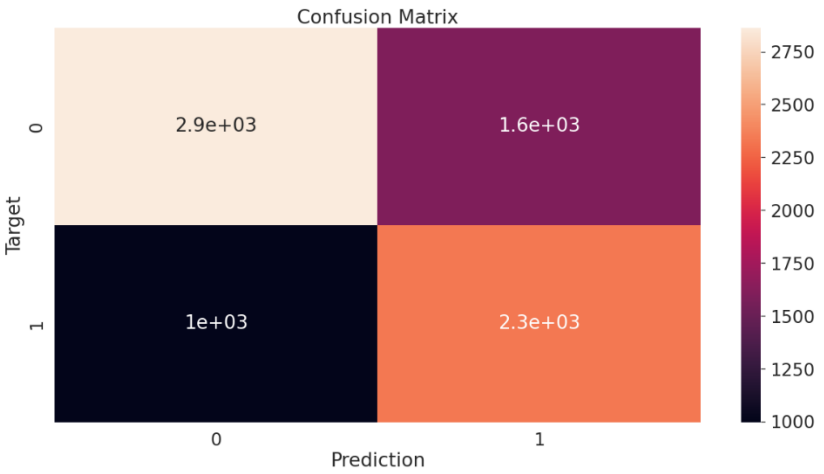
Customer input is crucial for the aviation business in general. There may be a variety of techniques to gather client feedback. The usual and simplest method is to use the customer feedback form that is available while travelling. The means of collecting feedback can be provided by different approaches.

Online forms, Social media and Polls etc are some of the approaches to extract customer feedback after the travel. Customers find social media such as facebook and twitter so comfortable modes of communication to the organizations.

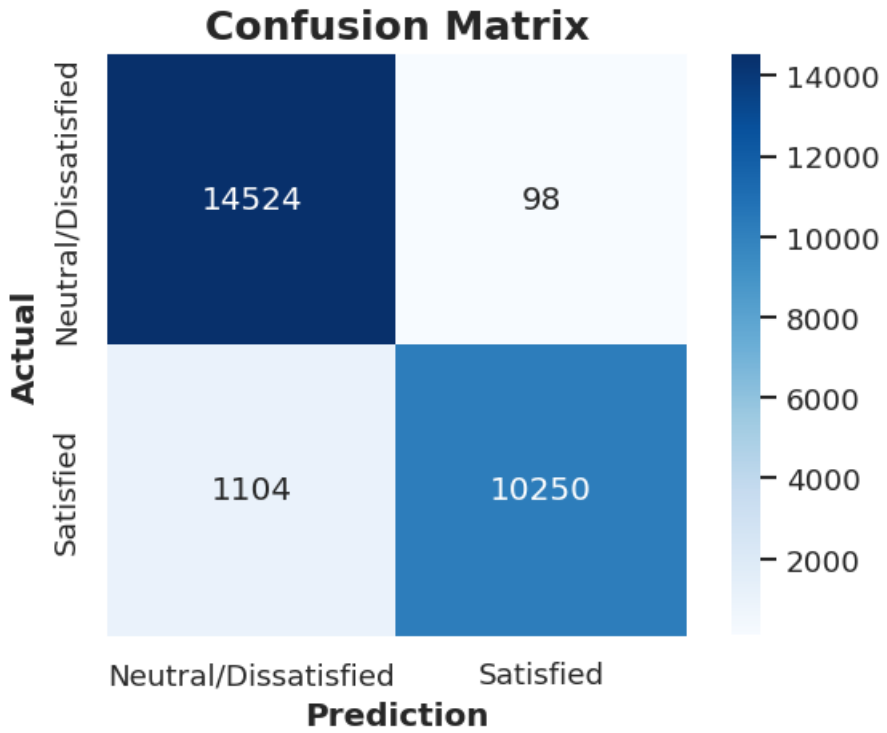
Algorithms

Supervised ML Algorithms used

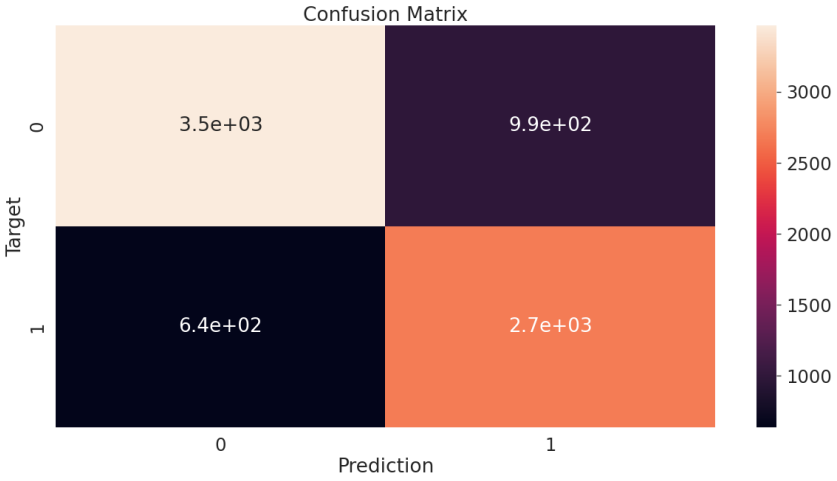
1) Logistic



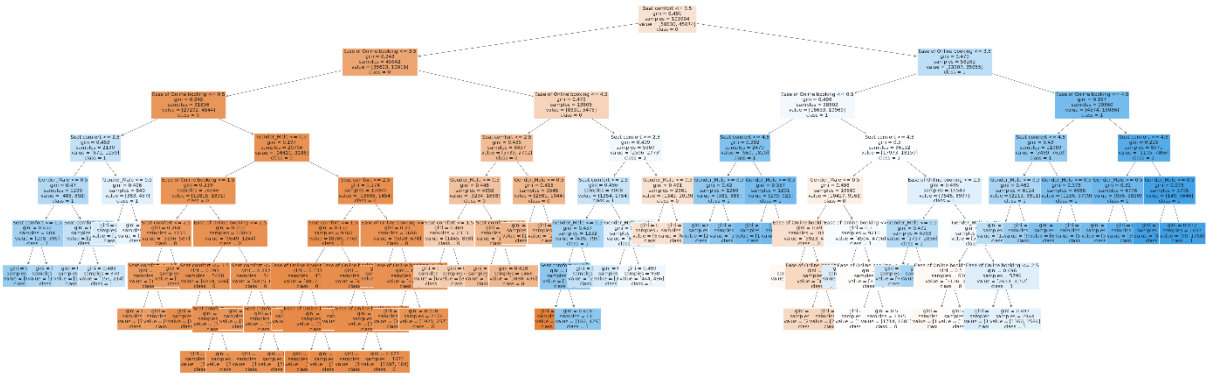
2) Linear



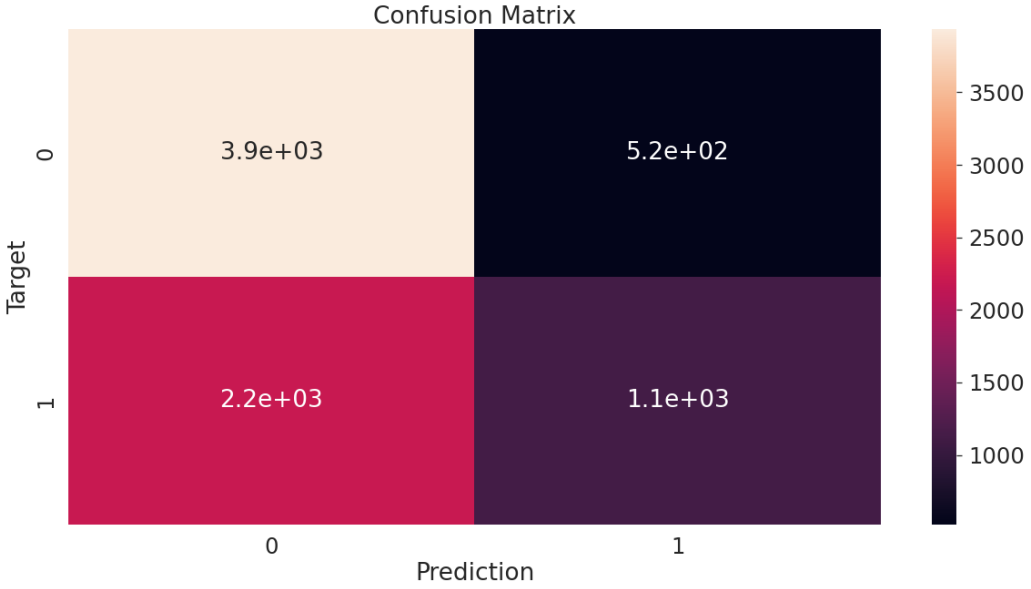
3) Naïve Bayes



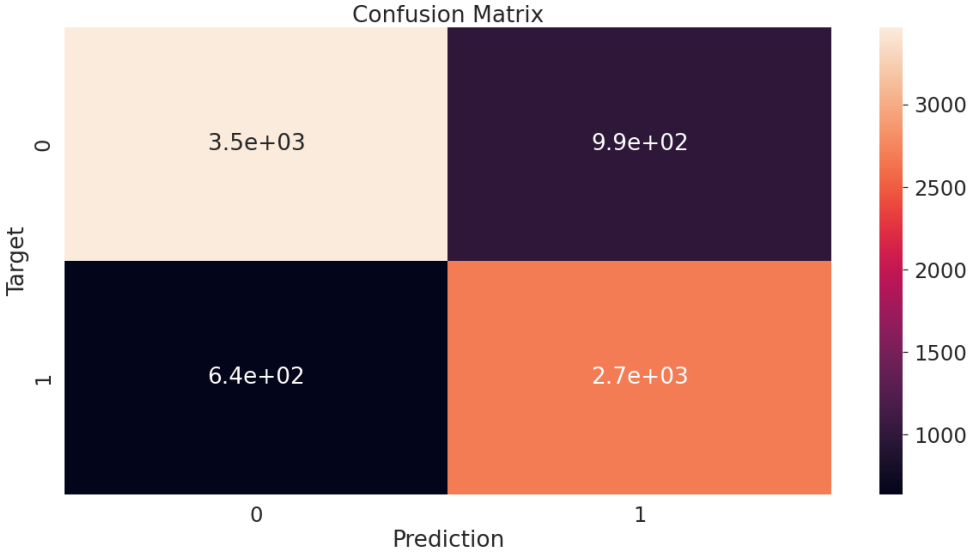
4) Decision Tree



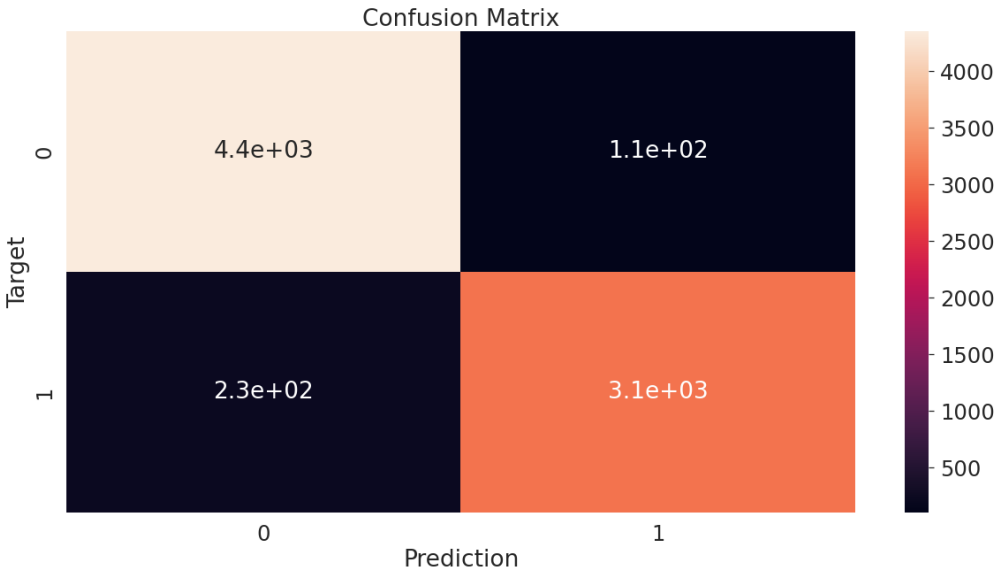
5) Support Vector Machine



6) K Nearest Neighbour (KNN)

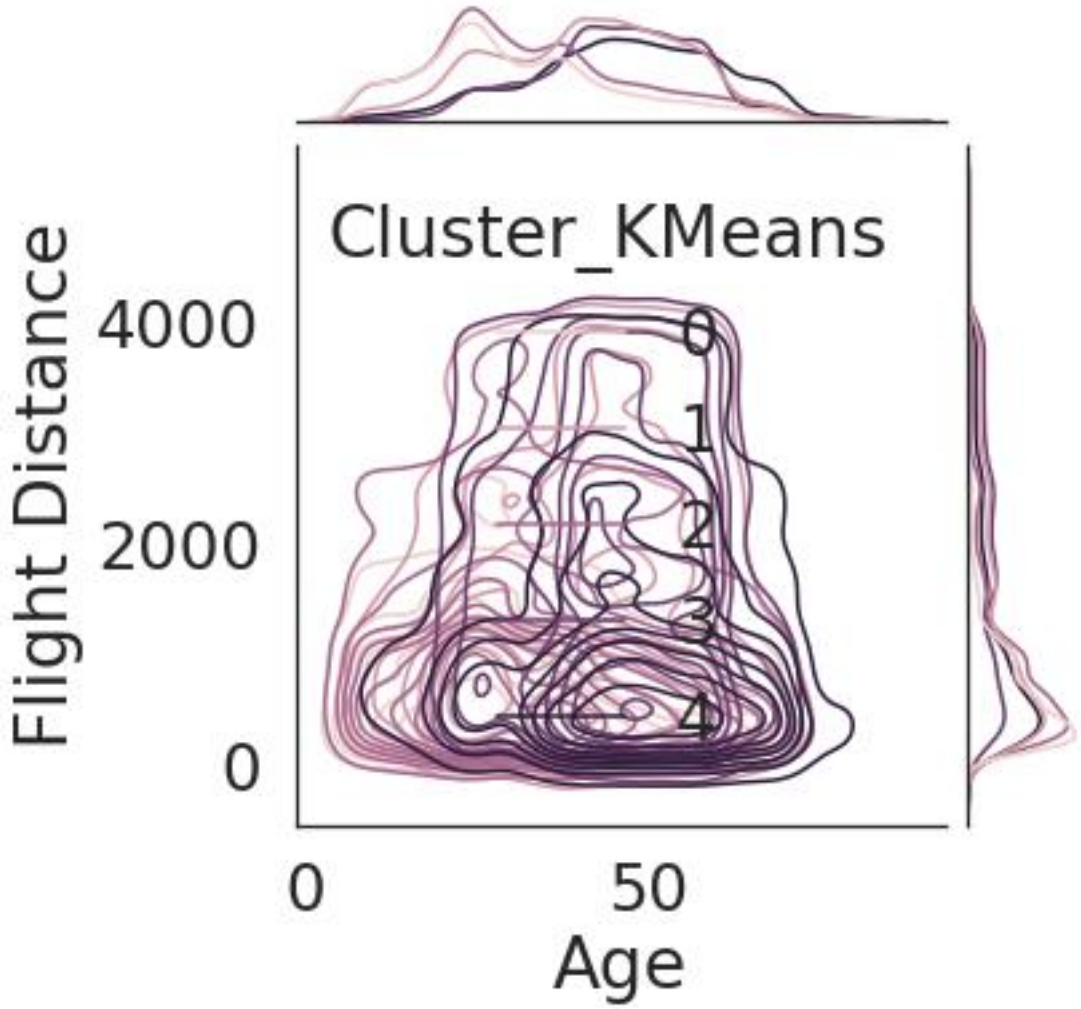


7) Random Forest

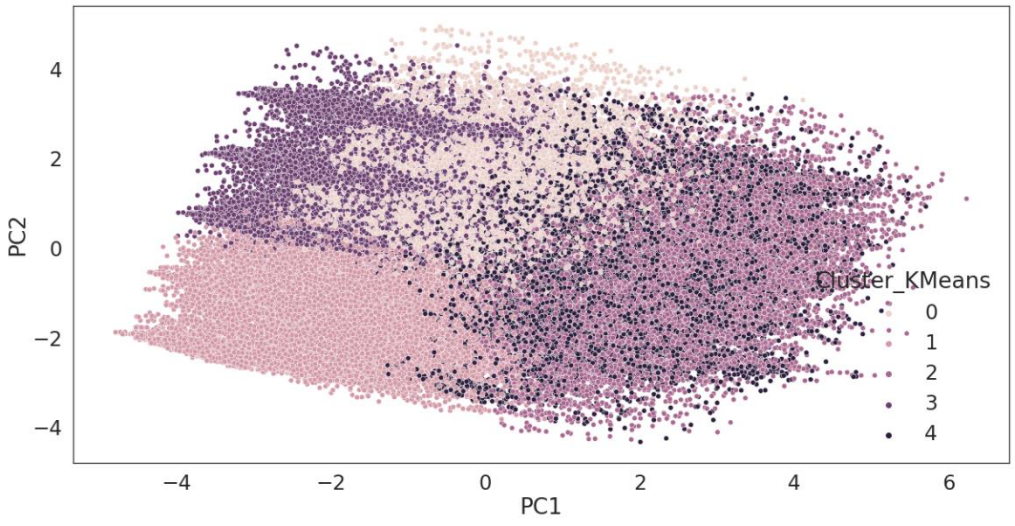


Unsupervised ML Algorithms used

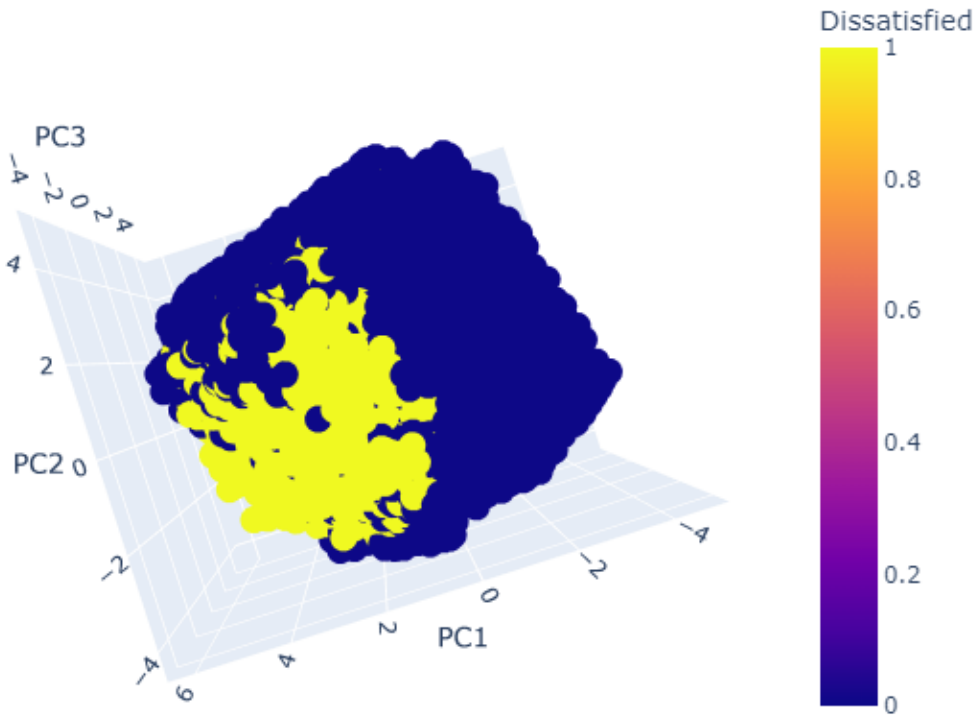
1) K Means Clustering



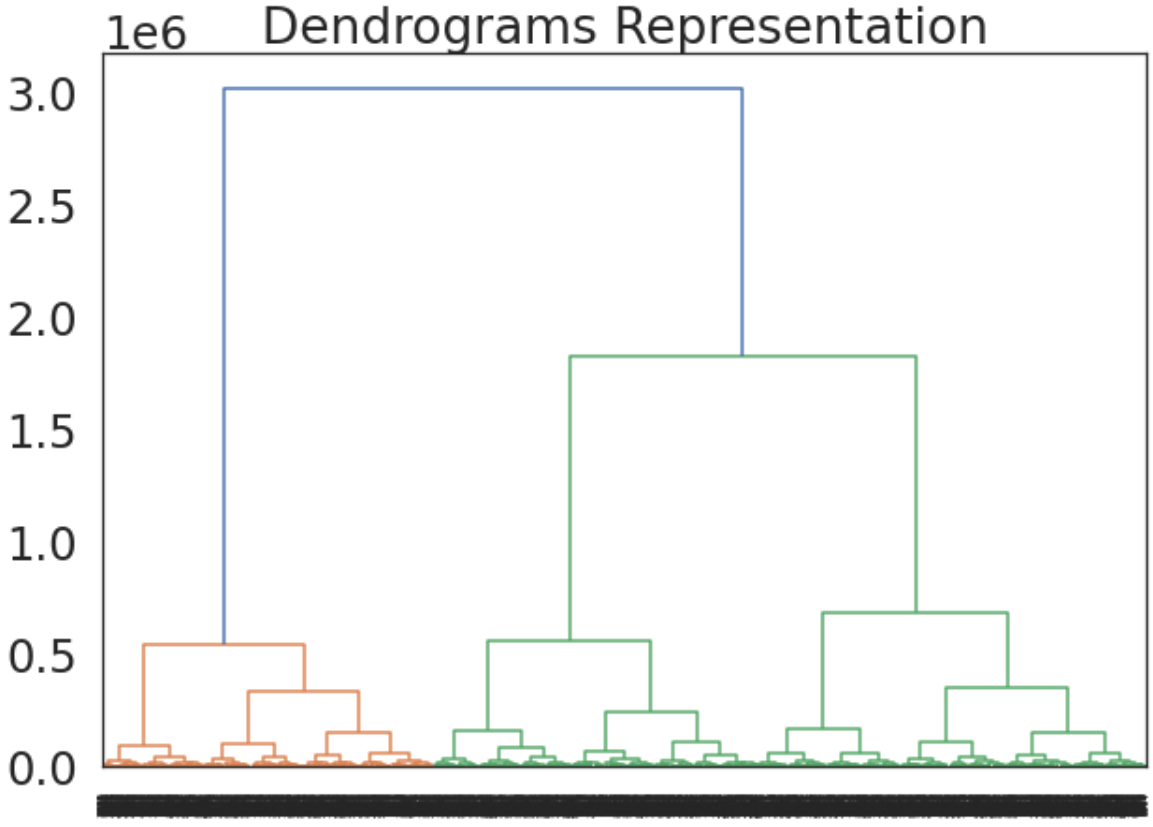
2) Principal Component Analysis (PCA)



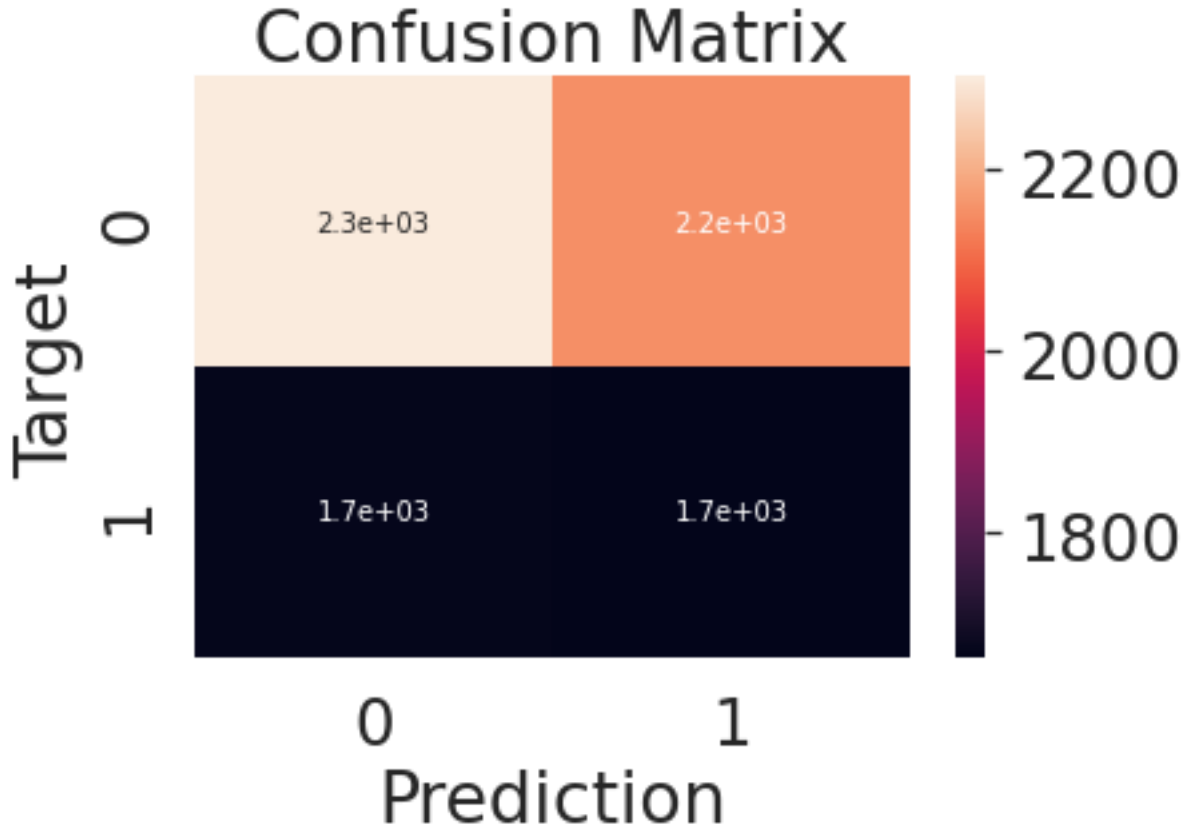
3) DB SCAN Clustering



4) Agglomerative Clustering



5) Birch Clustering



Result Analysis

Algorithm is analysed by the following factors

1) Accuracy
2) Precision
3) Specificity
4) Recall
5) F1-Score
6) Rightly_Classified
7) Wrongly Classified

ML Classification Algo	Rightly_Classified	Wrongly_Classified	Accuracy	Precision	Recall	Specificity	F1-Score
Random Forest	0.956857	0.043143	0.956857	0.966958	0.930972	0.976212	0.948624
Gaussian Naive Bayes	0.79096	0.20904	0.79096	0.731396	0.808223	0.778052	0.767893
KNN	0.79096	0.20904	0.79096	0.731396	0.808223	0.778052	0.767893
K-Means Clustering	0.79096	0.20904	0.79096	0.731396	0.808223	0.778052	0.767893
Decision_Tree	0.701609	0.298391	0.701609	0.68024	0.59882	0.781425	0.636938
Logistic Regression	0.66718	0.33282	0.66718	0.594148	0.70078	0.642056	0.643074
Agglomerative Clustering	0.508	0.492	0.508	0.453303	0.441242	0.562842	0.447191

RANDOM FOREST gives more accuracy with high F1score .So, we prefer to use random forest classifier
Supervised learning algorithms classify better compared to Unsupervisied since dataset is labelled one .

Copy of the CODE :

The copy of the ipynb notebook is linked below this

ML_project(airline).ip
ynb - Colaboratory.pc

ML_project(airline).ipynb - Colaboratory

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Using a dataset that contains an airline passenger satisfaction survey, the goal of this project is to determine what factors lead to customer satisfaction for an Airline.

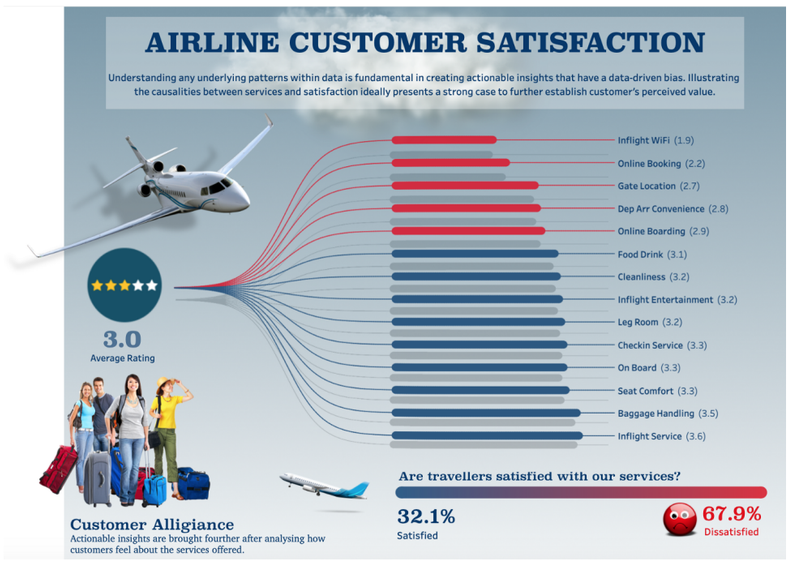
Survey: Several features measured by 5-point Likert scale such as: Food and drink, Seat comfort, Inflight entertainment.

Demographic: features such as Age and Gender.

Commercial: features such as Customer Type.

Service: features related to the service provided that generated the survey such as: Type of Travel, Departure Delay in Minutes.

```
import cv2
from google.colab.patches import cv2_imshow

img = cv2.imread('/content/drive/MyDrive/1_4XEebsOoxOhoalszUO-2FA.png')
cv2_imshow(img)
```



```
import pandas as pd
train_df = pd.read_csv("/content/drive/MyDrive/Airline_Dataset.csv")
train_df.head()
```

	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
0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3	1	5	3	5	5	4	3	4	4	5	5	25	18.0	neutral or dissatisfied
1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	3	3	1	3	1	1	1	5	3	1	4	1	1	6.0	neutral or dissatisfied
2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	2	2	5	5	5	5	4	3	4	4	4	5	0	0.0	satisfied
3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	5	5	2	2	2	2	2	5	3	1	4	2	11	9.0	neutral or dissatisfied
4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	3	3	3	4	5	5	3	3	4	4	3	3	3	0	0.0	satisfied

```
train_df.columns
```

```
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')
```

```
train_df.isnull().sum()
```

```
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             393
Satisfaction                           0
dtype: int64
```

```
type(train_df["Arrival Delay in Minutes"][0])
```

```
numpy.float64
```

```
len(train_df)
```

```
129880
```

```
import numpy as np
train_df.fillna(np.nanmedian(train_df["Arrival Delay in Minutes"]),inplace = True)
```

```
train_df.isnull().sum()
```

```
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
```

```
len(train_df)
```

```
129880
```

```
train_df.corr()
```

	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
id	1.000000	0.020322	0.095504	-0.023096	-0.002192	0.013400	-0.000113	-0.000510	0.055538	0.052164	0.001620	0.055502	0.044088	0.074569	0.079325	0.078793	0.024048	-0.017643	-0.035405
Age	0.020322	1.000000	0.099459	0.016116	0.036960	0.022565	-0.000398	0.023194	0.207572	0.159136	0.074947	0.057078	0.039119	-0.047991	0.033475	-0.051347	0.052565	-0.009041	-0.011206
Flight Distance	0.095504	0.099459	1.000000	0.006701	-0.018914	0.065165	0.005520	0.057066	0.214825	0.157662	0.130507	0.111194	0.134533	0.064855	0.073608	0.059316	0.095648	0.002402	-0.001973
Inflight wifi service	-0.023096	0.016116	0.006701	1.000000	0.344915	0.714807	0.338573	0.132214	0.457445	0.121513	0.207802	0.119928	0.160317	0.120376	0.043762	0.110029	0.131300	-0.015946	-0.017762
Departure/Arrival time convenient	-0.002192	0.036960	-0.018914	0.344915	1.000000	0.437620	0.447510	0.000687	0.072287	0.008666	-0.008380	0.067297	0.010617	0.070833	0.091132	0.072195	0.009862	0.000778	-0.001005
Ease of Online booking	0.013400	0.022565	0.065165	0.714807	0.437620	1.000000	0.460041	0.030514	0.404866	0.028561	0.046564	0.039064	0.109450	0.039148	0.008819	0.035373	0.015125	-0.005318	-0.007046
Gate location	-0.000113	-0.000398	0.005520	0.338573	0.447510	0.460041	1.000000	-0.002872	0.002756	0.002788	0.002741	-0.029019	-0.005181	0.000972	-0.039353	0.000310	-0.005918	0.005973	0.005643
Food and drink	-0.000510	0.023194	0.057066	0.132214	0.000687	0.030514	-0.002872	1.000000	0.233500	0.575846	0.623461	0.057404	0.033173	0.035321	0.085198	0.035210	0.658054	-0.029164	-0.031685
Online boarding	0.055538	0.207572	0.214825	0.457445	0.072287	0.404866	0.002756	0.233500	1.000000	0.419253	0.283922	0.154242	0.123225	0.083541	0.204238	0.074058	0.329377	-0.019404	-0.022663
Seat comfort	0.052164	0.159136	0.157662	0.121513	0.008666	0.028561	0.002788	0.575846	0.419253	1.000000	0.611837	0.130545	0.104272	0.074620	0.189979	0.068842	0.679613	-0.027999	-0.030397
Inflight entertainment	0.001620	0.074947	0.130507	0.207802	-0.008380	0.046564	0.002741	0.623461	0.283922	0.611837	1.000000	0.418574	0.300397	0.379123	0.119554	0.406094	0.692511	-0.027012	-0.030183
On-board service	0.055502	0.057078	0.111194	0.119928	0.067297	0.039064	-0.029019	0.057404	0.154242	0.130545	0.418574	1.000000	0.357721	0.520296	0.244619	0.551569	0.122084	-0.030486	-0.034671
Leg room service	0.044088	0.039119	0.134533	0.160317	0.010617	0.109450	-0.005181	0.033173	0.123225	0.104272	0.300397	0.357721	1.000000	0.371455	0.152693	0.369569	0.096695	0.014574	0.011384
Baggage handling	0.074569	-0.047991	0.064855	0.120376	0.070833	0.039148	0.000972	0.035321	0.083541	0.074620	0.379123	0.520296	0.371455	1.000000	0.234503	0.629237	0.097071	-0.004105	-0.007997
Checkin service	0.079325	0.033475	0.073608	0.043762	0.091132	0.008819	-0.039353	0.085198	0.204238	0.189979	0.119554	0.244619	0.152693	0.234503	1.000000	0.237601	0.176658	-0.018752	-0.021675
Inflight service	0.078793	-0.051347	0.059316	0.110029	0.072195	0.035373	0.000310	0.035210	0.074058	0.068842	0.406094	0.551569	0.369569	0.629237	0.237601	1.000000	0.090356	-0.054432	-0.059685
Cleanliness	0.024048	0.052565	0.095648	0.131300	0.009862	0.015125	-0.005918	0.658054	0.329377	0.679613	0.692511	0.122084	0.096695	0.097071	0.176658	0.090356	1.000000	-0.014543	-0.016547
Departure Delay in Minutes	-0.017643	-0.009041	0.002402	-0.015946	0.000778	-0.005318	0.005973	-0.029164	-0.019404	-0.027999	-0.027012	-0.030486	0.014574	-0.004105	-0.018752	-0.054432	-0.014543	1.000000	0.959382
Arrival Delay in Minutes	-0.035405	-0.011206	-0.001973	-0.017762	-0.001005	-0.007046	0.005643	-0.031685	-0.022663	-0.030397	-0.030183	-0.034671	0.011384	-0.007997	-0.021675	-0.059685	-0.016547	0.959382	1.000000

```
train_df.columns
```

```
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')
```

```
train_df["id"].head()
```

```
0     70172
1      5047
2    110028
3     24026
4    119299
Name: id, dtype: int64
```

```
train_df
```

	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
0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3	1	5	3	5	5	4	3	4	4	5	5	25	18.0	neutral or dissatisfied
1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	3	3	1	3	1	1	1	5	3	1	4	1	1	6.0	neutral or dissatisfied
2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	2	2	5	5	5	5	4	3	4	4	4	5	0	0.0	satisfied
3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	5	5	2	2	2	2	2	5	3	1	4	2	11	9.0	neutral or dissatisfied
4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	3	3	3	4	5	5	3	3	4	4	3	3	3	0	0.0	satisfied
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
129875	78463	Male	disloyal Customer	34	Business travel	Business	526	3	3	3	1	4	3	4	4	3	2	4	4	5	4	0	0.0	neutral or dissatisfied
129876	71167	Male	Loyal Customer	23	Business travel	Business	646	4	4	4	4	4	4	4	4	4	5	5	5	5	4	0	0.0	satisfied
129877	37675	Female	Loyal Customer	17	Personal Travel	Eco	828	2	5	1	5	2	1	2	2	4	3	4	5	4	2	0	0.0	neutral or dissatisfied
129878	90086	Male	Loyal Customer	14	Business travel	Business	1127	3	3	3	3	4	4	4	4	3	2	5	4	5	4	0	0.0	satisfied
129879	34799	Female	Loyal Customer	42	Personal Travel	Eco	264	2	5	2	5	4	2	2	1	1	2	1	1	1	1	0	0.0	neutral or dissatisfied

129880 rows × 24 columns

```
train_df.columns
```

```
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')
```

```
train_df
```

	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
0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3	1	5	3	5	5	4	3	4	4	5	5	25	18.0	neutral or dissatisfied
1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	3	3	1	3	1	1	1	5	3	1	4	1	1	6.0	neutral or dissatisfied
2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	2	2	5	5	5	5	4	3	4	4	4	5	0	0.0	satisfied
3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	5	5	2	2	2	2	2	5	3	1	4	2	11	9.0	neutral or dissatisfied
4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	3	3	3	4	5	5	3	3	4	4	3	3	3	0	0.0	satisfied
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
129875	78463	Male	disloyal Customer	34	Business travel	Business	526	3	3	3	1	4	3	4	4	3	2	4	4	5	4	0	0.0	neutral or dissatisfied
129876	71167	Male	Loyal Customer	23	Business travel	Business	646	4	4	4	4	4	4	4	4	4	5	5	5	5	4	0	0.0	satisfied
			Loyal		Personal																			neutral or

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129880 entries, 0 to 129879
Data columns (total 24 columns):
 #   Column                            Non-Null Count   Dtype  
---  ------                            --------------   -----  
 0   id                                129880 non-null  int64  
 1   Gender                            129880 non-null  object 
 2   Customer Type                     129880 non-null  object 
 3   Age                               129880 non-null  int64  
 4   Type of Travel                    129880 non-null  object 
 5   Class                             129880 non-null  object 
 6   Flight Distance                   129880 non-null  int64  
 7   Inflight wifi service             129880 non-null  int64  
 8   Departure/Arrival time convenient 129880 non-null  int64  
 9   Ease of Online booking            129880 non-null  int64  
 10  Gate location                     129880 non-null  int64  
 11  Food and drink                    129880 non-null  int64  
 12  Online boarding                   129880 non-null  int64  
 13  Seat comfort                      129880 non-null  int64  
 14  Inflight entertainment            129880 non-null  int64  
 15  On-board service                  129880 non-null  int64  
 16  Leg room service                  129880 non-null  int64  
 17  Baggage handling                  129880 non-null  int64  
 18  Checkin service                   129880 non-null  int64  
 19  Inflight service                  129880 non-null  int64  
 20  Cleanliness                       129880 non-null  int64  
 21  Departure Delay in Minutes        129880 non-null  int64  
 22  Arrival Delay in Minutes          129880 non-null  float64
 23  Satisfaction                      129880 non-null  object 
dtypes: float64(1), int64(18), object(5)
memory usage: 23.8+ MB
```

```
train_df
```

	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
0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3	1	5	3	5	5	4	3	4	4	5	5	25	18.0	neutral or dissatisfied
1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	3	3	1	3	1	1	1	5	3	1	4	1	1	6.0	neutral or dissatisfied
2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	2	2	5	5	5	5	4	3	4	4	4	5	0	0.0	satisfied
3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	5	5	2	2	2	2	2	5	3	1	4	2	11	9.0	neutral or dissatisfied

`train_df`

	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
0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3	1	5	3	5	5	4	3	4	4	5	5	25	18.0	neutral or dissatisfied
1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	3	3	1	3	1	1	1	5	3	1	4	1	1	6.0	neutral or dissatisfied
2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	2	2	5	5	5	5	4	3	4	4	4	5	0	0.0	satisfied
3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	5	5	2	2	2	2	2	5	3	1	4	2	11	9.0	neutral or dissatisfied
4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	3	3	3	4	5	5	3	3	4	4	3	3	3	0	0.0	satisfied
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
129875	78463	Male	disloyal Customer	34	Business travel	Business	526	3	3	3	1	4	3	4	4	3	2	4	4	5	4	0	0.0	neutral or dissatisfied
129876	71167	Male	Loyal Customer	23	Business travel	Business	646	4	4	4	4	4	4	4	4	4	5	5	5	5	4	0	0.0	satisfied
129877	37675	Female	Loyal Customer	17	Personal Travel	Eco	828	2	5	1	5	2	1	2	2	4	3	4	5	4	2	0	0.0	neutral or dissatisfied
129878	90086	Male	Loyal Customer	14	Business travel	Business	1127	3	3	3	3	4	4	4	4	3	2	5	4	5	4	0	0.0	satisfied
129879	34799	Female	Loyal Customer	42	Personal Travel	Eco	264	2	5	2	5	4	2	2	1	1	2	1	1	1	1	0	0.0	neutral or dissatisfied

129880 rows × 24 columns

GENDER OF PASSENGERS

```
# Gender Column

train_df["Gender"].unique()
```

```
array(['Male', 'Female'], dtype=object)
```

```
train_df["Customer Type"].unique()
```

```
array(['Loyal Customer', 'disloyal Customer'], dtype=object)
```

```
train_df["Satisfaction"].unique()
```

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

Type of Travel

```
train_df["Type of Travel"].unique()
```

```
array(['Personal Travel', 'Business travel'], dtype=object)
```

Class

```
train_df["Class"].unique()
```

```
array(['Eco Plus', 'Business', 'Eco'], dtype=object)
```

```
train_df['Flight Distance'].max()
```

```
4983
```

Inflight wifi service: Satisfaction level of the inflight wifi service (0:Not Applicable;1-5)

```
al=train_df["Inflight wifi service"].value_counts().sum()
al
train_df[["Inflight wifi service",'Gender']].value_counts()/al *100
```

```
Inflight wifi service  Gender
2                      Female    12.739452
3                      Female    12.570065
                       Male      12.210502
2                      Male      12.145057
4                      Female     9.611949
                       Male       9.463351
1                      Female     8.741146
                       Male       8.450108
5                      Female     5.535879
                       Male       5.517401
0                      Female     1.539883
                       Male       1.475208
dtype: float64
```

Departure/Arrival time convenient: Satisfaction level of Departure/Arrival time convenient

```
al=train_df["Departure/Arrival time convenient"].value_counts().sum()
al
train_df[["Departure/Arrival time convenient",'Age']].value_counts()/al *100
```

```
Departure/Arrival time convenient  Age
4                                  39     0.615953
                                   25     0.584386
                                   40     0.566677
5                                  39     0.564367
2                                  39     0.561287
4                                  45     0.558208
                                   41     0.542039
                                   44     0.540499
                                   43     0.536649
5                                  40     0.535879
3                                  39     0.535879
4                                  49     0.534339
                                   46     0.533569
                                   38     0.525100
5                                  41     0.522020
                                   44     0.518171
4                                  26     0.512781
                                   52     0.511241
                                   47     0.510471
                                   37     0.505852
                                   42     0.505082
5                                  25     0.505082
                                   48     0.503542
4                                  36     0.503542
                                   48     0.497382
5                                  47     0.493532
4                                  27     0.485833
                                   24     0.485063
5                                  42     0.481983
4                                  23     0.479674
                                   51     0.478134
                                   50     0.478134
                                   22     0.473514
5                                  45     0.473514
```

```
                                51    0.463505
                                43    0.461195
      1                         39    0.460425
      3                         42    0.458885
      5                         46    0.457345
                                49    0.456575
      4                         30    0.453496
      3                         40    0.452726
                                36    0.451186
      4                         54    0.449646
      5                         52    0.448876
                                23    0.446566
      4                         35    0.445796
      3                         44    0.443486
                                41    0.439637
      4                         53    0.437327
      3                         43    0.435787
      5                         38    0.434247
      4                         33    0.434247
      2                         25    0.429627
      0                         25    0.426548
      4                         56    0.426548
                                55    0.425778
```

Ease of Online booking: Satisfaction level of online booking

```
al=train_df["Ease of Online booking"].value_counts().sum()
al
train_df[["Ease of Online booking",'Customer Type']].value_counts()/al *100
```

```
     Ease of Online booking  Customer Type
     3                       Loyal Customer       18.748075
     2                       Loyal Customer       18.578688
     4                       Loyal Customer       15.204804
     1                       Loyal Customer       14.050662
     5                       Loyal Customer       11.583770
     3                       disloyal Customer     4.652756
     2                       disloyal Customer     4.558824
     4                       disloyal Customer     3.615645
     0                       Loyal Customer        3.524792
     1                       disloyal Customer     2.800277
     5                       disloyal Customer     1.831691
     0                       disloyal Customer     0.850015
     dtype: float64
```

Gate location: Satisfaction level of Gate location

```
al=train_df["Gate location"].value_counts().sum()
al
train_df["Gate location"].value_counts()/al *100
```

```
     3    27.500000
     4    23.457037
     2    18.706498
     1    16.931783
     5    13.403911
     0     0.000770
     Name: Gate location, dtype: float64
```

Food and drink: Satisfaction level of Food and drink

```
al=train_df["Food and drink"].value_counts().sum()
al
train_df["Food and drink"].value_counts()/al *100
```

```
     4    23.531722
     5    21.525254
     3    21.399754
     2    21.083308
     1    12.358331
     0     0.101632
     Name: Food and drink, dtype: float64
```

Online boarding: Satisfaction level of online boarding

```
al=train_df["Online boarding"].value_counts().sum()
al
train_df["Online boarding"].value_counts()/al *100
```

```
4    29.618109
3    20.878503
5    20.033877
2    16.887897
1    10.210194
0     2.371420
Name: Online boarding, dtype: float64
```

Seat comfort: Satisfaction level of Seat comfort

```
al=train_df["Seat comfort"].value_counts().sum()
al
train_df["Seat comfort"].value_counts()/al *100
```

```
4    30.609794
5    25.529720
3    17.961195
2    14.266246
1    11.632276
0     0.000770
Name: Seat comfort, dtype: float64
```

Inflight entertainment: Satisfaction level of inflight entertainment

```
al=train_df["Inflight entertainment"].value_counts().sum()
al
train_df["Inflight entertainment"].value_counts()/al *100
```

```
4    28.326917
5    24.287034
3    18.389282
2    16.914075
1    12.068833
0     0.013859
Name: Inflight entertainment, dtype: float64
```

On-board service: Satisfaction level of On-board service

```
al=train_df["On-board service"].value_counts().sum()
al
train_df["On-board service"].value_counts()/al *100
```

```
4    29.799045
5    22.707114
3    21.975670
2    14.129196
1    11.385125
0     0.003850
Name: On-board service, dtype: float64
```

Leg room service: Satisfaction level of Leg room service

```
al=train_df["Leg room service"].value_counts().sum()
al
train_df["Leg room service"].value_counts()/al *100
```

```
4    27.630120
5    23.795042
3    19.291654
2    18.894364
1     9.928395
0     0.460425
Name: Leg room service, dtype: float64
```

Baggage handling: Satisfaction level of baggage handling

```
al=train_df["Baggage handling"].value_counts().sum()
al
train_df["Baggage handling"].value_counts()/al *100
```

```
4    36.003234
5    26.084078
3    19.903757
2    11.057900
1     6.951032
Name: Baggage handling, dtype: float64
```

Check-in service: Satisfaction level of Check-in service

```
al=train_df["Checkin service"].value_counts().sum()
al
train_df["Checkin service"].value_counts()/al *100
```

```
4    27.974284
3    27.296735
5    19.928395
1    12.402217
2    12.397598
0     0.000770
Name: Checkin service, dtype: float64
```

Inflight service: Satisfaction level of inflight service

```
al=train_df["Inflight service"].value_counts().sum()
al
train_df["Inflight service"].value_counts()/al *100
```

```
4    36.435941
5    26.228827
3    19.491839
2    11.016323
1     6.823221
0     0.003850
Name: Inflight service, dtype: float64
```

Cleanliness: Satisfaction level of Cleanliness

```
al=train_df["Cleanliness"].value_counts().sum()
al
train_df["Cleanliness"].value_counts()/al *100
```

```
4    26.154142
3    23.590237
5    21.878657
2    15.485833
1    12.880351
0     0.010779
Name: Cleanliness, dtype: float64
```

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129880 entries, 0 to 129879
Data columns (total 24 columns):
 #   Column                            Non-Null Count   Dtype
---  ------                            --------------   -----
 0   id                                129880 non-null  int64
 1   Gender                            129880 non-null  object
 2   Customer Type                     129880 non-null  object
 3   Age                               129880 non-null  int64
 4   Type of Travel                    129880 non-null  object
 5   Class                             129880 non-null  object
 6   Flight Distance                   129880 non-null  int64
 7   Inflight wifi service             129880 non-null  int64
 8   Departure/Arrival time convenient 129880 non-null  int64
 9   Ease of Online booking            129880 non-null  int64
 10  Gate location                     129880 non-null  int64
 11  Food and drink                    129880 non-null  int64
 12  Online boarding                   129880 non-null  int64
 13  Seat comfort                      129880 non-null  int64
 14  Inflight entertainment            129880 non-null  int64
 15  On-board service                  129880 non-null  int64
 16  Leg room service                  129880 non-null  int64
```

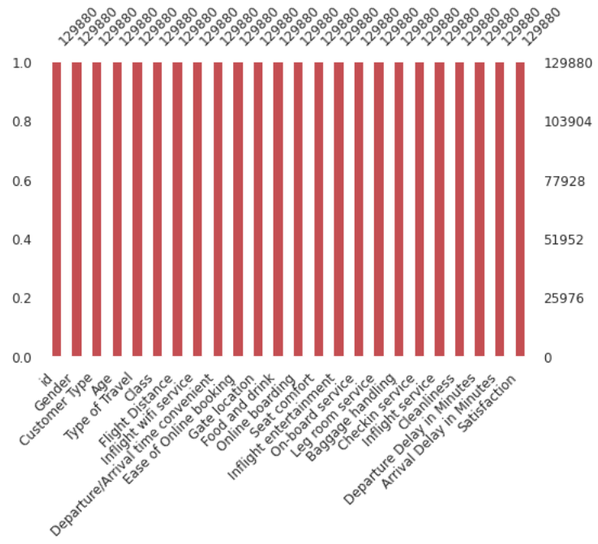
```
 17  Baggage handling                 129880 non-null  int64
 18  Checkin service                  129880 non-null  int64
 19  Inflight service                 129880 non-null  int64
 20  Cleanliness                      129880 non-null  int64
 21  Departure Delay in Minutes       129880 non-null  int64
 22  Arrival Delay in Minutes         129880 non-null  float64
 23  Satisfaction                     129880 non-null  object
dtypes: float64(1), int64(18), object(5)
memory usage: 23.8+ MB
```

DATA VISUALIZATION

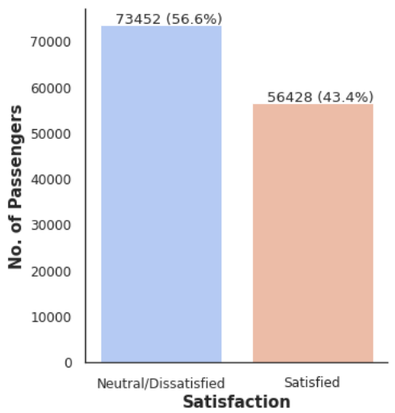
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import math as math

from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV, Lasso, lars_path
from sklearn.naive_bayes import MultinomialNB,GaussianNB,BernoulliNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier,VotingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, KFold, cross_val_score, GridSearchCV, RandomizedSearchCV
from sklearn.metrics import confusion_matrix, precision_score, recall_score, precision_recall_curve,f1_score, roc_auc_score, roc_curve, log_loss,classification_report

from ipywidgets import interactive

pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 500)

import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

```
msno.bar(train_df, color = 'r',fontsize=12,figsize = (8, 5))
```

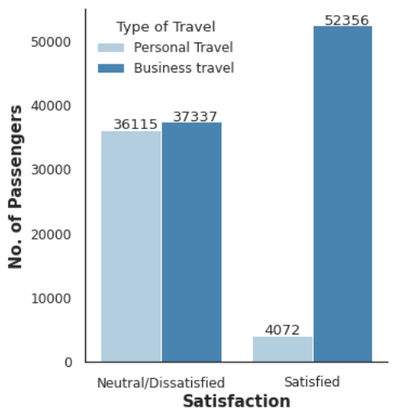
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f957d78bca0>
```



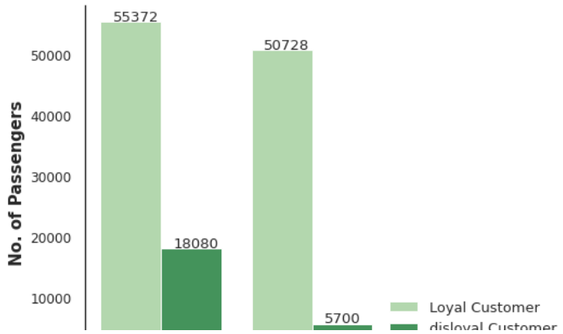
```
sns.set(style='white',font_scale=1.1)
fig = plt.figure(figsize=[5,6])
ax = sns.countplot(data=train_df,x='Satisfaction',palette='coolwarm')
ax.set_xticklabels(['Neutral/Dissatisfied','Satisfied'])
for p in ax.patches:
        ax.annotate(str(p.get_height())+' ('+str((p.get_height()/len(train_df)*100).round(1))+'%)', (p.get_x()+0.1, p.get_height()+400))
plt.xlabel('Satisfaction',weight='bold',fontsize='15')
plt.ylabel('No. of Passengers',weight='bold',fontsize='15')
sns.despine()
plt.savefig('targetplot1.png',transparent=True, bbox_inches='tight')
```



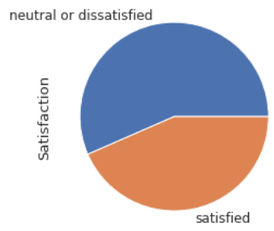

```
sns.set(style='white',font_scale=1.1)
fig = plt.figure(figsize=[5,6])
ax = sns.countplot(data=train_df,x='Satisfaction',hue='Type of Travel',palette='Blues')
ax.set_xticklabels(['Neutral/Dissatisfied','Satisfied'])
for p in ax.patches:
        ax.annotate(p.get_height(), (p.get_x()+0.08, p.get_height()+200))
plt.xlabel('Satisfaction',weight='bold',fontsize='15')
plt.ylabel('No. of Passengers',weight='bold',fontsize='15')
sns.despine()
# plt.savefig('targetplot2.png',transparent=True, bbox_inches='tight')
```



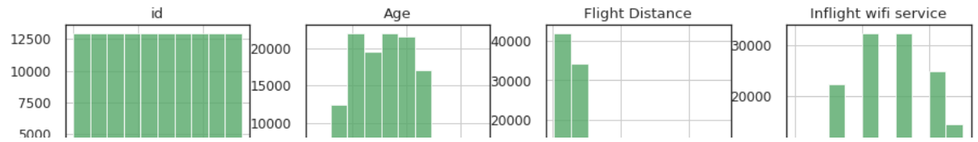

```
sns.set(style='white',font_scale=1.1)
fig = plt.figure(figsize=[5,6])
ax = sns.countplot(data=train_df,x='Satisfaction',hue='Customer Type',palette='Greens')
ax.set_xticklabels(['Neutral/Dissatisfied','Satisfied'])
for p in ax.patches:
        ax.annotate(p.get_height(), (p.get_x()+0.08, p.get_height()+200))
plt.xlabel('Satisfaction',weight='bold',fontsize='15')
plt.ylabel('No. of Passengers',weight='bold',fontsize='15')
plt.legend(loc="upper right", bbox_to_anchor=(1.6, 0.2),fontsize=13)
sns.despine()
# plt.savefig('targetplot3.png',transparent=True, bbox_inches='tight')
```




```
train_df['Satisfaction'].value_counts().head(10).plot.pie()
import matplotlib.pyplot as plt
plt.gca().set_aspect('equal')
```




```
hist = train_df.hist(figsize = (15, 22), layout=(6, 4) , color='g',alpha=0.8 )
```




```
df1 = pd.get_dummies(train_df,columns=['Gender','Customer Type','Type of Travel','Class'],drop_first=True)
df1
```

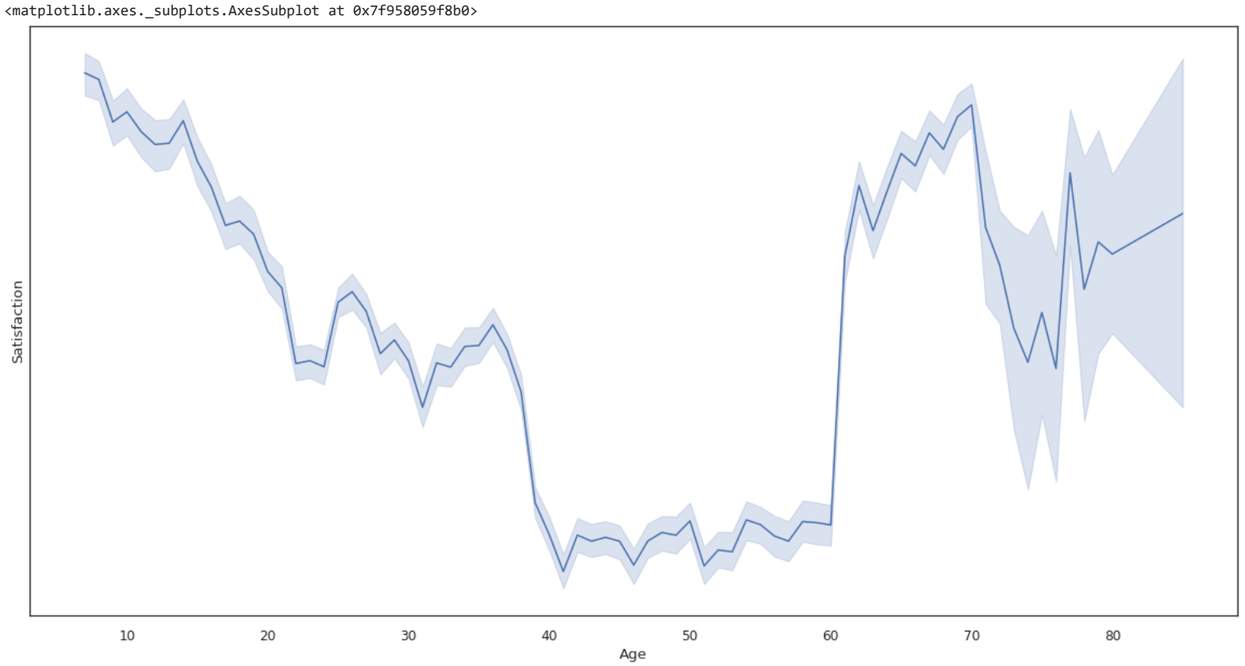
	id	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Seat comfort	Inflight entertainment	On-board service	Leg room service	Baggage handling	Checkin service	Inflight service	Cleanliness	Departure Delay in Minutes	Arrival Delay in Minutes	Satisfaction	Gender_Male	Customer Type_disloyal Customer	Type of Travel_Personal Travel	Class_Eco	Class_Eco Plus
0	70172	13	460	3	4	3	1	5	3	5	5	4	3	4	4	5	5	25	18.0	neutral or dissatisfied	1	0	1	0	1
1	5047	25	235	3	2	3	3	1	3	1	1	1	5	3	1	4	1	1	6.0	neutral or dissatisfied	1	1	0	0	0
2	110028	26	1142	2	2	2	2	5	5	5	5	4	3	4	4	4	5	0	0.0	satisfied	0	0	0	0	0
3	24026	25	562	2	5	5	5	2	2	2	2	2	5	3	1	4	2	11	9.0	neutral or dissatisfied	0	0	0	0	0
4	119299	61	214	3	3	3	3	4	5	5	3	3	4	4	3	3	3	0	0.0	satisfied	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
129875	78463	34	526	3	3	3	1	4	3	4	4	3	2	4	4	5	4	0	0.0	neutral or dissatisfied	1	1	0	0	0
129876	71167	23	646	4	4	4	4	4	4	4	4	4	5	5	5	5	4	0	0.0	satisfied	1	0	0	0	0
129877	37675	17	828	2	5	1	5	2	1	2	2	4	3	4	5	4	2	0	0.0	neutral or dissatisfied	0	0	1	1	0
129878	90086	14	1127	3	3	3	3	4	4	4	4	3	2	5	4	5	4	0	0.0	satisfied	1	0	0	0	0
129879	34799	42	264	2	5	2	5	4	2	2	1	1	2	1	1	1	1	0	0.0	neutral or dissatisfied	0	0	1	1	0

129880 rows × 25 columns

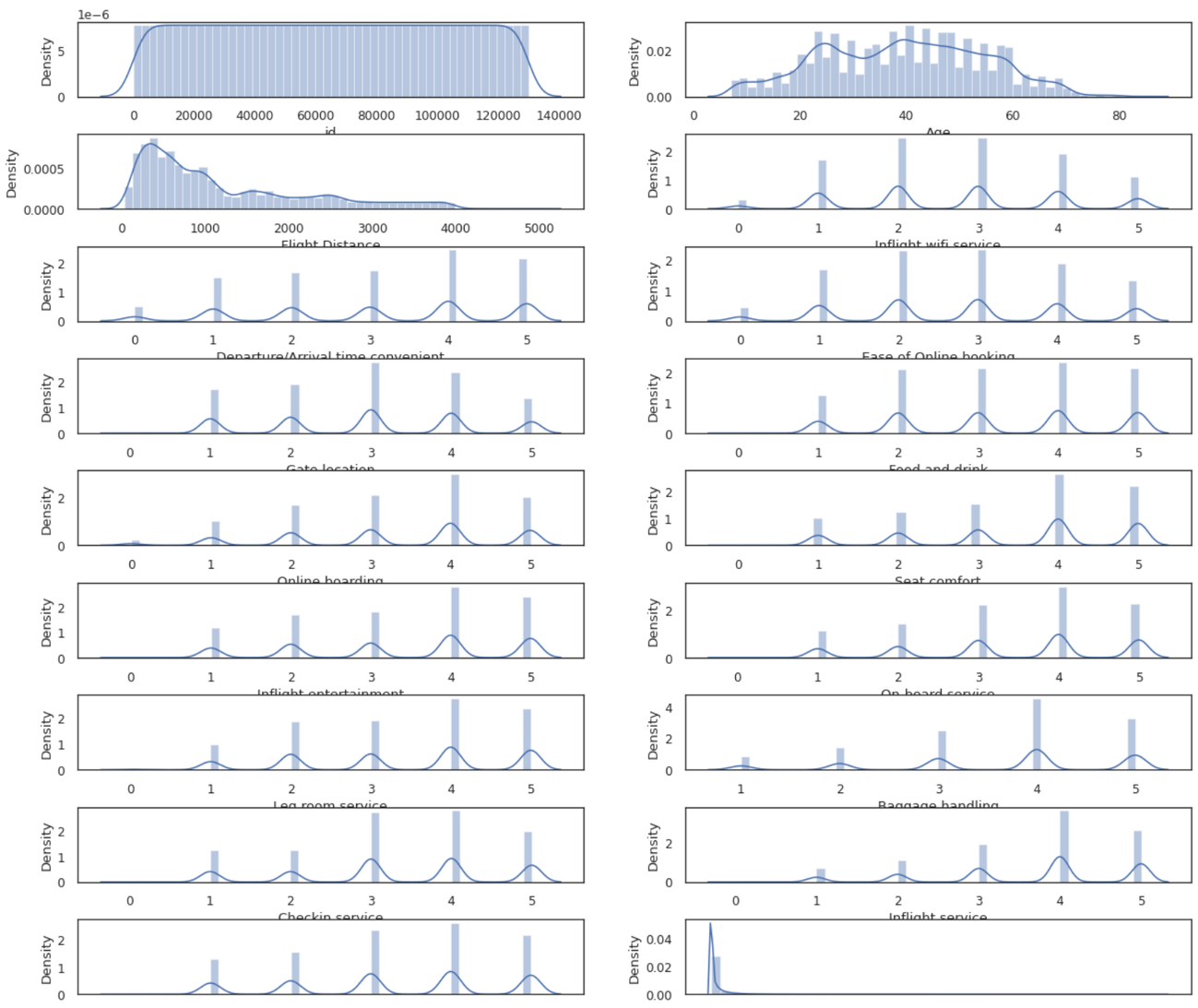
Cleanliness Departure Delay in Minutes Arrival Delay in Minutes

```
plt.rcParams['figure.figsize']=(20,10)
sns.lineplot(x='Age',y='Satisfaction',data = train_df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f958059f8b0>
```



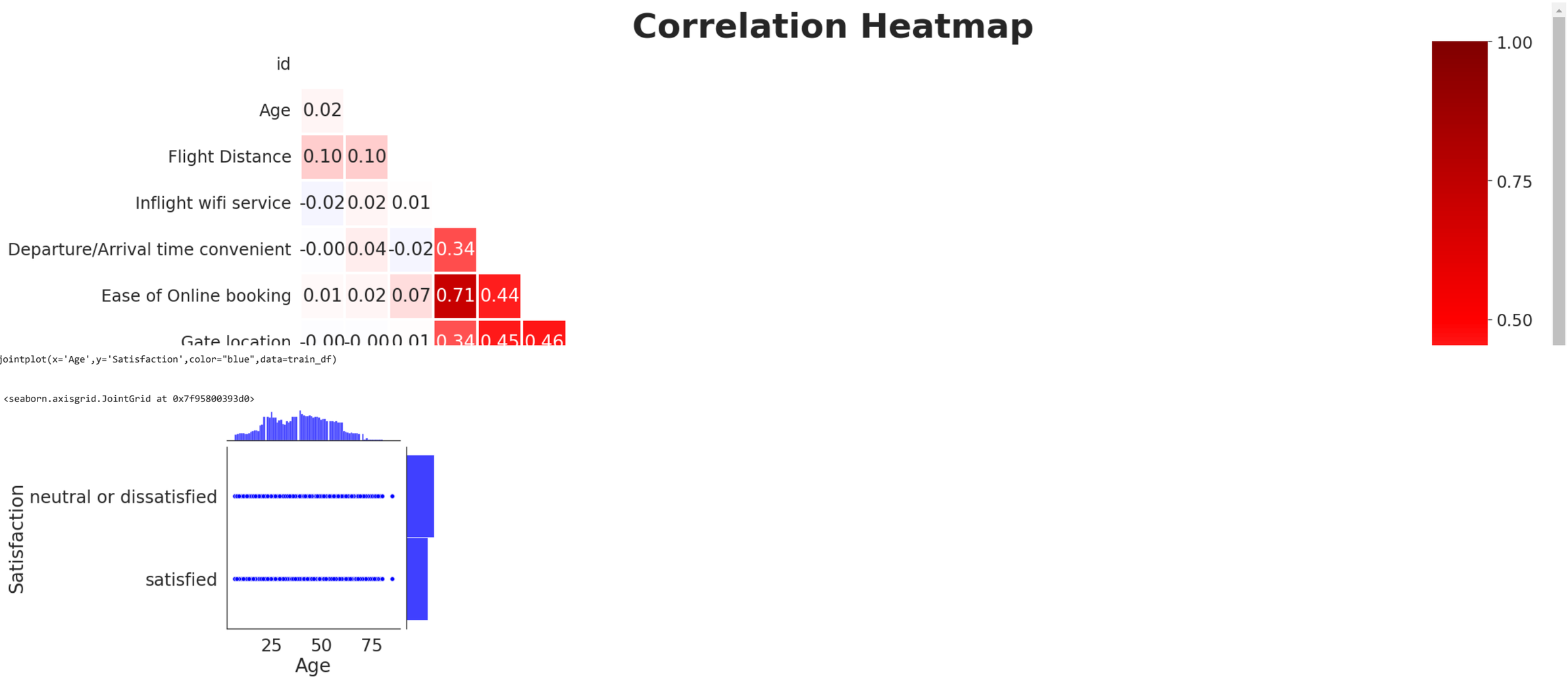
```
def plot_numerical_columns(train_df):
    train_df = train_df[train_df.select_dtypes([np.int64,np.float64]).columns]
    lcols = 2
    lrows = math.ceil(len(train_df.columns)/lcols)
    plt.figure(figsize=(20,20))
    plt.subplots_adjust(wspace=0.2, hspace=0.5)
    for i, column in enumerate(train_df.columns):
        plt.subplot(lrows,lcols,i+1)
        sns.distplot(train_df[column])


plot_numerical_columns(train_df)
```




```
corr_matrix = df1.corr()
corr_matrix
```

	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	Gender_Male	Customer Type_disloyal Customer	Type of Travel_Personal Travel	Class_Eco	Class
id	1.000000	0.020322	0.095504	-0.023096	-0.002192	0.013400	-0.000113	-0.000510	0.055538	0.052164	0.001620	0.055502	0.044088	0.074569	0.079325	0.078793	0.024048	-0.017643	-0.035405	-0.001361	0.001467	-0.000935	-0.082223	-0.04
Age	0.020322	1.000000	0.099459	0.016116	0.036960	0.022565	-0.000398	0.023194	0.207572	0.159136	0.074947	0.057078	0.039119	-0.047991	0.033475	-0.051347	0.052565	-0.009041	-0.011206	0.008996	-0.284172	-0.044808	-0.132597	-0.01
Flight Distance	0.095504	0.099459	1.000000	0.006701	-0.018914	0.065165	0.005520	0.057066	0.214825	0.157662	0.130507	0.111194	0.134533	0.064855	0.073608	0.059316	0.095648	0.002402	-0.001973	0.003616	-0.226021	-0.266792	-0.403881	-0.12
Inflight wifi service	-0.023096	0.016116	0.006701	1.000000	0.344915	0.714807	0.338573	0.132214	0.457445	0.121513	0.207802	0.119928	0.160317	0.120376	0.043762	0.110029	0.131300	-0.015946	-0.017762	0.005901	-0.005757	-0.105865	-0.037176	0.00
Departure/Arrival time convenient	-0.002192	0.036960	-0.018914	0.344915	1.000000	0.437620	0.447510	0.000687	0.072287	0.008666	-0.008380	0.067297	0.010617	0.070833	0.091132	0.072195	0.009862	0.000778	-0.001005	0.008995	-0.206873	0.257102	0.079799	0.02
Ease of Online booking	0.013400	0.022565	0.065165	0.714807	0.437620	1.000000	0.460041	0.030514	0.404866	0.028561	0.046564	0.039064	0.109450	0.039148	0.008819	0.035373	0.015125	-0.005318	-0.007046	0.005893	-0.018059	-0.134078	-0.099132	-0.01
Gate location	-0.000113	-0.000398	0.005520	0.338573	0.447510	0.460041	1.000000	-0.002872	0.002756	0.002788	0.002741	-0.029019	-0.005181	0.000972	-0.039353	0.000310	-0.005918	0.005973	0.005643	-0.000863	0.004465	-0.029869	-0.005188	-0.00
Food and drink	-0.000510	0.023194	0.057066	0.132214	0.000687	0.030514	-0.002872	1.000000	0.233500	0.575846	0.623461	0.057404	0.033173	0.035321	0.085198	0.035210	0.658054	-0.029164	-0.031685	0.001730	-0.056997	-0.068986	-0.080233	-0.01
Online boarding	0.055538	0.207572	0.214825	0.457445	0.072287	0.404866	0.002756	0.233500	1.000000	0.419253	0.283922	0.154242	0.123225	0.083541	0.204238	0.074058	0.329377	-0.019404	-0.022663	-0.045022	-0.189083	-0.224020	-0.292662	-0.07
Seat comfort	0.052164	0.159136	0.157662	0.121513	0.008666	0.028561	0.002788	0.575846	0.419253	1.000000	0.611837	0.130545	0.104272	0.074620	0.189979	0.068842	0.679613	-0.027999	-0.030397	-0.030756	-0.156239	-0.127717	-0.204940	-0.05
Inflight entertainment	0.001620	0.074947	0.130507	0.207802	-0.008380	0.046564	0.002741	0.623461	0.283922	0.611837	1.000000	0.418574	0.300397	0.379123	0.119554	0.406094	0.692511	-0.027012	-0.030183	0.003843	-0.106001	-0.152936	-0.176933	-0.04
On-board service	0.055502	0.057078	0.111194	0.119928	0.067297	0.039064	-0.029019	0.057404	0.154242	0.130545	0.418574	1.000000	0.357721	0.520296	0.244619	0.551569	0.122084	-0.030486	-0.034671	0.006447	-0.054172	-0.059794	-0.184657	-0.07
Leg room service	0.044088	0.039119	0.134533	0.160317	0.010617	0.109450	-0.005181	0.033173	0.123225	0.104272	0.300397	0.357721	1.000000	0.371455	0.152693	0.369569	0.096695	0.014574	0.011384	0.031047	-0.046841	-0.139612	-0.183162	-0.06
Baggage handling	0.074569	-0.047991	0.064855	0.120376	0.070833	0.039148	0.000972	0.035321	0.083541	0.074620	0.379123	0.520296	0.371455	1.000000	0.234503	0.629237	0.097071	-0.004105	-0.007997	0.036356	0.024874	-0.033012	-0.138829	-0.06
Checkin service	0.079325	0.033475	0.073608	0.043762	0.091132	0.008819	-0.039353	0.085198	0.204238	0.189979	0.119554	0.244619	0.152693	0.234503	1.000000	0.237601	0.176658	-0.018752	-0.021675	0.008462	-0.031243	0.016247	-0.129629	-0.06
Inflight service	0.078793	-0.051347	0.059316	0.110029	0.072195	0.035373	0.000310	0.035210	0.074058	0.068842	0.406094	0.551569	0.369569	0.629237	0.237601	1.000000	0.090356	-0.054432	-0.059685	0.038504	0.023292	-0.023538	-0.134774	-0.06
Cleanliness	0.024048	0.052565	0.095648	0.131300	0.009862	0.015125	-0.005918	0.658054	0.329377	0.679613	0.692511	0.122084	0.096695	0.097071	0.176658	0.090356	1.000000	-0.014543	-0.016547	0.002867	-0.081302	-0.084615	-0.124709	-0.03
Departure Delay in Minutes	-0.017643	-0.009041	0.002402	-0.015946	0.000778	-0.005318	0.005973	-0.029164	-0.019404	-0.027999	-0.027012	-0.030486	0.014574	-0.004105	-0.018752	-0.054432	-0.014543	1.000000	0.959382	0.003491	0.003859	-0.005913	0.008775	0.00
Arrival Delay in Minutes	-0.035405	-0.011206	-0.001973	-0.017762	-0.001005	-0.007046	0.005643	-0.031685	-0.022663	-0.030397	-0.030183	-0.034671	0.011384	-0.007997	-0.021675	-0.059685	-0.016547	0.959382	1.000000	0.001286	0.004769	-0.005972	0.012184	0.00
Gender_Male	-0.001361	0.008996	0.003616	0.005901	0.008995	0.005893	-0.000863	0.001730	-0.045022	-0.030756	0.003843	0.006447	0.031047	0.036356	0.008462	0.038504	0.002867	0.003491	0.001286	1.000000	-0.030958	0.009503	-0.002631	-0.01
Customer																								

```
sns.set(style='white',font_scale=2.2)
fig = plt.figure(figsize=[35,30])
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
cmap = sns.diverging_palette(150, 0, as_cmap=True)
sns.heatmap(corr_matrix,cmap='seismic',linewidth=3,linecolor='white',vmax = 1, vmin=-1,mask=mask, annot=True,fmt='0.2f')
plt.title('Correlation Heatmap', weight='bold',fontsize=50)
plt.savefig('heatmap.png',transparent=True, bbox_inches='tight')
```

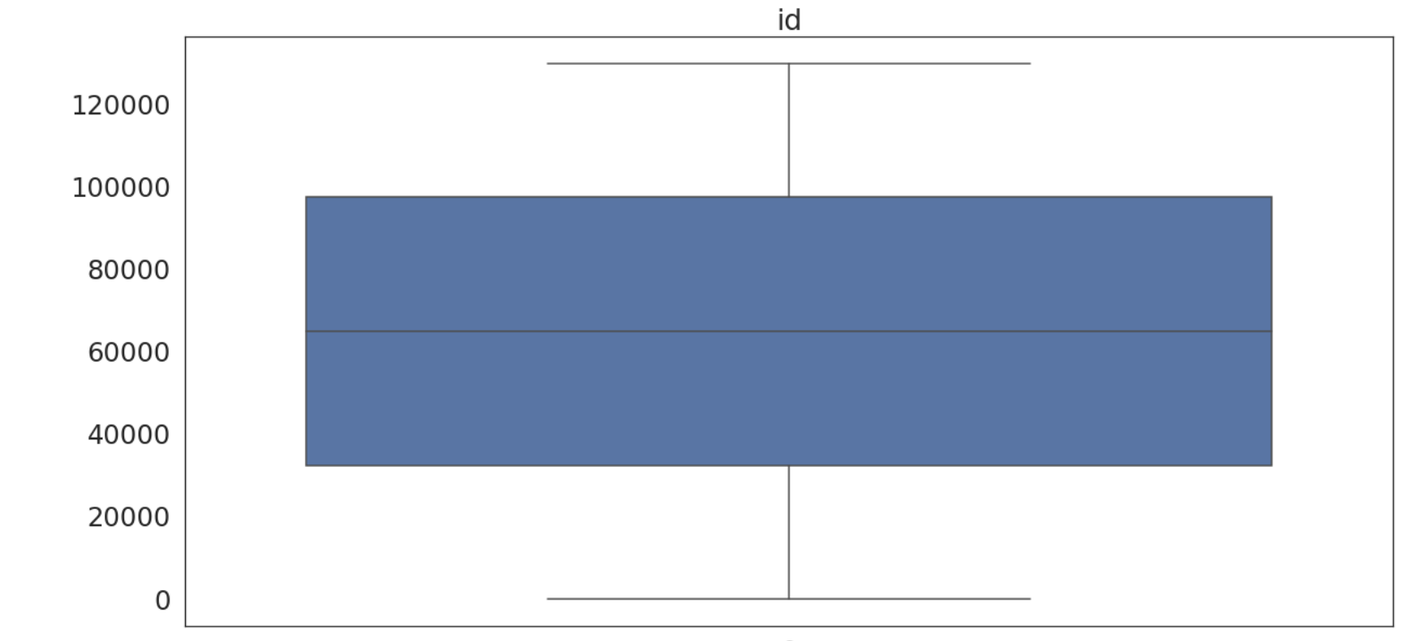



```
sns.jointplot(x='Age',y='Satisfaction',color="blue",data=train_df)
```

```
<seaborn.axisgrid.JointGrid at 0x7f95800393d0>
```


FEATURE ENGINEERING

```
for col in train_df.select_dtypes(exclude = 'object').keys().to_list():
    sns.boxplot(data = train_df[col])
    plt.title(col)
    plt.show()
```




```
Q1 = np.percentile(train_df['Flight Distance'], 25,
                   interpolation = 'midpoint')

Q3 = np.percentile(train_df['Flight Distance'], 75,
                   interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", train_df.shape)
```

```
# Upper bound
upper = np.where(train_df['Flight Distance'] >= (Q3+1.5*IQR))
# Lower bound
lower = np.where(train_df['Flight Distance'] <= (Q1-1.5*IQR))

''' Removing the Outliers '''
Df = train_df.copy()
Df.drop(upper[0], inplace = True)
Df.drop(lower[0], inplace = True)
print("New Shape: ", Df.shape)
```

```
Old Shape:  (129880, 24)
New Shape:  (127016, 24)
```

```
sns.boxplot(train_df["Flight Distance"])
```

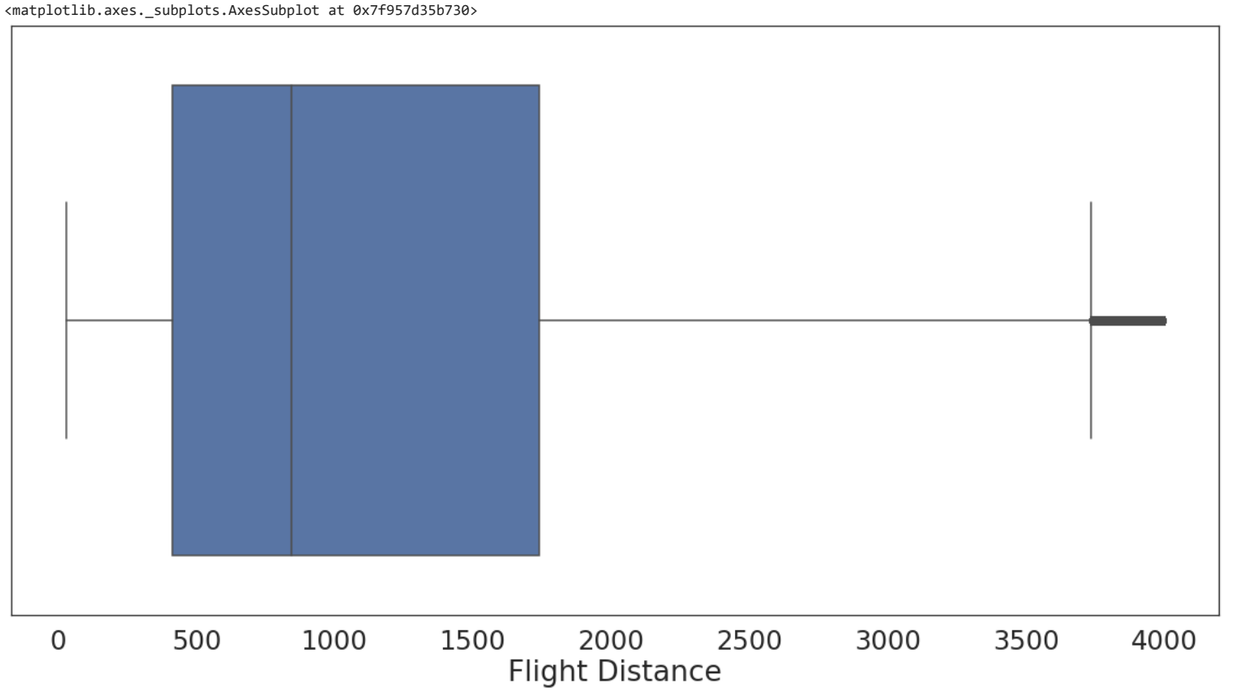
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f957c748a60>
```

```
# Flight Distance has outliers
train_df["Flight Distance"] = train_df["Flight Distance"]
```

```
train_df = train_df[(train_df["Flight Distance"]>0)&(train_df["Flight Distance"]<4000)]
```

```
sns.boxplot(train_df["Flight Distance"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f957d35b730>
```



LABEL ENCODING

```
train_df["Gender"] = train_df["Gender"].map({"Male":0,"Female":1})
train_df["Customer Type"] = train_df["Customer Type"].map({"Loyal Customer":0,"disloyal Customer":1})
train_df["Satisfaction"] = train_df["Satisfaction"].map({"neutral or dissatisfied":0,"satisfied":1})
train_df["Type of Travel"] = train_df["Type of Travel"].map({'Personal Travel':0,'Business travel':1})
train_df["Class"] = train_df["Class"].map({"Eco Plus":0,"Business":1,"Eco":2})
```

```
train_df
```

	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
0	70172	0	0	13	0	0	460	3	4	3	1	5	3	5	5	4	3	4	4	5	5	25	18.0	0
1	5047	0	1	25	1	1	235	3	2	3	3	1	3	1	1	1	5	3	1	4	1	1	6.0	0
2	110028	1	0	26	1	1	1142	2	2	2	2	5	5	5	5	4	3	4	4	4	5	0	0.0	1
3	24026	1	0	25	1	1	562	2	5	5	5	2	2	2	2	2	5	3	1	4	2	11	9.0	0
4	119299	0	0	61	1	1	214	3	3	3	3	4	5	5	3	3	4	4	3	3	3	0	0.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

```
cor = train_df.corr()
```

```
irrelevant = []
for i in train_df.columns:
    for j in train_df.columns:
        if i != j:
            if cor[i][j] >= 0.8:
                irrelevant.append([i,j])
print(irrelevant)
```

```
[['Departure Delay in Minutes', 'Arrival Delay in Minutes'], ['Arrival Delay in Minutes', 'Departure Delay in Minutes']]
```

```
df1
```

	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	Satisfaction	Gender_Male	Customer Type_disloyal Customer	Type of Travel_Personal Travel	Class_Eco	Class_Eco Plus
0	70172	13	460	3	4	3	1	5	3	5	5	4	3	4	4	5	5	25	18.0	neutral or dissatisfied	1	0	1	0	1
1	5047	25	235	3	2	3	3	1	3	1	1	1	5	3	1	4	1	1	6.0	neutral or dissatisfied	1	1	0	0	0
2	110028	26	1142	2	2	2	2	5	5	5	5	4	3	4	4	4	5	0	0.0	satisfied	0	0	0	0	0
3	24026	25	562	2	5	5	5	2	2	2	2	2	5	3	1	4	2	11	9.0	neutral or dissatisfied	0	0	0	0	0
4	119299	61	214	3	3	3	3	4	5	5	3	3	4	4	3	3	3	0	0.0	satisfied	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
129875	78463	34	526	3	3	3	1	4	3	4	4	3	2	4	4	5	4	0	0.0	neutral or dissatisfied	1	1	0	0	0
129876	71167	23	646	4	4	4	4	4	4	4	4	4	5	5	5	5	4	0	0.0	satisfied	1	0	0	0	0
129877	37675	17	828	2	5	1	5	2	1	2	2	4	3	4	5	4	2	0	0.0	neutral or dissatisfied	0	0	1	1	0
129878	90086	14	1127	3	3	3	3	4	4	4	4	3	2	5	4	5	4	0	0.0	satisfied	1	0	0	0	0
129879	34799	42	264	2	5	2	5	4	2	2	1	1	2	1	1	1	1	0	0.0	neutral or dissatisfied	0	0	1	1	0

129880 rows × 25 columns

```
df1 = df1.drop(['id','Departure Delay in Minutes', 'Arrival Delay in Minutes','Flight Distance','Age','Gate location','Departure/Arrival time convenient'],axis=1)
```

```
df1["Satisfaction"] = df1["Satisfaction"].map({"neutral or dissatisfied":0,"satisfied":1})
```

```
df1
```

	Inflight wifi service	Ease of Online booking	Food and drink	Online boarding	Seat comfort	Inflight entertainment	On-board service	Leg room service	Baggage handling	Checkin service	Inflight service	Cleanliness	Satisfaction	Gender_Male	Customer Type_disloyal Customer	Type of Travel_Personal Travel	Class_Eco	Class_Eco Plus
0	3	3	5	3	5	5	4	3	4	4	5	5	0	1	0	1	0	1
1	3	3	1	3	1	1	1	5	3	1	4	1	0	1	1	0	0	0
2	2	2	5	5	5	5	4	3	4	4	4	5	1	0	0	0	0	0
3	2	5	2	2	2	2	2	5	3	1	4	2	0	0	0	0	0	0
4	3	3	4	5	5	3	3	4	4	3	3	3	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
129875	3	3	4	3	4	4	3	2	4	4	5	4	0	1	1	0	0	0
129876	4	4	4	4	4	4	4	5	5	5	5	4	1	1	0	0	0	0

MODEL SELECTION

```
df1.isnull().sum()
```

```
Inflight wifi service             0
Ease of Online booking            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
Satisfaction                      0
Gender_Male                       0
Customer Type_disloyal Customer   0
Type of Travel_Personal Travel    0
Class_Eco                         0
Class_Eco Plus                    0
dtype: int64
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
import random
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import AgglomerativeClustering
```

```
df3=train_df.sample(frac=0.30)
```

```
y = df3['Satisfaction']
x = df3.drop('Satisfaction',axis=1)
```

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
```

```
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=42)
```

Intlight entertainment

Logistic

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()
clf.fit(x_train, y_train)
```

```
LogisticRegression()
```

```
y_pred = clf.predict(x_test)
```

```
logistic_preds = pd.DataFrame({"Predicted":y_pred,"Actual":y_test})
logistic_preds
```

	Predicted	Actual
0	1	1
1	1	1
2	1	0
3	0	0
4	1	0
...	...	...
7783	0	0
7784	0	1
7785	1	1
7786	1	0
7787	0	0

7788 rows × 2 columns

```
logistic_TP = len(logistic_preds[(logistic_preds["Predicted"]==logistic_preds["Actual"])&(logistic_preds["Predicted"]==1)])
logistic_FP = len(logistic_preds[(logistic_preds["Predicted"]!=logistic_preds["Actual"])&(logistic_preds["Predicted"]==1)])
logistic_FN = len(logistic_preds[(logistic_preds["Predicted"]!=logistic_preds["Actual"])&(logistic_preds["Predicted"]==0)])
logistic_TN = len(logistic_preds[(logistic_preds["Predicted"]==logistic_preds["Actual"])&(logistic_preds["Predicted"]==0)])
print(logistic_TP,logistic_FP,logistic_FN,logistic_TN)
print("Rightly Classified: ",(logistic_TP+logistic_TN),"/",(logistic_TP+logistic_FP+logistic_FN+logistic_TN))
print("Wrongly Classified: ",(logistic_FP+logistic_FN),"/",(logistic_TP+logistic_FP+logistic_FN+logistic_TN))

logistic_Accuracy = (logistic_TP+logistic_TN)/(logistic_TP+logistic_FP+logistic_FN+logistic_TN)
logistic_Precision = (logistic_TP)/(logistic_TP+logistic_FP)
logistic_Recall = (logistic_TP)/(logistic_TP+logistic_FN)
logistic_Specificity = (logistic_TN)/(logistic_TN+logistic_FP)
logistic_F1 = (2*logistic_Precision*logistic_Recall)/(logistic_Precision+logistic_Recall)
```

```
2335 1595 997 2861
Rightly Classified:  5196 / 7788
Wrongly Classified:  2592 / 7788
```

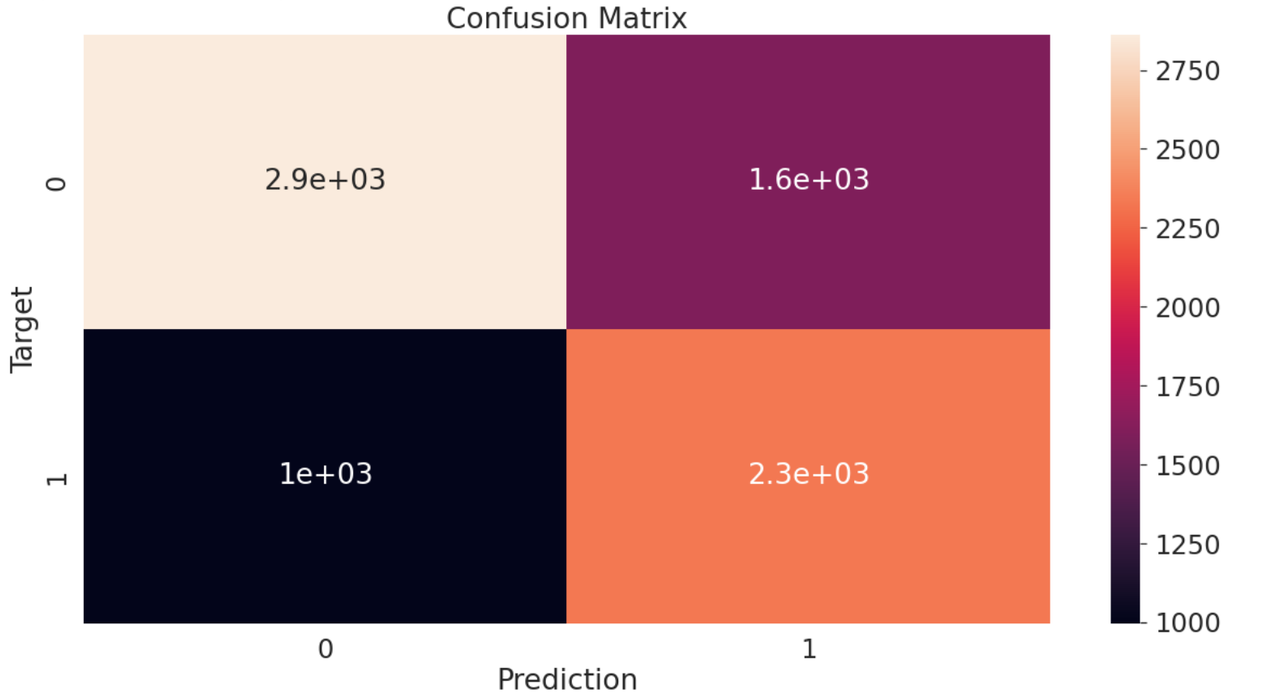
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
precision=precision_score(y_test,y_pred)
print("precision:",precision)
recall =recall_score(y_test,y_pred)
print("recall:",recall)
```

```
Accuracy: 66.72%
precision: 0.594147582697201
recall: 0.7007803121248499
```

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
cf = confusion_matrix(y_test, y_pred)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
```

```
plt.ylabel('Target')
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```




```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

NB

```
#Model and Training
model = GaussianNB()
model.fit(x_train, y_train)
nb1 =model.fit(x_train, y_train).predict(x_test)
```

```
nb_preds = pd.DataFrame({"Predicted":nb1,"Actual":y_test})
nb_preds
```

	Predicted	Actual
0	1	1
1	1	1
2	1	0
3	0	0
4	0	0
...	...	...
7783	0	0
7784	0	1
7785	1	1
7786	0	0
7787	0	0

7788 rows × 2 columns

```
nb_TP = len(nb_preds[(nb_preds["Predicted"]==nb_preds["Actual"])&(nb_preds["Predicted"]==1)])
nb_FP = len(nb_preds[(nb_preds["Predicted"]!=nb_preds["Actual"])&(nb_preds["Predicted"]==1)])
nb_FN = len(nb_preds[(nb_preds["Predicted"]!=nb_preds["Actual"])&(nb_preds["Predicted"]==0)])
nb_TN = len(nb_preds[(nb_preds["Predicted"]==nb_preds["Actual"])&(nb_preds["Predicted"]==0)])
```

```
print(nb_TP,nb_FP,nb_FN,nb_TN)
print("Rightly Classified: ",(nb_TP+nb_TN),"/",(nb_TP+nb_FP+nb_FN+nb_TN))
print("Wrongly Classified: ",(nb_FP+nb_FN),"/",(nb_TP+nb_FP+nb_FN+nb_TN))

nb_Accuracy = (nb_TP+nb_TN)/(nb_TP+nb_FP+nb_FN+nb_TN)
nb_Precision = (nb_TP)/(nb_TP+nb_FP)
nb_Recall = (nb_TP)/(nb_TP+nb_FN)
nb_Specificity = (nb_TN)/(nb_TN+nb_FP)
nb_F1 = (2*nb_Precision*nb_Recall)/(nb_Precision+nb_Recall)
```

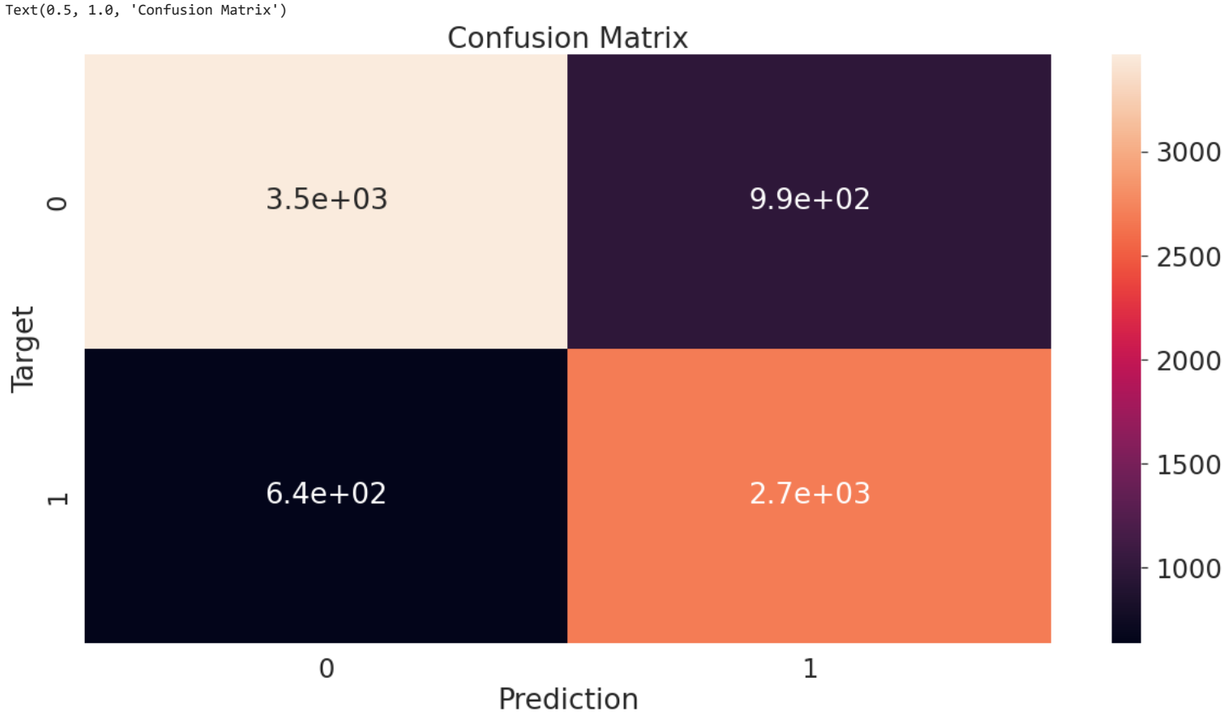
```
2693 989 639 3467
Rightly Classified:  6160 / 7788
Wrongly Classified:  1628 / 7788
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
accuracy = accuracy_score(y_test, nb1)
print("Accuracy: {:.2f}%".format(accuracy * 100))
precision=precision_score(y_test,nb1)
print("precision:",precision)
recall =recall_score(y_test,nb1)
print("recall:",recall)
```

```
Accuracy: 79.10%
precision: 0.7313959804454101
recall: 0.8082232893157263
```

```
from sklearn.metrics import confusion_matrix
cf = confusion_matrix(y_test, nb1)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```



SVM

```
model1= svm.SVC()
model1.fit(x_train, y_train)
```

```
y_pred2 = model1.fit(x_train, y_train).predict(x_test)

svc_preds = pd.DataFrame({"Predicted":y_pred2,"Actual":y_test})
svc_preds
```

	Predicted	Actual
0	0	1
1	0	1
2	1	0
3	0	0
4	0	0
...	...	...
7783	0	0
7784	0	1
7785	1	1
7786	0	0
7787	0	0

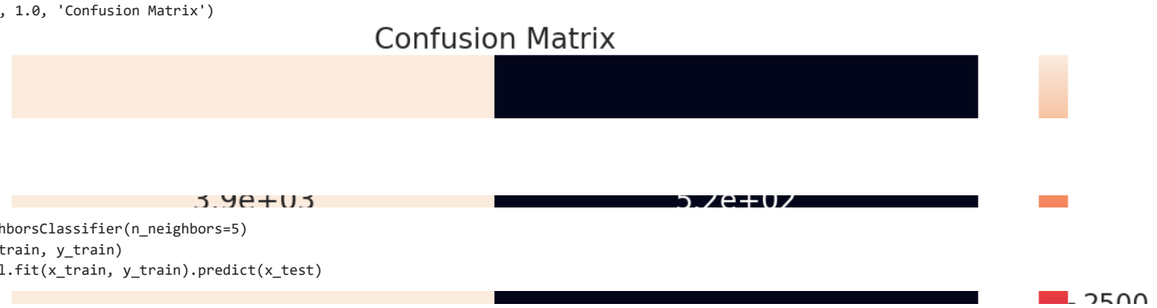
7788 rows × 2 columns

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
accuracy = accuracy_score(y_test, y_pred2)
print("Accuracy: {:.2f}%".format(accuracy * 100))
precision=precision_score(y_test,y_pred2)
print("precision:",precision)
recall =recall_score(y_test,y_pred2)
print("recall:",recall)
```

```
Accuracy: 64.93%
precision: 0.6828971393791844
recall: 0.336734693877551
```

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
cf = confusion_matrix(y_test, y_pred2)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```



KNN

```
model1= KNeighborsClassifier(n_neighbors=5)
model1.fit(x_train, y_train)
y_pred3 =model.fit(x_train, y_train).predict(x_test)
```

```
knn_preds = pd.DataFrame({"Predicted":y_pred3,"Actual":y_test})
knn_preds
```

	Predicted	Actual
0	1	1
1	1	1
2	1	0
3	0	0
4	0	0
...	...	...
7783	0	0
7784	0	1
7785	1	1
7786	0	0
7787	0	0

7788 rows × 2 columns

```
knn_TP = len(knn_preds[(knn_preds["Predicted"]==knn_preds["Actual"])&(knn_preds["Predicted"]==1)])
knn_FP = len(knn_preds[(knn_preds["Predicted"]!=knn_preds["Actual"])&(knn_preds["Predicted"]==1)])
knn_FN = len(knn_preds[(knn_preds["Predicted"]!=knn_preds["Actual"])&(knn_preds["Predicted"]==0)])
knn_TN = len(knn_preds[(knn_preds["Predicted"]==knn_preds["Actual"])&(knn_preds["Predicted"]==0)])
print(knn_TP,knn_FP,knn_FN,knn_TN)
print("Rightly Classified: ",(knn_TP+knn_TN),"/",(knn_TP+knn_FP+knn_FN+knn_TN))
print("Wrongly Classified: ",(knn_FP+knn_FN),"/",(knn_TP+knn_FP+knn_FN+knn_TN))

knn_Accuracy = (knn_TP+knn_TN)/(knn_TP+knn_FP+knn_FN+knn_TN)
knn_Precision = (knn_TP)/(knn_TP+knn_FP)
knn_Recall = (knn_TP)/(knn_TP+knn_FN)
knn_Specificity = (knn_TN)/(knn_TN+knn_FP)
knn_F1 = (2*knn_Precision*knn_Recall)/(knn_Precision+knn_Recall)
```

```
2693 989 639 3467
Rightly Classified:  6160 / 7788
Wrongly Classified:  1628 / 7788
```

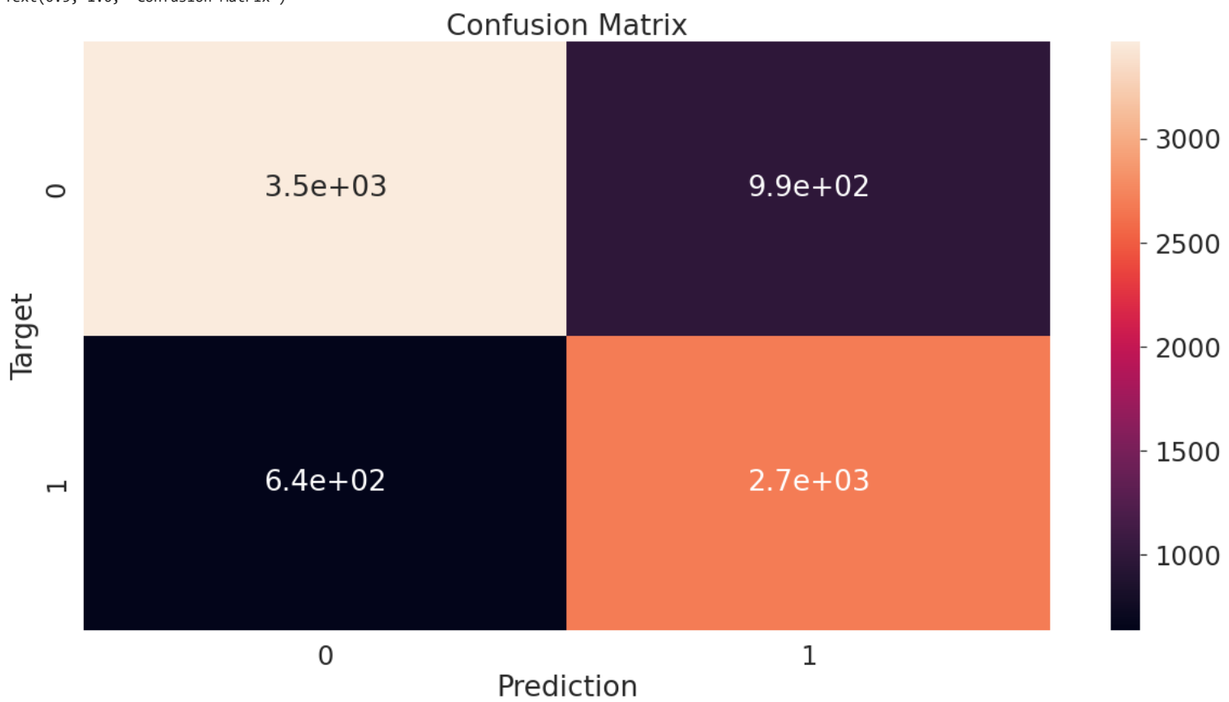
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
accuracy = accuracy_score(y_test, y_pred3)
print("Accuracy: {:.2f}%".format(accuracy * 100))
precision=precision_score(y_test,y_pred3)
print("precision:",precision)
recall =recall_score(y_test,y_pred3)
print("recall:",recall)
```

```
Accuracy: 79.10%
precision: 0.7313959804454101
recall: 0.8082232893157263
```

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
cf = confusion_matrix(y_test, y_pred3)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
```

```
plt.ylabel('Target')
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```



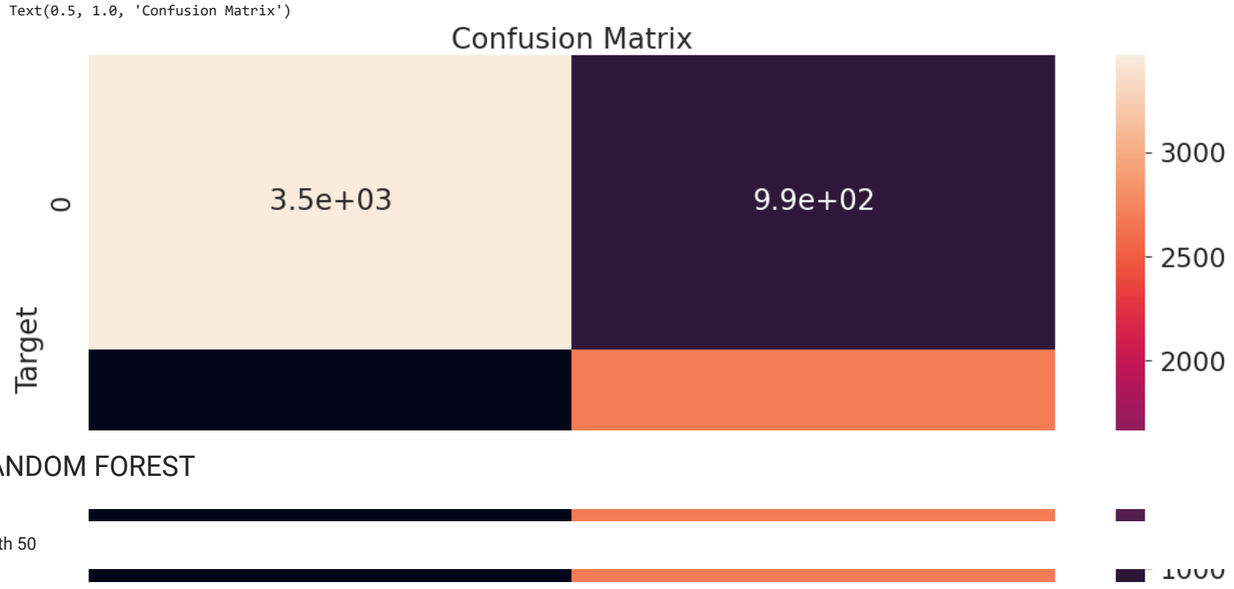

```
model1= KNeighborsClassifier(n_neighbors=11)
model1.fit(x_train, y_train)
y_pred4 =model.fit(x_train, y_train).predict(x_test)
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
accuracy = accuracy_score(y_test, y_pred4)
print("Accuracy: {:.2f}%".format(accuracy * 100))
precision=precision_score(y_test,y_pred4)
print("precision:",precision)
recall =recall_score(y_test,y_pred4)
print("recall:",recall)
```

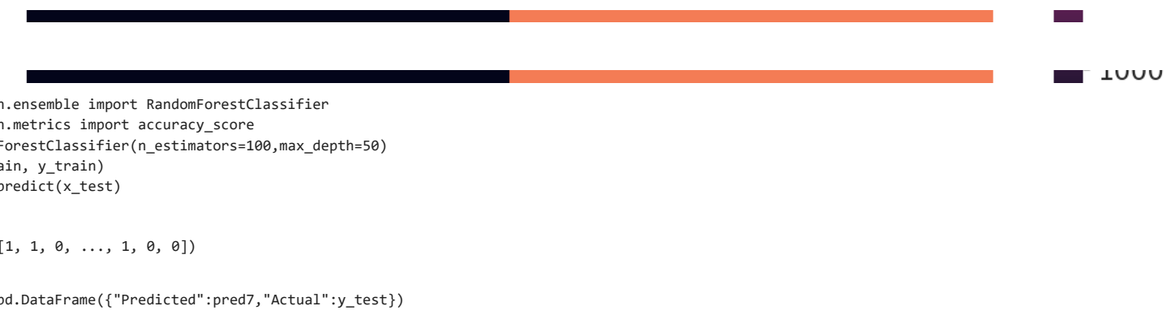
```
Accuracy: 79.10%
precision: 0.7313959804454101
recall: 0.8082232893157263
```

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
cf = confusion_matrix(y_test, y_pred4)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```



RANDOM FOREST




```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
rf = RandomForestClassifier(n_estimators=100,max_depth=50)
rf.fit(x_train, y_train)
pred7 = rf.predict(x_test)
pred7
```

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

```
rf_preds = pd.DataFrame({"Predicted":pred7,"Actual":y_test})
rf_preds
```

	Predicted	Actual
0	1	1
1	1	1
2	0	0
3	0	0
4	0	0
...	...	...
7783	0	0
7784	1	1
7785	1	1
7786	0	0
7787	0	0

7788 rows × 2 columns

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
accuracy = accuracy_score(y_test, pred7)
print("Accuracy: {:.2f}%".format(accuracy * 100))
precision=precision_score(y_test,pred7)
print("precision:",precision)
recall =recall_score(y_test,pred7)
print("recall:",recall)
```

```
Accuracy: 95.69%
precision: 0.9669576059850374
recall: 0.9309723889555822
```

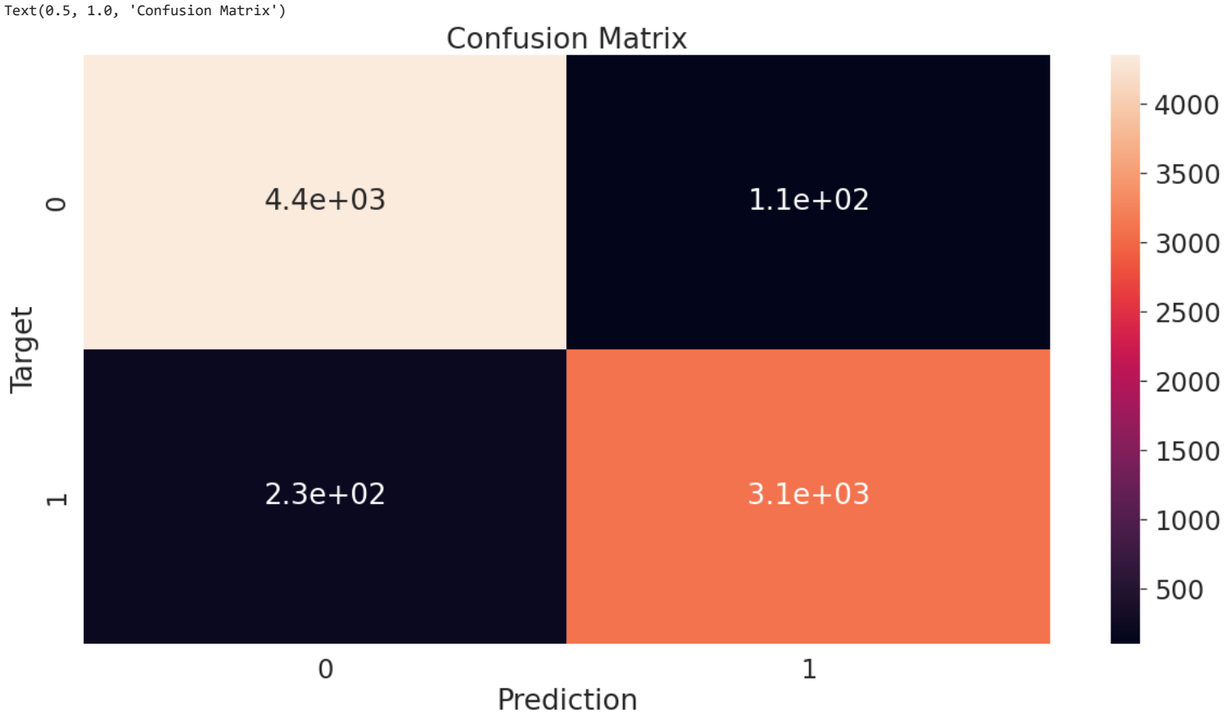
```
rf_TP = len(rf_preds[(rf_preds["Predicted"]==rf_preds["Actual"])&(rf_preds["Predicted"]==1)])
rf_FP = len(rf_preds[(rf_preds["Predicted"]!=rf_preds["Actual"])&(rf_preds["Predicted"]==1)])
```

```
rf_FN = len(rf_preds[(rf_preds["Predicted"]!=rf_preds["Actual"])&(rf_preds["Predicted"]==0)])
rf_TN = len(rf_preds[(rf_preds["Predicted"]==rf_preds["Actual"])&(rf_preds["Predicted"]==0)])
print(rf_TP,rf_FP,rf_FN,rf_TN)
print("Rightly Classified: ",(rf_TP+rf_TN),"/",(rf_TP+rf_FP+rf_FN+rf_TN))
print("Wrongly Classified: ",(rf_FP+rf_FN),"/",(rf_TP+rf_FP+rf_FN+rf_TN))

rf_Accuracy = (rf_TP+rf_TN)/(rf_TP+rf_FP+rf_FN+rf_TN)
rf_Precision = (rf_TP)/(rf_TP+rf_FP)
rf_Recall = (rf_TP)/(rf_TP+rf_FN)
rf_Specificity = (rf_TN)/(rf_TN+rf_FP)
rf_F1 = (2*rf_Precision*rf_Recall)/(rf_Precision+rf_Recall)
```

```
3102 106 230 4350
Rightly Classified:  7452 / 7788
Wrongly Classified:  336 / 7788
```

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
cf = confusion_matrix(y_test, pred7)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```



depth 100

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
rf = RandomForestClassifier(n_estimators=100,max_depth=100)
rf.fit(x_train, y_train)
pred7 = rf.predict(x_test)
pred7
```

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

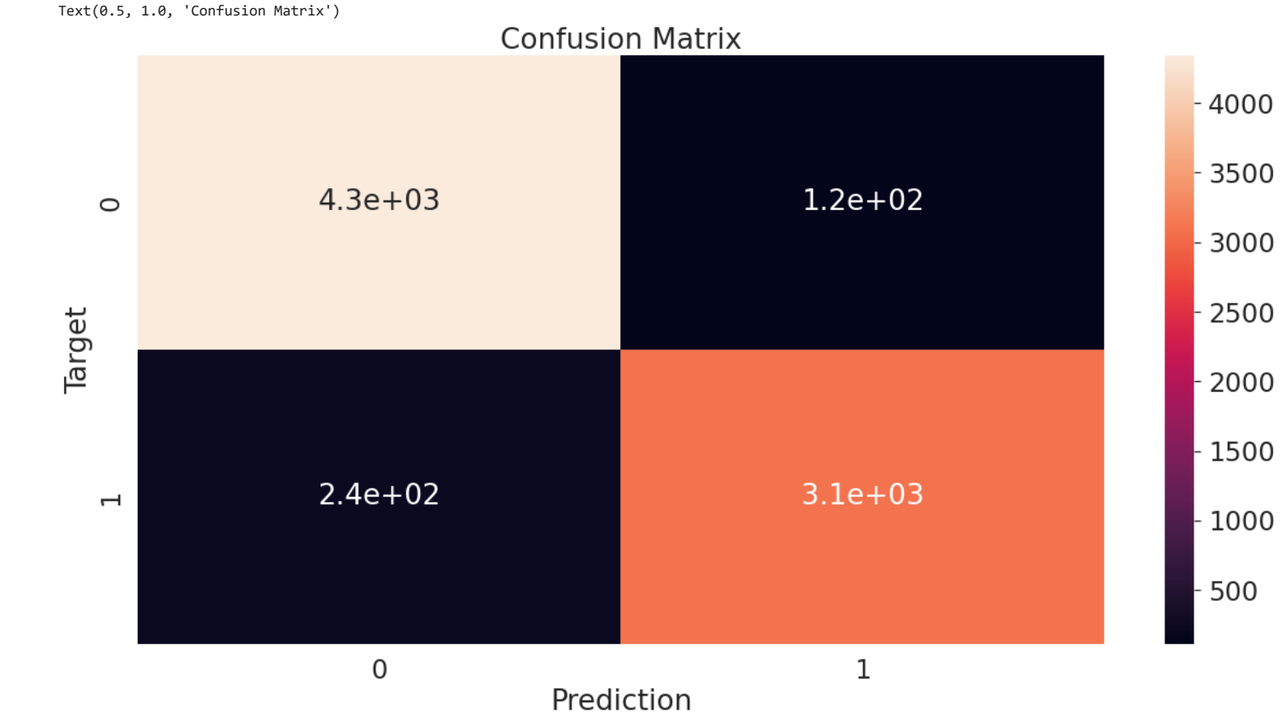
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
accuracy = accuracy_score(y_test, pred7)
print("Accuracy: {:.2f}%".format(accuracy * 100))
```

```
precision=precision_score(y_test,pred7)
print("precision:",precision)
recall =recall_score(y_test,pred7)
print("recall:",recall)
```

```
Accuracy: 95.49%
precision: 0.9641856119588913
recall: 0.929171668667467
```

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
cf = confusion_matrix(y_test, pred7)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```



depth 250

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
rf = RandomForestClassifier(n_estimators=100,max_depth=250)
rf.fit(x_train, y_train)
pred7 = rf.predict(x_test)
pred7
```

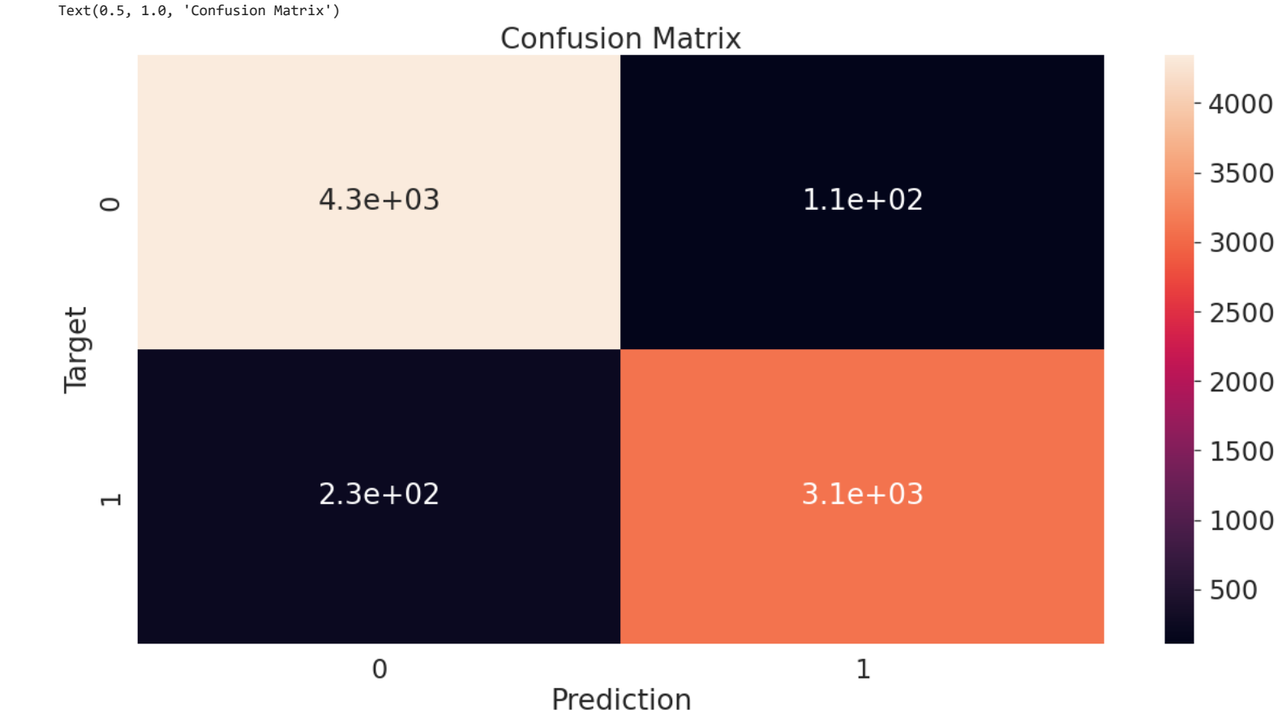
```
array([1, 1, 0, ..., 1, 0, 0])
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
accuracy = accuracy_score(y_test, pred7)
print("Accuracy: {:.2f}%".format(accuracy * 100))
precision=precision_score(y_test,pred7)
print("precision:",precision)
recall =recall_score(y_test,pred7)
print("recall:",recall)
```

```
Accuracy: 95.62%
precision: 0.965452847805789
recall: 0.9309723889555822
```

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
cf = confusion_matrix(y_test, pred7)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```



depth 500

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
rf = RandomForestClassifier(n_estimators=100,max_depth=500)
rf.fit(x_train, y_train)
pred7 = rf.predict(x_test)
pred7
```

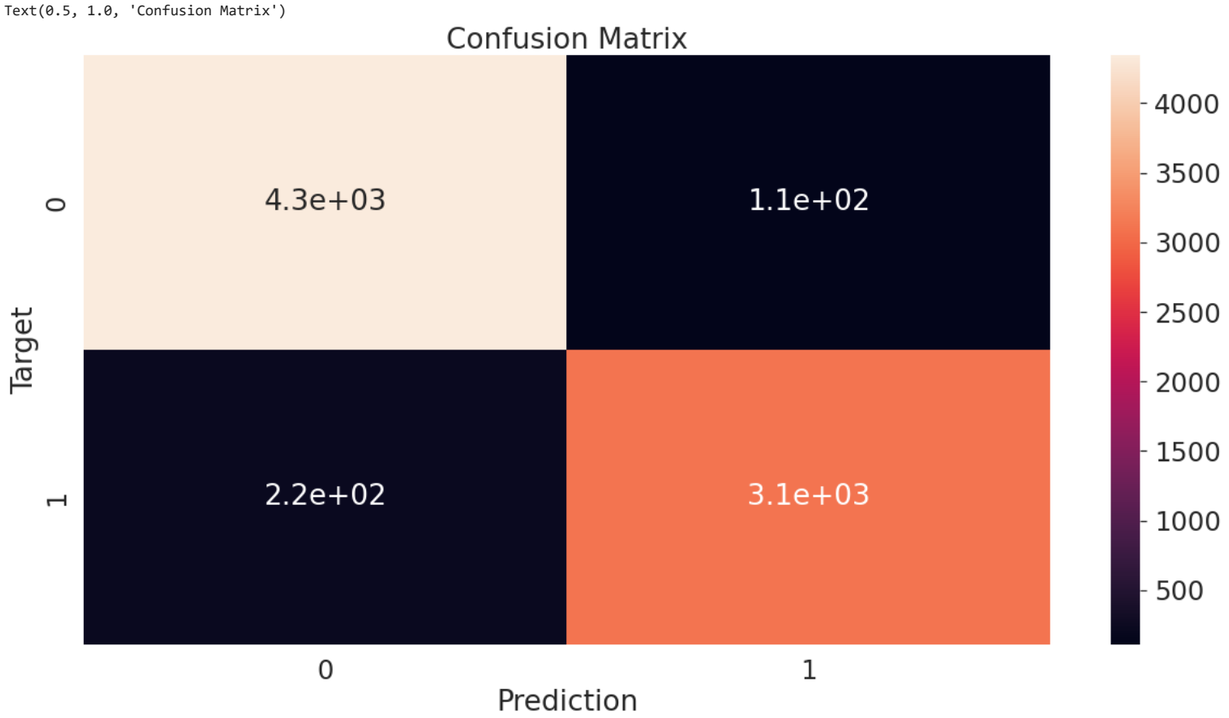
```
array([1, 1, 0, ..., 1, 0, 0])
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
accuracy = accuracy_score(y_test, pred7)
print("Accuracy: {:.2f}%".format(accuracy * 100))
precision=precision_score(y_test,pred7)
print("precision:",precision)
recall =recall_score(y_test,pred7)
print("recall:",recall)
```

```
Accuracy: 95.74%
precision: 0.9655493482309124
recall: 0.9336734693877551
```

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
cf = confusion_matrix(y_test, pred7)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```



Decision tree

```
y1 = df1['Satisfaction']
X1 = df1[['Ease of Online booking','Seat comfort','Gender_Male']]
```

```
from sklearn.model_selection import train_test_split
X1_train, X1_test, y1_train, y1_test= train_test_split(X1, y1, test_size=0.2, random_state=42)
```

```
from sklearn import tree
dc = tree.DecisionTreeClassifier()
dc.fit(X1_train, y1_train)
decision_cls = dc.predict(X1_test)
decision_cls
# clf.fit(x_train, y_train)
```

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

```
decision_preds = pd.DataFrame({"Predicted":decision_cls,"Actual":y1_test})
decision_preds
```

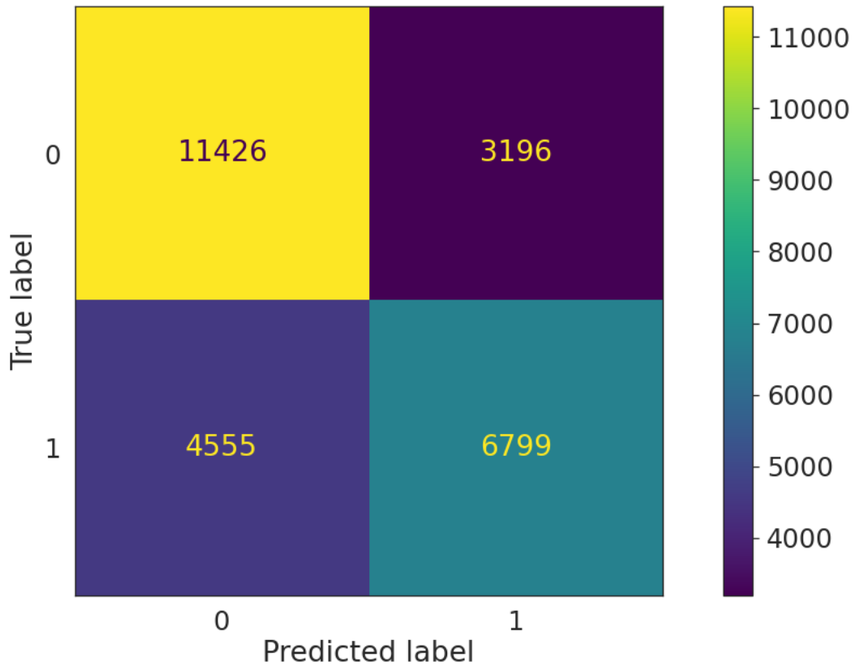
	Predicted	Actual
103044	0	0
43282	0	1
65543	0	0
65083	0	0
76496	1	1

```
decision_TP = len(decision_preds[(decision_preds["Predicted"]==decision_preds["Actual"])&(decision_preds["Predicted"]==1)])
decision_FP = len(decision_preds[(decision_preds["Predicted"]!=decision_preds["Actual"])&(decision_preds["Predicted"]==1)])
decision_FN = len(decision_preds[(decision_preds["Predicted"]!=decision_preds["Actual"])&(decision_preds["Predicted"]==0)])
decision_TN = len(decision_preds[(decision_preds["Predicted"]==decision_preds["Actual"])&(decision_preds["Predicted"]==0)])
print(decision_TP,decision_FP,decision_FN,decision_TN)
print("Rightly Classified: ",(decision_TP+decision_TN),"/",(decision_TP+decision_FP+decision_FN+decision_TN))
print("Wrongly Classified: ",(decision_FP+decision_FN),"/",(decision_TP+decision_FP+decision_FN+decision_TN))

decision_Accuracy = (decision_TP+decision_TN)/(decision_TP+decision_FP+decision_FN+decision_TN)
decision_Precision = (decision_TP)/(decision_TP+decision_FP)
decision_Recall = (decision_TP)/(decision_TP+decision_FN)
decision_Specificity = (decision_TN)/(decision_TN+decision_FP)
decision_F1 = (2*decision_Precision*decision_Recall)/(decision_Precision+decision_Recall)
```

```
6799 3196 4555 11426
Rightly Classified:  18225 / 25976
Wrongly Classified:  7751 / 25976
```

```
print("DECISION TREE")
print(confusion_matrix(y1_test, decision_cls))
cm = confusion_matrix(y1_test, decision_cls)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=dc.classes_)
disp.plot()
plt.show()
```

```
DECISION TREE
[[11426  3196]
 [ 4555  6799]]
```




```
from sklearn.tree import plot_tree
plt.figure(figsize=(90,30))
plot_tree(dc,class_names=[ '0','1'],feature_names=X1.columns, filled=True,fontsize=20)
```

```
[Text(0.5540081521739131, 0.9444444444444444, 'Seat comfort <= 3.5\ngini = 0.491\nsamples = 103904\nvalue = [58830, 45074]\nclass = 0'),
 Text(0.3111413043478261, 0.8333333333333334, 'Ease of Online booking <= 3.5\ngini = 0.343\nsamples = 45642\nvalue = [35623, 10019]\nclass = 0'),
 Text(0.14945652173913043, 0.7222222222222222, 'Ease of Online booking <= 0.5\ngini = 0.245\nsamples = 31836\nvalue = [27292, 4544]\nclass = 0'),
 Text(0.07065217391304347, 0.6111111111111112, 'Seat comfort <= 2.5\ngini = 0.483\nsamples = 2130\nvalue = [871, 1259]\nclass = 1'),
 Text(0.043478260869565216, 0.5, 'Gender_Male <= 0.5\ngini = 0.47\nsamples = 1290\nvalue = [488, 802]\nclass = 1'),
 Text(0.021739130434782608, 0.3888888888888889, 'Seat comfort <= 1.5\ngini = 0.452\nsamples = 604\nvalue = [208, 396]\nclass = 1'),
 Text(0.010869565217391304, 0.2777777777777778, 'gini = 0.437\nsamples = 276\nvalue = [89, 187]\nclass = 1'),
 Text(0.03260869565217391, 0.2777777777777778, 'gini = 0.462\nsamples = 328\nvalue = [119, 209]\nclass = 1'),
 Text(0.06521739130434782, 0.3888888888888889, 'Seat comfort <= 1.5\ngini = 0.483\nsamples = 686\nvalue = [280, 406]\nclass = 1'),
 Text(0.05434782608695652, 0.2777777777777778, 'gini = 0.477\nsamples = 316\nvalue = [124, 192]\nclass = 1'),
 Text(0.07608695652173914, 0.2777777777777778, 'gini = 0.488\nsamples = 370\nvalue = [156, 214]\nclass = 1'),
 Text(0.09782608695652174, 0.5, 'Gender_Male <= 0.5\ngini = 0.496\nsamples = 840\nvalue = [383, 457]\nclass = 1'),
 Text(0.08695652173913043, 0.3888888888888889, 'gini = 0.494\nsamples = 446\nvalue = [198, 248]\nclass = 1'),
 Text(0.10869565217391304, 0.3888888888888889, 'gini = 0.498\nsamples = 394\nvalue = [185, 209]\nclass = 1'),
 Text(0.22826086956521738, 0.6111111111111112, 'Gender_Male <= 0.5\ngini = 0.197\nsamples = 29706\nvalue = [26421, 3285]\nclass = 0'),
 Text(0.15760869565217392, 0.5, 'Ease of Online booking <= 1.5\ngini = 0.219\nsamples = 14646\nvalue = [12815, 1831]\nclass = 0'),
 Text(0.13043478260869565, 0.3888888888888889, 'Seat comfort <= 2.5\ngini = 0.264\nsamples = 3753\nvalue = [3166, 587]\nclass = 0'),
 Text(0.11956521739130435, 0.2777777777777778, 'Seat comfort <= 1.5\ngini = 0.275\nsamples = 2106\nvalue = [1760, 346]\nclass = 0'),
 Text(0.10869565217391304, 0.16666666666666666, 'gini = 0.258\nsamples = 938\nvalue = [795, 143]\nclass = 0'),
 Text(0.13043478260869565, 0.16666666666666666, 'gini = 0.287\nsamples = 1168\nvalue = [965, 203]\nclass = 0'),
 Text(0.14130434782608695, 0.2777777777777778, 'gini = 0.25\nsamples = 1647\nvalue = [1406, 241]\nclass = 0'),
 Text(0.18478260869565216, 0.3888888888888889, 'Ease of Online booking <= 2.5\ngini = 0.202\nsamples = 10893\nvalue = [9649, 1244]\nclass = 0'),
 Text(0.16304347826086957, 0.2777777777777778, 'Seat comfort <= 1.5\ngini = 0.193\nsamples = 5420\nvalue = [4834, 586]\nclass = 0'),
 Text(0.15217391304347827, 0.16666666666666666, 'gini = 0.166\nsamples = 1427\nvalue = [1297, 130]\nclass = 0'),
 Text(0.17391304347826086, 0.16666666666666666, 'Seat comfort <= 2.5\ngini = 0.202\nsamples = 3993\nvalue = [3537, 456]\nclass = 0'),
 Text(0.16304347826086957, 0.05555555555555555, 'gini = 0.191\nsamples = 1688\nvalue = [1508, 180]\nclass = 0'),
 Text(0.18478260869565216, 0.05555555555555555, 'gini = 0.211\nsamples = 2305\nvalue = [2029, 276]\nclass = 0'),
 Text(0.20652173913043478, 0.2777777777777778, 'Seat comfort <= 1.5\ngini = 0.212\nsamples = 5473\nvalue = [4815, 658]\nclass = 0'),
 Text(0.1956521739130435, 0.16666666666666666, 'gini = 0.237\nsamples = 1447\nvalue = [1248, 199]\nclass = 0'),
 Text(0.21739130434782608, 0.16666666666666666, 'Seat comfort <= 2.5\ngini = 0.202\nsamples = 4026\nvalue = [3567, 459]\nclass = 0'),
 Text(0.20652173913043478, 0.05555555555555555, 'gini = 0.203\nsamples = 1736\nvalue = [1537, 199]\nclass = 0'),
 Text(0.22826086956521738, 0.05555555555555555, 'gini = 0.201\nsamples = 2290\nvalue = [2030, 260]\nclass = 0'),
 Text(0.29891304347826086, 0.5, 'Seat comfort <= 2.5\ngini = 0.174\nsamples = 15060\nvalue = [13606, 1454]\nclass = 0'),
 Text(0.2717391304347826, 0.3888888888888889, 'Seat comfort <= 1.5\ngini = 0.152\nsamples = 9362\nvalue = [8586, 776]\nclass = 0'),
 Text(0.25, 0.2777777777777778, 'Ease of Online booking <= 1.5\ngini = 0.133\nsamples = 4328\nvalue = [4017, 311]\nclass = 0'),
 Text(0.2391304347826087, 0.16666666666666666, 'gini = 0.171\nsamples = 1056\nvalue = [956, 100]\nclass = 0'),
 Text(0.2608695652173913, 0.16666666666666666, 'Ease of Online booking <= 2.5\ngini = 0.121\nsamples = 3272\nvalue = [3061, 211]\nclass = 0'),
 Text(0.25, 0.05555555555555555, 'gini = 0.101\nsamples = 1675\nvalue = [1586, 89]\nclass = 0'),
 Text(0.2717391304347826, 0.05555555555555555, 'gini = 0.141\nsamples = 1597\nvalue = [1475, 122]\nclass = 0'),
 Text(0.29347826086956524, 0.2777777777777778, 'Ease of Online booking <= 1.5\ngini = 0.168\nsamples = 5034\nvalue = [4569, 465]\nclass = 0'),
 Text(0.2826086956521739, 0.16666666666666666, 'gini = 0.187\nsamples = 1229\nvalue = [1101, 128]\nclass = 0'),
 Text(0.30434782608695654, 0.16666666666666666, 'Ease of Online booking <= 2.5\ngini = 0.161\nsamples = 3805\nvalue = [3468, 337]\nclass = 0'),
 Text(0.29347826086956524, 0.05555555555555555, 'gini = 0.146\nsamples = 1934\nvalue = [1781, 153]\nclass = 0'),
 Text(0.31521739130434784, 0.05555555555555555, 'gini = 0.177\nsamples = 1871\nvalue = [1687, 184]\nclass = 0'),
 Text(0.3260869565217391, 0.3888888888888889, 'Ease of Online booking <= 1.5\ngini = 0.21\nsamples = 5698\nvalue = [5020, 678]\nclass = 0'),
 Text(0.31521739130434784, 0.2777777777777778, 'gini = 0.231\nsamples = 1479\nvalue = [1282, 197]\nclass = 0'),
 Text(0.33695652173913043, 0.2777777777777778, 'Ease of Online booking <= 2.5\ngini = 0.202\nsamples = 4219\nvalue = [3738, 481]\nclass = 0'),
 Text(0.32608695652173914, 0.16666666666666666, 'gini = 0.195\nsamples = 2041\nvalue = [1817, 224]\nclass = 0'),
 Text(0.34782608695652173, 0.16666666666666666, 'gini = 0.208\nsamples = 2178\nvalue = [1921, 257]\nclass = 0'),
 Text(0.4728260869565217, 0.7222222222222222, 'Ease of Online booking <= 4.5\ngini = 0.479\nsamples = 13806\nvalue = [8331, 5475]\nclass = 0'),
 Text(0.41847826086956524, 0.6111111111111112, 'Seat comfort <= 2.5\ngini = 0.435\nsamples = 8437\nvalue = [5735, 2702]\nclass = 0'),
 Text(0.391304347826087, 0.5, 'Gender_Male <= 0.5\ngini = 0.448\nsamples = 4892\nvalue = [3234, 1658]\nclass = 0'),
 Text(0.3695652173913043, 0.3888888888888889, 'Seat comfort <= 1.5\ngini = 0.469\nsamples = 2313\nvalue = [1445, 868]\nclass = 0'),
 Text(0.358695652173913, 0.2777777777777778, 'gini = 0.472\nsamples = 999\nvalue = [617, 382]\nclass = 0'),
 Text(0.3804347826086957, 0.2777777777777778, 'gini = 0.466\nsamples = 1314\nvalue = [828, 486]\nclass = 0'),
 Text(0.41304347826086957, 0.3888888888888889, 'Seat comfort <= 1.5\ngini = 0.425\nsamples = 2579\nvalue = [1789, 790]\nclass = 0'),
 Text(0.40217391304347827, 0.2777777777777778, 'gini = 0.434\nsamples = 1113\nvalue = [759, 354]\nclass = 0'),
 Text(0.42391304347826086, 0.2777777777777778, 'gini = 0.418\nsamples = 1466\nvalue = [1030, 436]\nclass = 0'),
 Text(0.44565217391304346, 0.5, 'Gender_Male <= 0.5\ngini = 0.416\nsamples = 3545\nvalue = [2501, 1044]\nclass = 0'),
 Text(0.43478260869565216, 0.3888888888888889, 'gini = 0.408\nsamples = 1818\nvalue = [1298, 520]\nclass = 0'),
 Text(0.45652173913043476, 0.3888888888888889, 'gini = 0.423\nsamples = 1727\nvalue = [1203, 524]\nclass = 0'),
 Text(0.5271739130434783, 0.6111111111111112, 'Seat comfort <= 2.5\ngini = 0.499\nsamples = 5369\nvalue = [2596, 2773]\nclass = 1'),
 Text(0.5, 0.5, 'Seat comfort <= 1.5\ngini = 0.486\nsamples = 3008\nvalue = [1254, 1754]\nclass = 1'),
 Text(0.4782608695652174, 0.3888888888888889, 'Gender_Male <= 0.5\ngini = 0.457\nsamples = 1232\nvalue = [436, 796]\nclass = 1'),
 Text(0.4673913043478261, 0.2777777777777778, 'Seat comfort <= 0.5\ngini = 0.419\nsamples = 612\nvalue = [183, 429]\nclass = 1'),
 Text(0.45652173913043476, 0.16666666666666666, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]\nclass = 0'),
 Text(0.4782608695652174, 0.16666666666666666, 'gini = 0.418\nsamples = 611\nvalue = [182, 429]\nclass = 1'),
 Text(0.4891304347826087, 0.2777777777777778, 'gini = 0.483\nsamples = 620\nvalue = [253, 367]\nclass = 1'),
 Text(0.5217391304347826, 0.3888888888888889, 'Gender_Male <= 0.5\ngini = 0.497\nsamples = 1776\nvalue = [818, 958]\nclass = 1'),
 Text(0.5108695652173914, 0.2777777777777778, 'gini = 0.493\nsamples = 846\nvalue = [374, 472]\nclass = 1'),
 Text(0.532608695652174, 0.2777777777777778, 'gini = 0.499\nsamples = 930\nvalue = [444, 486]\nclass = 1'),
 Text(0.5543478260869565, 0.5, 'Gender_Male <= 0.5\ngini = 0.491\nsamples = 2361\nvalue = [1342, 1019]\nclass = 0'),
 Text(0.5434782608695652, 0.3888888888888889, 'gini = 0.491\nsamples = 1186\nvalue = [672, 514]\nclass = 0'),
 Text(0.5652173913043478, 0.3888888888888889, 'gini = 0.49\nsamples = 1175\nvalue = [670, 505]\nclass = 0'),
 Text(0.796875, 0.8333333333333334, 'Ease of Online booking <= 3.5\ngini = 0.479\nsamples = 58262\nvalue = [23207, 35055]\nclass = 1'),
 Text(0.6807065217391305, 0.7222222222222222, 'Ease of Online booking <= 0.5\ngini = 0.499\nsamples = 38602\nvalue = [18633, 19969]\nclass = 1'),
 Text(0.6195652173913043, 0.6111111111111112, 'Seat comfort <= 4.5\ngini = 0.392\nsamples = 2470\nvalue = [660, 1810]\nclass = 1'),
 Text(0.5978260869565217, 0.5, 'Gender_Male <= 0.5\ngini = 0.42\nsamples = 1269\nvalue = [381, 888]\nclass = 1'),
 Text(0.5869565217391305, 0.3888888888888889, 'gini = 0.425\nsamples = 678\nvalue = [208, 470]\nclass = 1'),
 Text(0.6086956521739131, 0.3888888888888889, 'gini = 0.414\nsamples = 591\nvalue = [173, 418]\nclass = 1'),
 Text(0.6413043478260869, 0.5, 'Gender_Male <= 0.5\ngini = 0.357\nsamples = 1201\nvalue = [279, 922]\nclass = 1'),
 Text(0.6304347826086957, 0.3888888888888889, 'gini = 0.36\nsamples = 620\nvalue = [146, 474]\nclass = 1'),
 Text(0.6521739130434783, 0.3888888888888889, 'gini = 0.353\nsamples = 581\nvalue = [133, 448]\nclass = 1'),
 Text(0.7418478260869565, 0.6111111111111112, 'Seat comfort <= 4.5\ngini = 0.5\nsamples = 36132\nvalue = [17973, 18159]\nclass = 1'),
 Text(0.6956521739130435, 0.5, 'Gender_Male <= 0.5\ngini = 0.498\nsamples = 19589\nvalue = [10427, 9162]\nclass = 0'),
 Text(0.6739130434782609, 0.3888888888888889, 'Ease of Online booking <= 1.5\ngini = 0.49\nsamples = 10379\nvalue = [5923, 4456]\nclass = 0'),
 Text(0.6630434782608695, 0.2777777777777778, 'gini = 0.5\nsamples = 2869\nvalue = [1475, 1394]\nclass = 0'),
 Text(0.6847826086956522, 0.2777777777777778, 'Ease of Online booking <= 2.5\ngini = 0.483\nsamples = 7510\nvalue = [4448, 3062]\nclass = 0'),
 Text(0.6739130434782609, 0.16666666666666666, 'gini = 0.484\nsamples = 3771\nvalue = [2219, 1552]\nclass = 0'),
 Text(0.6956521739130435, 0.16666666666666666, 'gini = 0.482\nsamples = 3739\nvalue = [2229, 1510]\nclass = 0'),
```

```
Text(0.717391304347826, 0.3888888888888889, 'Ease of Online booking <= 1.5\ngini = 0.5\nsamples = 9210\nvalue = [4504, 4706]\nclass = 1'),
```

UNSUPERVISED:

```
Text(0.7880434782608695, 0.5, 'Ease of Online booking <= 1.5\ngini = 0.496\nsamples = 16543\nvalue = [7546, 8997]\nclass = 1'),
```

```
airline = pd.read_csv("/content/drive/MyDrive/Airline_Dataset.csv")
airline
```

	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
0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3	1	5	3	5	5	4	3	4	4	5	5	25	18.0	neutral or dissatisfied
1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	3	3	1	3	1	1	1	5	3	1	4	1	1	6.0	neutral or dissatisfied
2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	2	2	5	5	5	5	4	3	4	4	4	5	0	0.0	satisfied
3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	5	5	2	2	2	2	2	5	3	1	4	2	11	9.0	neutral or dissatisfied
4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	3	3	3	4	5	5	3	3	4	4	3	3	3	0	0.0	satisfied
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
129875	78463	Male	disloyal Customer	34	Business travel	Business	526	3	3	3	1	4	3	4	4	3	2	4	4	5	4	0	0.0	neutral or dissatisfied
129876	71167	Male	Loyal Customer	23	Business travel	Business	646	4	4	4	4	4	4	4	4	4	5	5	5	5	4	0	0.0	satisfied
129877	37675	Female	Loyal Customer	17	Personal Travel	Eco	828	2	5	1	5	2	1	2	2	4	3	4	5	4	2	0	0.0	neutral or dissatisfied
129878	90086	Male	Loyal Customer	14	Business travel	Business	1127	3	3	3	3	4	4	4	4	3	2	5	4	5	4	0	0.0	satisfied
129879	34799	Female	Loyal Customer	42	Personal Travel	Eco	264	2	5	2	5	4	2	2	1	1	2	1	1	1	1	0	0.0	neutral or dissatisfied

129880 rows × 24 columns


```
airline.isnull().sum()
```

```
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             393
Satisfaction                           0
dtype: int64
```

```
airline["Arrival Delay in Minutes"].fillna("0.0", inplace = True)
```

```
airline.isnull().sum()
```

```
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
```

```
train_df.isnull().sum()
```

```
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
```

```
airline
```

	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
0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3	1	5	3	5	5	4	3	4	4	5	5	25	18.0	neutral or dissatisfied
2	110028	Female	Loyal	26	Business	Business	1142	2	2	2	2	5	5	5	5	4	3	4	4	4	5	0	0.0	satisfied
3	24026	Female	Loyal	14	Business	Business	1197	3	3	3	3	4	4	4	4	3	3	5	4	5	4	0	0.0	satisfied

KMeans

```
def delay_transformation(x):
    if x<=10:
        x = "less_than_10min"
    elif x<=40:
        x = "10_to_40min"
    elif x<=120:
        x = "41_to_120min"
    elif x<=240:
        x = "121_240min"
    elif x<=450:
        x = "241_450min"
    else:
        x = "more_than_450min"

    return x
```

```
def transform_dataset(df, clean=True, transform=True, scaler=True):

    if clean==True:

        df.set_index("id", inplace=True)

        df.Gender = df.Gender.apply(lambda x: 1 if x=="Female" else 0)

        df.sort_values(by="id",inplace=True)

        df.rename(columns={"Customer Type":"Loyal"}, inplace=True)
        df["Loyal"] = df["Loyal"].apply(lambda x: 1 if x=='Loyal Customer' else 0)

        df.rename(columns={"Satisfaction":"Dissatisfied"}, inplace=True)
        df["Dissatisfied"] = df['Dissatisfied'].apply(lambda x: 1 if x=='neutral or dissatisfied' else 0)

        df.rename(columns={'Type of Travel':"Business Travel"}, inplace=True)
        df["Business Travel"] = df["Business Travel"].apply(lambda x: 1 if x=='Business travel' else 0)

    if transform == True:
        df["Flight Distance"] = np.log(df["Flight Distance"]+1)

        df["Departure Delay in Minutes"] = df["Departure Delay in Minutes"].apply(delay_transformation)
        df.rename(columns={"Departure Delay in Minutes":"Departure_Delay"},inplace=True)

        df.drop(columns="Arrival Delay in Minutes", inplace=True)

        df = pd.get_dummies(df, columns=["Class","Departure_Delay"],drop_first=True)

    if scaler==True:
        df.loc[:,["Age","Flight Distance"]] = StandardScaler().fit_transform(df.loc[:,["Age","Flight Distance"]])

        df.loc[:,"Inflight wifi service":"Cleanliness"] = StandardScaler().fit_transform(df.loc[:,"Inflight wifi service":"Cleanliness"])

    return df


airline_train_transf = transform_dataset(pd.read_csv("/content/drive/MyDrive/Airline_Dataset.csv"))


# Creating PCA model:
from sklearn.decomposition import PCA
def pca_transformation(df, n):
    pca_model = PCA(n_components=n)

    df = pca_model.fit_transform(df)

    columns_list = []
    for i in range(1,n+1):
        columns_list.append("PC"+str(i))

    df = pd.DataFrame(df, columns=columns_list)

    return df
```

```
airline_train_pca = pca_transformation(airline_train_transf, 3)
```

```
airline_train_pca
```

	PC1	PC2	PC3
0	-2.162549	0.838906	0.665462
1	-2.388978	1.684433	-1.133255
2	-1.814272	-1.177756	2.138611
3	-2.580912	1.895519	-0.919618
4	-1.186164	-0.106768	1.505057
...	...	...	...
129875	-1.192401	-0.720638	-1.468321
129876	0.163928	-1.466539	-2.698131
129877	0.468562	0.728907	-1.275233
129878	-1.313740	-0.862425	-0.959772
129879	2.392872	-0.072719	0.027915

129880 rows × 3 columns

```
X_airline_train = airline_train_transf.drop(columns='Dissatisfied').copy()

print(X_airline_train.shape)

Y_airline_train = airline_train_transf['Dissatisfied'].copy()
print(Y_airline_train.shape)
```

```
(129880, 26)
(129880,)
```

```
X_airline_train = transform_dataset(pd.read_csv("/content/drive/MyDrive/Airline_Dataset.csv").copy()).drop("Dissatisfied",axis=1)

# Applying PCA without target column
X_airline_train = pca_transformation(X_airline_train, 6)

Y_airline_train = transform_dataset(pd.read_csv("/content/drive/MyDrive/Airline_Dataset.csv"))["Dissatisfied"]
```

```
X_airline_train.shape
```

```
(129880, 6)
```

```
airline_test_transf = transform_dataset(pd.read_csv("/content/drive/MyDrive/Airline_Dataset.csv"))
```

```
X_airline_test = airline_test_transf.drop(columns='Dissatisfied').copy()

print(X_airline_test.shape)

Y_airline_test = airline_test_transf['Dissatisfied'].copy()
print(Y_airline_test.shape)
```

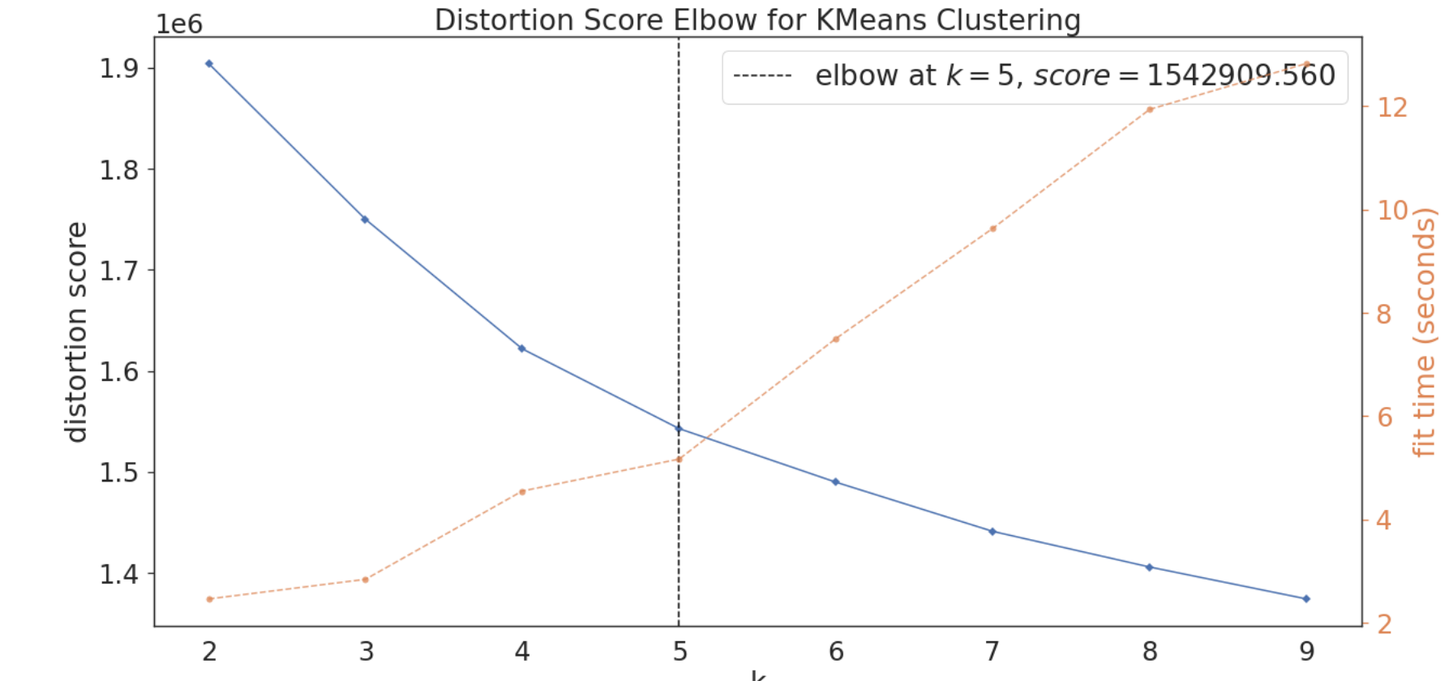
```
(129880, 26)
(129880,)
```

```
Y_airline_test.value_counts()/Y_airline_test.shape[0]
```

```
1    0.565537
0    0.434463
Name: Dissatisfied, dtype: float64
```

```
from sklearn.cluster import KMeans
```

```
from yellowbrick.cluster import KElbowVisualizer

# Visualizing the elbow choosing the best number of clusters
visualizer=KElbowVisualizer(KMeans(n_clusters = 9), k=(2,10))#, metric='silhouette')
visualizer.fit(airline_test_transf)
visualizer.poof()
```




```
kmeans_model = KMeans(n_clusters=5)

# Fitting the model
kmeans_model.fit(X_airline_train)
```

```
KMeans(n_clusters=5)
```

```
print("Number of clusters defined:", len(set(kmeans_model.labels_)))
```

```
Number of clusters defined: 5
```

```
airline_train_cluster = transform_dataset(pd.read_csv("/content/drive/MyDrive/Airline_Dataset.csv"),transform=False,scaler=False)

airline_train_cluster["Cluster_KMeans"] = kmeans_model.labels_

airline_train_cluster
```

id	Gender	Loyal	Age	Business 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	Dissatisfied	Cluster_KMeans
1	0	0	48	1	Business	821	3	3	3	3	5	3	5	5	3	2	5	4	5	5	2	5.0	1	0
2	1	1	35	1	Business	821	2	2	2	2	3	5	4	5	5	5	5	3	5	5	26	39.0	0	3
3	0	1	41	1	Business	853	4	4	4	4	5	5	5	3	3	3	3	4	3	5	0	0.0	0	1
4	0	1	50	1	Business	1905	2	2	2	2	4	4	5	5	5	5	5	3	5	4	0	0.0	0	3
5	1	1	49	1	Business	3470	3	3	3	3	4	5	4	3	3	4	3	3	3	5	0	1.0	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
129876	0	1	28	0	Eco Plus	447	4	4	4	2	4	4	1	4	5	4	4	4	5	4	2	3.0	1	1
129877	0	1	41	0	Eco Plus	308	3	5	3	4	2	3	2	2	5	5	5	5	4	2	0	0.0	1	2
129878	0	1	42	0	Eco Plus	337	2	5	2	1	3	2	3	3	3	4	5	4	4	3	6	14.0	1	0
129879	0	1	50	0	Eco Plus	337	5	4	4	1	3	4	4	3	4	5	5	3	4	3	31	22.0	0	1
129880	1	1	20	0	Eco Plus	337	3	1	3	2	2	3	2	2	4	4	1	4	2	2	0	0.0	1	2

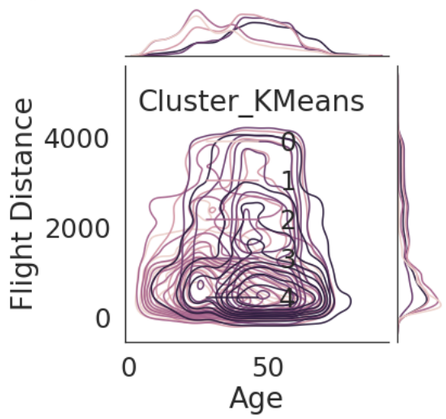
129880 rows × 24 columns

```
airline_train_cluster.groupby("Cluster_KMeans").mean()
```

Cluster_KMeans	Gender	Loyal	Age	Business Travel	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	Dissatisfied
0	0.488336	0.649506	30.806029	0.470276	660.546943	2.013678	2.830242	1.800363	2.885751	4.021779	2.100482	3.843212	4.073658	3.282679	3.022929	3.731043	3.154796	3.772697	4.026604	12.520738	13.001865	0.860077
1	0.508306	0.871127	41.107082	0.796231	1355.885147	4.174335	4.158760	4.151313	3.947285	3.714600	4.125020	4.030858	4.161642	3.944306	3.867014	4.118285	3.619078	4.137292	3.871968	13.486319	13.643233	0.264094
2	0.453372	0.690631	32.945627	0.571548	894.768073	2.361856	3.044434	2.588922	2.963619	1.623672	2.283906	1.729805	1.793971	3.217708	3.260873	3.677899	3.027821	3.694584	1.601835	17.451630	18.010081	0.846676
3	0.513161	0.952447	45.402745	0.918825	1802.390189	2.124223	1.975389	2.029283	1.904643	3.597214	4.117215	4.220707	4.345013	4.270471	4.154632	4.328828	3.840988	4.334626	3.919993	13.955241	14.045546	0.152797
4	0.577864	0.910996	46.602960	0.671453	1203.568010	2.572313	2.954472	2.781692	2.903113	3.203949	3.456403	3.490388	2.492519	2.104690	2.332609	2.215854	2.848536	2.182955	3.143098	15.926239	16.559400	0.758567

```
plt.figure(figsize=(16,16))
sns.jointplot(data=airline_train_cluster, x="Age",y="Flight Distance",hue="Cluster_KMeans", kind='kde')
```

```
<seaborn.axisgrid.JointGrid at 0x7f958b1931f0>
<Figure size 1152x1152 with 0 Axes>
```




```
cluster_airline_train = pd.merge(X_airline_train, Y_airline_train, on=Y_airline_train.index)
```

```
cluster_airline_train.set_index("key_0",inplace=True)
```

```
cluster_airline_train["Cluster_KMeans"] = kmeans_model.labels_
```

```
cluster_airline_train
```

key_0	PC1	PC2	PC3	PC4	PC5	PC6	Dissatisfied	Cluster_KMeans
1	-2.258988	0.843163	0.652522	0.833312	0.559848	-0.150832	1	0
2	-2.322035	1.684684	-1.134204	-0.331493	-0.834505	0.574279	0	3
3	-1.752557	-1.177035	2.139542	-0.117485	0.263683	-0.322498	0	1
4	-2.515199	1.895787	-0.921051	-0.881764	-0.411805	0.229877	0	3
5	-1.103689	-0.107485	1.510183	-1.577467	-0.522737	-0.639739	0	3
...	...	...	...	...	...	...	...	...
129876	-1.271182	-0.718189	-1.476919	0.666214	-0.388351	0.601913	1	1
129877	0.101743	-1.465987	-2.702344	0.194580	1.499609	0.255843	1	2
129878	0.401991	0.729840	-1.280273	0.577797	1.210200	0.377829	1	0
129879	-1.241250	-0.863243	-0.957370	-0.508831	-0.165570	1.983280	0	1
129880	2.365820	-0.074425	0.032870	-0.459529	-0.900625	0.566900	1	2

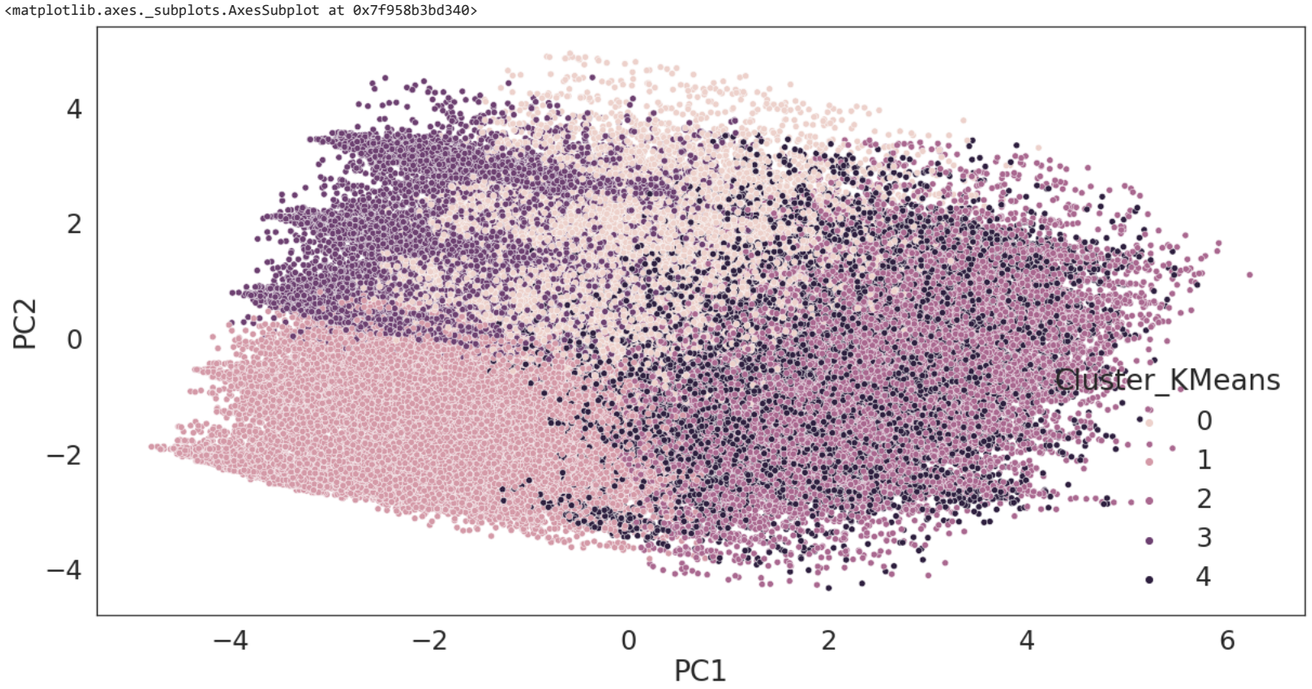
129880 rows × 8 columns

```
cluster_airline_train.groupby('Cluster_KMeans').mean()
```

Cluster_KMeans	PC1	PC2	PC3	PC4	PC5	PC6	Dissatisfied
0	-0.028159	1.168413	0.280558	1.391093	0.116005	-0.145682	0.860077
1	-1.900414	-1.333343	-0.002164	0.122270	-0.024461	0.006542	0.264094
2	2.188859	0.536884	1.408576	0.000469	0.125284	0.010202	0.846676

```
sns.scatterplot(data=cluster_airline_train, x="PC1",y="PC2", hue="Cluster_KMeans")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f958b3bd340>
```



DBSCAN

```
from sklearn.cluster import DBSCAN


dbscan_model = DBSCAN(eps=1.1)

# Model fitting
dbscan_model.fit(X_airline_train)

# Storing the result in a column of our new DataFrame
airline_train_cluster["Cluster_DBSCAN"] = dbscan_model.labels_


print("Number of clusters:", len(set(dbscan_model.labels_)))
```

```
Number of clusters: 2
```

```
# Looking at the different clusters created:
airline_train_cluster.groupby("Cluster_DBSCAN").mean()
```

Cluster_DBSCAN	Gender	Loyal	Age	Business Travel	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	Dissatisfied	Cluster_KMeans
-1	0.554245	0.820755	44.337264	0.558962	833.596698	2.122642	2.601415	1.867925	2.735849	2.825472	2.408019	2.794811	2.872642	2.754717	2.636792	2.632075	2.768868	2.712264	2.778302	13.511792	14.273159	0.424528	2.313679
0	0.507230	0.816895	39.411877	0.691015	1191.484736	2.730681	3.059093	2.759787	2.977714	3.206016	3.255400	3.443479	3.359667	3.385081	3.353217	3.635390	3.308027	3.645239	3.287990	14.717649	15.093797	0.565999	1.980712

```
airline_train_cluster.groupby("Cluster_DBSCAN").count()
```

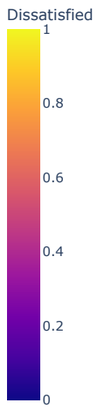
Cluster_DBSCAN	Gender	Loyal	Age	Business Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Seat comfort	Inflight entertainment	On-board service	Leg room service	Baggage handling	Checkin service	Inflight service	Cleanliness	Departure Delay in Minutes	Arrival Delay in Minutes	Dissatisfied	Cluster_KMeans
-1	424	424	424	424	424	424	424	424	424	424	424	424	424	424	424	424	424	424	424	424	424	421	424	424
0	129456	129456	129456	129456	129456	129456	129456	129456	129456	129456	129456	129456	129456	129456	129456	129456	129456	129456	129456	129456	129456	129066	129456	129456

Conclusion: the DBSCAN doesn't perform well in this case, either creating a huge number of clusters or creating not enough clusters. Since this model is too slow to run and it provokes very different results for a slight modification in eps, it is hard to adopt and optimise this model.

```
train_df.describe()
```

	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
count	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000	129790.000000
mean	64944.101972	0.507404	0.183157	39.427614	0.690554	1.376608	1187.970945	2.728708	3.057701	2.756846	2.976886	3.204708	3.252562	3.441390	3.358078	3.383073	3.350690	3.631952	3.306218
std	37502.505737	0.499947	0.386797	15.120185	0.462267	0.616189	993.763866	1.329365	1.526768	1.401724	1.278459	1.329926	1.350732	1.319321	1.334129	1.287073	1.316369	1.180044	1.266187
min	1.000000	0.000000	0.000000	7.000000	0.000000	0.000000	31.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000
25%	32465.250000	0.000000	0.000000	27.000000	0.000000	1.000000	414.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	3.000000	3.000000
50%	64960.500000	1.000000	0.000000	40.000000	1.000000	1.000000	843.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	4.000000	4.000000	4.000000	4.000000	4.000000	3.000000
75%	97430.750000	1.000000	0.000000	51.000000	1.000000	2.000000	1742.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	5.000000	4.000000	4.000000	4.000000	5.000000	4.000000
max	129880.000000	1.000000	1.000000	85.000000	1.000000	2.000000	3999.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000

```
import plotly.express as px
fig = px.scatter_3d(cluster_airline_train,x='PC1',y='PC2',z='PC3',color='Dissatisfied')
fig.show()
```



Agglomerative Clustering

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

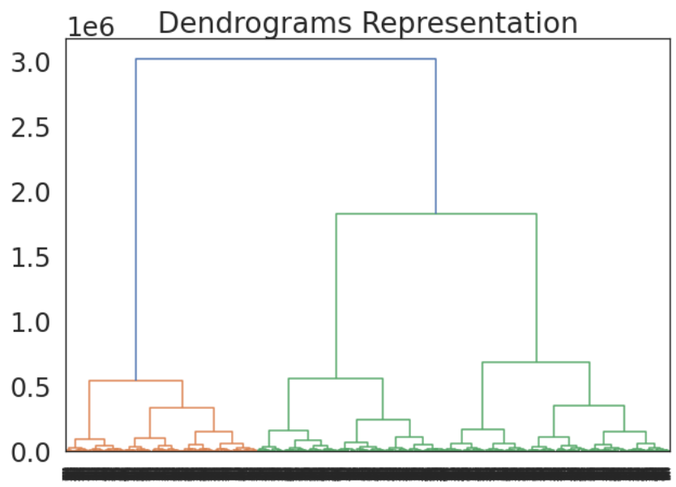
```
df2=train_df[:5000]
# df = df.sample(frac=0.50)
```

```
y1 = df2['Satisfaction']
x1 = df2.drop('Satisfaction',axis=1)
```

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
```

```
from sklearn.model_selection import train_test_split
x1_train, x1_test, y1_train, y1_test= train_test_split(x1, y1, test_size=0.2, random_state=42)
```

```
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(10, 7))
plt.title("Dendrograms Representation")
dend = shc.dendrogram(shc.linkage(df2, method='ward'))
```




```
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='ward')
cluster.fit_predict(df2)
```

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

```
cluster.fit(x1_train)
cluster_y_preds = cluster.fit_predict(x1_test)
```

```
agcl_preds = pd.DataFrame({"Predicted":cluster_y_preds,"Actual":y1_test})
agcl_preds
```

```
agcl_TP = len(agcl_preds[(agcl_preds["Predicted"]==agcl_preds["Actual"])&(agcl_preds["Predicted"]==1)])
agcl_FP = len(agcl_preds[(agcl_preds["Predicted"]!=agcl_preds["Actual"])&(agcl_preds["Predicted"]==1)])
agcl_FN = len(agcl_preds[(agcl_preds["Predicted"]!=agcl_preds["Actual"])&(agcl_preds["Predicted"]==0)])
agcl_TN = len(agcl_preds[(agcl_preds["Predicted"]==agcl_preds["Actual"])&(agcl_preds["Predicted"]==0)])
print(agcl_TP,agcl_FP,agcl_FN,agcl_TN)
print("Rightly Classified: ",(agcl_TP+agcl_TN),"/",(agcl_TP+agcl_FP+agcl_FN+agcl_TN))
print("Wrongly Classified: ",(agcl_FP+agcl_FN),"/",(agcl_TP+agcl_FP+agcl_FN+agcl_TN))

agcl_Accuracy = (agcl_TP+agcl_TN)/(agcl_TP+agcl_FP+agcl_FN+agcl_TN)
agcl_Precision = (agcl_TP)/(agcl_TP+agcl_FP)
agcl_Recall = (agcl_TP)/(agcl_TP+agcl_FN)
agcl_Specificity = (agcl_TN)/(agcl_TN+agcl_FP)
agcl_F1 = (2*agcl_Precision*agcl_Recall)/(agcl_Precision+agcl_Recall)
```

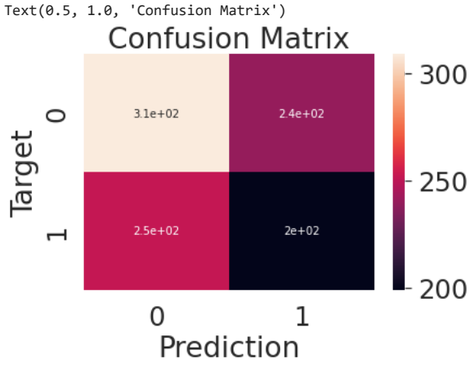
```
199 240 252 309
Rightly Classified:  508 / 1000
Wrongly Classified:  492 / 1000
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
accuracy = accuracy_score(y1_test, cluster_y_preds)
print("Accuracy: {:.2f}%".format(accuracy * 100))
precision=precision_score(y1_test,cluster_y_preds)
print("precision:",precision)
recall =recall_score(y1_test,cluster_y_preds)
print("recall:",recall)
```

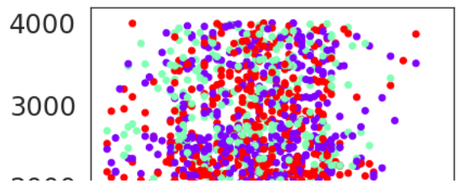
```
Accuracy: 50.80%
precision: 0.4533029612756264
recall: 0.44124168514412415
```

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
cf = confusion_matrix(y1_test, cluster_y_preds)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix')
```

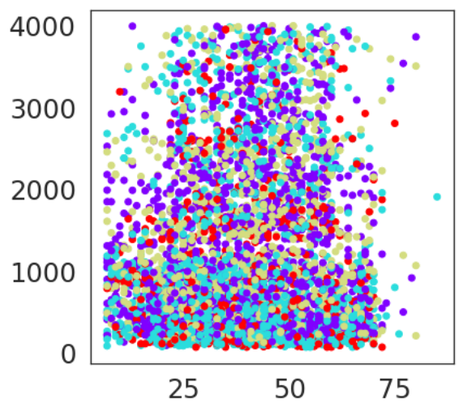
```
Text(0.5, 1.0, 'Confusion Matrix')
```



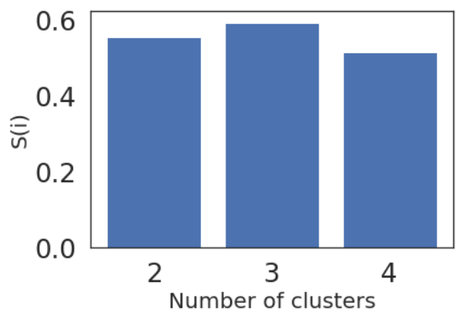

```
ac3 = AgglomerativeClustering(n_clusters = 3, affinity='euclidean', linkage='ward')

plt.figure(figsize =(6, 6))
plt.scatter(df2['Age'], df2['Flight Distance'],
           c = ac3.fit_predict(df2), cmap ='rainbow')
plt.show()
```




```
ac2 = AgglomerativeClustering(n_clusters = 4, affinity='euclidean', linkage='ward')

plt.figure(figsize =(6, 6))
plt.scatter(df2['Age'], df2['Flight Distance'], c = ac2.fit_predict(df2), cmap ='rainbow')
plt.show()
```




```
from sklearn.metrics import silhouette_score

silhouette_scores = []
silhouette_scores.append(
        silhouette_score(df2, cluster.fit_predict(df2)))
silhouette_scores.append(
        silhouette_score(df2, ac3.fit_predict(df2)))
silhouette_scores.append(
        silhouette_score(df2, ac2.fit_predict(df2)))


plt.bar([2,3,4], silhouette_scores)
plt.xlabel('Number of clusters', fontsize = 20)
plt.ylabel('S(i)', fontsize = 20)
plt.show()
```

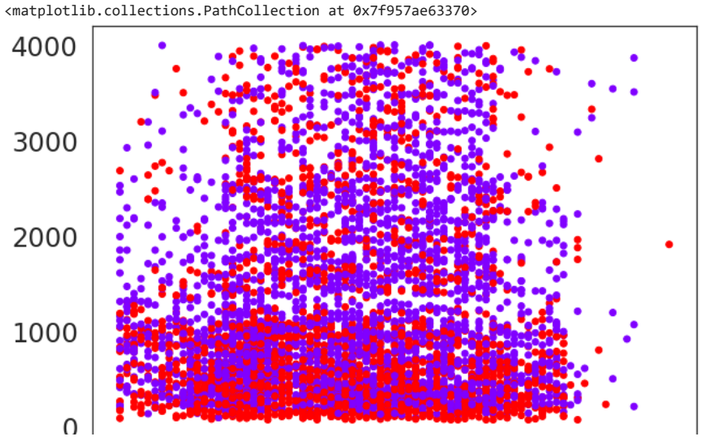



```
from sklearn.cluster import AgglomerativeClustering
cluster1 = AgglomerativeClustering(n_clusters=2, affinity='manhattan', linkage='average')
cluster1.fit_predict(df2)
```

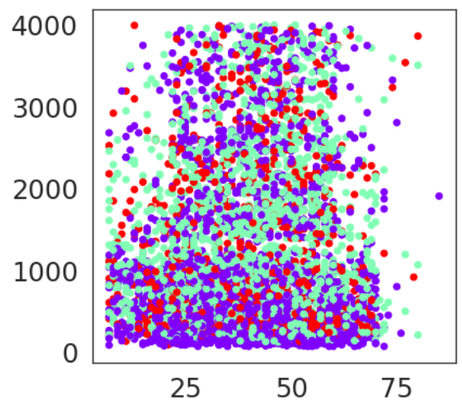
```
array([0, 1, 0, ..., 1, 0, 1])
```

```
plt.figure(figsize=(10, 7))
plt.scatter(df2['Age'], df2['Flight Distance'], c=cluster1.labels_,cmap ='rainbow')
```

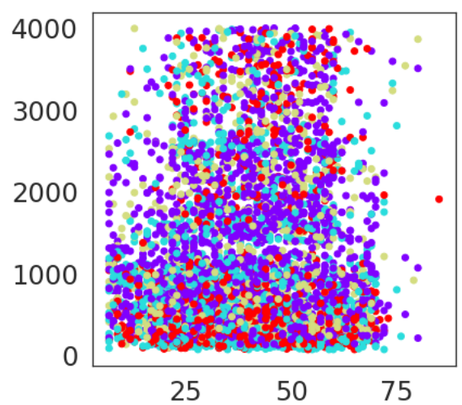
```
<matplotlib.collections.PathCollection at 0x7f957ae63370>
```




```
ac33 = AgglomerativeClustering(n_clusters = 3, affinity='manhattan', linkage='average')

plt.figure(figsize =(6, 6))
plt.scatter(df2['Age'], df2['Flight Distance'],
           c = ac33.fit_predict(df2), cmap ='rainbow')
plt.show()
```




```
ac22 = AgglomerativeClustering(n_clusters = 4, affinity='manhattan', linkage='average')

plt.figure(figsize =(6, 6))
plt.scatter(df2['Age'], df2['Flight Distance'], c = ac22.fit_predict(df2), cmap ='rainbow')
plt.show()
```




```
from sklearn.metrics import silhouette_score

silhouette_scores1 = []
silhouette_scores1.append(
```

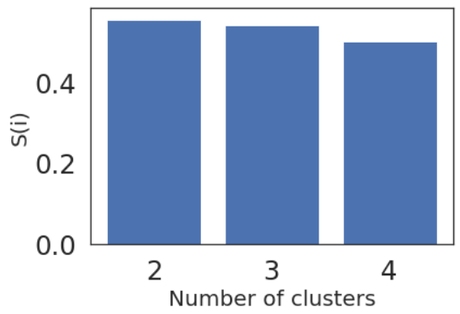
```
        silhouette_score(df2, cluster.fit_predict(df2)))
silhouette_scores1.append(
        silhouette_score(df2, ac33.fit_predict(df2)))
silhouette_scores1.append(
        silhouette_score(df2, ac22.fit_predict(df2)))

silhouette_scores
```

```
[0.5538002722363019, 0.5930506012762949, 0.5143587643228175]
```

```
silhouette_scores1
```

```
[0.5538002722363019, 0.5402608117850713, 0.5001590229130339]
```

```
plt.bar([2,3,4], silhouette_scores1)
plt.xlabel('Number of clusters', fontsize = 20)
plt.ylabel('S(i)', fontsize = 20)
plt.show()
```



BIRCH CLUSTERING

BALANCE ITERATIVE REDUCING AND CLUSTERING USING HIERARCHIES

```
from sklearn.cluster import Birch

Birch_model = Birch(n_clusters=2)
Birch_model.fit(x_train)
Birch_y_preds = Birch_model.fit_predict(x_test)
```

```
birch_preds = pd.DataFrame({"Predicted":Birch_y_preds,"Actual":y_test})
birch_preds
```

	Predicted	Actual
0	0	1
1	0	1
2	0	0
3	0	0
4	0	0
...	...	...
7783	0	0
7784	1	1
7785	1	1
7786	0	0
7787	0	0

7788 rows × 2 columns

```
birch_TP = len(birch_preds[(birch_preds["Predicted"]==birch_preds["Actual"])&(birch_preds["Predicted"]==1)])
birch_FP = len(birch_preds[(birch_preds["Predicted"]!=birch_preds["Actual"])&(birch_preds["Predicted"]==1)])
birch_FN = len(birch_preds[(birch_preds["Predicted"]!=birch_preds["Actual"])&(birch_preds["Predicted"]==0)])
birch_TN = len(birch_preds[(birch_preds["Predicted"]==birch_preds["Actual"])&(birch_preds["Predicted"]==0)])
print(birch_TP,birch_FP,birch_FN,birch_TN)
```

```
print("Rightly Classified: ",(birch_TP+birch_TN),"/",(birch_TP+birch_FP+birch_FN+birch_TN))
print("Wrongly Classified: ",(birch_FP+birch_FN),"/",(birch_TP+birch_FP+birch_FN+birch_TN))

birch_Accuracy = (birch_TP+birch_TN)/(birch_TP+birch_FP+birch_FN+birch_TN)
birch_Precision = (birch_TP)/(birch_TP+birch_FP)
birch_Recall = (birch_TP)/(birch_TP+birch_FN)
birch_Specificity = (birch_TN)/(birch_TN+birch_FP)
birch_F1 = (2*birch_Precision*birch_Recall)/(birch_Precision+birch_Recall)
```

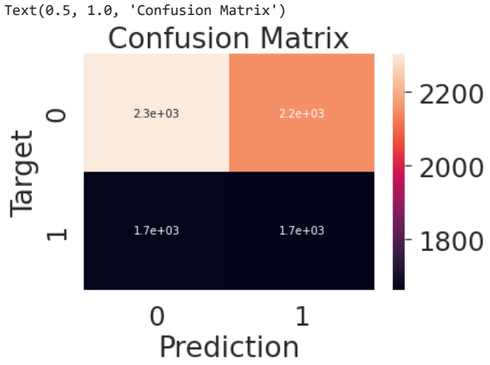
```
1663 2153 1669 2303
Rightly Classified:  3966 / 7788
Wrongly Classified:  3822 / 7788
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
accuracy = accuracy_score(y_test, Birch_y_preds)
print("Accuracy: {:.2f}%".format(accuracy * 100))
precision=precision_score(y_test,Birch_y_preds)
print("precision:",precision)
recall =recall_score(y_test,Birch_y_preds)
print("recall:",recall)
```

```
Accuracy: 50.92%
precision: 0.4357966457023061
recall: 0.49909963985594236
```

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
cf = confusion_matrix(y_test, Birch_y_preds)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```



Metrics

```
comparison = pd.DataFrame({"ML Classification Algo":["Decision_Tree","Logistic Regression","Random Forest","Gaussian Naive Bayes","KNN","K-Means Clustering","Agglomerative C
 "Rightly_Classified":[(decision_TP+decision_TN)/(decision_TP+decision_FP+decision_FN+decision_TN),(logistic_TP+logistic_TN)/(logistic_TP+logistic_FP+logistic_FN+logistic_TN
 "Wrongly_Classified":[(decision_FP+decision_FN)/(decision_TP+decision_FP+decision_FN+decision_TN),(logistic_FP+logistic_FN)/(logistic_TP+logistic_FP+logistic_FN+logistic_TN
 "Accuracy":[decision_Accuracy,logistic_Accuracy,rf_Accuracy,nb_Accuracy,knn_Accuracy,nb_Accuracy,agcl_Accuracy],
 "Precision":[decision_Precision,logistic_Precision,rf_Precision,nb_Precision,knn_Precision,nb_Precision,agcl_Precision],
 "Recall":[decision_Recall,logistic_Recall,rf_Recall,nb_Recall,knn_Recall,nb_Recall,agcl_Recall],
 "Specificity":[decision_Specificity,logistic_Specificity,rf_Specificity,nb_Specificity,knn_Specificity,nb_Specificity,agcl_Specificity],
 "F1-Score":[decision_F1,logistic_F1,rf_F1,nb_F1,knn_F1,nb_F1,agcl_F1]})

comparison.sort_values(by="Accuracy",ascending=False).style.background_gradient(cmap='rainbow')
```

	ML Classification Algo	Rightly_Classified	Wrongly_Classified	Accuracy	Precision	Recall	Specificity	F1-Score
2	Random Forest	0.956857	0.043143	0.956857	0.966958	0.930972	0.976212	0.948624

FROM ALL THE METRICS WE ANALYSED ACCUARACY OF RANDOM FOREST GOT HIGH ACCURACY RATE.SO, RANDOM FOREST IS BEST CLASSIFIER FOR THIS CLASSIFICATION PROBLEM

5	K-Means Clustering	0.790960	0.209040	0.790960	0.731396	0.808223	0.778052	0.767893