ABOUT THE DATASET

This dataset contains a survey about air passengers satisfication.Need to predict Airline passenger satisfaction level that is :1.Satisfied and 2.Neutral or dissatisfied. The main aim of this problem statement is to predict whether the customer is satisfied with his/her airplane experience or not. This problem comes under classification and this classification is done on the basis of passenger information and ratings given by him/her.

ABSTRACT

My goal is to build a predective classification model to identify which factors and features have the most significant impact on passenger satisfaction.Provide insights to airlines for improving customer experience and increasing passenger satisfaction.Select the best predictive models for predicting passengers satisfaction.

FEATURES

There is the following information about the passengers of some airline:

Gender: male or female

Customer type: regular or non-regular airline customer

Age: the actual age of the passenger

Type of travel: the purpose of the passenger's flight (personal or business travel)

Class: business, economy, economy plus

Flight distance

Inflight wifi service: satisfaction level with Wi-Fi service on board (0: not rated; 1-5)

Departure/Arrival time convenient: departure/arrival time satisfaction level (0: not rated; 1-5)

Ease of Online booking: online booking satisfaction rate (0: not rated; 1-5)

Gate location: level of satisfaction with the gate location (0:not rated; 1-5)

Food and drink: food and drink satisfaction level (0: not rated; 1-5)

Online boarding: satisfaction level with online boarding (0: not rated; 1-5)

Seat comfort: seat satisfaction level (0: not rated; 1-5)

On-board service: level of satisfaction with on-board service (0: not rated; 1-5)

Leg room service: level of satisfaction with leg room service (0: not rated; 1-5)

Baggage handling: level of satisfaction with baggage handling (0: not rated; 1-5)

Checkin service: level of satisfaction with checkin service (0: not rated; 1-5)

Inflight service: level of satisfaction with inflight service (0: not rated; 1-5)

Cleanliness: level of satisfaction with cleanliness (0: not rated; 1-5)

Inflight entertainment: satisfaction with inflight entertainment (0: not rated; 1-5) Departure delay in minutes:

Arrival delay in minutes:

Satisfaction: Airline satisfaction level(Satisfaction, neutral or dissatisfaction).

DATASET

```
import pandas as pd
import numpy as np
```

```
df=pd.read_csv('/content/drive/MyDrive/Datasets/test.csv')
df
```

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

25976 rows × 25 columns

DATA PREPROCESSING AND CLEANING

```
df.shape
```

```
(25976, 25)
```

This dataset contains 25976 rows and 25 columns

DATATYPES OF COLUMN VALUES

```
df.dtypes
```

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

```
Arrival Delay in Minutes             float64
satisfaction                          object
dtype: object
```

MISSING VALUES HANDLING

```
df.isna().sum()
```

```
Unnamed: 0                            0
id                                    0
Gender                                0
Customer Type                         0
Age                                   0
Type of Travel                        0
Class                                 0
Flight Distance                       0
Inflight wifi service                 0
Departure/Arrival time convenient     0
Ease of Online booking                0
Gate location                         0
Food and drink                        0
Online boarding                       0
Seat comfort                          0
Inflight entertainment                0
On-board service                      0
Leg room service                      0
Baggage handling                      0
Checkin service                       0
Inflight service                      0
Cleanliness                           0
Departure Delay in Minutes            0
Arrival Delay in Minutes             83
satisfaction                          0
dtype: int64
```

```
df['Arrival Delay in Minutes'].fillna(df['Arrival Delay in Minutes'].mean(),inplace=True)
```

```
df.isna().sum()
```

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

REMOVE UNWANTED COLUMNS AND REPLACE COLUMN VALUES

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

```
df['satisfaction'].replace(['neutral or dissatisfied'],['dissatisfied'],inplace=True)
```

DATA VISUALIZATION

```
df.satisfaction.value_counts()
```

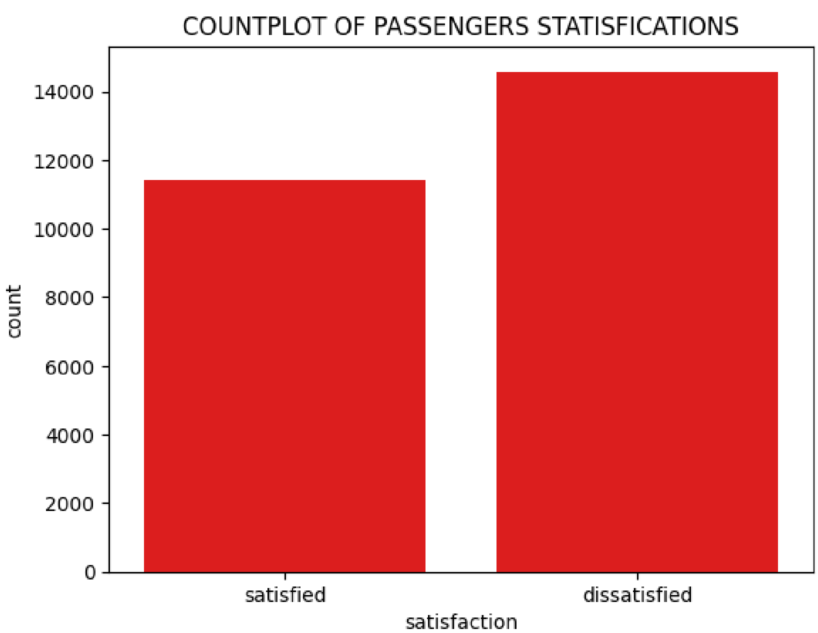
```
satisfaction
dissatisfied    14573
satisfied       11403
Name: count, dtype: int64
```

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

DATA VISUALIZATION

Data visualization is the graphical representation of the input features and output feature.

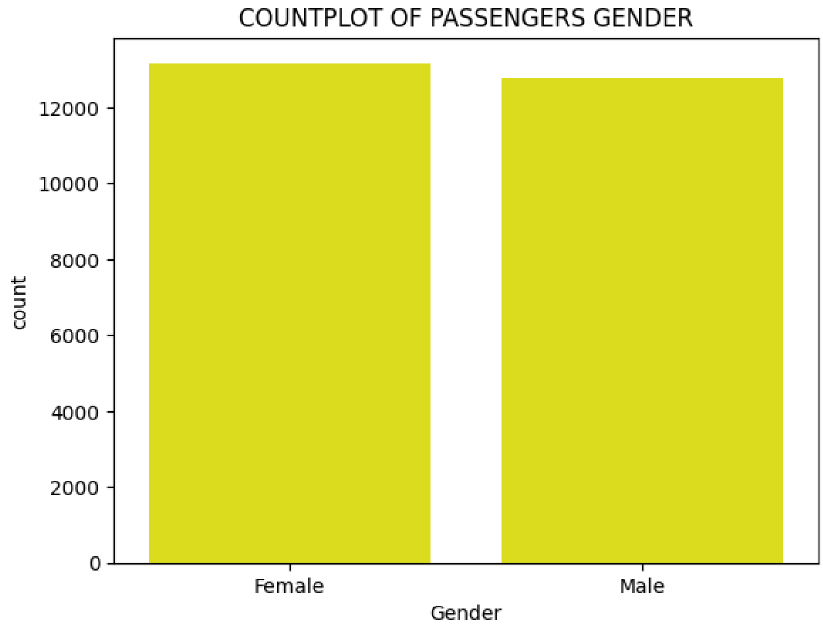
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(data=df,x='satisfaction',color='red')
plt.title('COUNTPLOT OF PASSENGERS STATISFICATIONS')
plt.show()
```




```
df.Gender.value_counts()
```

```
Gender
Female    13172
Male      12804
Name: count, dtype: int64
```

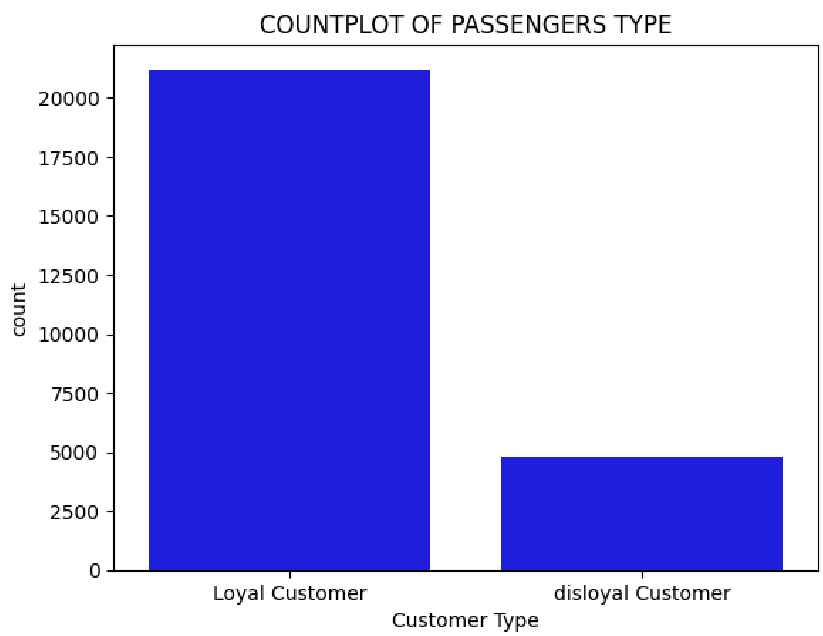
```
sns.countplot(data=df,x='Gender',color='yellow')
plt.title('COUNTPLOT OF PASSENGERS GENDER')
plt.show()
```




```
df['Customer Type'].value_counts()
```

```
Customer Type
Loyal Customer       21177
disloyal Customer     4799
Name: count, dtype: int64
```

```
sns.countplot(data=df,x='Customer Type',color='blue')
plt.title('COUNTPLOT OF PASSENGERS TYPE')
plt.show()
```

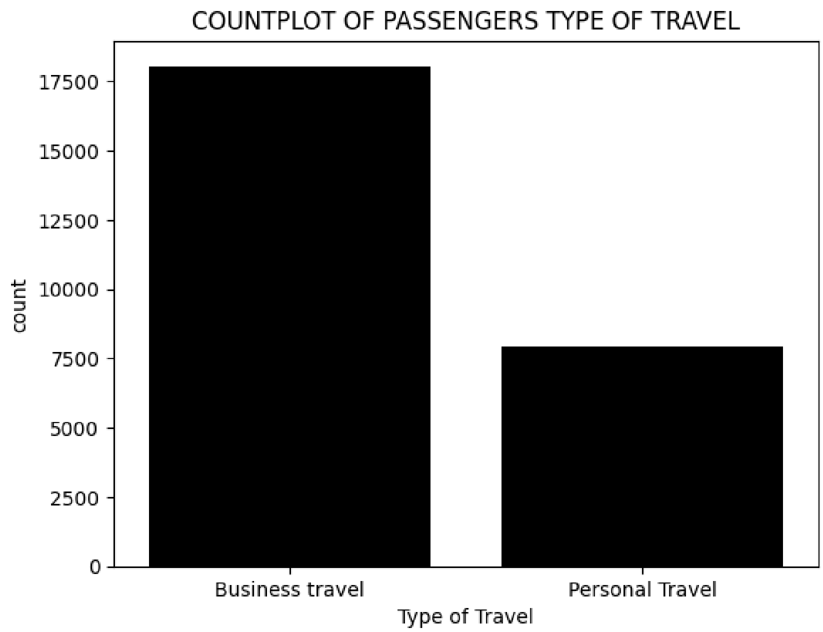



```
df['Type of Travel'].value_counts()
```

```
Type of Travel
Business travel    18038
Personal Travel     7938
Name: count, dtype: int64
```

```
sns.countplot(data=df,x='Type of Travel',color='black')
plt.title('COUNTPLOT OF PASSENGERS TYPE OF TRAVEL')
plt.show()
```

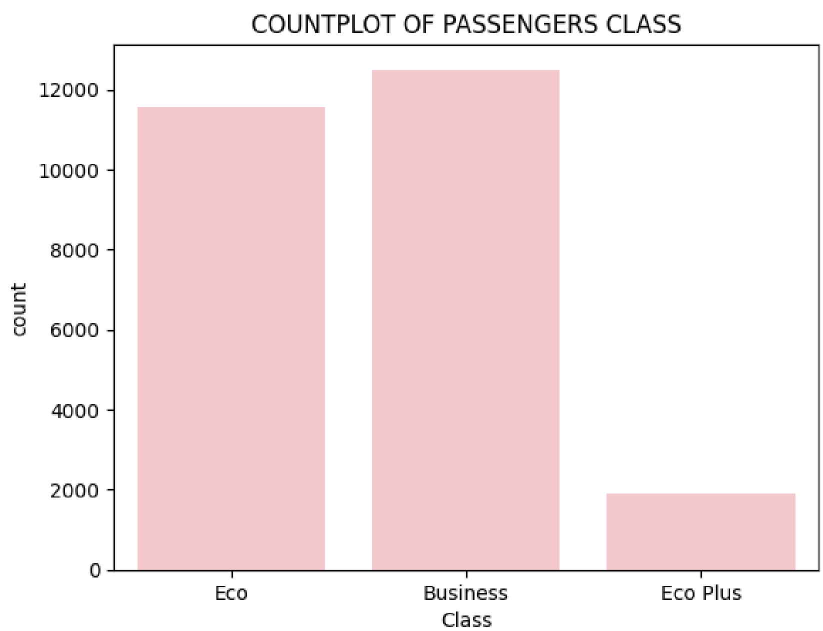
```
plt.show()
```




```
df.Class.value_counts()
```

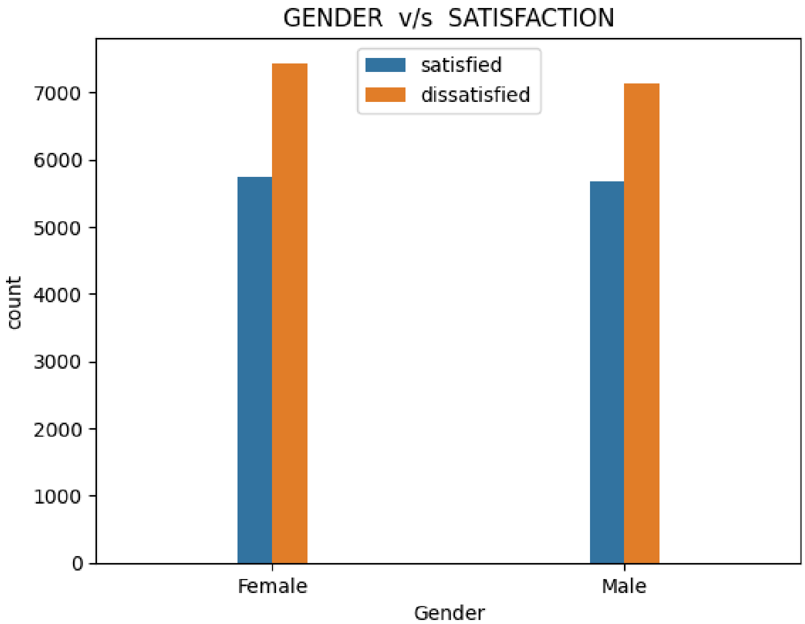
```
Class
Business    12495
Eco         11564
Eco Plus     1917
Name: count, dtype: int64
```

```
sns.countplot(data=df,x='Class',color='pink')
plt.title('COUNTPLOT OF PASSENGERS CLASS')
plt.show()
```



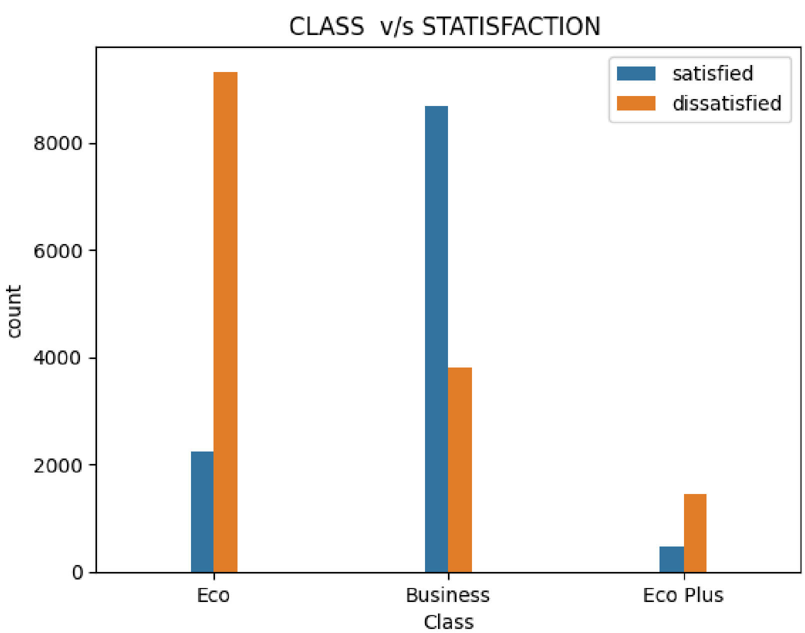
HOW THE GENDER RELATED TO SATISFACTION

```
sns.countplot(data=df,x='Gender',hue='satisfaction',width=0.2)
plt.title('GENDER  v/s  SATISFACTION')
plt.legend(loc='upper center')
plt.show()
```



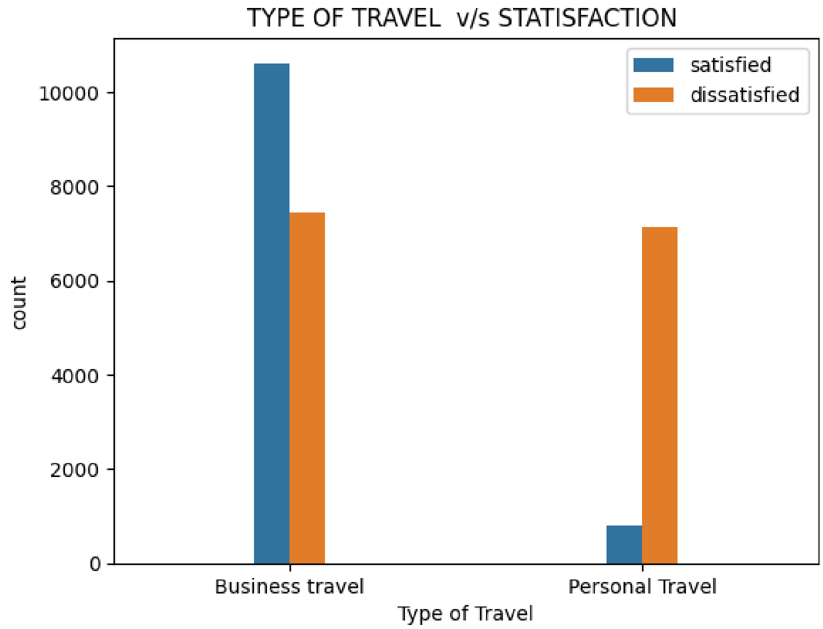
HOW THE CLASS IS RELATED TO SATISFACTION

```
sns.countplot(data=df,x='Class',hue='satisfaction',width=0.2)
plt.title('CLASS  v/s STATISFACTION ')
plt.legend(loc='upper right')
plt.show()
```

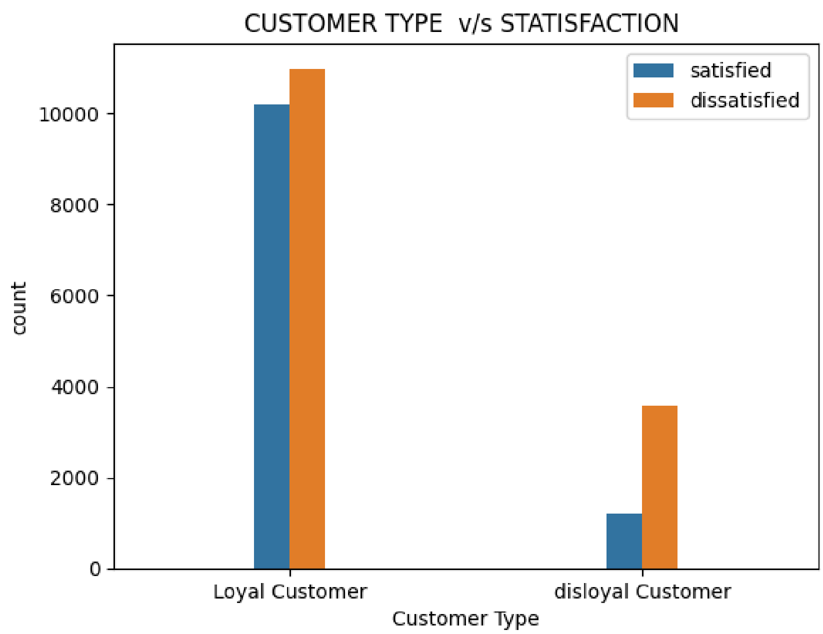


HOW THE TYPE OF TRAVEL IS RELATED TO SATISFACTION

```
sns.countplot(data=df,x='Type of Travel',hue='satisfaction',width=0.2)
plt.title('TYPE OF TRAVEL  v/s STATISFACTION ')
plt.legend(loc='upper right')
plt.show()
```




```
sns.countplot(data=df,x='Customer Type',hue='satisfaction',width=0.2)
plt.title('CUSTOMER TYPE  v/s STATISFACTION ')
plt.legend(loc='upper right')
plt.show()
```

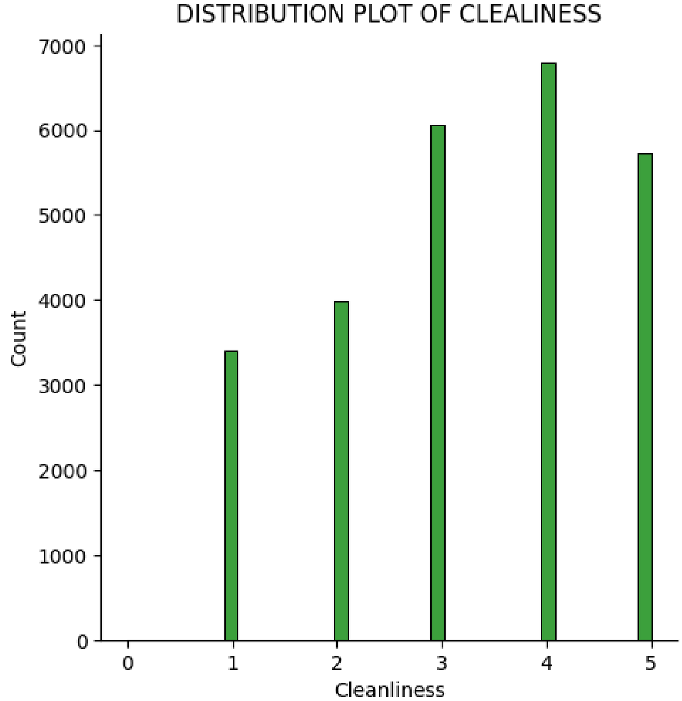



```
df.Cleanliness.value_counts()
```

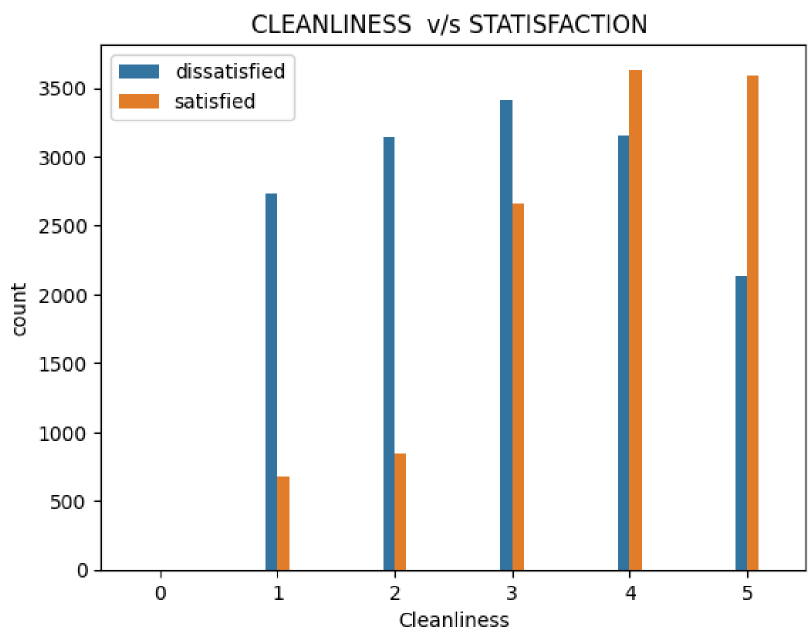
```
Cleanliness
4    6790
3    6065
5    5727
2    3981
1    3411
0       2
Name: count, dtype: int64
```

```
sns.displot(df['Cleanliness'],color='green')
plt.title('DISTRIBUTION PLOT OF CLEALINESS')
```

```
Text(0.5, 1.0, 'DISTRIBUTION PLOT OF CLEALINESS')
```




```
sns.countplot(data=df,x='Cleanliness',hue='satisfaction',width=0.2)
plt.title('CLEANLINESS  v/s STATISFACTION ')
plt.legend(loc='upper left')
plt.show()
```

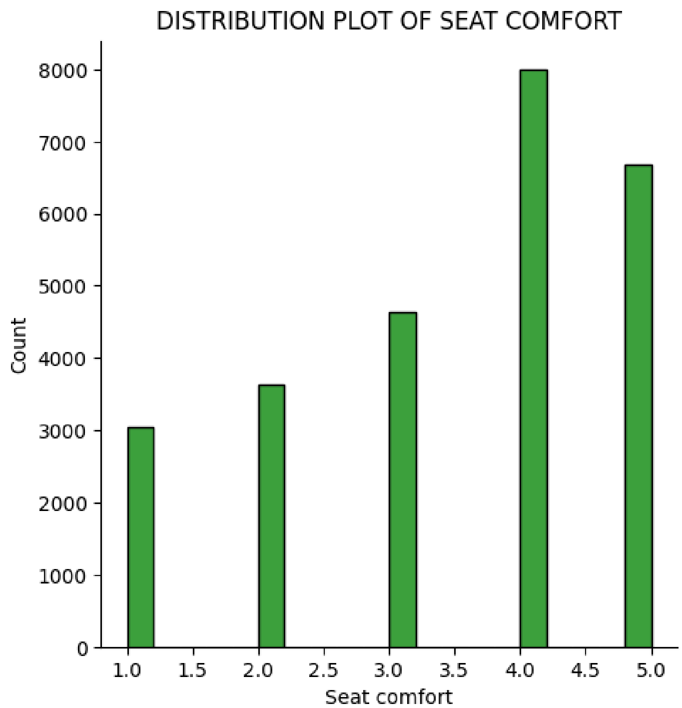



```
df['Seat comfort'].value_counts()
```

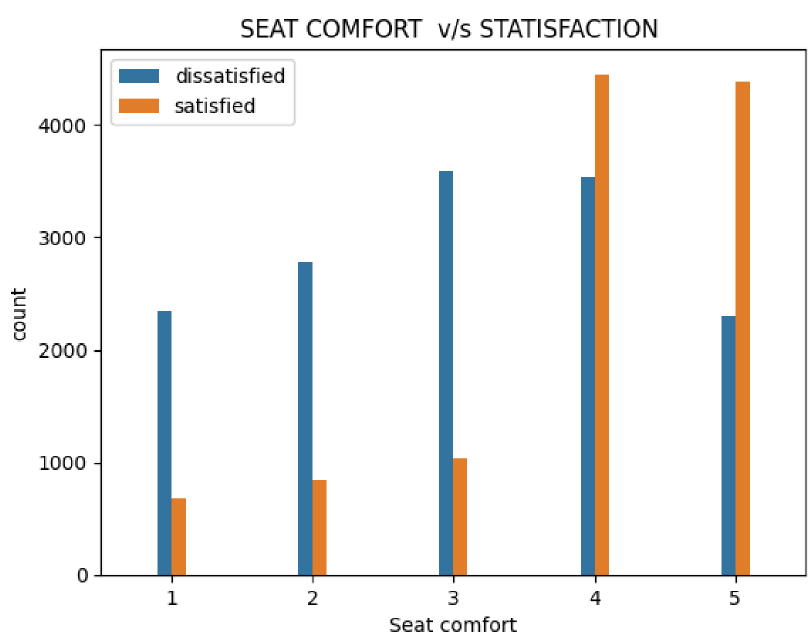
```
Seat comfort
4    7991
5    6688
3    4632
2    3632
1    3033
Name: count, dtype: int64
```

```
sns.displot(df['Seat comfort'],color='green')
plt.title('DISTRIBUTION PLOT OF SEAT COMFORT')
```

```
Text(0.5, 1.0, 'DISTRIBUTION PLOT OF SEAT COMFORT')
```




```
sns.countplot(data=df,x='Seat comfort',hue='satisfaction',width=0.2)
plt.title('SEAT COMFORT  v/s STATISFACTION ')
plt.legend(loc='upper left')
plt.show()
```

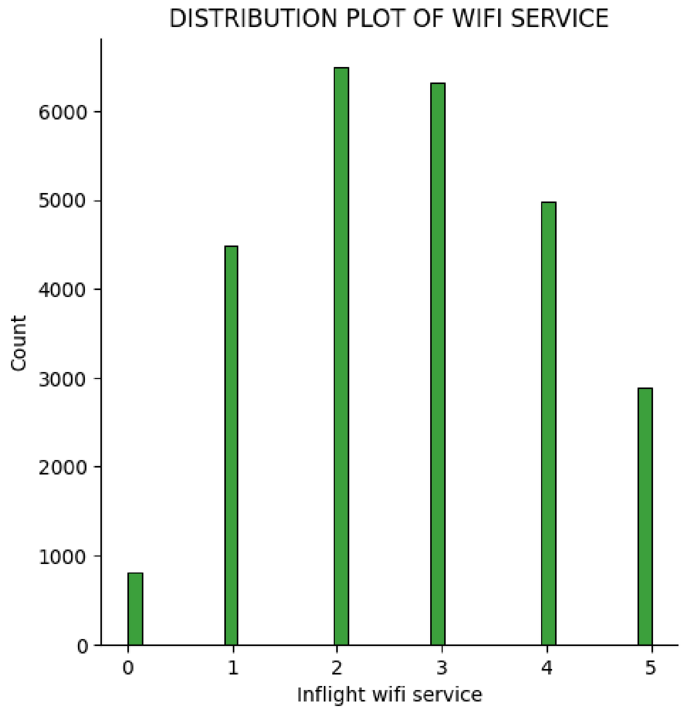



```
df['Inflight wifi service'].value_counts()
```

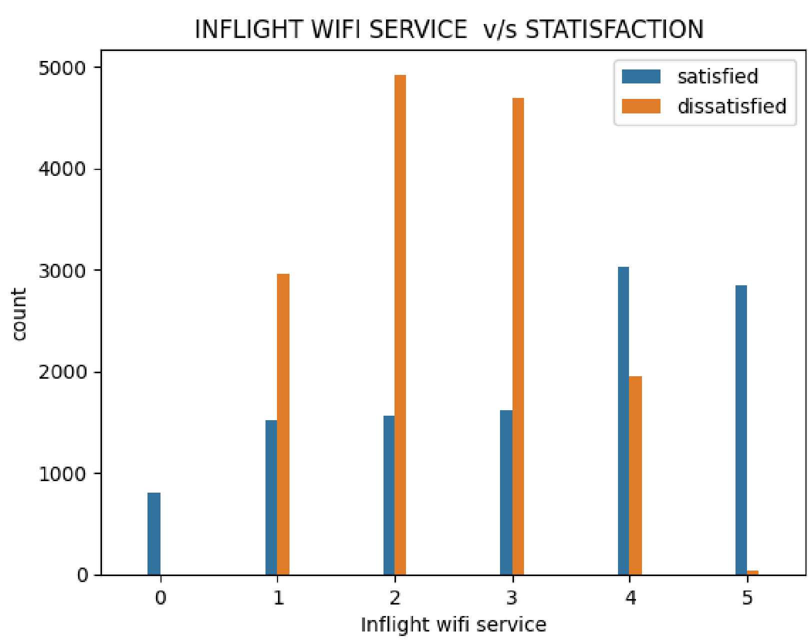
```
Inflight wifi service
2    6490
3    6317
4    4981
1    4488
5    2887
0     813
Name: count, dtype: int64
```

```
sns.displot(df['Inflight wifi service'],color='green')
plt.title('DISTRIBUTION PLOT OF WIFI SERVICE')
```

```
Text(0.5, 1.0, 'DISTRIBUTION PLOT OF WIFI SERVICE')
```




```
sns.countplot(data=df,x='Inflight wifi service',hue='satisfaction',width=0.2)
plt.title('INFLIGHT WIFI SERVICE  v/s STATISFACTION ')
plt.legend(loc='upper right')
plt.show()
```



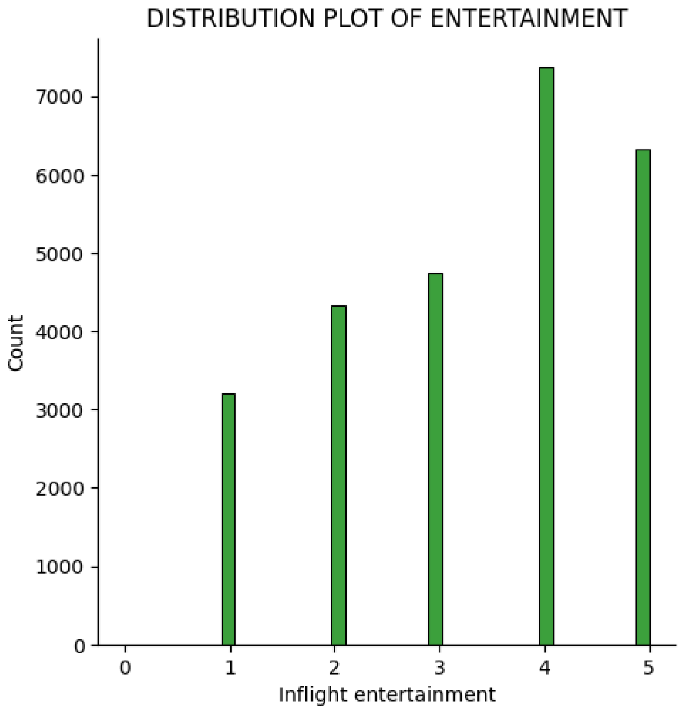

```
df['Inflight entertainment'].value_counts()
```

```
Inflight entertainment
4    7368
5    6331
3    4745
2    4331
1    3197
```

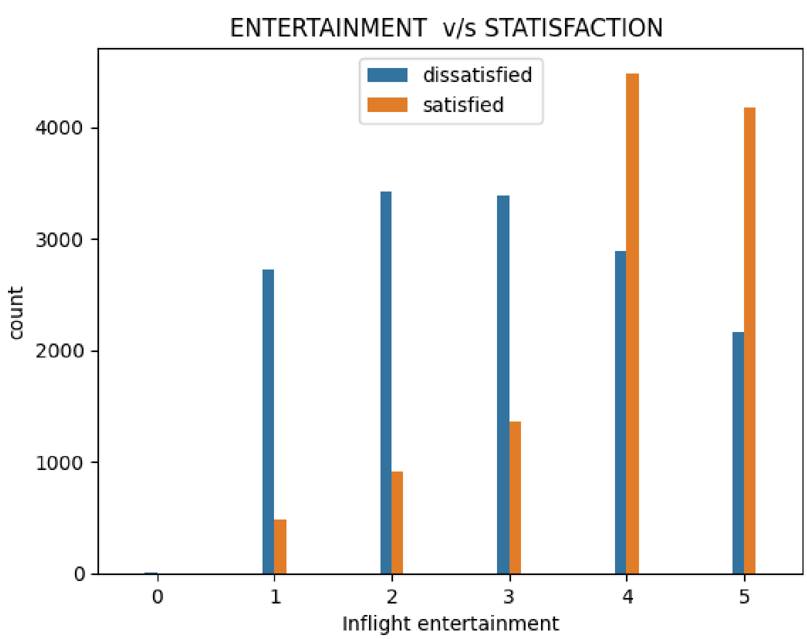
```
0       4
Name: count, dtype: int64
```

```
sns.displot(df['Inflight entertainment'],color='green')
plt.title('DISTRIBUTION PLOT OF ENTERTAINMENT')
```

```
Text(0.5, 1.0, 'DISTRIBUTION PLOT OF ENTERTAINMENT')
```




```
sns.countplot(data=df,x='Inflight entertainment',hue='satisfaction',width=0.2)
plt.title('ENTERTAINMENT  v/s STATISFACTION ')
plt.legend(loc='upper center')
plt.show()
```



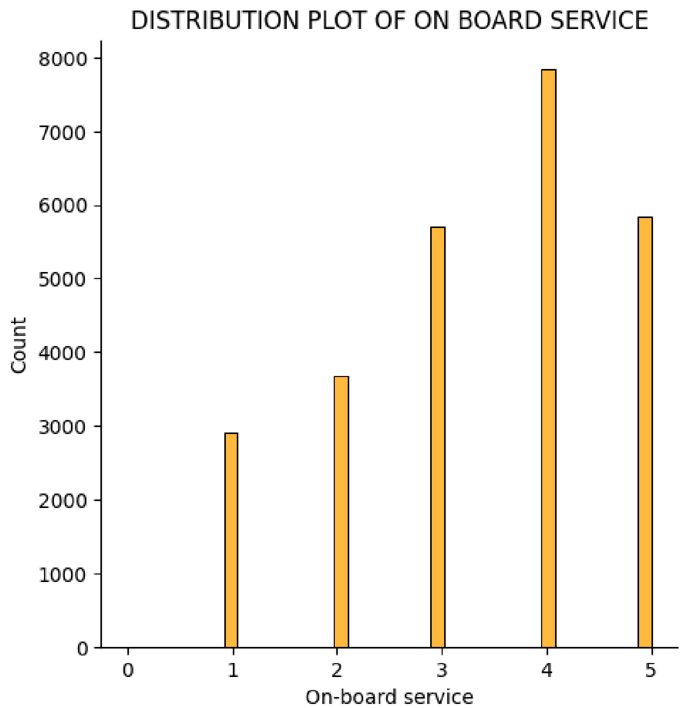

```
df['On-board service'].value_counts()
```

```
On-board service
4    7836
5    5844
3    5709
```

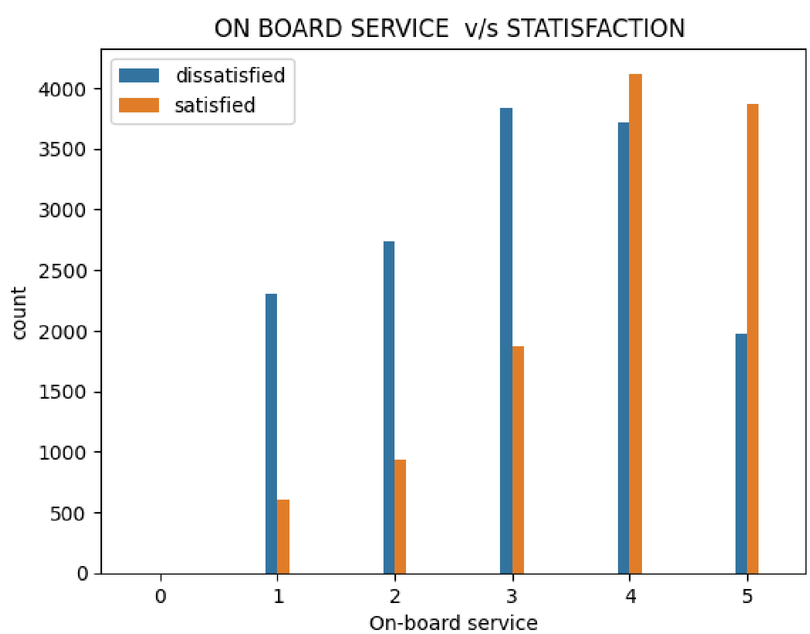
```
2    3670
1    2915
0       2
Name: count, dtype: int64
```

```
sns.displot(df['On-board service'],color='orange')
plt.title('DISTRIBUTION PLOT OF ON BOARD SERVICE')
```

```
Text(0.5, 1.0, 'DISTRIBUTION PLOT OF ON BOARD SERVICE')
```




```
sns.countplot(data=df,x='On-board service',hue='satisfaction',width=0.2)
plt.title('ON BOARD SERVICE  v/s STATISFACTION ')
plt.legend(loc='upper left')
plt.show()
```



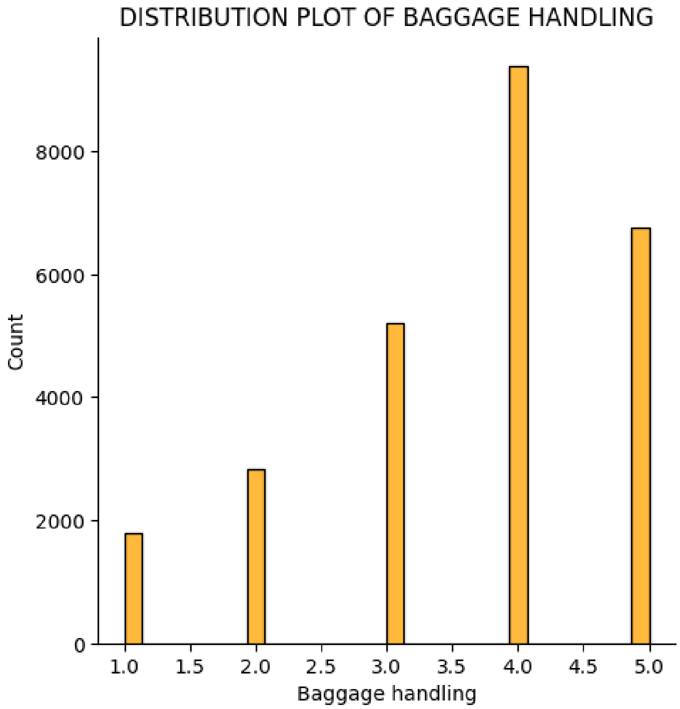

```
df['Baggage handling'].value_counts()
```

```
Baggage handling
4    9378
```

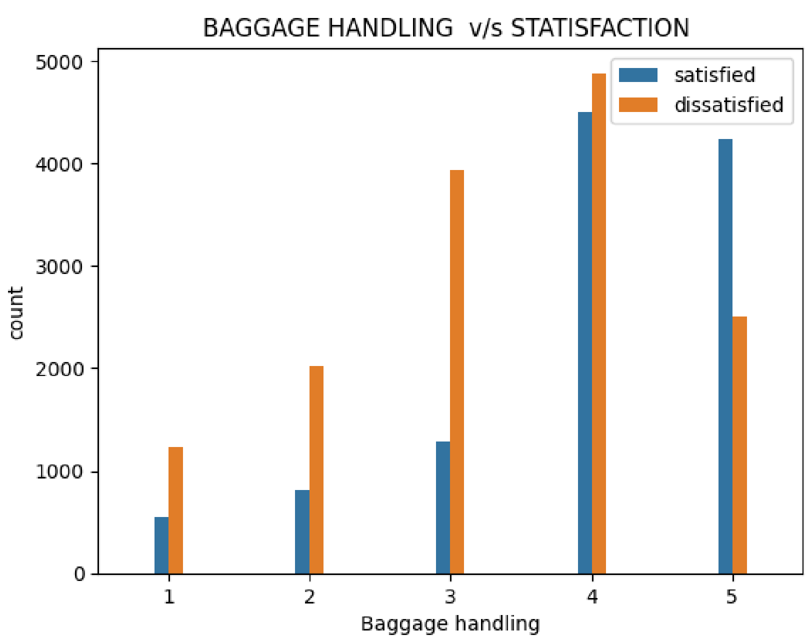
```
5    6747
3    5219
2    2841
1    1791
Name: count, dtype: int64
```

```
sns.displot(df['Baggage handling'],color='orange')
plt.title('DISTRIBUTION PLOT OF BAGGAGE HANDLING')
```

```
Text(0.5, 1.0, 'DISTRIBUTION PLOT OF BAGGAGE HANDLING')
```

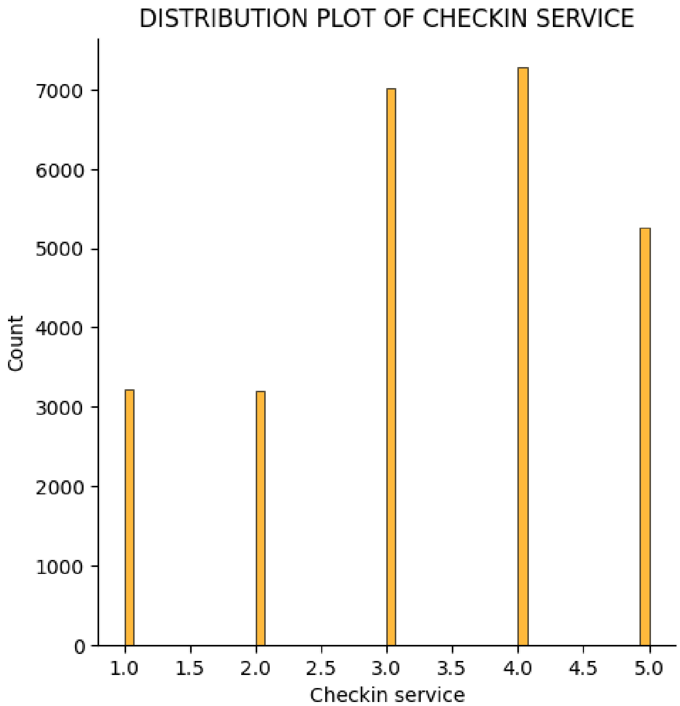



```
sns.countplot(data=df,x='Baggage handling',hue='satisfaction',width=0.2)
plt.title('BAGGAGE HANDLING  v/s STATISFACTION ')
plt.legend(loc='upper right')
plt.show()
```

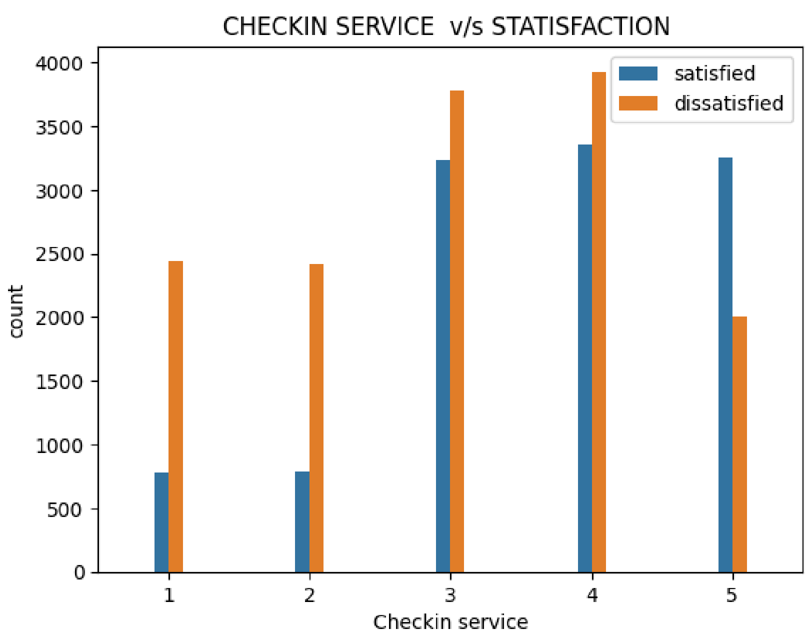



```
sns.displot(df['Checkin service'],color='orange')
plt.title('DISTRIBUTION PLOT OF CHECKIN SERVICE')
```

```
Text(0.5, 1.0, 'DISTRIBUTION PLOT OF CHECKIN SERVICE')
```




```
sns.countplot(data=df,x='Checkin service',hue='satisfaction',width=0.2)
plt.title('CHECKIN SERVICE  v/s STATISFACTION ')
plt.legend(loc='upper right')
plt.show()
```

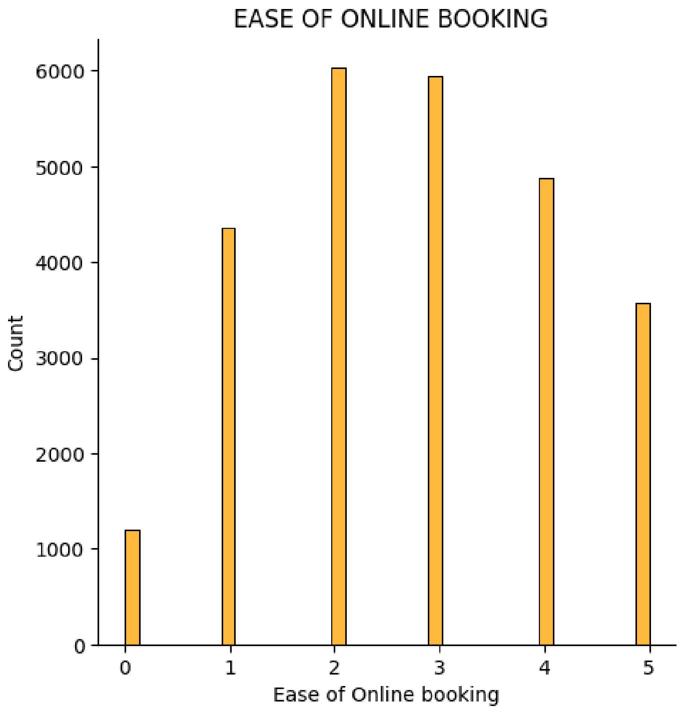



```
df['Ease of Online booking'].value_counts()
```

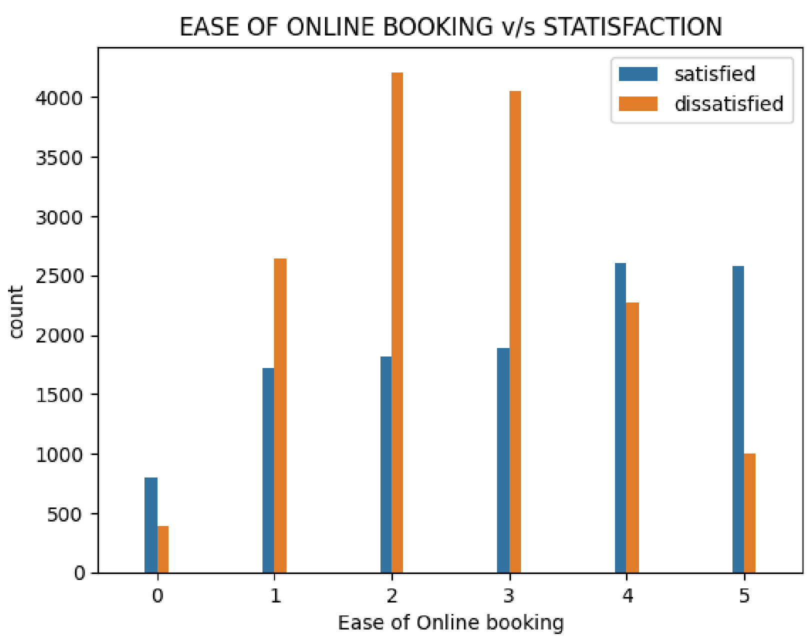
```
Ease of Online booking
2    6030
3    5944
4    4873
1    4361
5    3573
0    1195
Name: count, dtype: int64
```

```
sns.displot(df['Ease of Online booking'],color='orange')
plt.title(' EASE OF ONLINE BOOKING')
```

```
Text(0.5, 1.0, ' EASE OF ONLINE BOOKING')
```




```
sns.countplot(data=df,x='Ease of Online booking',hue='satisfaction',width=0.2)
plt.title(' EASE OF ONLINE BOOKING v/s STATISFACTION ')
plt.legend(loc='upper right')
plt.show()
```

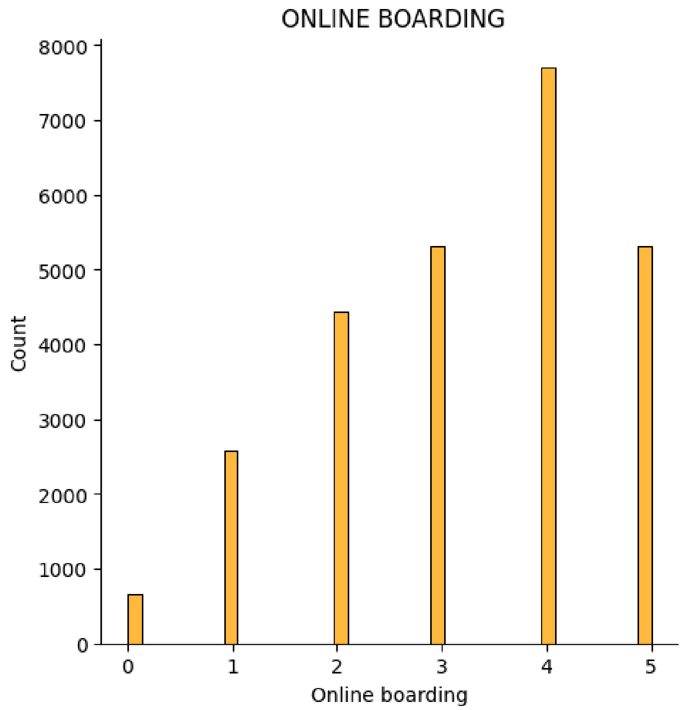



```
df['Online boarding'].value_counts()
```

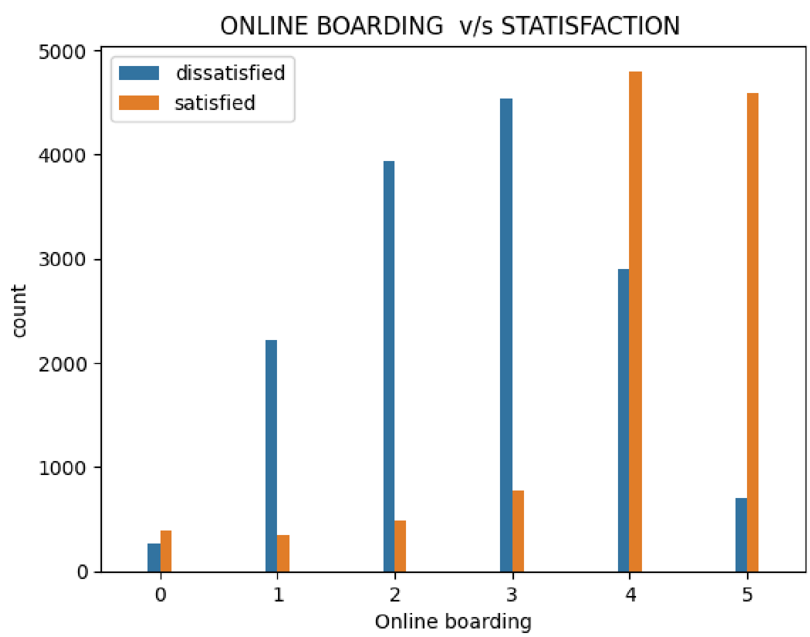
```
Online boarding
4    7706
3    5313
5    5307
2    4429
1    2569
0     652
Name: count, dtype: int64
```

```
sns.displot(df['Online boarding'],color='orange')
plt.title(' ONLINE BOARDING')
```

```
Text(0.5, 1.0, ' ONLINE BOARDING')
```




```
sns.countplot(data=df,x='Online boarding',hue='satisfaction',width=0.2)
plt.title('ONLINE BOARDING  v/s STATISFACTION ')
plt.legend(loc='upper left')
plt.show()
```



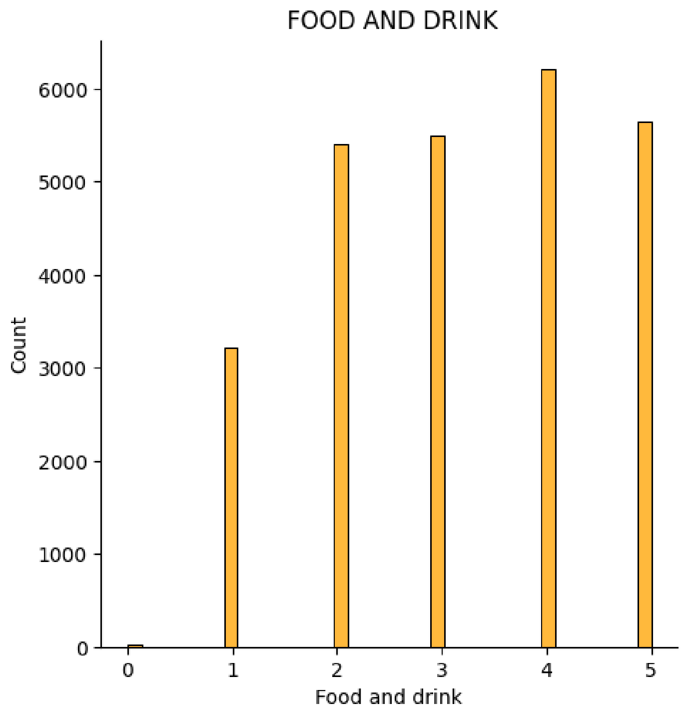

```
df['Food and drink'].value_counts()
```

```
Food and drink
4    6204
5    5644
3    5494
2    5395
```

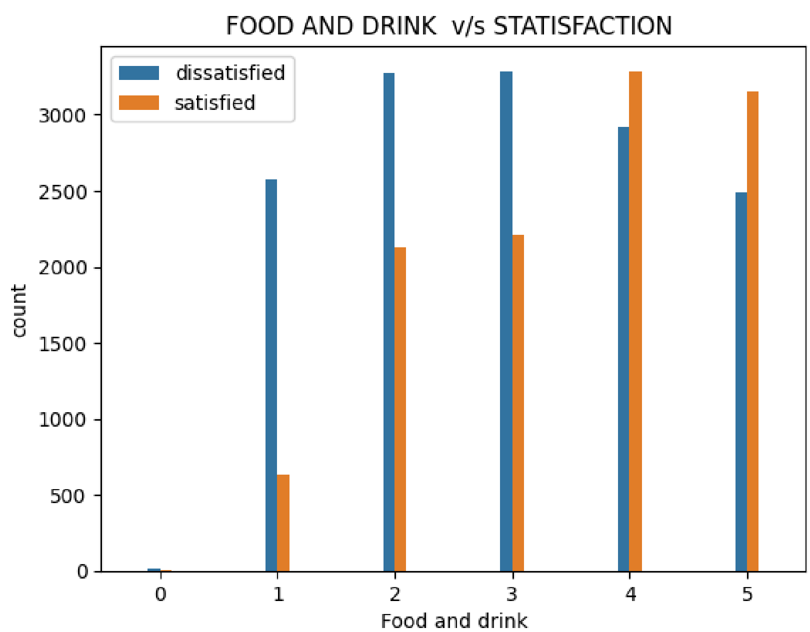
```
1    3214
0      25
Name: count, dtype: int64
```

```
sns.displot(df['Food and drink'],color='orange')
plt.title(' FOOD AND DRINK')
```

```
Text(0.5, 1.0, ' FOOD AND DRINK')
```




```
sns.countplot(data=df,x='Food and drink',hue='satisfaction',width=0.2)
plt.title('FOOD AND DRINK  v/s STATISFACTION ')
plt.legend(loc='upper left')
plt.show()
```



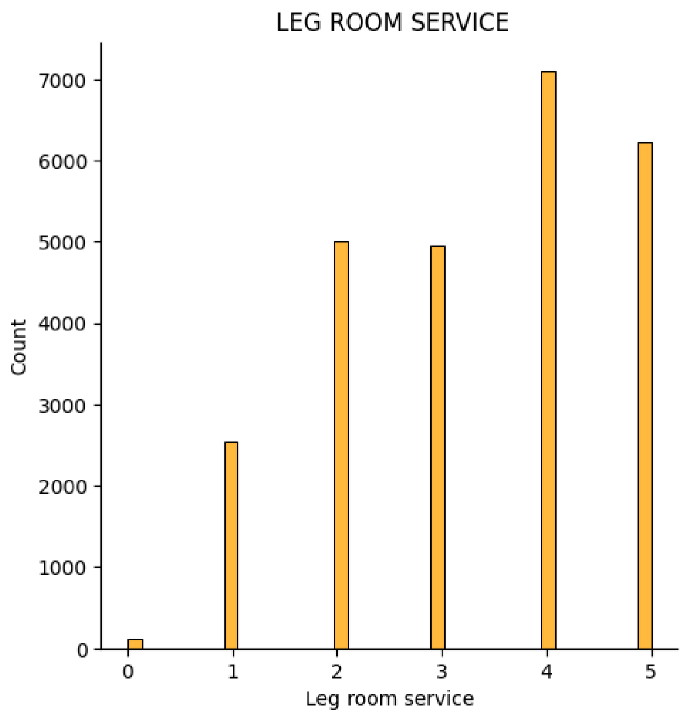

```
df['Leg room service'].value_counts()
```

```
Leg room service
4    7097
5    6238
```

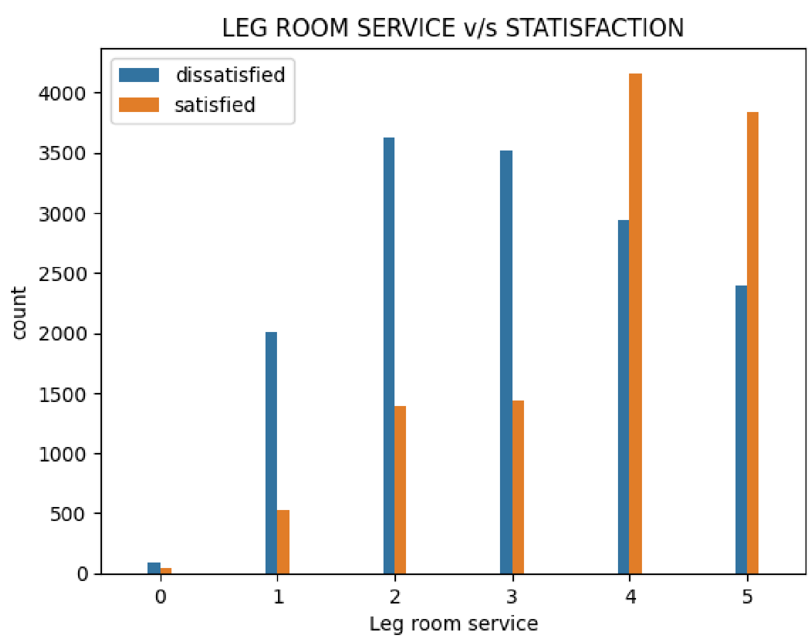
```
2    5015
3    4958
1    2542
0     126
Name: count, dtype: int64
```

```
sns.displot(df['Leg room service'],color='orange')
plt.title(' LEG ROOM SERVICE')
```

```
Text(0.5, 1.0, ' LEG ROOM SERVICE')
```



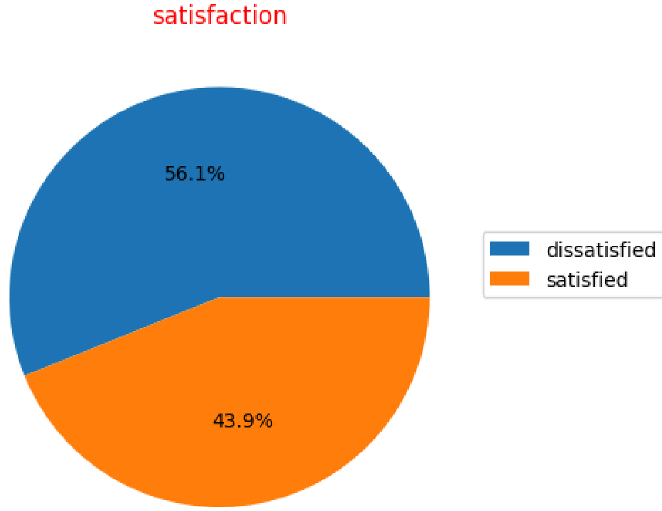

```
sns.countplot(data=df,x='Leg room service',hue='satisfaction',width=0.2)
plt.title(' LEG ROOM SERVICE v/s STATISFACTION ')
plt.legend(loc='upper left')
plt.show()
```



PLOT THE PIE CHART OF SATISFICATION LEVEL

```
plt.pie(df['satisfaction'].value_counts(),autopct='%1.1f%%')
plt.legend(df['satisfaction'].value_counts().index,loc=(1,0.5))
plt.title('satisfaction',color='red')
```

```
Text(0.5, 1.0, 'satisfaction')
```



LABEL ENCODING

LabelEncoding Machine learning models work only with numerical values.so categorical columns are converted in to numerical values.

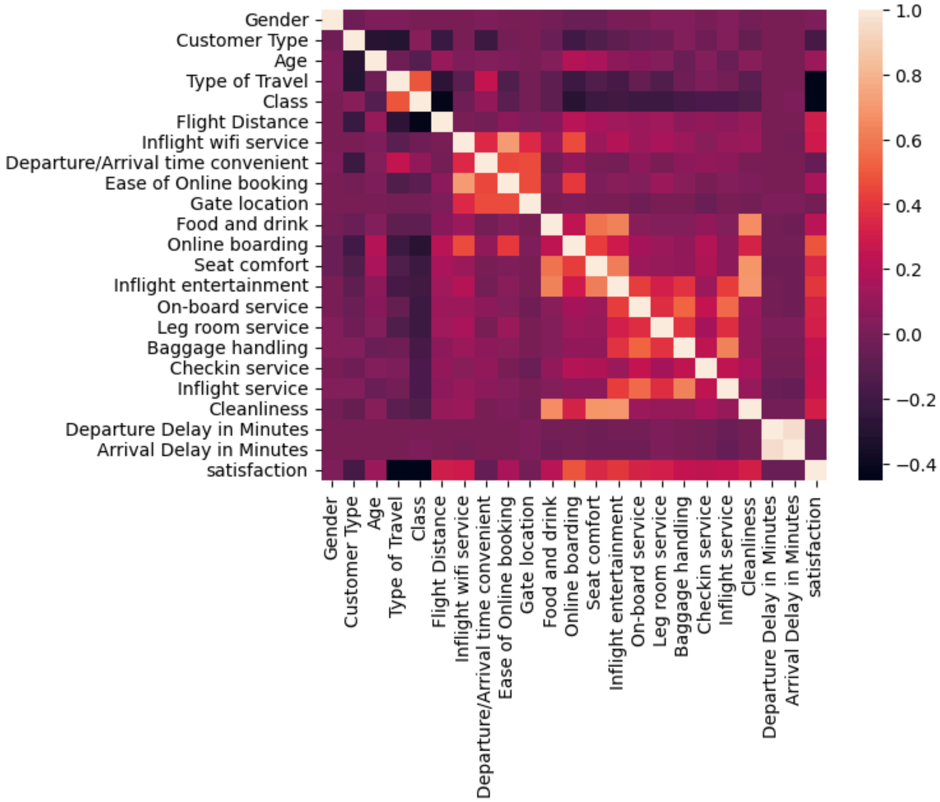
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Gender']=le.fit_transform(df['Gender'])
df['Customer Type']=le.fit_transform(df['Customer Type'])
df['Type of Travel'] = le.fit_transform(df['Type of Travel'])
df['Class'] = le.fit_transform(df['Class'])
df['satisfaction']=le.fit_transform(df['satisfaction'])
```

```
df
```

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease Onl book
0	0	0	52	0	1	160	5	4	
1	0	0	36	0	0	2863	1	1	
2	1	1	20	0	1	192	2	0	
3	1	0	44	0	0	3377	0	0	
4	0	0	49	0	1	1182	2	3	
...	...	...	...	...	...	...	...	...	
25971	1	1	34	0	0	526	3	3	
25972	1	0	23	0	0	646	4	4	
25973	0	0	17	1	1	828	2	5	
25974	1	0	14	0	0	1127	3	3	
25975	0	0	42	1	1	264	2	5	

25976 rows × 23 columns

```
correlation=df.corr()
heatmap=sns.heatmap(correlation)
```




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

```
satisfaction
0    14573
1    11403
Name: count, dtype: int64
```

```
x=df.drop(columns='satisfaction')
y=df.satisfaction
```

TRAIN TEST SPLIT AND SCALING

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train_scaled=sc.fit_transform(x_train)
x_test_scaled=sc.transform(x_test)
```

MODEL CREATION

K-Nearest Neighbors algorithm(KNN) The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today.

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn=KNeighborsClassifier(n_neighbors=5, metric='euclidean')
knn.fit(x_train_scaled,y_train)
y_pred_knn=knn.predict(x_test_scaled)
y_pred_knn
```

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

```
np.array(y_test)
```

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

NAIVE_BAYES

Naive Bayes classifier is a probabilistic machine learning model based on Bayes' theorem.

```
from sklearn.naive_bayes import BernoulliNB
naive=BernoulliNB()
naive.fit(x_train_scaled,y_train)
y_pred_naive=naive.predict(x_test_scaled)
y_pred_naive
```

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

```
np.array(y_test)
```

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

SUPPORT VECTOR MACHINE

A support vector machine (SVM) is a type of supervised learning algorithm used in machine learning to solve classification and regression tasks; SVMs are particularly good at solving binary classification problems, which require classifying the elements of a data set into two groups.

```
from sklearn.svm import SVC
sv=SVC(kernel='linear',C=0.1)
sv.fit(x_train_scaled,y_train)
y_pred_sv=sv.predict(x_test_scaled)
y_pred_sv
```

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

```
np.array(y_test)
```

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

DECISION TREE

A decision tree is a non-parametric supervised learning algorithm for classification and regression tasks. It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaf nodes. Decision trees are used for classification and regression tasks, providing easy-to-understand models.

```
from sklearn.tree import DecisionTreeClassifier
tree=DecisionTreeClassifier(max_depth=15,criterion='entropy',splitter='random',
min_samples_split=9,class_weight='balanced')
tree.fit(x_train_scaled,y_train)
y_pred_tree=tree.predict(x_test_scaled)
y_pred_tree
```

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

```
np.array(y_test)
```

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

```
#from sklearn.model_selection import GridSearchCV
#gs = GridSearchCV(estimator=DecisionTreeClassifier(),
#                     param_grid={"criterion": ["gini", "entropy"], "splitter": ["random", "best"], "max_depth": [10, 15, 20, 25, 30
#                    "class_weight": ["balanced", "None"], "min_samples_split": [3, 5, 6, 7, 9]}, cv=5)
```

```
#gs.fit(x_train_scaled, y_train)
```

```
#gs.best_params_
```

RANDOM FOREST

Random forest is a commonly-used machine learning algorithm that combines the output of multiple decision trees to reach a single result.

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=100,max_depth=100,min_samples_split=5,criterion='entropy',bootstrap=False)
rf.fit(x_train_scaled,y_train)
y_pred_rf=rf.predict(x_test_scaled)
y_pred_rf
```

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

```
np.array(y_test)
```

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

XGBOOST

XGBoost is a boosting algorithm that uses bagging, which trains multiple decision trees and then combines the results. It allows XGBoost to learn more quickly than other algorithms but also gives it an advantage in situations with many features to consider.

```
import xgboost
xb=xgboost.XGBClassifier(n_estimators=100)
xb.fit(x_train_scaled,y_train)
y_pred_xb=xb.predict(x_test_scaled)
y_pred_xb
```

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

```
np.array(y_test)
```

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

CLASSIFICATION REPORT AND CONFUSION MATRIX

A classification report is a performance evaluation metric in machine learning. It is used to show the precision, recall, F1 Score,accuracy score and support of your trained classification model.

Confusion Matrix - an overview | ScienceDirect Topics A confusion matrix is a table that is used to define the performance of a classification algorithm. A confusion matrix visualizes and summarizes the performance of a classification algorithm

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,ConfusionMatrixDisplay
print(classification_report(y_test,y_pred_knn))
```

```
              precision    recall  f1-score   support

           0       0.90      0.95      0.93      4353
           1       0.93      0.87      0.90      3440

    accuracy                           0.91      7793
   macro avg       0.92      0.91      0.91      7793
weighted avg       0.91      0.91      0.91      7793
```

```
print(accuracy_score(y_test,y_pred_knn))
```

```
0.9140254074169126
```

```
print(classification_report(y_test,y_pred_naive))
```

```
              precision    recall  f1-score   support

           0       0.84      0.88      0.86      4353
           1       0.84      0.79      0.81      3440

    accuracy                           0.84      7793
   macro avg       0.84      0.84      0.84      7793
weighted avg       0.84      0.84      0.84      7793
```

```
print(accuracy_score(y_test,y_pred_naive))
```

```
0.8404978827152573
```

```
print(classification_report(y_test,y_pred_sv))
```

```
              precision    recall  f1-score   support

           0       0.87      0.91      0.89      4353
           1       0.88      0.82      0.85      3440

    accuracy                           0.87      7793
   macro avg       0.87      0.86      0.87      7793
weighted avg       0.87      0.87      0.87      7793
```

```
print(accuracy_score(y_test,y_pred_sv))
```

```
0.8696265879635571
```

```
print(classification_report(y_test,y_pred_tree))
```

```
              precision    recall  f1-score   support

           0       0.94      0.94      0.94      4353
           1       0.93      0.93      0.93      3440

    accuracy                           0.94      7793
   macro avg       0.94      0.94      0.94      7793
weighted avg       0.94      0.94      0.94      7793
```

```
print(accuracy_score(y_test,y_pred_tree))
```

```
0.9359681765687156
```

```
print(classification_report(y_test,y_pred_rf))
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      4353
           1       0.96      0.94      0.95      3440

    accuracy                           0.95      7793
   macro avg       0.96      0.95      0.95      7793
weighted avg       0.95      0.95      0.95      7793
```

```
print(accuracy_score(y_test,y_pred_rf))
```

```
0.9547029385345823
```

```
print(classification_report(y_test,y_pred_xb))
```

```
              precision    recall  f1-score   support
```

```
              0       0.95      0.97      0.96      4353
              1       0.96      0.94      0.95      3440

       accuracy                           0.96      7793
      macro avg       0.96      0.95      0.96      7793
   weighted avg       0.96      0.96      0.96      7793
```

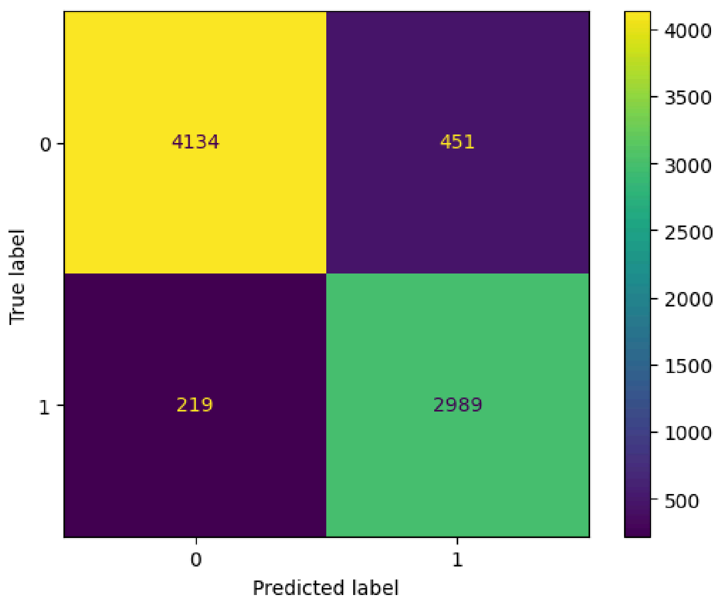
```
print(accuracy_score(y_test,y_pred_xb))
```

```
0.9557295008340818
```

CONFUSION MATRIX DISPLAY

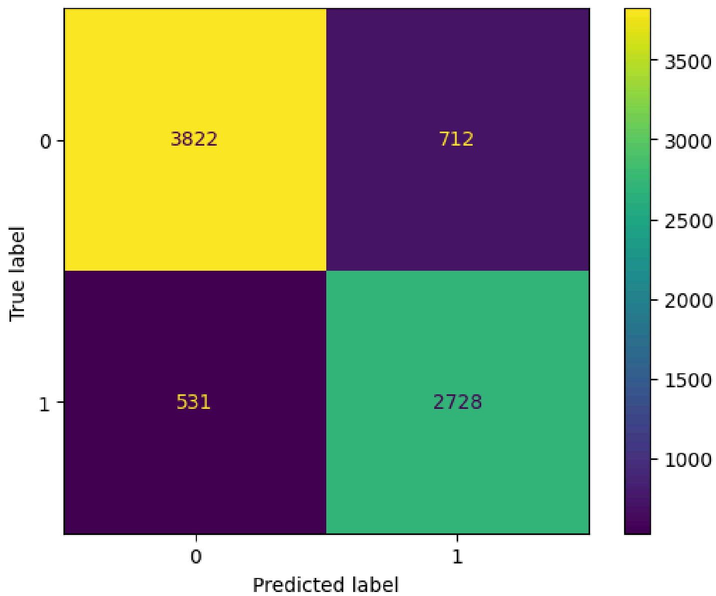
```
label=[0,1]
matx_knn=confusion_matrix(y_pred_knn,y_test)
print(matx_knn)
cmd=ConfusionMatrixDisplay(matx_knn,display_labels=label)
cmd.plot()
```

```
[[4134  451]
 [ 219 2989]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7c7d53d0bb20>
```



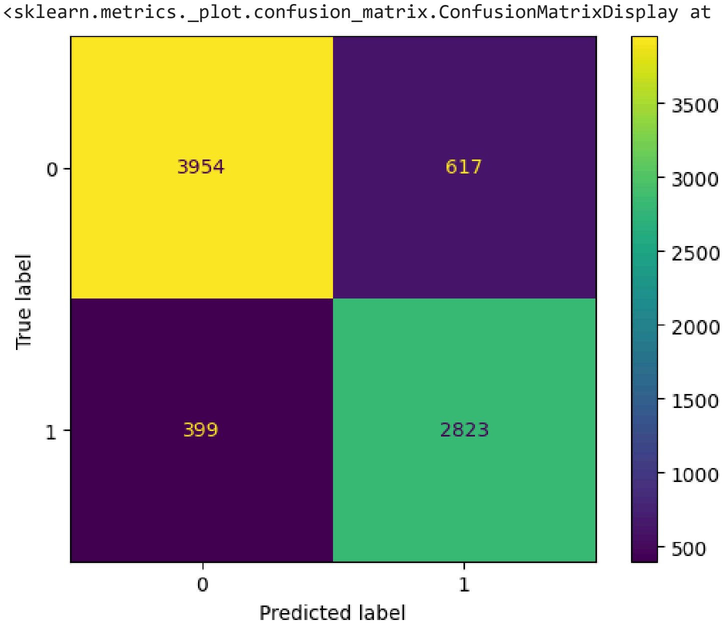

```
label=[0,1]
matx_naive=confusion_matrix(y_pred_naive,y_test)
print(matx_naive)
cmd=ConfusionMatrixDisplay(matx_naive,display_labels=label)
cmd.plot()
```

```
[[3822  712]
 [ 531 2728]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7c7d583969b0>
```



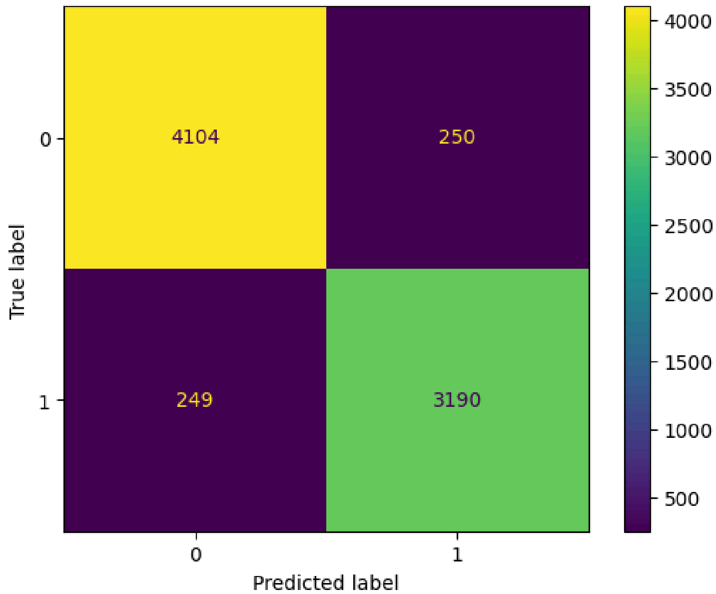

```
label=[0,1]
matx_sv=confusion_matrix(y_pred_sv,y_test)
print(matx_sv)
cmd=ConfusionMatrixDisplay(matx_sv,display_labels=label)
cmd.plot()
```

```
[[3954  617]
 [ 399 2823]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7c7d504c52d0>
```



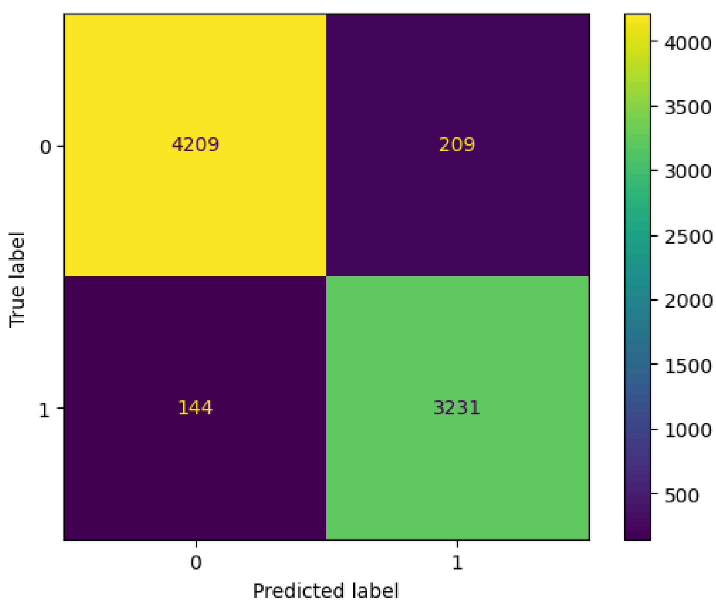

```
label=[0,1]
matx_tree=confusion_matrix(y_pred_tree,y_test)
print(matx_tree)
cmd=ConfusionMatrixDisplay(matx_tree,display_labels=label)
cmd.plot()
```

```
[[4104  250]
 [ 249 3190]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7c7d502dbb80>
```



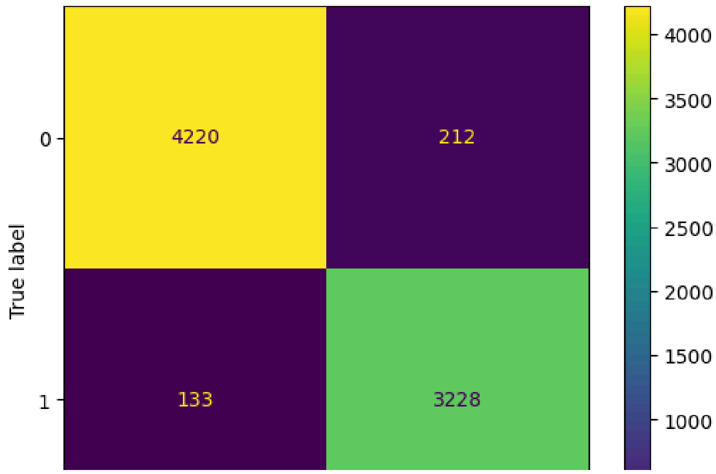

```
label=[0,1]
matx_rf=confusion_matrix(y_pred_rf,y_test)
print(matx_rf)
cmd=ConfusionMatrixDisplay(matx_rf,display_labels=label)
cmd.plot()
```

```
[[4209  209]
 [ 144 3231]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7c7d53ee7520>
```




```
label=[0,1]
matx_xb=confusion_matrix(y_pred_xb,y_test)
print(matx_xb)
cmd=ConfusionMatrixDisplay(matx_xb,display_labels=label)
cmd.plot()
```

```
[[4220  212]
 [ 133 3228]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7c7d5023fe20>
```



MODEL ACCURACY GRAPH

```
import plotly.express as px
model_names = ['tree', 'rf', 'xb', 'knn','naive','sv']
accuracy_scores = [93, 95, 95, 91,82,86]
data = {'Model': model_names, 'Accuracy Score': accuracy_scores}
df = pd.DataFrame(data)
colors = px.colors.qualitative.Pastel
fig = px.bar(df, x='Model', y='Accuracy Score', text='Accuracy Score',
             title='Comparison of Model Accuracy', color='Model',
             color_discrete_map={model: color for model, color in zip(model_names, colors)})

fig.update_traces(texttemplate='%{text:.2f}%', textposition='outside')
fig.update_layout(width=700, height=600)

fig.show()
```