Airline Passenger Satisfaction

Table of Contents

Data Features

Gender: Gender of the passengers (Female, Male)

Customer Type: The customer type (Loyal customer, disloyal customer)

Age: The actual age of the passengers

Type of Travel: Purpose of the flight of the passengers (Personal Travel, Business Travel)

Class: Travel class in the plane of the passengers (Business, Eco, Eco Plus)

Flight distance: The flight distance of this journey

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

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

Ease of Online booking: Satisfaction level of online booking

Gate location: Satisfaction level of Gate location

Food and drink: Satisfaction level of Food and drink

Online boarding: Satisfaction level of online boarding

Seat comfort: Satisfaction level of Seat comfort

Inflight entertainment: Satisfaction level of inflight entertainment

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

Leg room service: Satisfaction level of Leg room service

Baggage handling: Satisfaction level of baggage handling

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

Inflight service: Satisfaction level of inflight service

Cleanliness: Satisfaction level of Cleanliness

Departure Delay in Minutes: Minutes delayed when departure

Arrival Delay in Minutes: Minutes delayed when Arrival

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

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

plt.style.use('https://github.com/dhaitz/matplotlib-stylesheets/raw/master/pitayasmoothi

import warnings
warnings.filterwarnings('ignore')

# For displaying all of the columns in dataframes
pd.set_option('display.max_columns', None)

# For data modeling
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB

# For metrics and helpful functions
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Reading the Data

In [2]:
```
train = pd.read_csv('airline passenger satisfaction train.csv')
test = pd.read_csv('airline passenger satisfaction test.csv')
```

Statistical Information and General Information about the Data

train

In [3]:
```
train.describe()
```

Out[3]:

	Unnamed: 0	id	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Onli booki
count	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000(

mean	51951.500000	64924.210502	39.379706	1189.448375	2.729683	3.060296	2.7569
std	29994.645522	37463.812252	15.114964	997.147281	1.327829	1.525075	1.3989
min	0.000000	1.000000	7.000000	31.000000	0.000000	0.000000	0.0000
25%	25975.750000	32533.750000	27.000000	414.000000	2.000000	2.000000	2.0000
50%	51951.500000	64856.500000	40.000000	843.000000	3.000000	3.000000	3.0000
75%	77927.250000	97368.250000	51.000000	1743.000000	4.000000	4.000000	4.0000
max	103903.000000	129880.000000	85.000000	4983.000000	5.000000	5.000000	5.0000

In [4]:
```
test.describe()
```

Out[4]:

	Unnamed: 0	id	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	
count	25976.000000	25976.000000	25976.000000	25976.000000	25976.000000	25976.000000	25976.000000	2
mean	12987.500000	65005.657992	39.620958	1193.788459	2.724746	3.046812	2.756775	
std	7498.769632	37611.526647	15.135685	998.683999	1.335384	1.533371	1.412951	
min	0.000000	17.000000	7.000000	31.000000	0.000000	0.000000	0.000000	
25%	6493.750000	32170.500000	27.000000	414.000000	2.000000	2.000000	2.000000	
50%	12987.500000	65319.500000	40.000000	849.000000	3.000000	3.000000	3.000000	
75%	19481.250000	97584.250000	51.000000	1744.000000	4.000000	4.000000	4.000000	
max	25975.000000	129877.000000	85.000000	4983.000000	5.000000	5.000000	5.000000	

In [5]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103904 entries, 0 to 103903
Data columns (total 25 columns):
 #   Column                             Non-Null Count   Dtype
---  ------                             --------------   -----
 0   Unnamed: 0                         103904 non-null  int64
 1   id                                 103904 non-null  int64
 2   Gender                             103904 non-null  object
 3   Customer Type                      103904 non-null  object
 4   Age                                103904 non-null  int64
 5   Type of Travel                     103904 non-null  object
 6   Class                              103904 non-null  object
 7   Flight Distance                    103904 non-null  int64
 8   Inflight wifi service              103904 non-null  int64
 9   Departure/Arrival time convenient  103904 non-null  int64
 10  Ease of Online booking             103904 non-null  int64
 11  Gate location                      103904 non-null  int64
 12  Food and drink                     103904 non-null  int64
 13  Online boarding                    103904 non-null  int64
 14  Seat comfort                       103904 non-null  int64
 15  Inflight entertainment             103904 non-null  int64
 16  On-board service                   103904 non-null  int64
 17  Leg room service                   103904 non-null  int64
 18  Baggage handling                   103904 non-null  int64
 19  Checkin service                    103904 non-null  int64
 20  Inflight service                   103904 non-null  int64
 21  Cleanliness                        103904 non-null  int64
 22  Departure Delay in Minutes         103904 non-null  int64
```

```
 23  Arrival Delay in Minutes           103594 non-null  float64
 24  satisfaction                       103904 non-null  object 
dtypes: float64(1), int64(19), object(5)
memory usage: 19.8+ MB
```

In [6]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25976 entries, 0 to 25975
Data columns (total 25 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   Unnamed: 0                         25976 non-null  int64  
 1   id                                 25976 non-null  int64  
 2   Gender                             25976 non-null  object 
 3   Customer Type                      25976 non-null  object 
 4   Age                                25976 non-null  int64  
 5   Type of Travel                     25976 non-null  object 
 6   Class                              25976 non-null  object 
 7   Flight Distance                    25976 non-null  int64  
 8   Inflight wifi service              25976 non-null  int64  
 9   Departure/Arrival time convenient  25976 non-null  int64  
 10  Ease of Online booking             25976 non-null  int64  
 11  Gate location                      25976 non-null  int64  
 12  Food and drink                     25976 non-null  int64  
 13  Online boarding                    25976 non-null  int64  
 14  Seat comfort                       25976 non-null  int64  
 15  Inflight entertainment             25976 non-null  int64  
 16  On-board service                   25976 non-null  int64  
 17  Leg room service                   25976 non-null  int64  
 18  Baggage handling                   25976 non-null  int64  
 19  Checkin service                    25976 non-null  int64  
 20  Inflight service                   25976 non-null  int64  
 21  Cleanliness                        25976 non-null  int64  
 22  Departure Delay in Minutes         25976 non-null  int64  
 23  Arrival Delay in Minutes           25893 non-null  float64
 24  satisfaction                       25976 non-null  object 
dtypes: float64(1), int64(19), object(5)
memory usage: 5.0+ MB
```

In [7]:
```
train.columns
```

Out[7]:
```
Index(['Unnamed: 0', 'id', 'Gender', 'Customer Type', 'Age', 'Type of Travel',
       'Class', 'Flight Distance', 'Inflight wifi service',
       'Departure/Arrival time convenient', 'Ease of Online booking',
       'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort',
       'Inflight entertainment', 'On-board service', 'Leg room service',
       'Baggage handling', 'Checkin service', 'Inflight service',
       'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in Minutes',
       'satisfaction'],
      dtype='object')
```

In [8]:
```
test.columns
```

Out[8]:
```
Index(['Unnamed: 0', 'id', 'Gender', 'Customer Type', 'Age', 'Type of Travel',
       'Class', 'Flight Distance', 'Inflight wifi service',
       'Departure/Arrival time convenient', 'Ease of Online booking',
       'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort',
       'Inflight entertainment', 'On-board service', 'Leg room service',
       'Baggage handling', 'Checkin service', 'Inflight service',
       'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in Minutes',
       'satisfaction'],
      dtype='object')
```

In [9]:
```
train.head()
```

Out[9]:

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

In [10]:
```
test.head()
```

Out[10]:

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking
0	0	19556	Female	Loyal Customer	52	Business travel	Eco	160	5	4	3
1	1	90035	Female	Loyal Customer	36	Business travel	Business	2863	1	1	3
2	2	12360	Male	disloyal Customer	20	Business travel	Eco	192	2	0	2
3	3	77959	Male	Loyal Customer	44	Business travel	Business	3377	0	0	0
4	4	36875	Female	Loyal Customer	49	Business travel	Eco	1182	2	3	4

In [11]:
```
train.shape
```

Out[11]:
```
(103904, 25)
```

In [12]:
```
test.shape
```

Out[12]:
```
(25976, 25)
```

Data Cleaning

In [13]:
```
train.isnull().sum()
```

Out[13]:
```
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             310
satisfaction                           0
dtype: int64
```

In [14]:
```
test.isnull().sum()
```

Out[14]:
```
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
```

In [15]:
```
train.duplicated()
```

Out[15]:
```
0         False
1         False
2         False
3         False
4         False
          ...
103899    False
103900    False
103901    False
103902    False
103903    False
Length: 103904, dtype: bool
```

In [16]:
```
test.duplicated()
```

Out[16]:
```
0        False
1        False
2        False
3        False
```

```
4        False
         ...
25971    False
25972    False
25973    False
25974    False
25975    False
Length: 25976, dtype: bool
```

In [17]:
```
train.columns
```

Out[17]:
```
Index(['Unnamed: 0', 'id', 'Gender', 'Customer Type', 'Age', 'Type of Travel',
       'Class', 'Flight Distance', 'Inflight wifi service',
       'Departure/Arrival time convenient', 'Ease of Online booking',
       'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort',
       'Inflight entertainment', 'On-board service', 'Leg room service',
       'Baggage handling', 'Checkin service', 'Inflight service',
       'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in Minutes',
       'satisfaction'],
      dtype='object')
```

In [18]:
```
test.columns
```

Out[18]:
```
Index(['Unnamed: 0', 'id', 'Gender', 'Customer Type', 'Age', 'Type of Travel',
       'Class', 'Flight Distance', 'Inflight wifi service',
       'Departure/Arrival time convenient', 'Ease of Online booking',
       'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort',
       'Inflight entertainment', 'On-board service', 'Leg room service',
       'Baggage handling', 'Checkin service', 'Inflight service',
       'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in Minutes',
       'satisfaction'],
      dtype='object')
```

fill the missing value

In [19]:
```
train.fillna(train['Arrival Delay in Minutes'].mean(), inplace=True)
```

In [20]:
```
test.fillna(test['Arrival Delay in Minutes'].mean(), inplace=True)
```

In [21]:
```
train.isnull().sum()
```

Out[21]:
```
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
```

In [22]:
```
test.isnull().sum()
```

Out[22]:
```
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
```

Exploratory Data Analyst

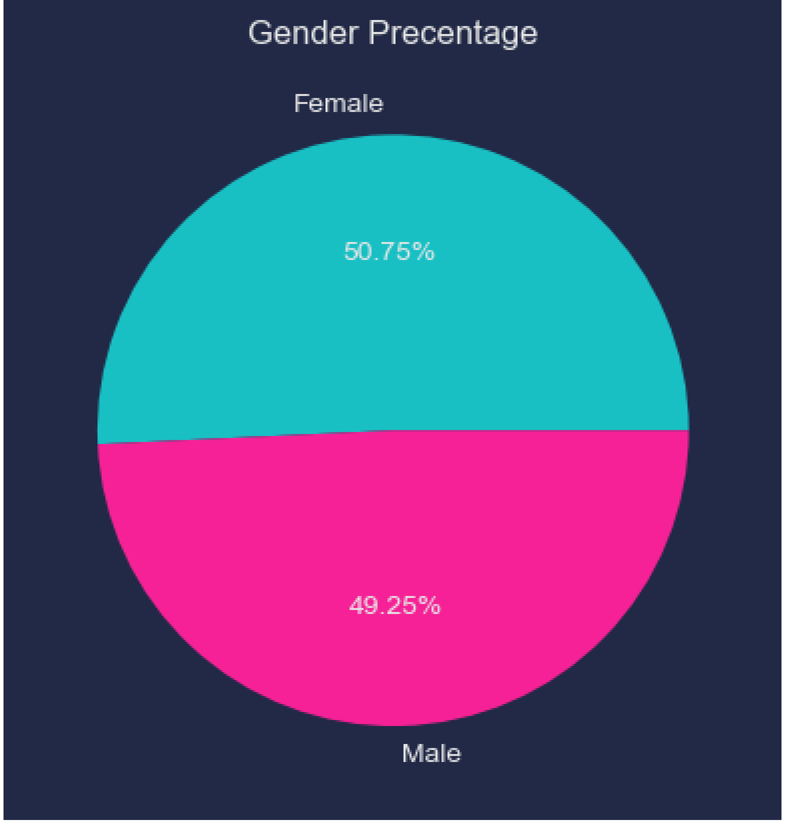
In [23]:
```
train.head()
```

Out[23]:

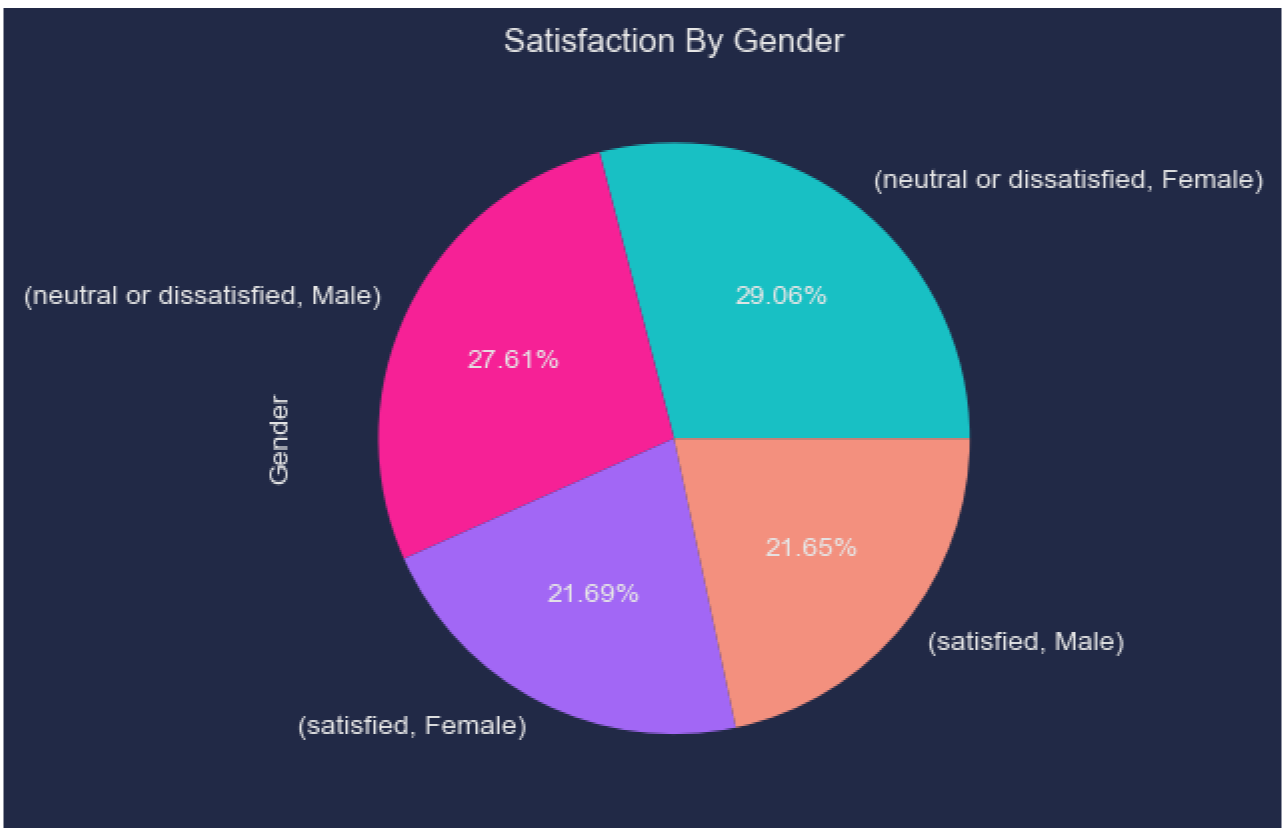
	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3
1	1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	3
2	2	110028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

Satisfaction by Gender

In [24]:
```
train.groupby('Gender').size().plot.pie(autopct='%1.2f%%')
plt.title("Gender Precentage")
plt.show()
```



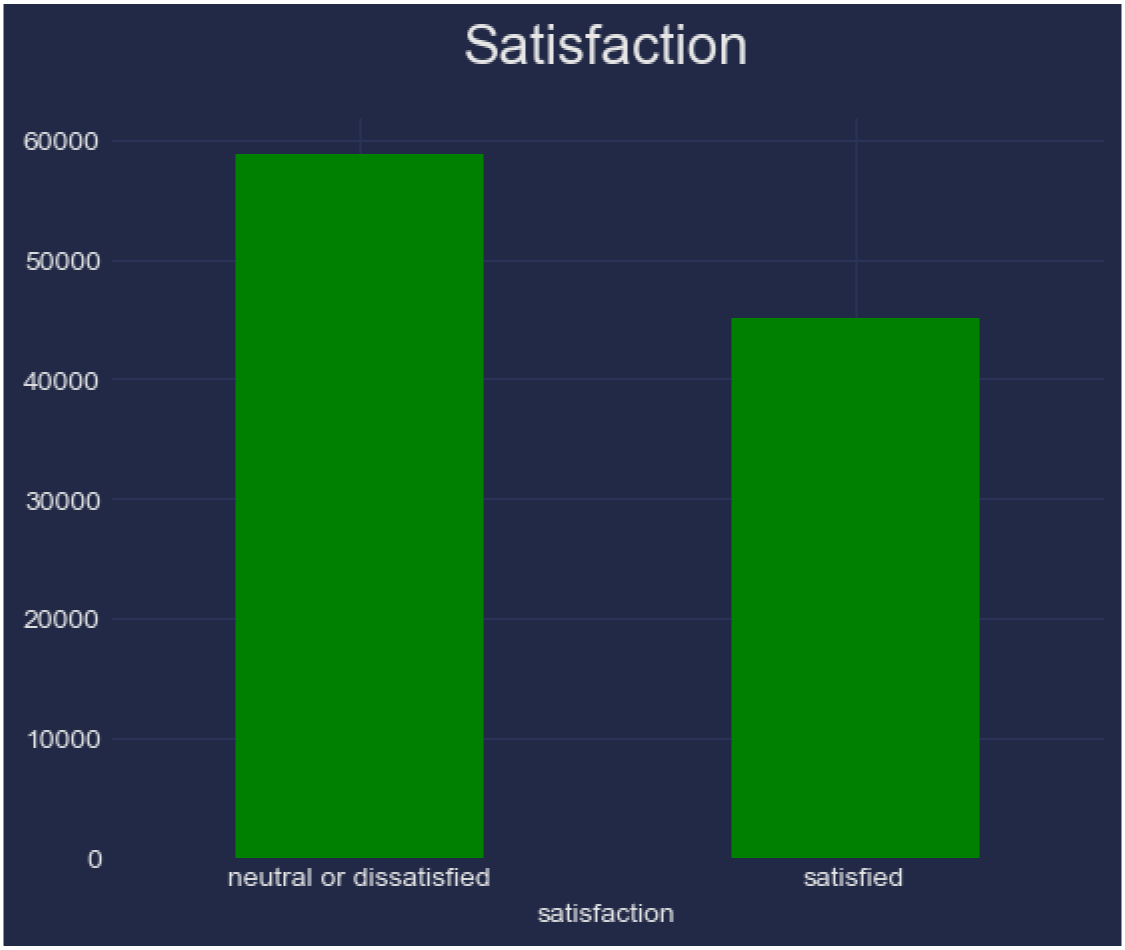
In [25]:
```
train.groupby('satisfaction')['Gender'].value_counts().plot.pie(autopct = '%1.2f%%')
plt.title('Satisfaction By Gender')
plt.show()
```



Bisa kita lihat, kepuasan berdasarkan jenis kelamin antara laki-laki dan perempuan. dari hasil diatas menenunjukkan bahwa, berdasarkan jenis kelamin, tingkat kepuasan atau satisfied jauh lebih kecil daripada tingkat ketidakpuasan.

In [26]:
```
train.groupby('satisfaction').size().plot(kind="bar", color = "green")
plt.title('Satisfaction', fontsize = 20, pad = 20)
```

```
plt.xticks(rotation = 0)
plt.show()
```



Tingkat ketidakpuasan jauh lebih banyak daripada tingkat kepuasan

In [27]:
```
train.head()
```

Out[27]:

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

Mencari Arrival dan departure delay in minutes tertinggi dan terendah

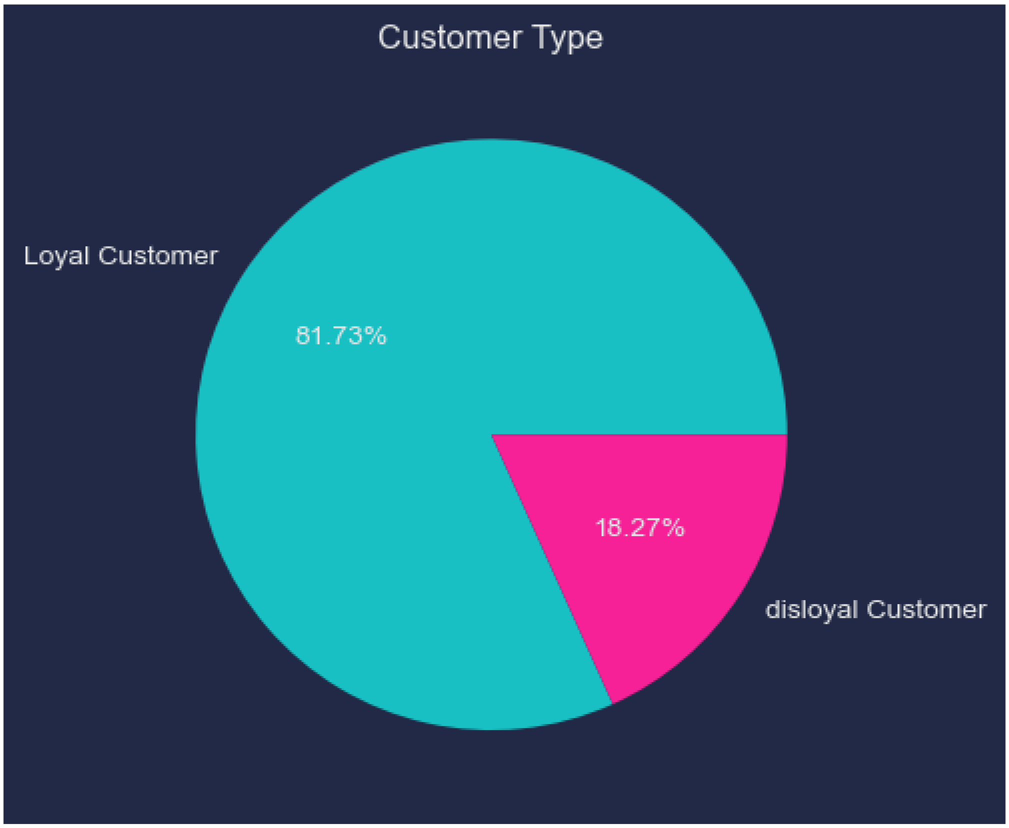
In [28]:
```
print("Arrival Delay in Minutes Tertinggi: ", train['Arrival Delay in Minutes'].max(), "
print("Departure Delay in Minutes Tertinggi: ", train['Departure Delay in Minutes'].max(

# Convert to Hours
convert_arrival = train['Arrival Delay in Minutes'].max()
```

```
convert_arrival = round(convert_arrival/60)
print(f"Arrival Convert to Hours: {convert_arrival} Hours")

convert_dpt = train['Departure Delay in Minutes'].max()
convert_dpt = round(convert_dpt/60)
print(f"Departure Convert to Hours: {convert_dpt} Hours")
```

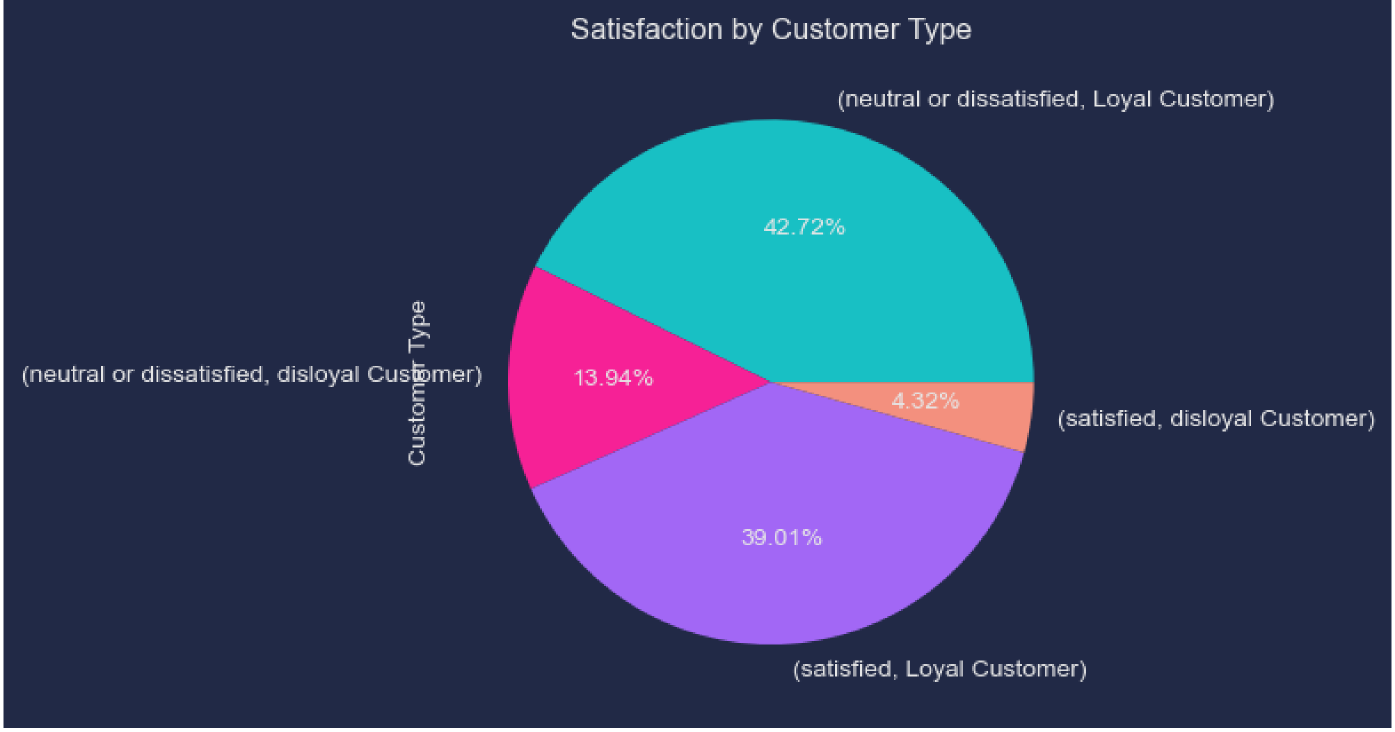
```
Arrival Delay in Minutes Tertinggi:  1584.0 Minutes
Departure Delay in Minutes Tertinggi:  1592 Minutes
Arrival Convert to Hours: 26 Hours
Departure Convert to Hours: 27 Hours
```

Customer type

In [29]:
```
train.groupby('Customer Type').size().plot.pie(autopct='%1.2f%%')
plt.title("Customer Type")
plt.show()
```

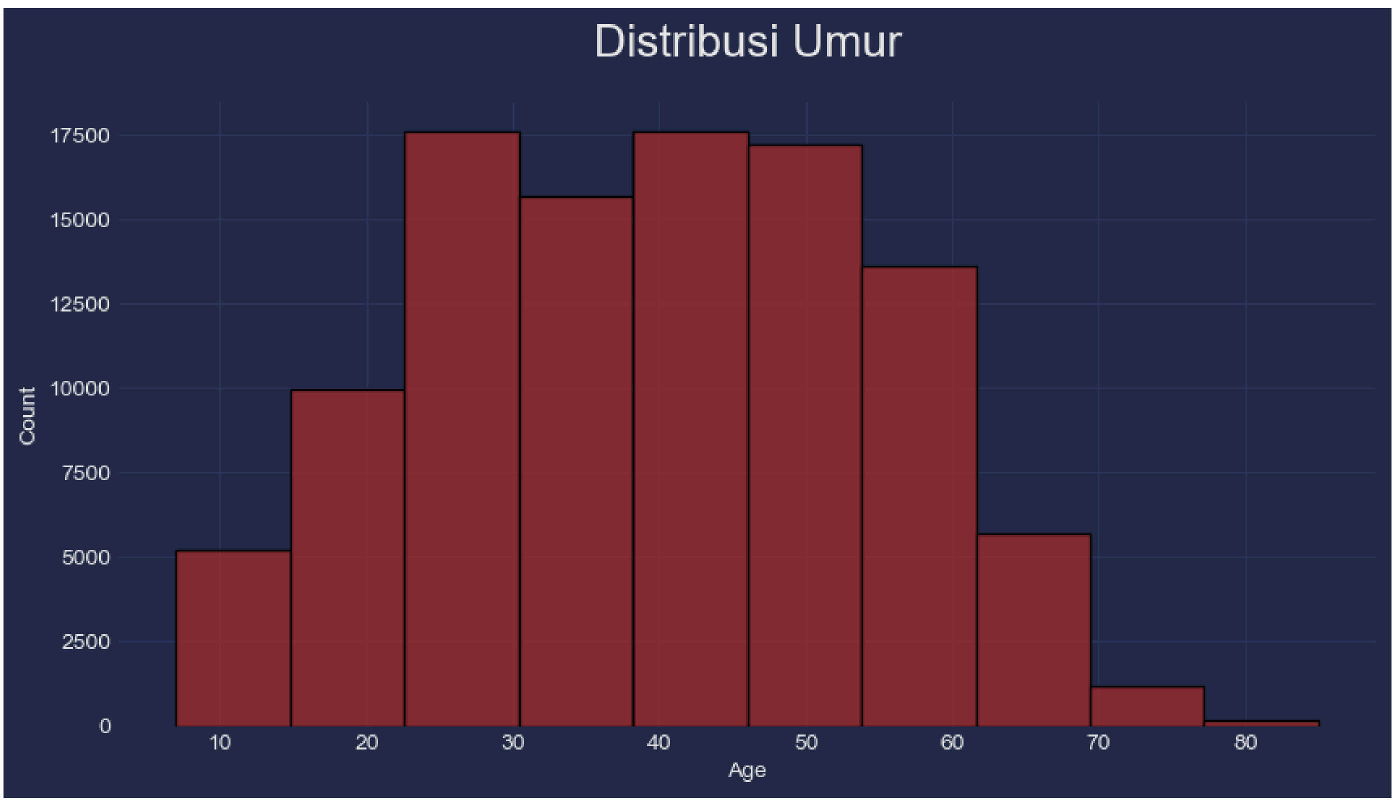


In [30]:
```
train.groupby('satisfaction')['Customer Type'].value_counts().plot.pie(autopct='%1.2f%%'
plt.title("Satisfaction by Customer Type")
plt.show()
```



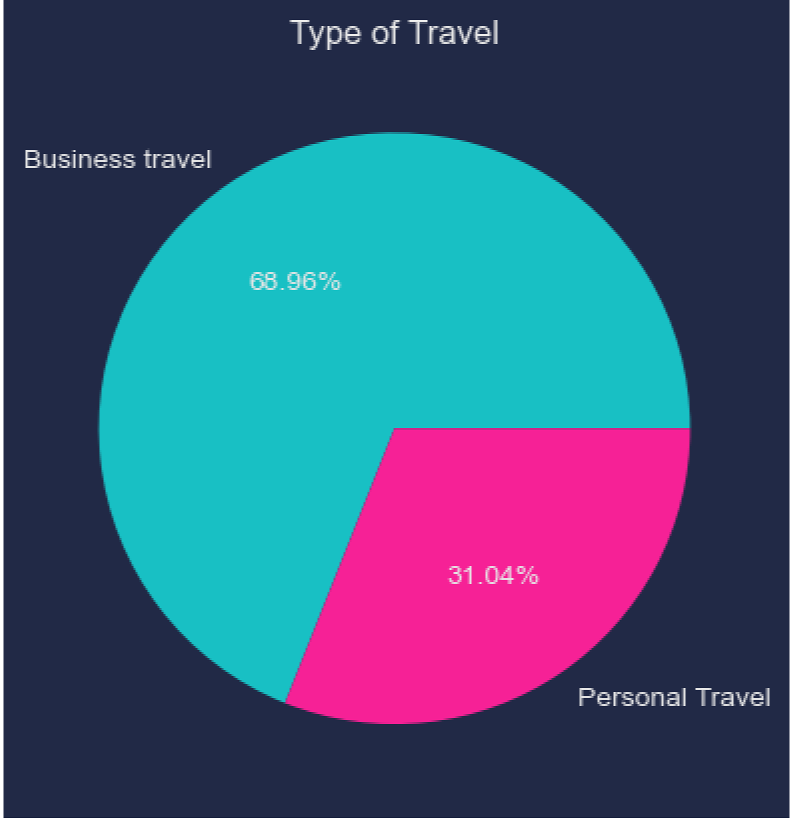
In [31]:

```
plt.figure(figsize = (10,5))
sns.histplot(train['Age'], bins = 10, color="brown")
plt.title("Distribusi Umur", fontsize = 20, pad = 20)
plt.xlabel("Age")
plt.ylabel("Count")
plt.show()
```

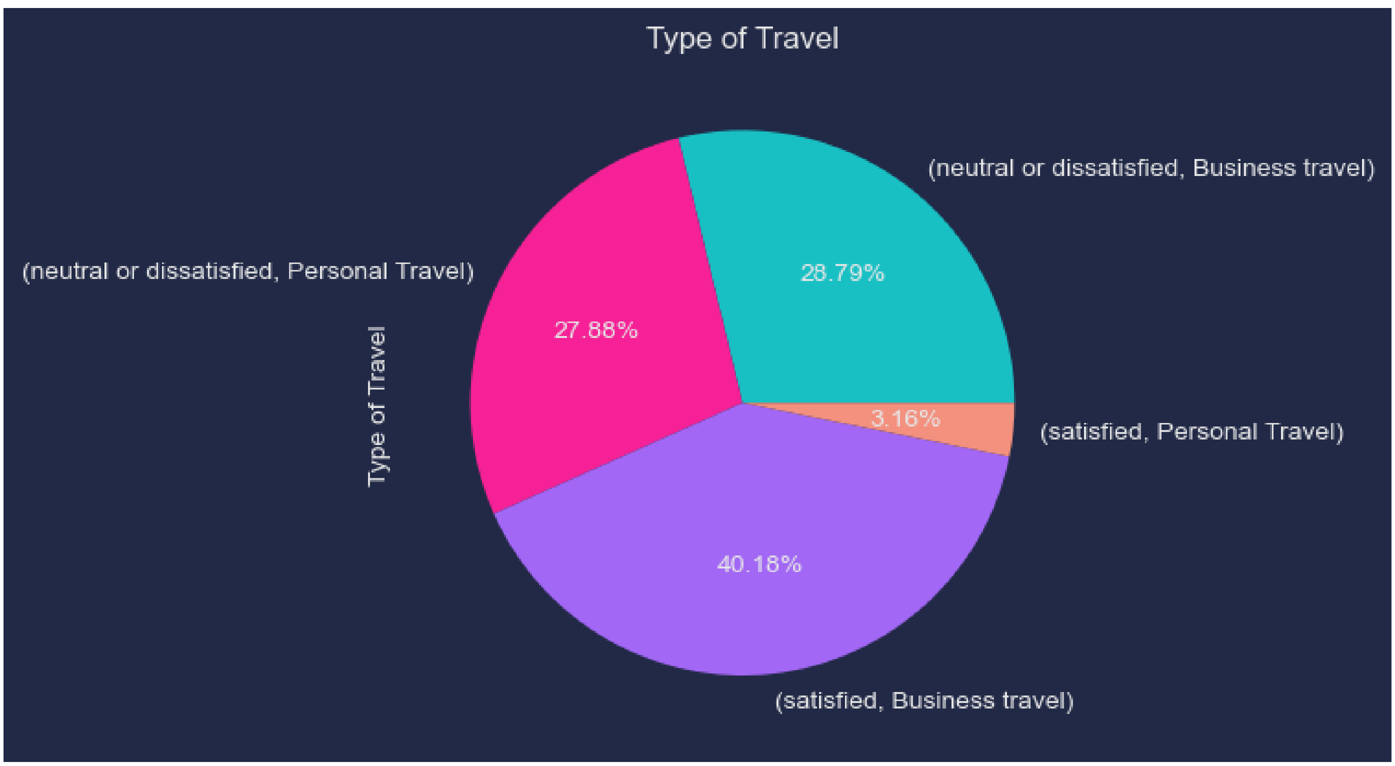


In [32]:

```
train.groupby('Type of Travel').size().plot.pie(autopct="%1.2f%%")
plt.title("Type of Travel")
plt.show()
```

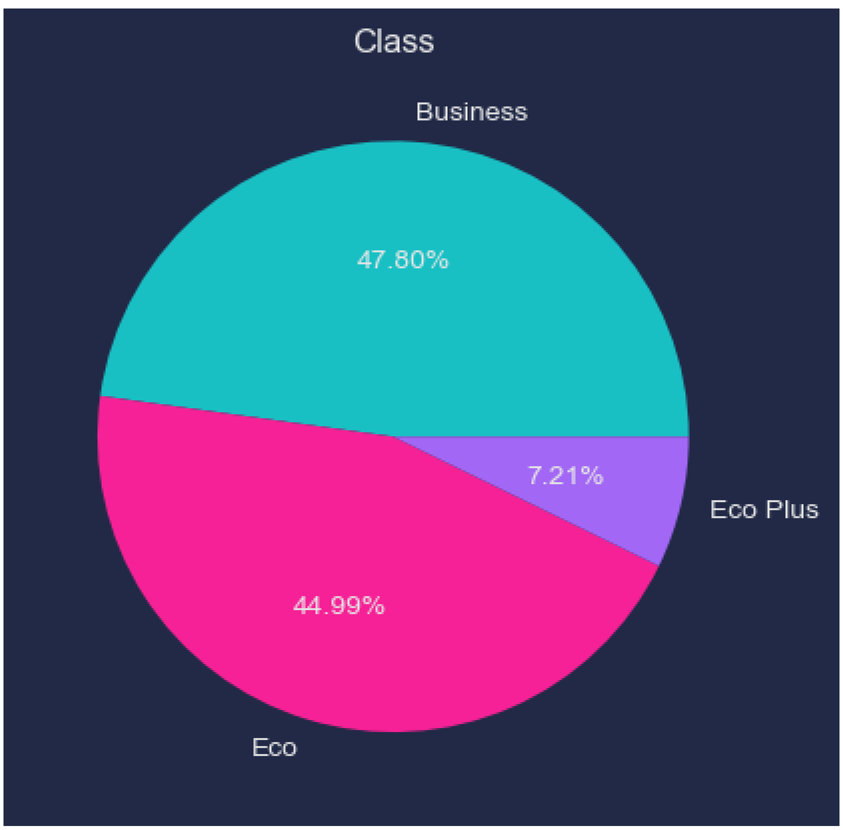


In [33]:
```
train.groupby('satisfaction')['Type of Travel'].value_counts().plot.pie(autopct="%1.2f%%
plt.title("Type of Travel")
plt.show()
```

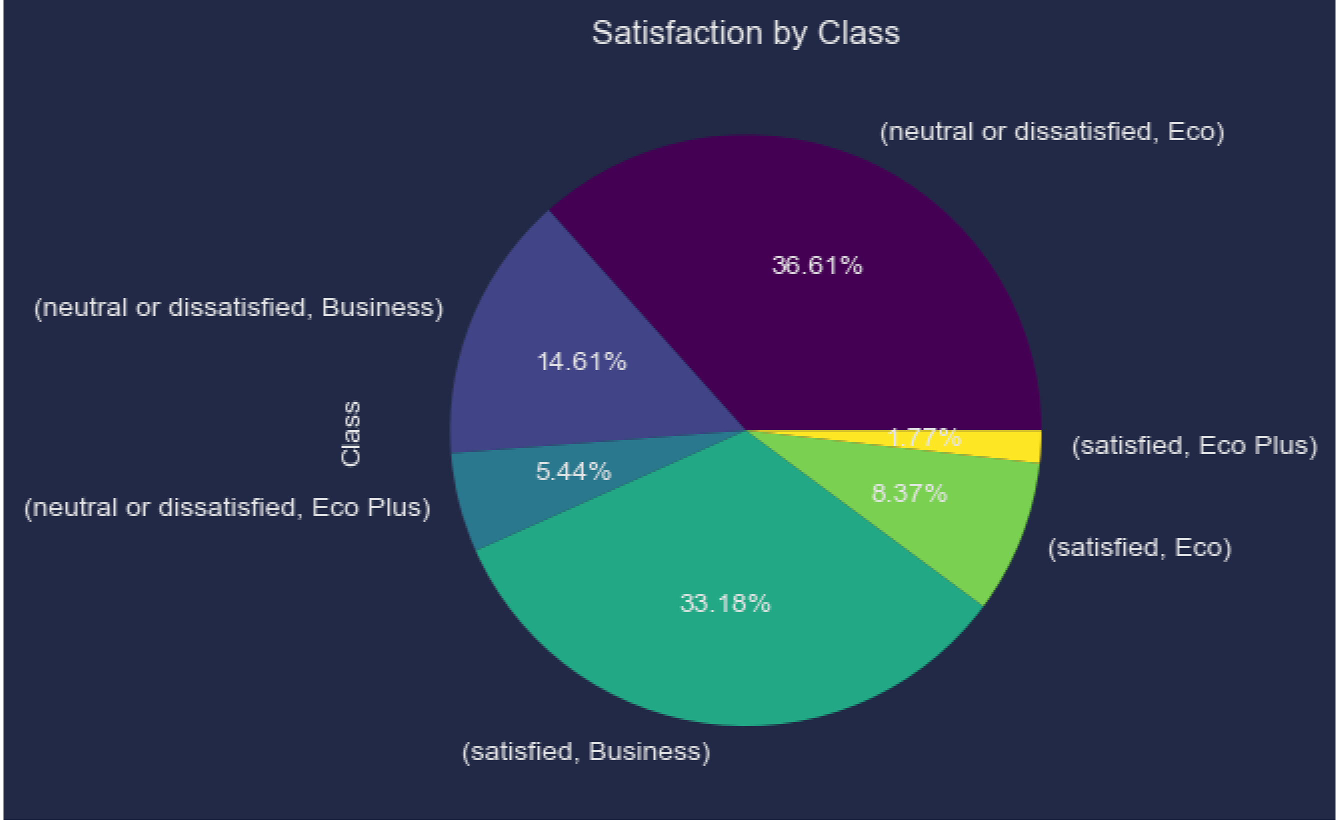


In [34]:
```
plt.figure(figsize = (7,4))
sns.histplot(data = train, x = 'Type of Travel', hue ='satisfaction', multiple = "dodge"
plt.title(f"Satisfaction by", pad = 20)
plt.show()
```

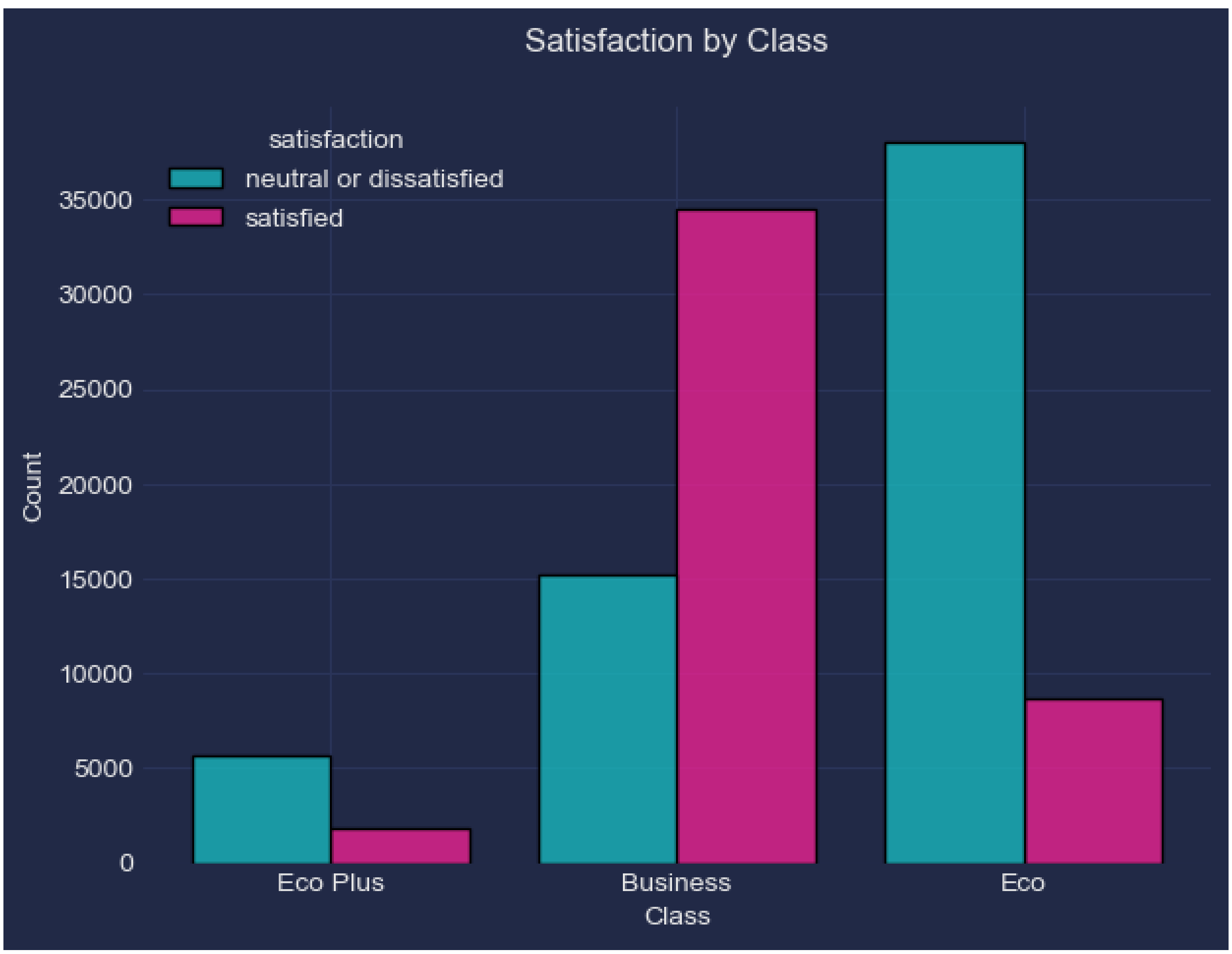
In [35]:
```
train.groupby('Class').size().plot.pie(autopct="%1.2f%%")
plt.title("Class")
plt.show()
```



In [36]:
```
train.groupby('satisfaction')['Class'].value_counts().plot.pie(autopct="%1.2f%%", cmap =
plt.title("Satisfaction by Class")
plt.show()
```



In [37]:
```
plt.figure(figsize = (7,5))
sns.histplot(data = train, x = 'Class', hue = 'satisfaction', multiple = "dodge", shrink
plt.title("Satisfaction by Class", pad = 20)
plt.show()
```



Tingkat kepuasan tertinggi pada Airline diperoleh oleh class = Business Class, namun bukan berarti jika memiliki tingkat kepuasan paling tinggi mendapat tingkat ketidakpuasan paling rendah. tingkat ketidakpuasan business class bukan menjadi yang paling rendah, diantara 3 Class, Eco plus Class yang memiliki ketidakpuasan paling rendah diantara Class yang lain.

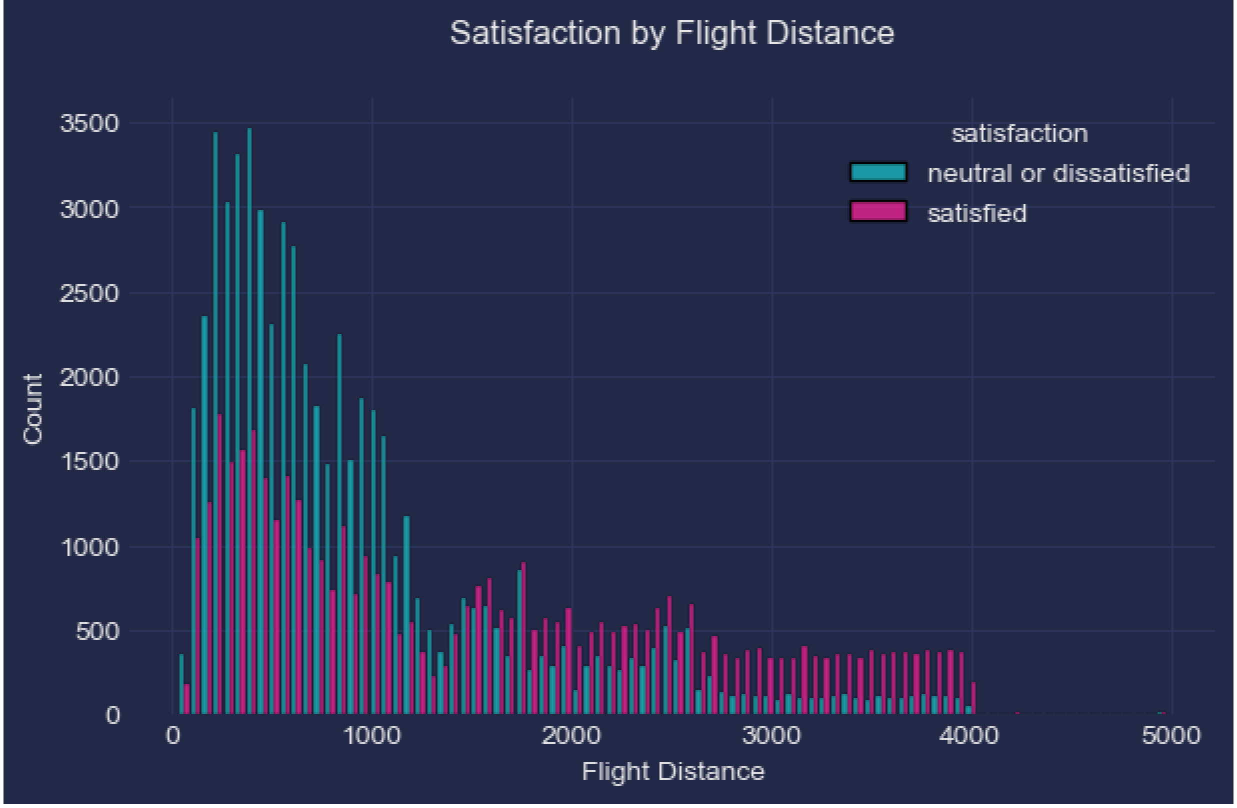
In [38]:
```
train.head()
```

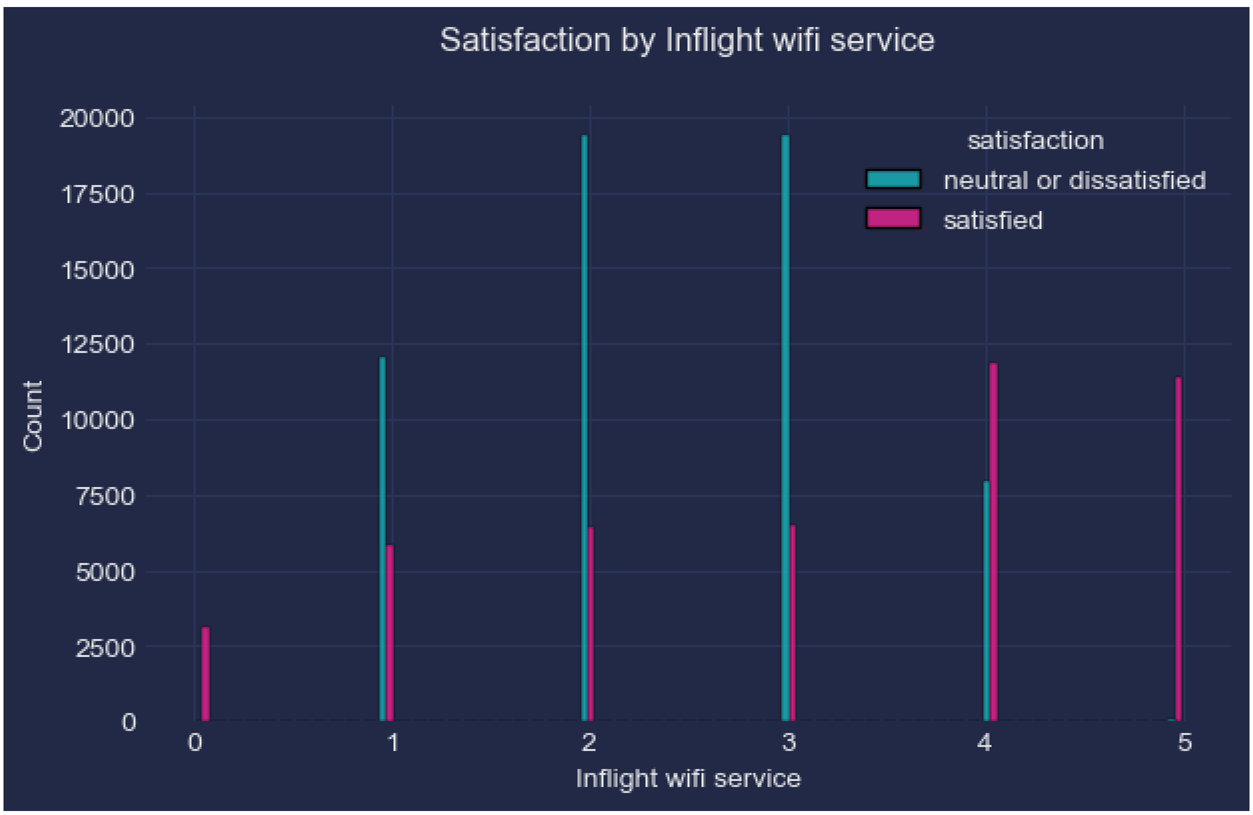
Out[38]:

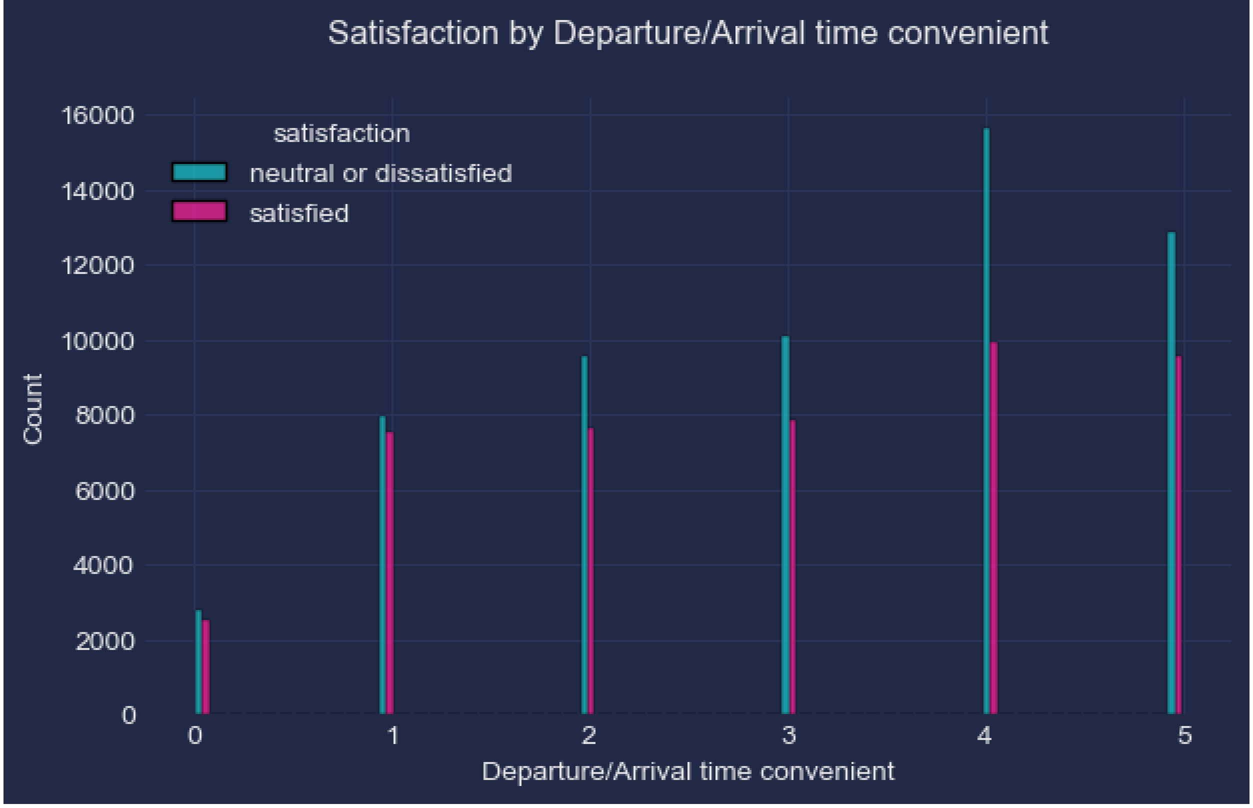
	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3
1	1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	3
2	2	110028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

In [39]:
```
columns_int = train.iloc[:, 6:20]
columns_int
stfct = train['satisfaction']
```

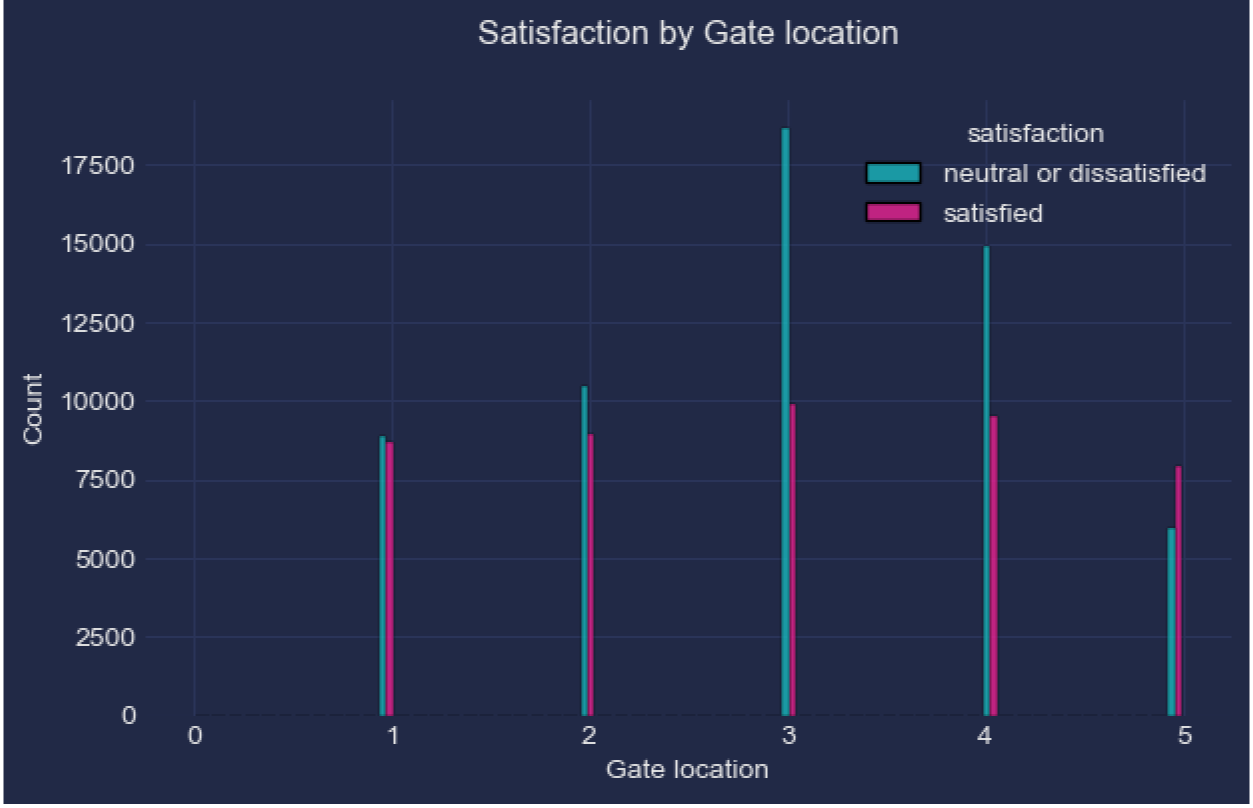
In [40]:
```
for col in columns_int:
    plt.figure(figsize = (7,4))
    sns.histplot(data = columns_int, x = col, hue = stfct, multiple = "dodge", shrink=.8
    plt.title(f"Satisfaction by {col}", pad = 20)
    plt.show()
```

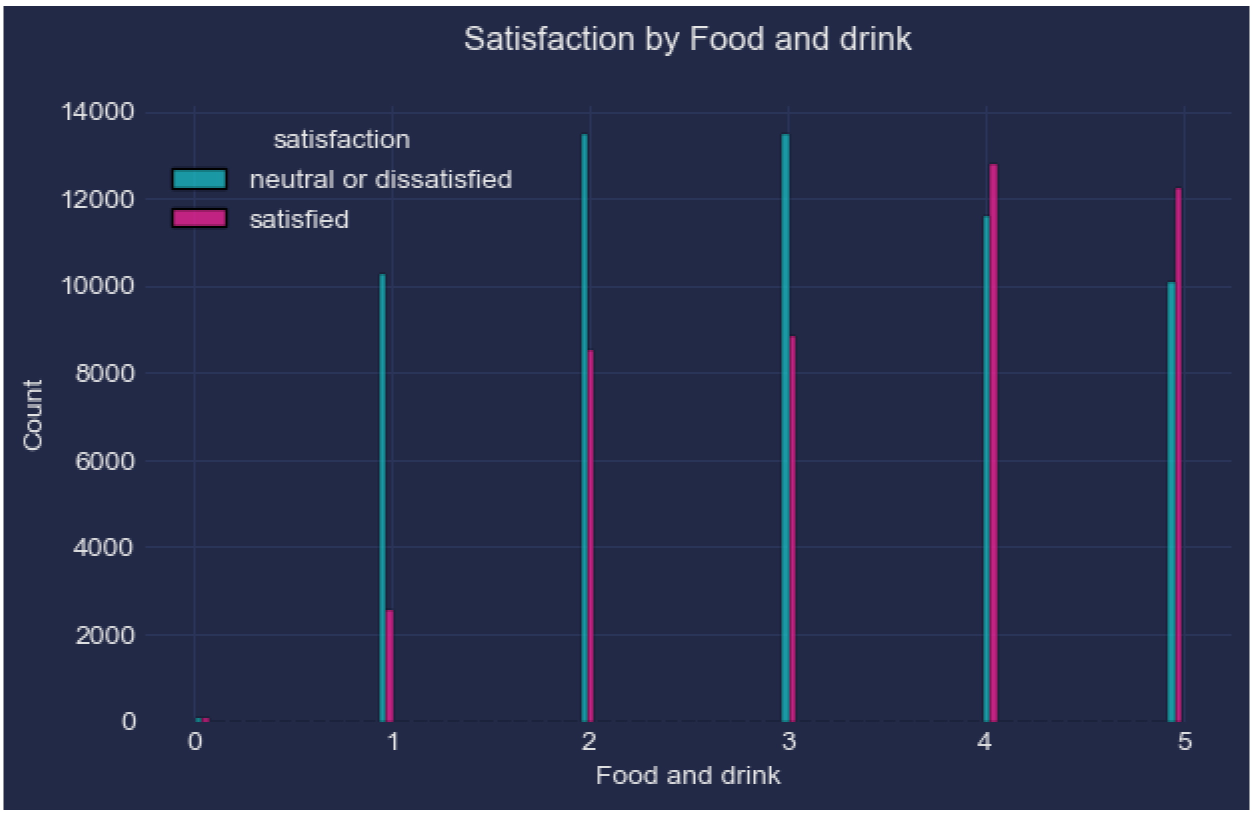

Satisfaction by Flight Distance
satisfaction
neutral or dissatisfied
satisfied
Count
Flight Distance
0
500
1000
1500
2000
2500
3000
3500
4000
5000


Satisfaction by Inflight wifi service
satisfaction
neutral or dissatisfied
satisfied
Count
Inflight wifi service
0
2500
5000
7500
10000
12500
15000
17500
20000
1
2
3
4
5

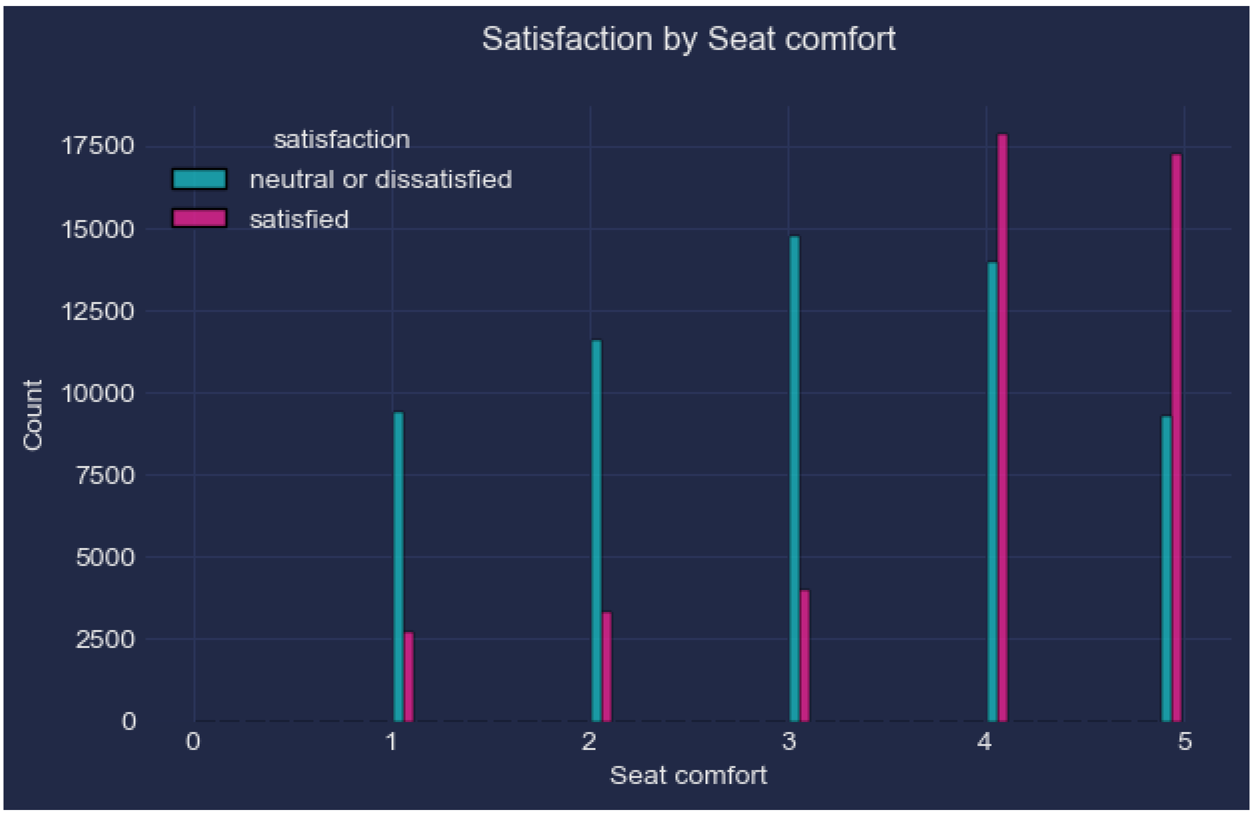

Satisfaction by Departure/Arrival time convenient
satisfaction
neutral or dissatisfied
satisfied
Count
0
2000
4000
6000
8000
10000
12000
14000
16000
0
1
2
3
4
5
Departure/Arrival time convenient

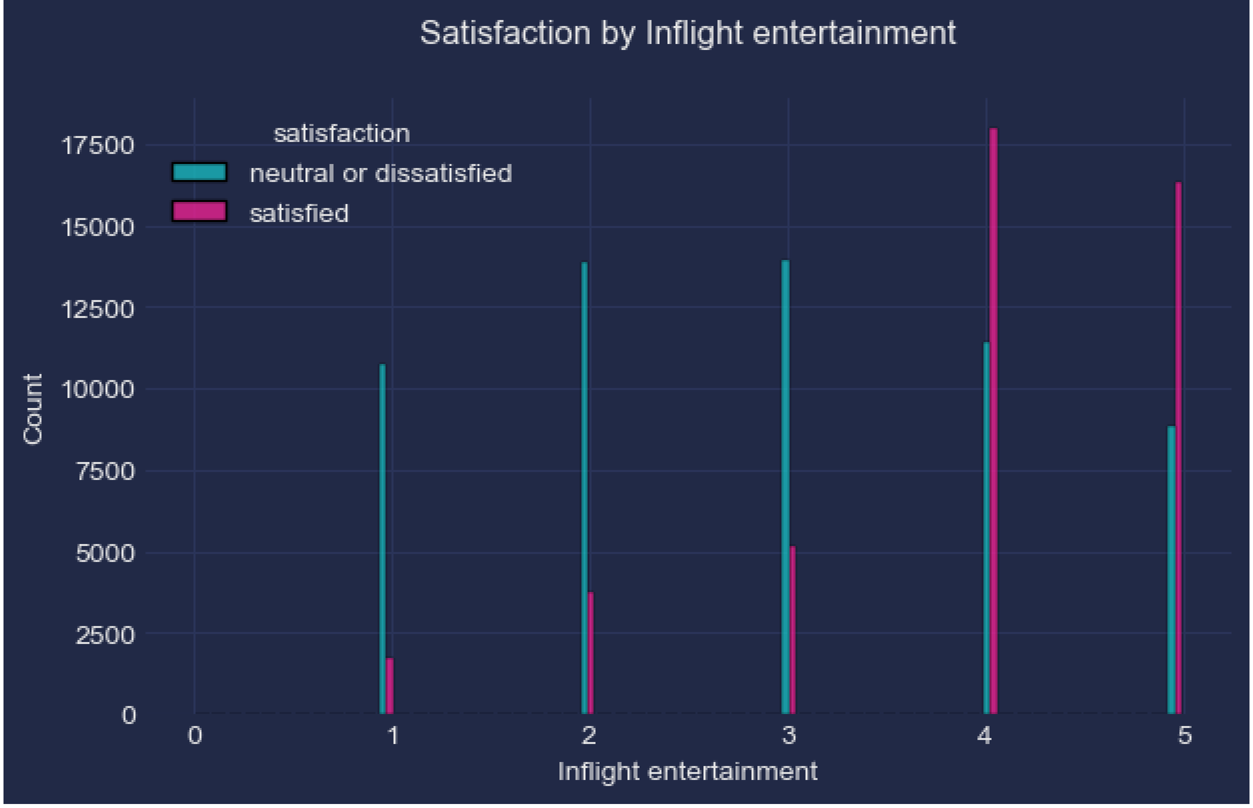

Satisfaction by Ease of Online booking
satisfaction
neutral or dissatisfied
satisfied
Count
0
2000
4000
6000
8000
10000
12000
14000
16000
0
1
2
3
4
5
Ease of Online booking

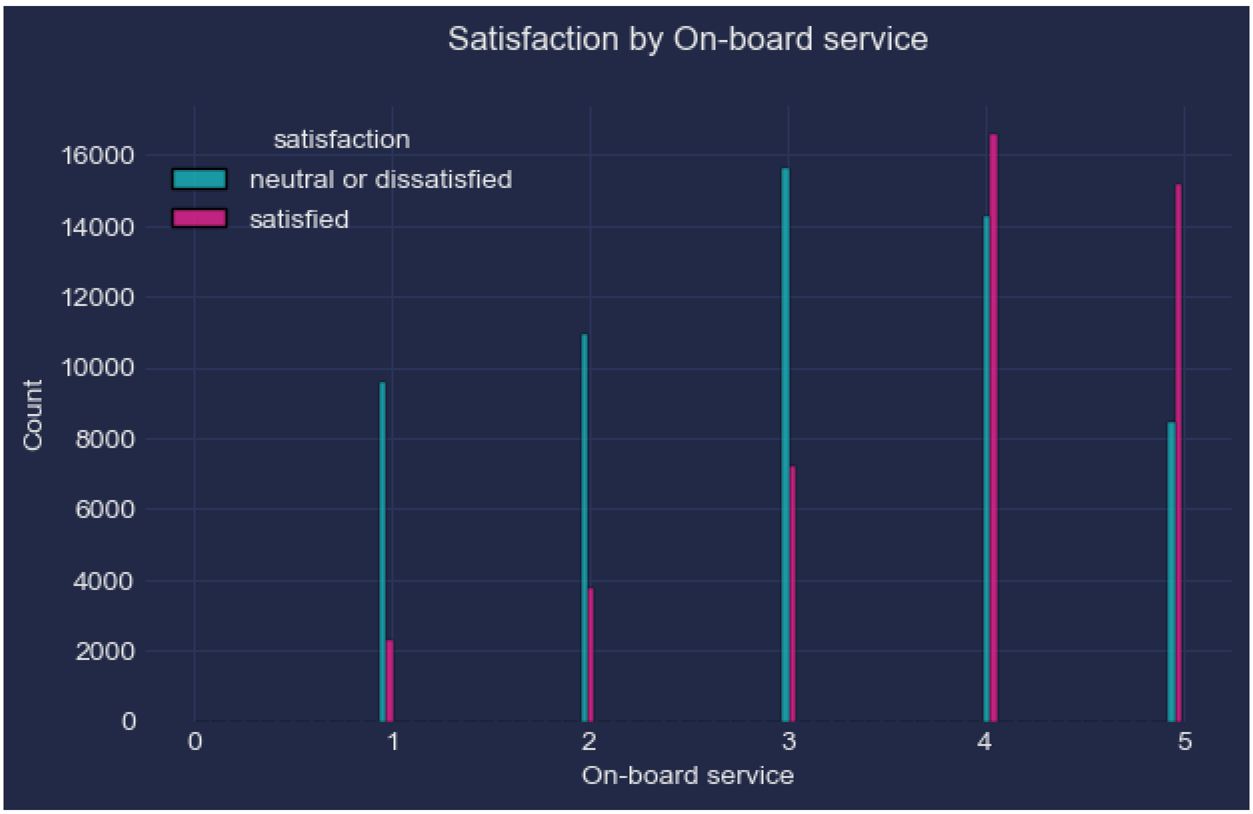

Satisfaction by Gate location
satisfaction
neutral or dissatisfied
satisfied
Count
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Gate location


Satisfaction by Food and drink
satisfaction
neutral or dissatisfied
satisfied
Count
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Food and drink

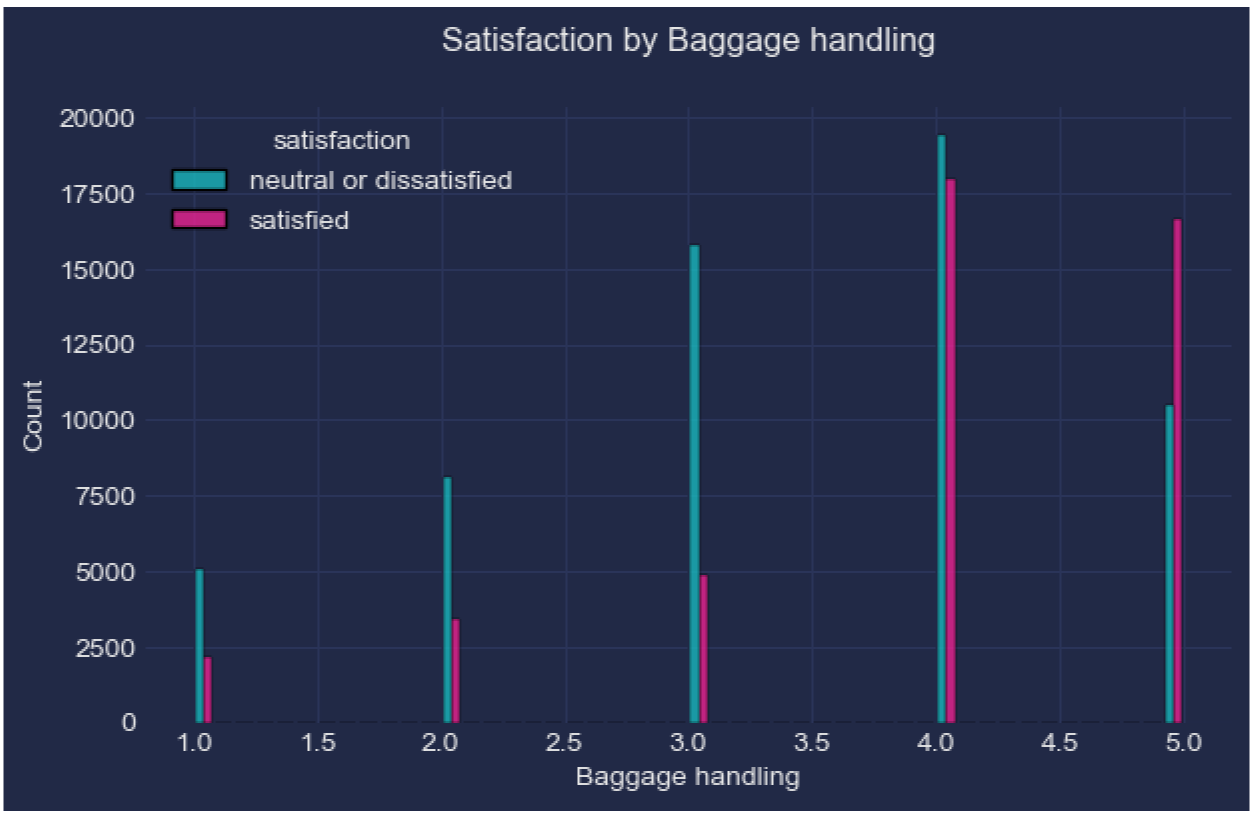

Satisfaction by Online boarding
satisfaction
neutral or dissatisfied
satisfied
Count
Online boarding
0
2500
5000
7500
10000
12500
15000
17500
20000
0
1
2
3
4
5


Satisfaction by Seat comfort
satisfaction
neutral or dissatisfied
satisfied
Count
Seat comfort
0
2500
5000
7500
10000
12500
15000
17500
0
1
2
3
4
5


Satisfaction by Inflight entertainment
satisfaction
neutral or dissatisfied
satisfied
Count
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Inflight entertainment

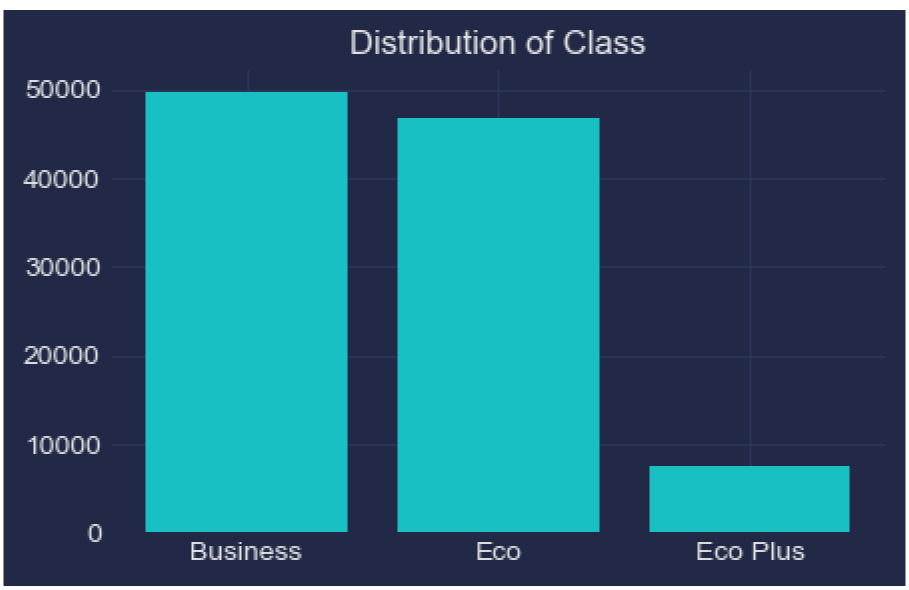

Satisfaction by On-board service
satisfaction
neutral or dissatisfied
satisfied
Count
16000
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
On-board service

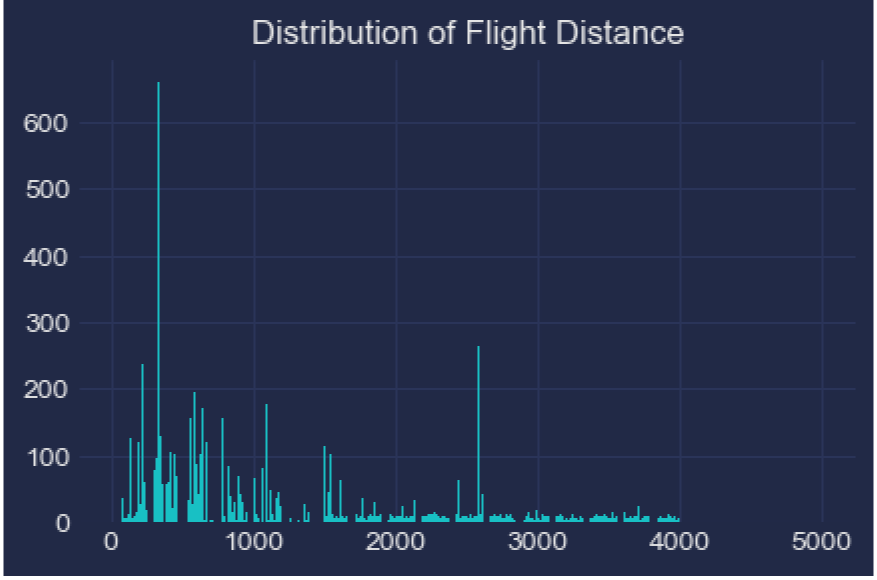
Satisfaction by Leg room service
satisfaction
neutral or dissatisfied
satisfied
Count
0
2000
4000
6000
8000
10000
12000
14000
16000
0
1
2
3
4
5
Leg room service

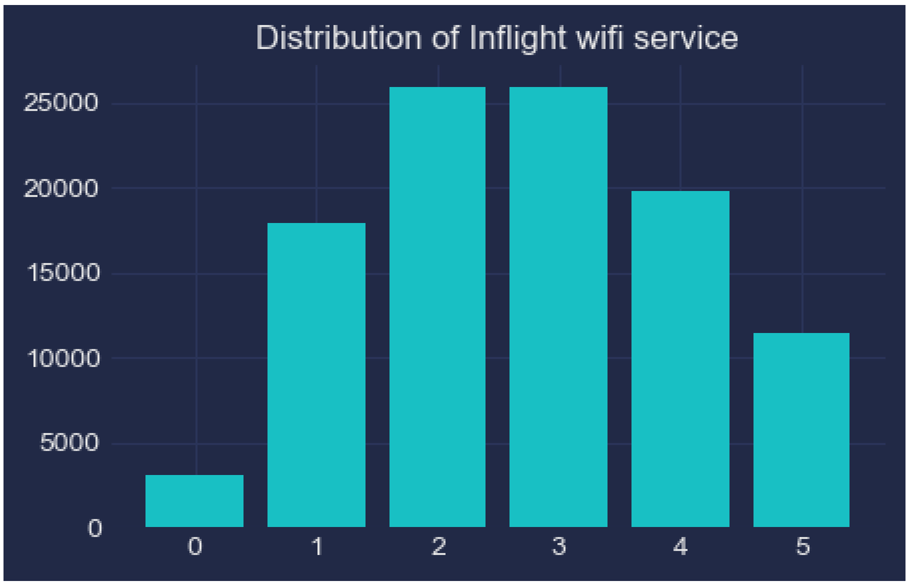


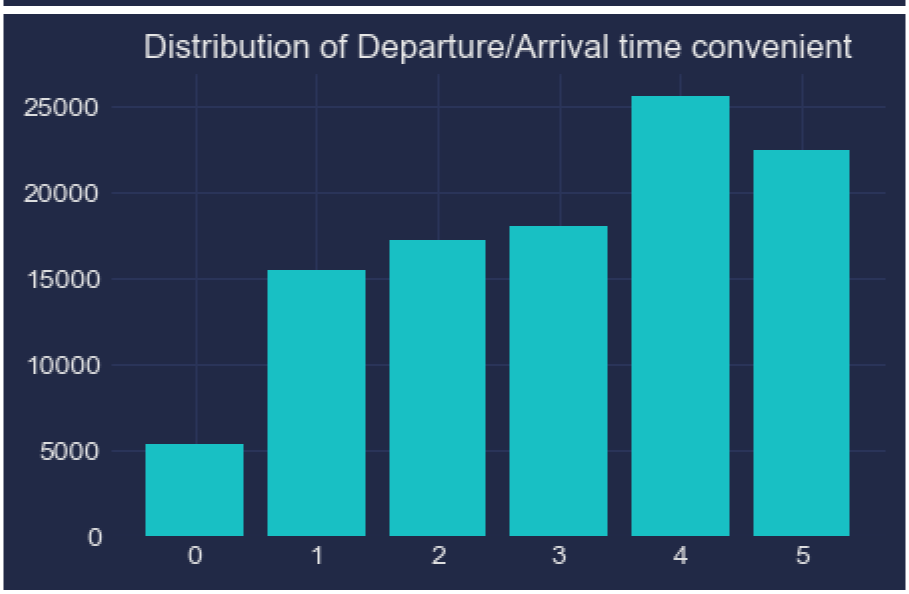
Satisfaction by Baggage handling
satisfaction
neutral or dissatisfied
satisfied
Count
0
2500
5000
7500
10000
12500
15000
17500
20000
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Baggage handling

In [41]:
```
for col in columns_int:
    plt.figure(figsize = (5,3))
    valuecounts = columns_int[col].value_counts()
    plt.bar(x = valuecounts.index,
           height = valuecounts.values)
    plt.title(f"Distribution of {col}")
    plt.show()
```

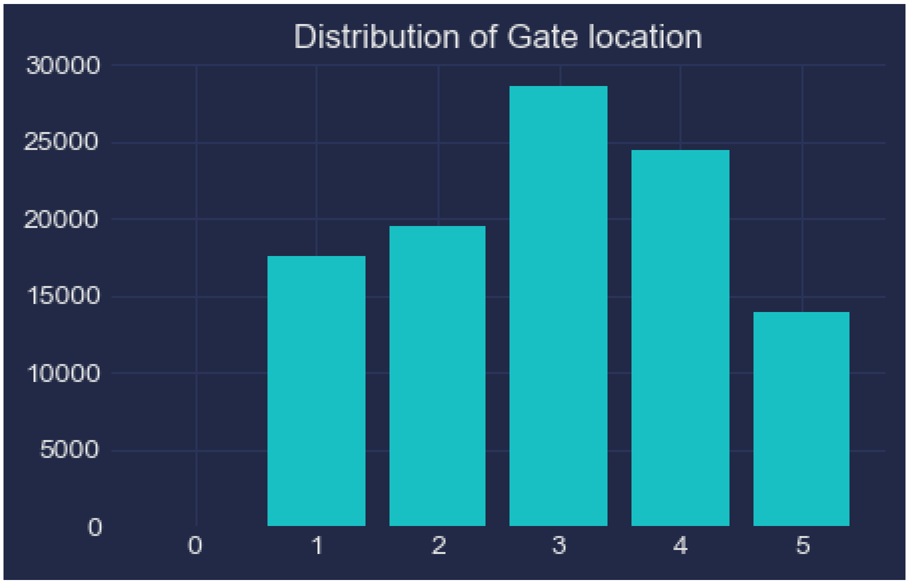


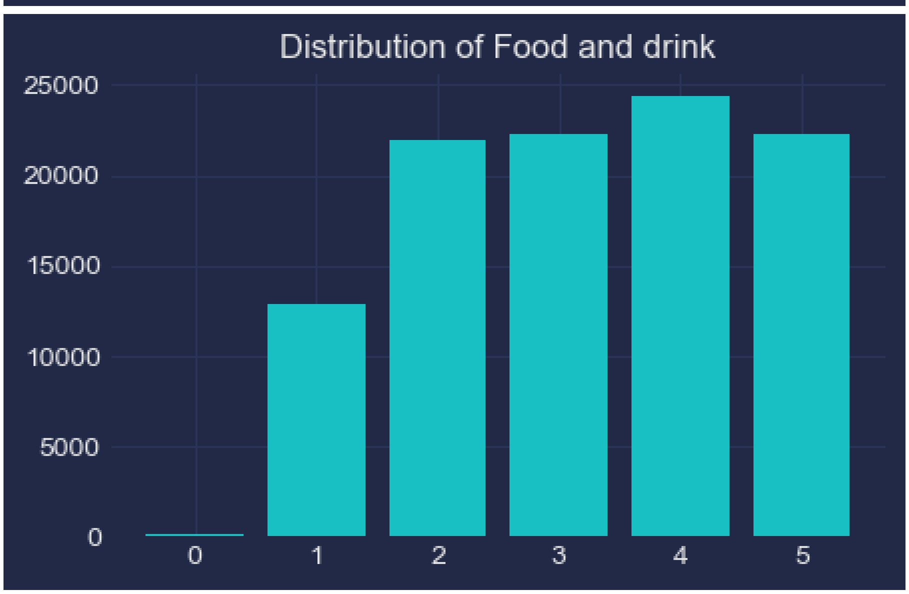

Distribution of Flight Distance
600
500
400
300
200
100
0
0
1000
2000
3000
4000
5000

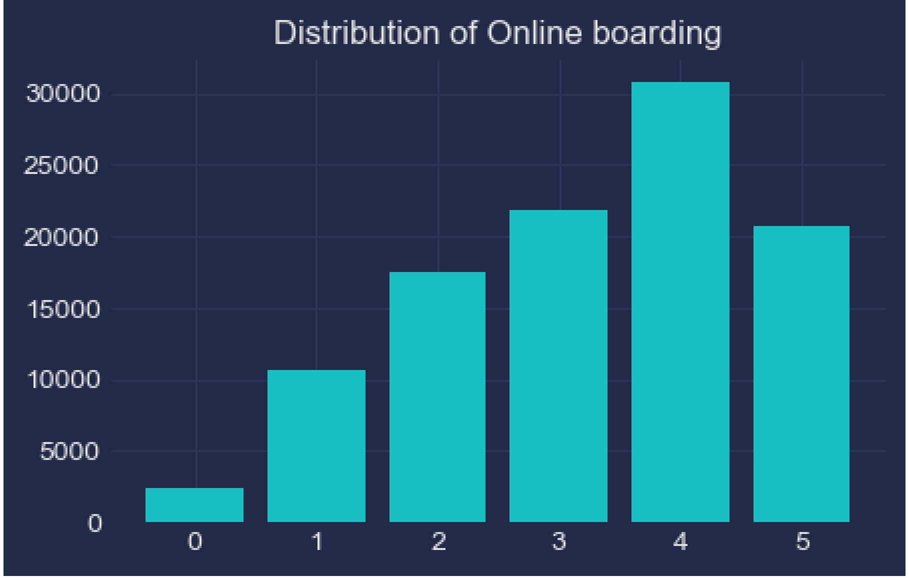
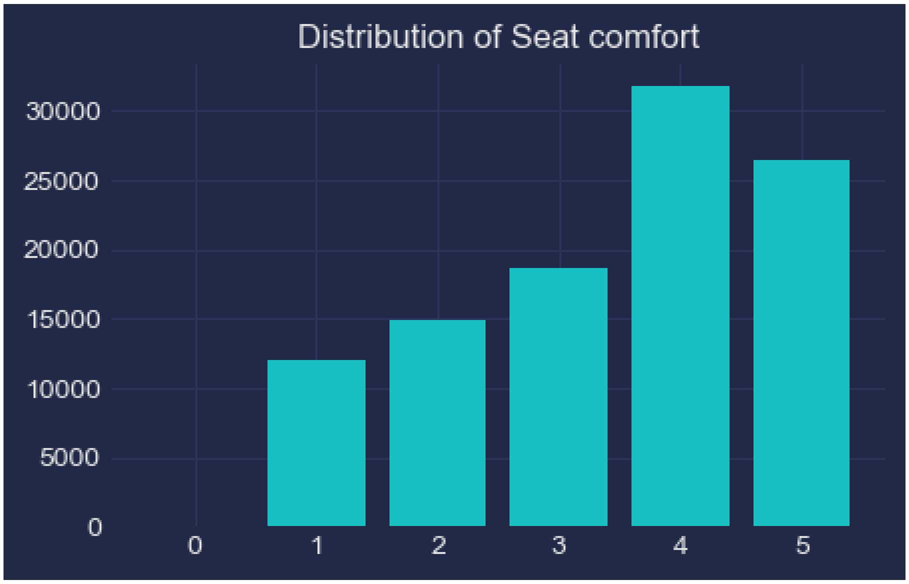
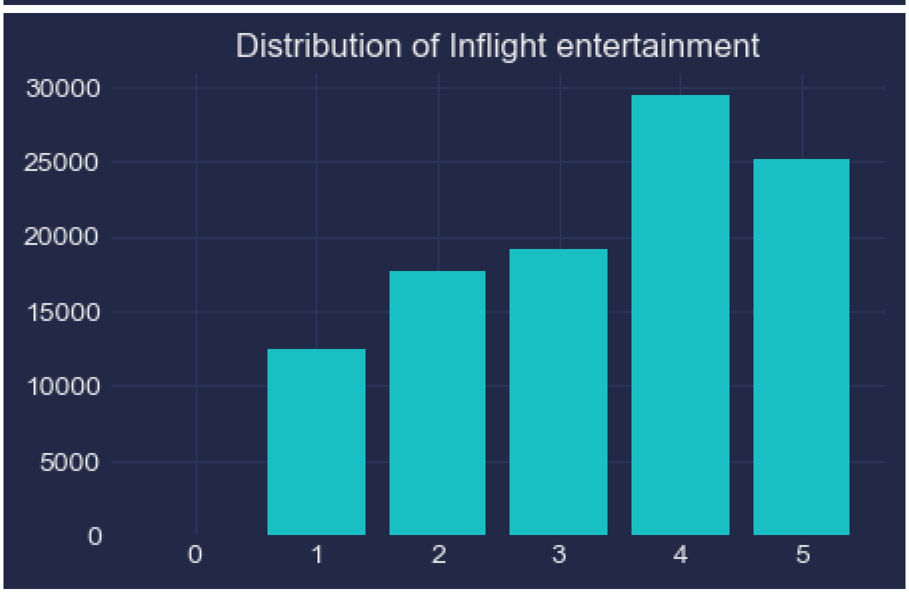

Distribution of Inflight wifi service
25000
20000
15000
10000
5000
0
0
1
2
3
4
5

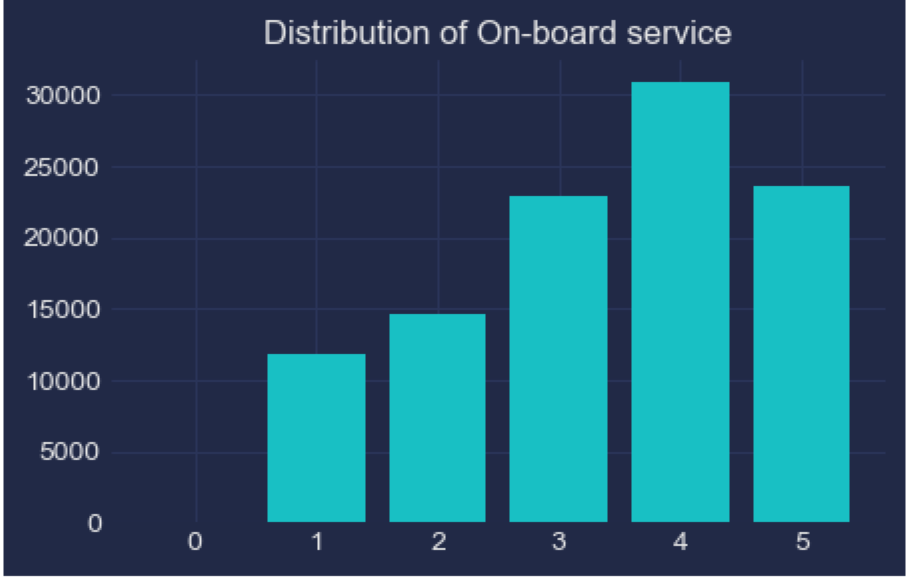

Distribution of Departure/Arrival time convenient
25000
20000
15000
10000
5000
0
0
1
2
3
4
5

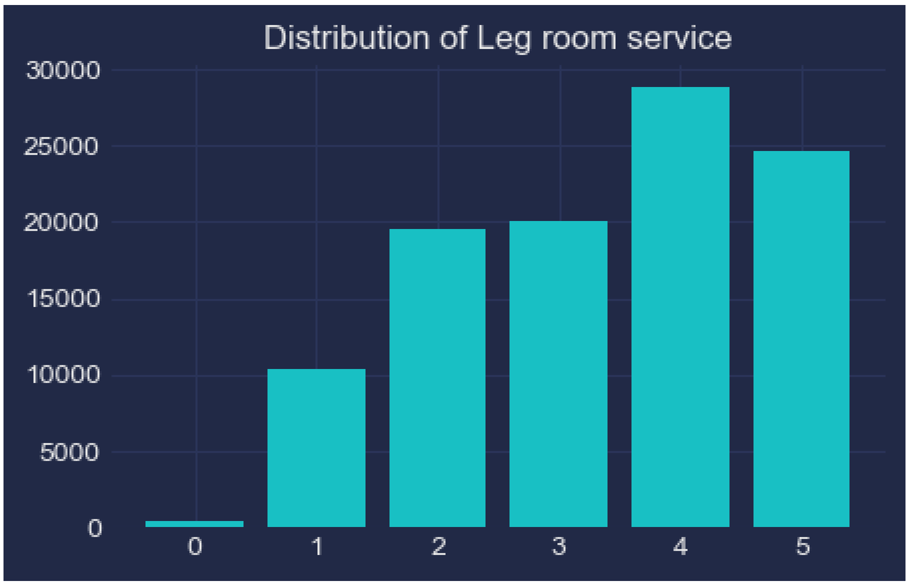

Distribution of Ease of Online booking
25000
20000
15000
10000
5000
0
0
1
2
3
4
5

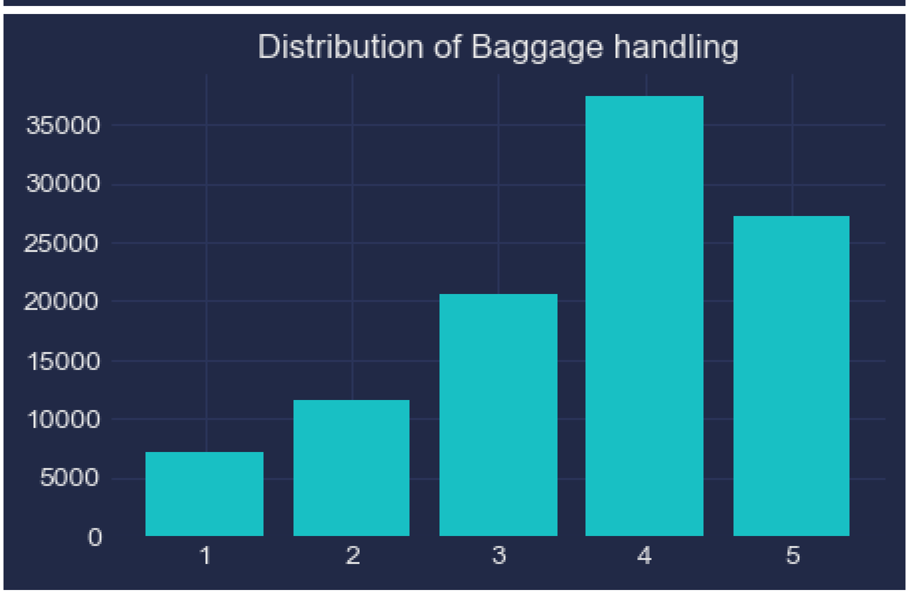

Distribution of Gate location
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5

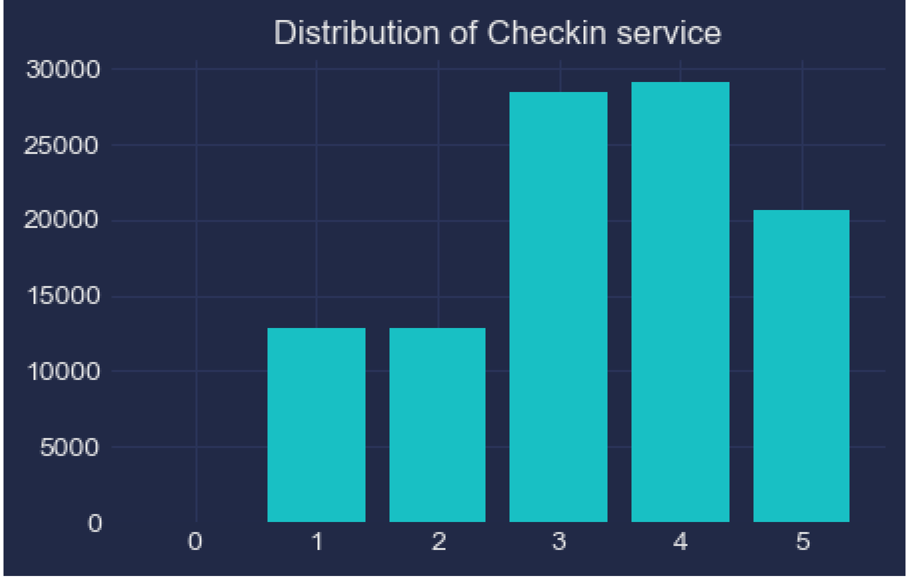

Distribution of Food and drink
25000
20000
15000
10000
5000
0
0
1
2
3
4
5


Distribution of Online boarding
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5

Distribution of Seat comfort
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5

Distribution of Inflight entertainment
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5

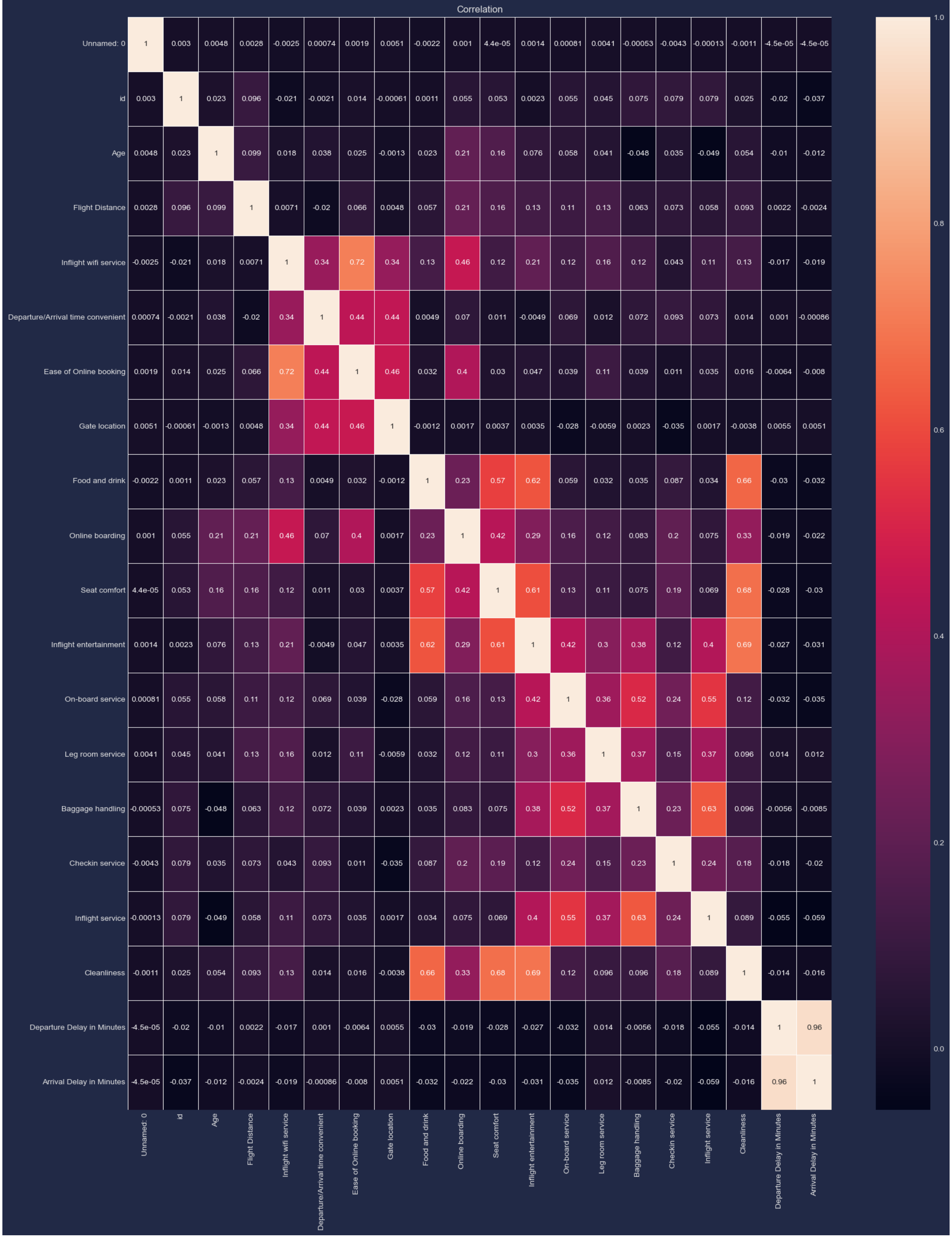

Distribution of On-board service
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5


Distribution of Leg room service
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5

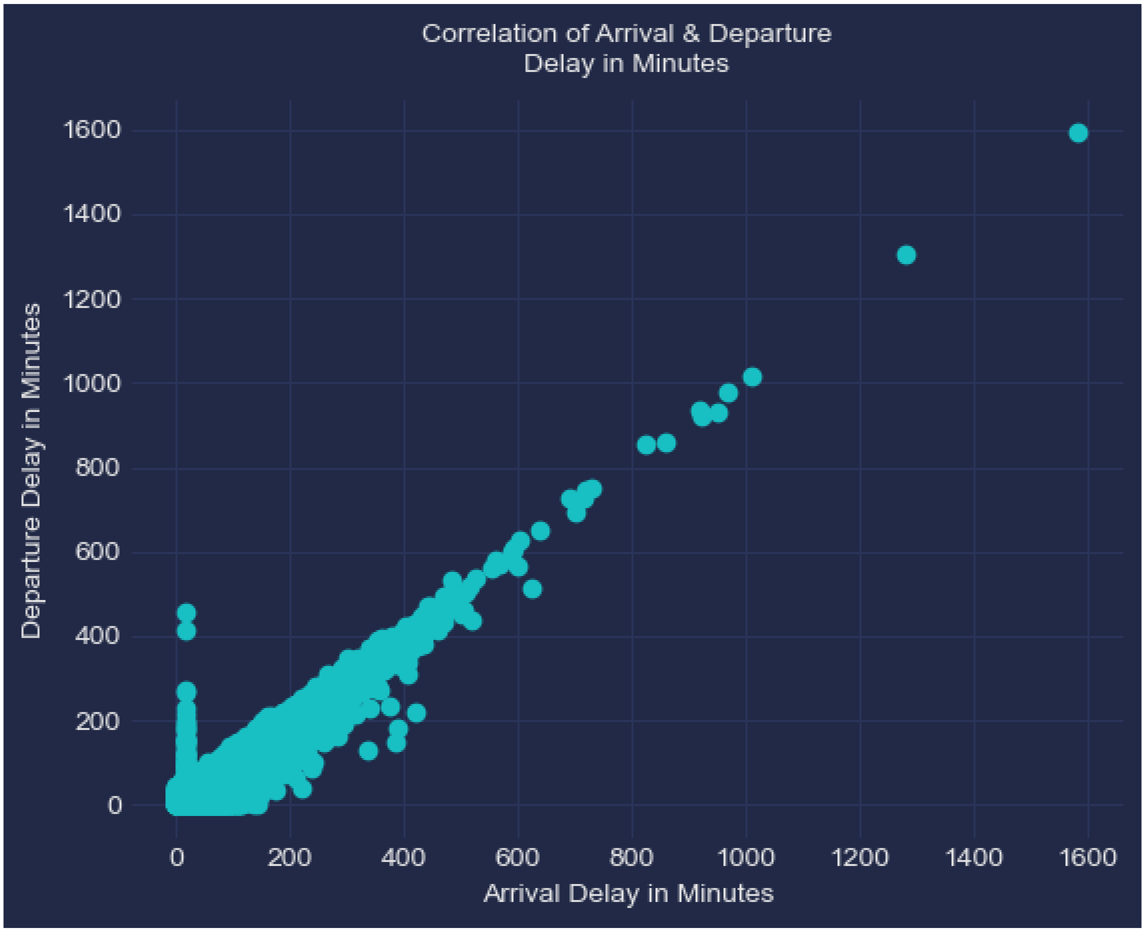

Distribution of Baggage handling
35000
30000
25000
20000
15000
10000
5000
0
1
2
3
4
5



In [42]:
```
train_corr = train.corr()
plt.figure(figsize = (20,25))
sns.heatmap(train_corr, annot = True, linewidth = 0.5)
plt.title("Correlation")
plt.show()
```



In [43]:
```python
plt.figure()
plt.scatter(data = train,
            x = 'Arrival Delay in Minutes',
            y = 'Departure Delay in Minutes')
plt.title('Correlation of Arrival & Departure\nDelay in Minutes', fontsize = 10, pad = 1
plt.xlabel('Arrival Delay in Minutes')
plt.ylabel('Departure Delay in Minutes')
plt.show()
```



Data Preprocessing

Delete unnecessary column

In [44]:
```
alldata = [train, test]

for data in alldata:
    data.drop(["Unnamed: 0", "id"], axis = 1, inplace = True)
```

In [45]:
```
train.columns
```

Out[45]:
```
Index(['Gender', 'Customer Type', 'Age', 'Type of Travel', 'Class',
       'Flight Distance', 'Inflight wifi service',
       'Departure/Arrival time convenient', 'Ease of Online booking',
       'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort',
       'Inflight entertainment', 'On-board service', 'Leg room service',
       'Baggage handling', 'Checkin service', 'Inflight service',
       'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in Minutes',
       'satisfaction'],
      dtype='object')
```

In [46]:
```
test.columns
```

Out[46]:
```
Index(['Gender', 'Customer Type', 'Age', 'Type of Travel', 'Class',
       'Flight Distance', 'Inflight wifi service',
       'Departure/Arrival time convenient', 'Ease of Online booking',
       'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort',
       'Inflight entertainment', 'On-board service', 'Leg room service',
       'Baggage handling', 'Checkin service', 'Inflight service',
       'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in Minutes',
       'satisfaction'],
      dtype='object')
```

In [47]:
```
from sklearn.preprocessing import LabelEncoder

data_categorical = ['Gender', 'Customer Type','Type of Travel', 'Class','satisfaction']

LE = LabelEncoder()

train['Gender'] = LE.fit_transform(train['Gender'])
print("Gender")
print(LE.classes_)
print(np.sort(train['Gender'].unique()))
print('')

train['Customer Type'] = LE.fit_transform(train['Customer Type'])
print("Customer Type")
print(LE.classes_)
print(np.sort(train['Customer Type'].unique()))
print('')

train['Type of Travel'] = LE.fit_transform(train['Type of Travel'])
print("Type of Travel")
print(LE.classes_)
print(np.sort(train['Type of Travel'].unique()))
print('')

train['Class'] = LE.fit_transform(train['Class'])
print("Class")
print(LE.classes_)
print(np.sort(train['Class'].unique()))
print('')

train['satisfaction'] = LE.fit_transform(train['satisfaction'])
print("Satisfaction")
print(LE.classes_)
print(np.sort(train['satisfaction'].unique()))
print('')
```

```
Gender
['Female' 'Male']
[0 1]

Customer Type
['Loyal Customer' 'disloyal Customer']
[0 1]

Type of Travel
['Business travel' 'Personal Travel']
[0 1]

Class
['Business' 'Eco' 'Eco Plus']
[0 1 2]

Satisfaction
['neutral or dissatisfied' 'satisfied']
[0 1]
```

In [48]:
```
test['Gender'] = LE.fit_transform(test['Gender'])
print("Gender")
print(LE.classes_)
print(np.sort(test['Gender'].unique()))
print('')

test['Customer Type'] = LE.fit_transform(test['Customer Type'])
print("Customer Type")
print(LE.classes_)
```

```
print(np.sort(test['Customer Type'].unique()))
print('')

test['Type of Travel'] = LE.fit_transform(test['Type of Travel'])
print("Type of Travel")
print(LE.classes_)
print(np.sort(test['Type of Travel'].unique()))
print('')

test['Class'] = LE.fit_transform(test['Class'])
print("Class")
print(LE.classes_)
print(np.sort(test['Class'].unique()))
print('')

test['satisfaction'] = LE.fit_transform(test['satisfaction'])
print("Satisfaction")
print(LE.classes_)
print(np.sort(test['satisfaction'].unique()))
print('')
```

```
Gender
['Female' 'Male']
[0 1]

Customer Type
['Loyal Customer' 'disloyal Customer']
[0 1]

Type of Travel
['Business travel' 'Personal Travel']
[0 1]

Class
['Business' 'Eco' 'Eco Plus']
[0 1 2]

Satisfaction
['neutral or dissatisfied' 'satisfied']
[0 1]
```

Untuk Label Class/Target = Satisfaction

- 0 = Neutral or dissatisfied
- 1 = Satisfied

In [49]:
```
train.head()
```

Out[49]:

	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	Onli boardir
0	1	0	13	1	2	460	3	4	3	1	5	
1	1	1	25	0	0	235	3	2	3	3	1	
2	0	0	26	0	0	1142	2	2	2	2	5	
3	0	0	25	0	0	562	2	5	5	5	2	
4	1	0	61	0	0	214	3	3	3	3	4	

In [50]:
```
# Data Splitting
x_train = train.drop(['satisfaction'], axis = 1)
y_train = train['satisfaction']

x_test = test.drop(['satisfaction'], axis = 1)
y_test = test['satisfaction']
```

Model Building

In [51]:
```
def make_results(model_name:str, model_object, metric:str):
    '''
    Arguments:
        model_name (string): what you want the model to be called in the output table
        model_object: a fit GridSearchCV object
        metric (string): precision, recall, f1, accuracy, or auc

    Returns a pandas df with the F1, recall, precision, accuracy, and auc scores
    for the model with the best mean 'metric' score across all validation folds.
    '''

    # Create dictionary that maps input metric to actual metric name in GridSearchCV
    metric_dict = {'auc': 'mean_test_roc_auc',
                   'precision': 'mean_test_precision',
                   'recall': 'mean_test_recall',
                   'f1': 'mean_test_f1',
                   'accuracy': 'mean_test_accuracy'
                  }

    # Get all the results from the CV and put them in a df
    cv_results = pd.DataFrame(model_object.cv_results_)

    # Isolate the row of the df with the max(metric) score
    best_estimator_results = cv_results.iloc[cv_results[metric_dict[metric]].idxmax(), :

    # Extract Accuracy, precision, recall, and f1 score from that row
    auc = best_estimator_results.mean_test_roc_auc
    f1 = best_estimator_results.mean_test_f1
    recall = best_estimator_results.mean_test_recall
    precision = best_estimator_results.mean_test_precision
    accuracy = best_estimator_results.mean_test_accuracy

    # Create table of results
    table = pd.DataFrame()
    table = pd.DataFrame({'model': [model_name],
                          'precision': [precision],
                          'recall': [recall],
                          'F1': [f1],
                          'accuracy': [accuracy],
                          'auc': [auc]
                         })

    return table
```

In [52]:
```
#Define a path to the folder where you want to save the model
path = '/home/ramawan/work/'
```

In [53]:
```
def write_pickle(path, model_object, save_as:str):
    '''
    In:
        path:         path of folder where you want to save the pickle
        model_object: a model you want to pickle
        save_as:      filename for how you want to save the model
```

```
    Out: A call to pickle the model in the folder indicated
    '''

    with open(path + save_as + '.pickle', 'wb') as to_write:
        pickle.dump(model_object, to_write)
```

In [54]:
```
def read_pickle(path, saved_model_name:str):
    '''
    In:
        path:             path to folder where you want to read from
        saved_model_name: filename of pickled model you want to read in

    Out:
        model: the pickled model
    '''
    with open(path + saved_model_name + '.pickle', 'rb') as to_read:
        model = pickle.load(to_read)

    return model
```

In [55]:
```
def get_scores(model_name:str, model, X_test_data, y_test_data):
    '''
    Generate a table of test scores.

    In:
        model_name (string):  How you want your model to be named in the output table
        model:                A fit GridSearchCV object
        X_test_data:          numpy array of X_test data
        y_test_data:          numpy array of y_test data

    Out: pandas df of precision, recall, f1, accuracy, and AUC scores for your model
    '''

    preds = model.best_estimator_.predict(X_test_data)

    auc = roc_auc_score(y_test_data, preds)
    accuracy = accuracy_score(y_test_data, preds)
    precision = precision_score(y_test_data, preds)
    recall = recall_score(y_test_data, preds)
    f1 = f1_score(y_test_data, preds)

    table = pd.DataFrame({'model': [model_name],
                          'precision': [precision],
                          'recall': [recall],
                          'f1': [f1],
                          'accuracy': [accuracy],
                          'AUC': [auc]
                         })

    return table
```

In [56]:
```
DT = DecisionTreeClassifier()
RF = RandomForestClassifier()
```

In [57]:
```
from sklearn.tree import plot_tree
from sklearn.model_selection import GridSearchCV
```

In [58]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, Con
from sklearn.metrics import roc_auc_score, roc_curve
```

In [59]:
```
cv_params = {'max_depth':[4, 6, 8, None],
             'min_samples_leaf': [2, 5, 1],
             'min_samples_split': [2, 4, 6]
```

```
                       }
scoring = {'accuracy', 'precision', 'recall', 'f1', 'roc_auc'}
dt2 = GridSearchCV(DT, cv_params, scoring = scoring, cv= 4, refit = 'roc_auc')
```

In [60]:
```
dt2.fit(x_train, y_train)
```

Out[60]:

```
▸ GridSearchCV
▸ estimator: DecisionTreeClassifier
  ▸ DecisionTreeClassifier
```

In [61]:
```
dt2.best_params_
```

Out[61]:
```
{'max_depth': 8, 'min_samples_leaf': 5, 'min_samples_split': 6}
```

In [62]:
```
dt2.best_score_
```

Out[62]:
```
0.9843450716614454
```

In [63]:
```
dt2_cv_result = make_results('Decision Tree', dt2, 'auc')
dt2_cv_result
```

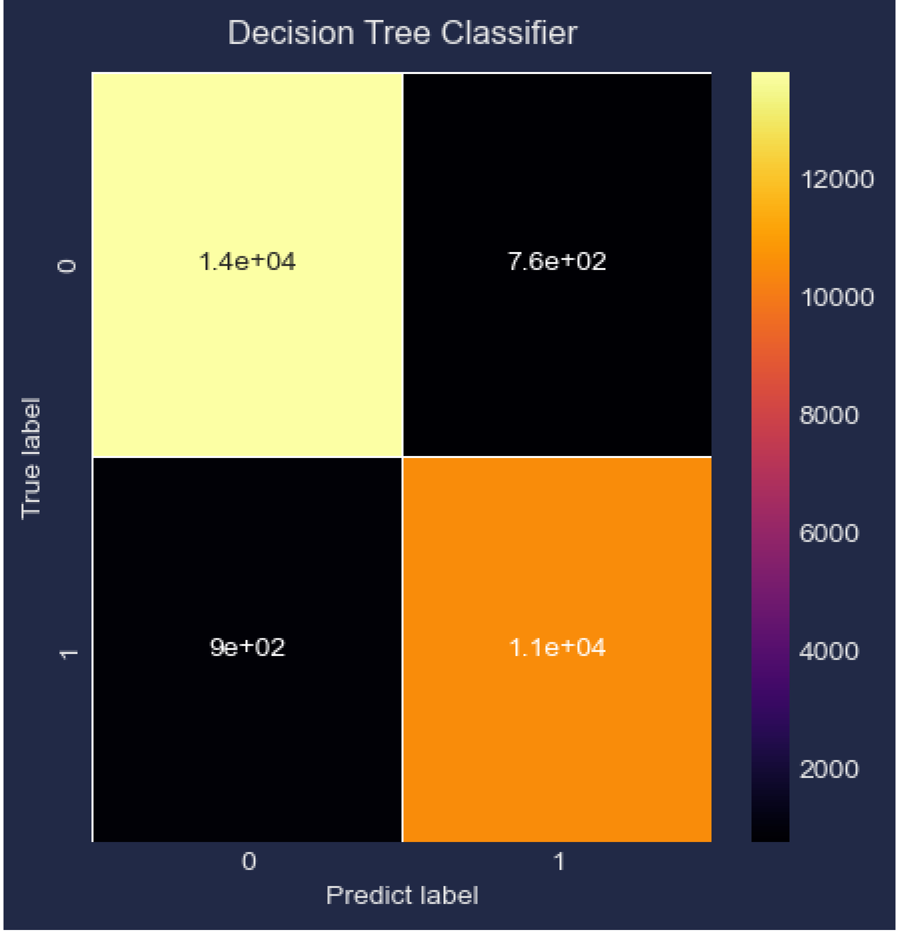
Out[63]:

	model	precision	recall	F1	accuracy	auc
0	Decision Tree	0.935744	0.916446	0.925947	0.936499	0.984345

In [64]:
```
dt2_pred = dt2.best_estimator_.predict(x_test)
dt2_cm = confusion_matrix(y_test, dt2_pred)
print(dt2_cm)
```

```
[[13811   762]
 [  900 10503]]
```

In [65]:
```
plt.figure(figsize = (5,5))
sns.heatmap(dt2_cm, annot=True, linewidth = 0.5, cmap = "inferno")
plt.title("Decision Tree Classifier", fontsize = 12, pad = 10)
plt.xlabel("Predict label")
plt.ylabel("True label")
plt.show()
```



In [66]:
```
#Random Forest
RF.fit(x_train,y_train)
```

Out[66]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

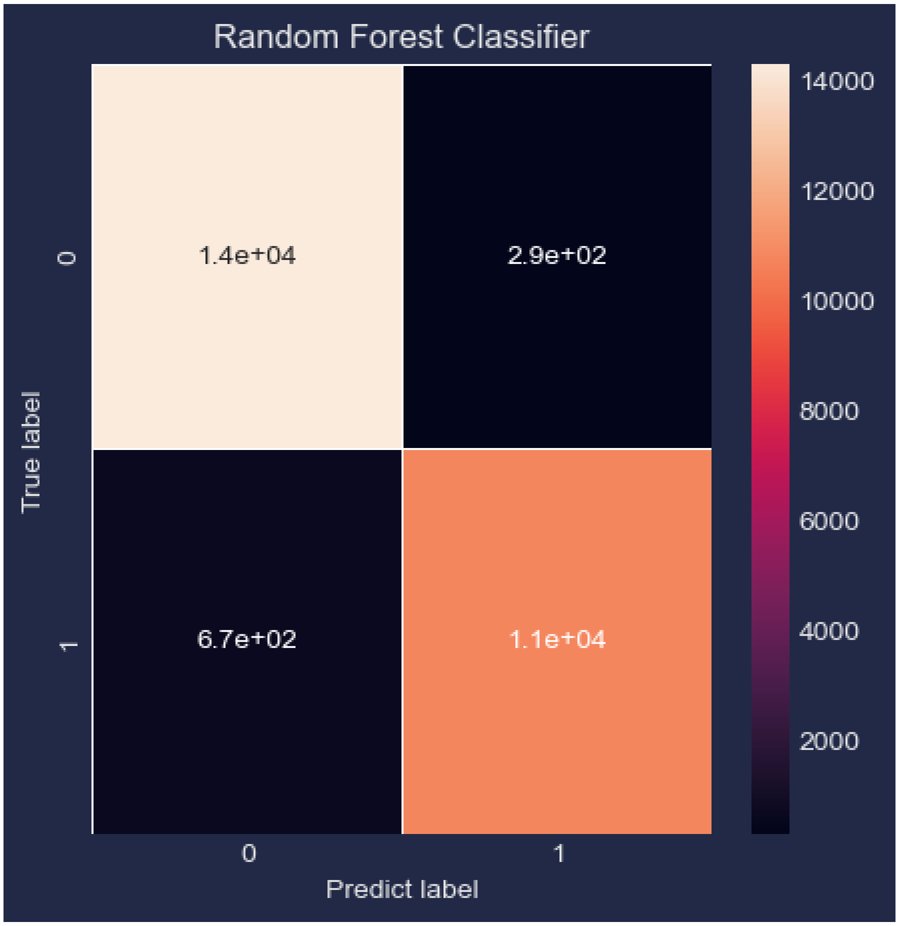
In [67]:
```
print("Accuracy score training data: ", RF.score(x_train, y_train))
print("Accuracy score testing data: ", RF.score(x_test, y_test))
print('')

y_pred = RF.predict(x_test)

rf_cr = classification_report(y_test, y_pred)
print(rf_cr)

rf_auc = roc_auc_score(y_test, y_pred)
print("Auc Score :", rf_auc)
```

```
Accuracy score training data:  0.9999807514628888
Accuracy score testing data:  0.9628888204496459

              precision    recall  f1-score   support

           0       0.96      0.98      0.97     14573
           1       0.97      0.94      0.96     11403

    accuracy                           0.96     25976
   macro avg       0.96      0.96      0.96     25976
weighted avg       0.96      0.96      0.96     25976

Auc Score : 0.9605155425678735
```

In [68]:
```
rf_cm = confusion_matrix(y_test, y_pred)
```

```
print(rf_cm)

plt.figure(figsize = (5,5))
sns.heatmap(rf_cm, annot = True, linewidth = 0.5)
plt.title("Random Forest Classifier")
plt.xlabel("Predict label")
plt.ylabel("True label")
plt.show()
```

```
[[14281   292]
 [  672 10731]]
```



In [69]:
```
xgb = XGBClassifier()
xgb_params = {'max_depth':[4, 6, 8, None],
             'learning_rate': [0.2, 0.3],
             'n_estimators': [50, 75],
              'min_child_weight' : [2,4]
             }
scoring = {'accuracy', 'precision', 'recall', 'f1', 'roc_auc'}
xgb = GridSearchCV(xgb, xgb_params, scoring = scoring, refit = 'roc_auc')
```

In [70]:
```
%%time
xgb.fit(x_train, y_train)
```

```
CPU times: total: 59min 7s
Wall time: 6min 3s
```

Out[70]:

```
▸ GridSearchCV
▸ estimator: XGBClassifier
  ▸ XGBClassifier
```

In [71]:
```
xgb.best_params_
```

Out[71]:
```
{'learning_rate': 0.2,
```

```
 'max_depth': 8,
 'min_child_weight': 4,
 'n_estimators': 75}
```

In [72]:
```
xgb.best_score_
```

Out[72]:
```
0.995157155934289
```

In [73]:
```
xgb_results = make_results('XGB Classifier', xgb, 'auc')
xgb_results
```

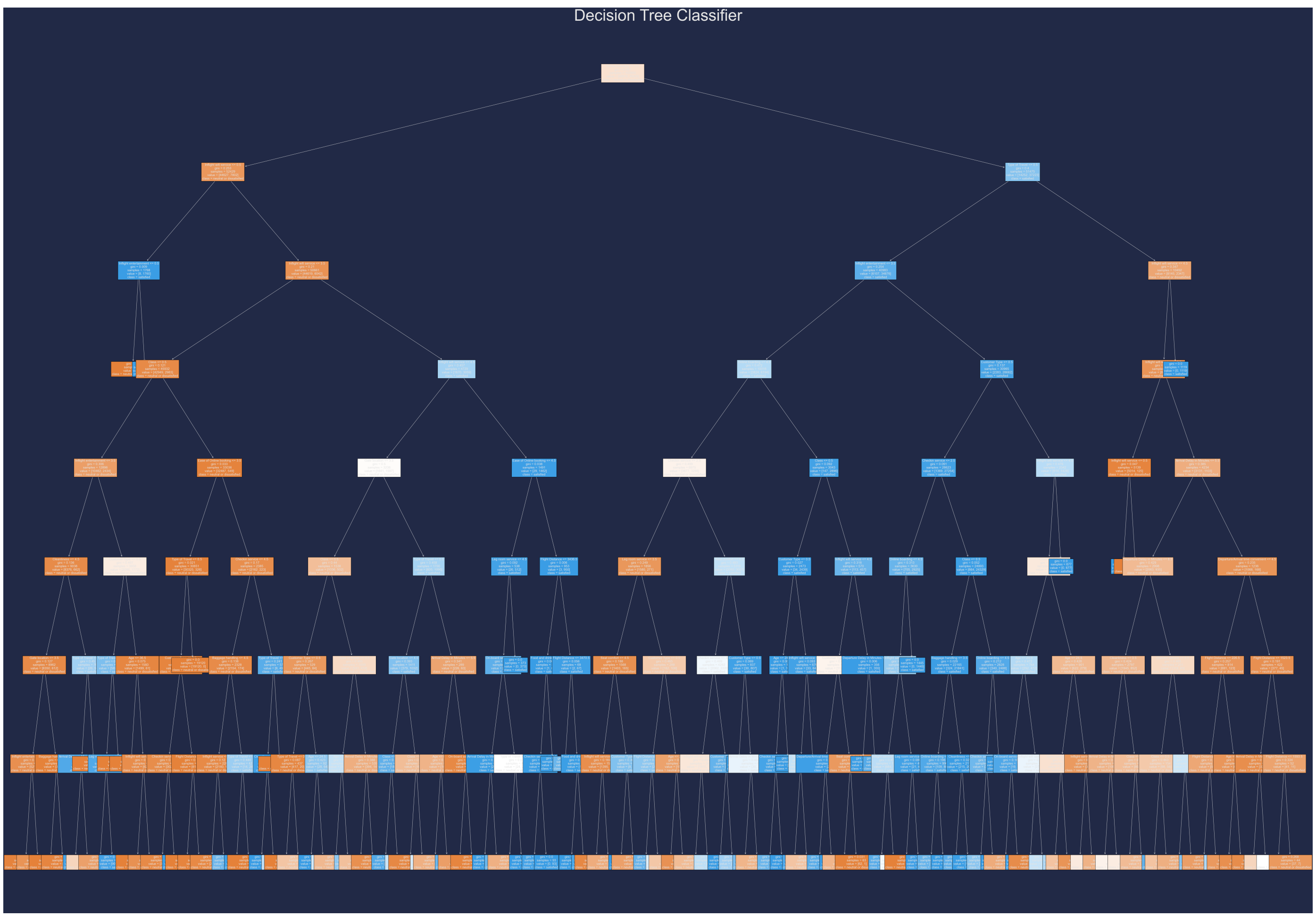
Out[73]:

	model	precision	recall	F1	accuracy	auc
0	XGB Classifier	0.97211	0.943187	0.957426	0.963649	0.995157

In [74]:
```
print("Accuracy Score Data Training :", xgb.score(x_train, y_train))
print("Accuracy Score Data Testing :", xgb.score(x_test, y_test))
```

```
Accuracy Score Data Training : 0.9977135905982246
Accuracy Score Data Testing : 0.9954021075701107
```

In [75]:
```
# Plot the tree
plt.figure(figsize=(100,70))
plot_tree(dt2.best_estimator_, max_depth=8, fontsize=14, feature_names=train.columns,
          class_names={0:'neutral or dissatisfied', 1:'satisfied'}, filled=True);
plt.title("Decision Tree Classifier", fontsize = 70)
plt.show()
```



In [76]:
```
from xgboost import plot_tree as plot_tree_xgb
```

In [77]:
```
dt2_importance = pd.DataFrame(dt2.best_estimator_.feature_importances_,
                              columns=['Feature_importances'],
                              index=x_train.columns)
```

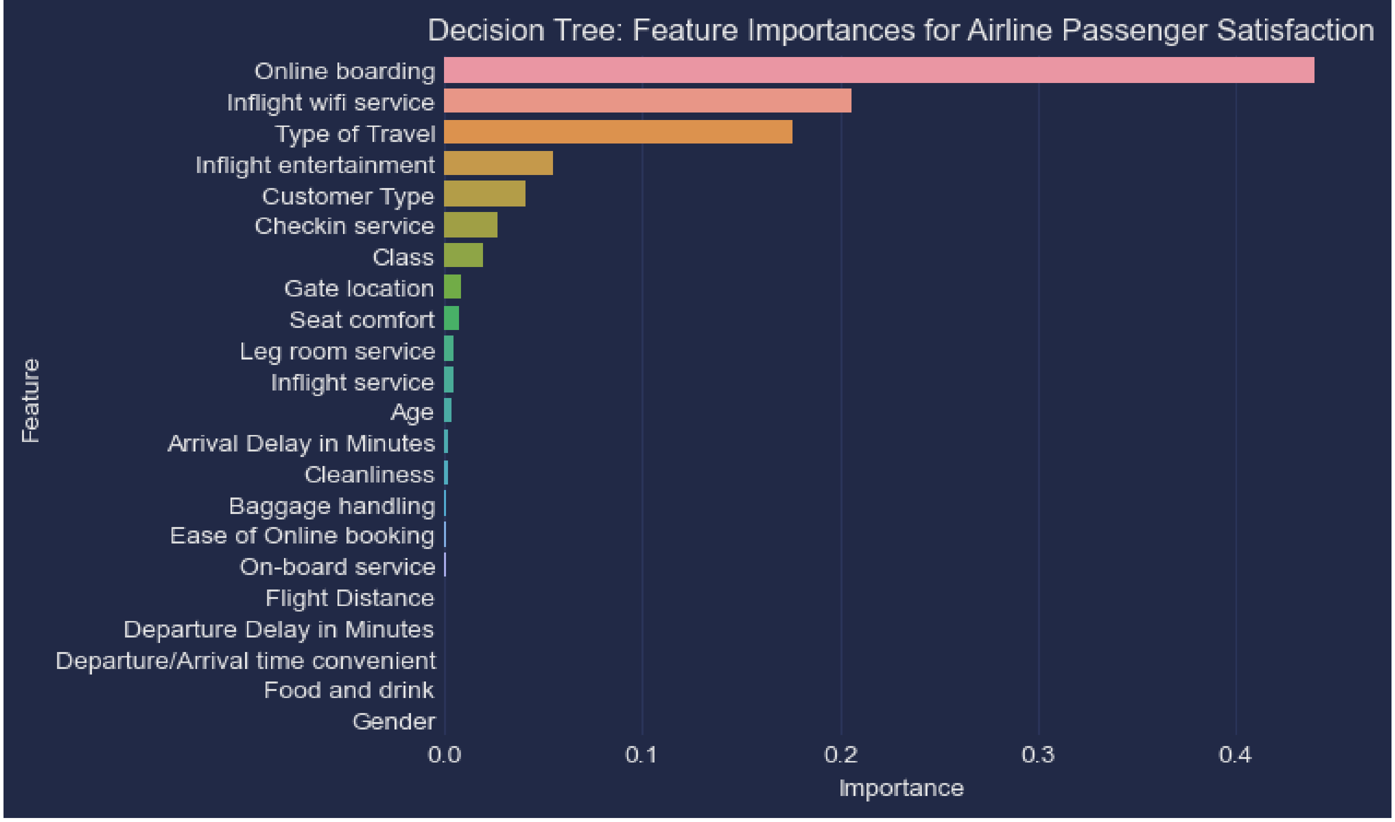
```
dt2_importance = dt2_importance.sort_values(by='Feature_importances', ascending=False)
dt2_importance
```

Out[77]:

	Feature_importances
Online boarding	0.440042
Inflight wifi service	0.205992
Type of Travel	0.175846
Inflight entertainment	0.054725
Customer Type	0.040992
Checkin service	0.027088
Class	0.019916
Gate location	0.008129
Seat comfort	0.007364
Leg room service	0.004883
Inflight service	0.004730
Age	0.003356
Arrival Delay in Minutes	0.002082
Cleanliness	0.001728
Baggage handling	0.001182
Ease of Online booking	0.000762
On-board service	0.000748
Flight Distance	0.000283
Departure Delay in Minutes	0.000111
Departure/Arrival time convenient	0.000038
Food and drink	0.000003
Gender	0.000000

In [78]:

```
sns.barplot(data=dt2_importance, x="Feature_importances", y=dt2_importance.index, orient
plt.title("Decision Tree: Feature Importances for Airline Passenger Satisfaction", fonts
plt.ylabel("Feature")
plt.xlabel("Importance")
plt.show()
```



dari barplot diatas menunjukkan bahwa dari hasil model Decision tree menunjukkan bahwa, "online boarding", "inflight wifi service", "type of travel" memiliki kepentingan paling tinggi dalam urutan tersebut.

In [79]:

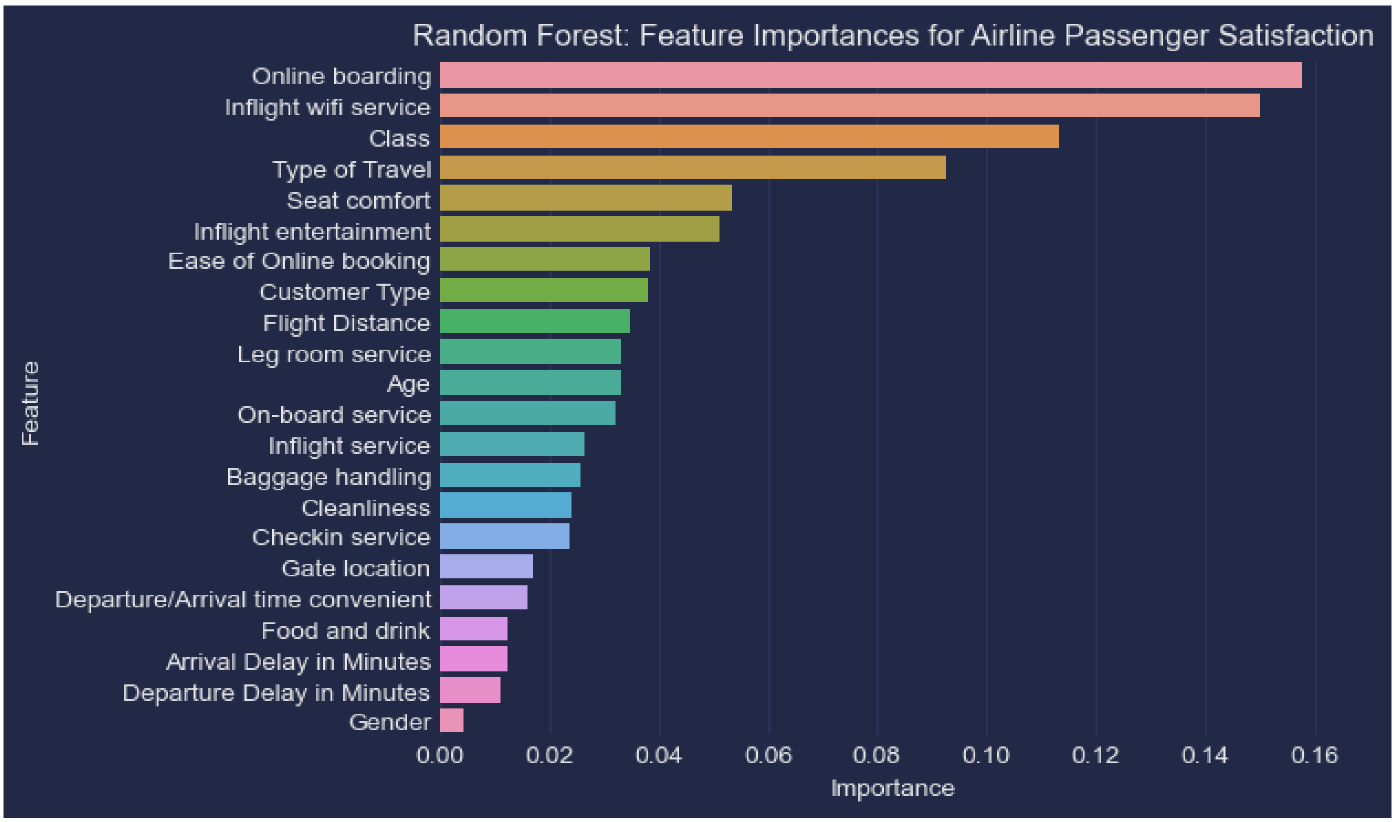
```
rf_importance = pd.DataFrame(RF.feature_importances_,
                             columns=['Feature_importances'],
                             index=x_train.columns)

rf_importance = rf_importance.sort_values(by='Feature_importances', ascending=False)
rf_importance
```

Out[79]:

	Feature_importances
Online boarding	0.157862
Inflight wifi service	0.150039
Class	0.113255
Type of Travel	0.092588
Seat comfort	0.053350
Inflight entertainment	0.051149
Ease of Online booking	0.038407
Customer Type	0.038242
Flight Distance	0.034816
Leg room service	0.033026
Age	0.032985
On-board service	0.031903
Inflight service	0.026355
Baggage handling	0.025734
Cleanliness	0.023881

Checkin service	0.023565
Gate location	0.016898
Departure/Arrival time convenient	0.015932
Food and drink	0.012514
Arrival Delay in Minutes	0.012321
Departure Delay in Minutes	0.010869
Gender	0.004309

In [80]:
```
sns.barplot(data=rf_importance, x="Feature_importances", y=rf_importance.index, orient='
plt.title("Random Forest: Feature Importances for Airline Passenger Satisfaction", fonts
plt.ylabel("Feature")
plt.xlabel("Importance")
plt.show()
```



dari barplot diatas menunjukkan bahwa dari hasil model Random forest menunjukkan bahwa, "online boarding", "inflight wifi service", "Class" memiliki kepentingan paling tinggi dalam urutan tersebut. namun pada model Random Forest nilai yang dihasilkan tidak sebesar model decision tree dan xgboost, ini dikarenakan pada model Random Forest saya tidak menggunakan GridSearchCV. alasan tidak menggunakan GridSearchCV karena terlalu lama saat pemrosesan, dikarenakan komputasi saya kurang memadai.

In [81]:
```
xgb_importance = pd.DataFrame(xgb.best_estimator_.feature_importances_,
                              columns=['Feature_importances'],
                              index=x_train.columns)

xgb_importance = xgb_importance.sort_values(by='Feature_importances', ascending=False)
xgb_importance
```

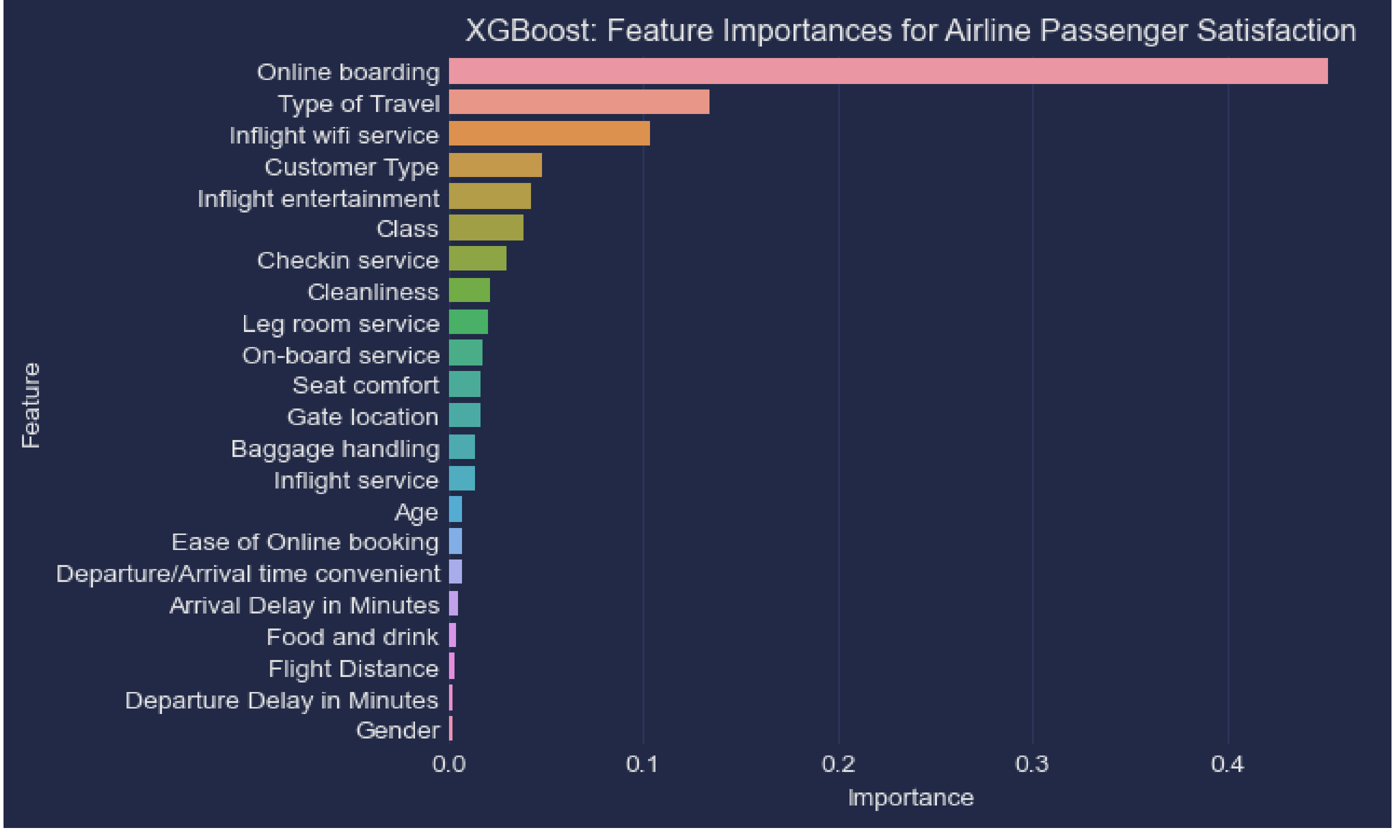
Out[81]:

	Feature_importances
Online boarding	0.452311
Type of Travel	0.133898

Inflight wifi service	0.103772
Customer Type	0.047521
Inflight entertainment	0.042332
Class	0.038401
Checkin service	0.029413
Cleanliness	0.021345
Leg room service	0.019723
On-board service	0.017335
Seat comfort	0.016497
Gate location	0.015913
Baggage handling	0.013580
Inflight service	0.012997
Age	0.007028
Ease of Online booking	0.006755
Departure/Arrival time convenient	0.006377
Arrival Delay in Minutes	0.004884
Food and drink	0.003470
Flight Distance	0.002757
Departure Delay in Minutes	0.002222
Gender	0.001469

In [82]:

```
sns.barplot(data=xgb_importance, x="Feature_importances", y=xgb_importance.index, orient
plt.title("XGBoost: Feature Importances for Airline Passenger Satisfaction", fontsize=12
plt.ylabel("Feature")
plt.xlabel("Importance")
plt.show()
```



dari barplot diatas menunjukkan bahwa dari hasil model XGBoost menunjukkan bahwa, "online boarding", "type of travel", "Inflight wifi service" memiliki kepentingan paling tinggi dalam urutan tersebut.

Kesimpulan dan Rekomendasi

Dari hasil ketiga model diatas menunjukkan bahwa variable paling penting yang mempengaruhi kepuasan pelanggan adalah "online boarding", "type of travel", "Inflight wifi service".

untuk dapat mempertahankan pelanggan, beberapa rekomendasi dapat disampaikan kepada stakeholders:

- Perbaiki sistem "Online Boarding" dari segi database, server, pengalaman pengguna baik website ataupun aplikasi mobile. perbaiki tampilannya juga bila diperlukan.
- Menyelidiki apakah adanya bug didalam website, server, database, aplikasi sehingga membuat pelanggan merasa tidak nyaman ketika melakukan online boarding.
- Perbaiki layanan wifi saya pesawat berada diatas, karena layanan wifi dapat membuat pelanggan merasa nyaman, tidak bosan saat perjalanan panjang. ini akan membuat pelanggan juga merasa senang ketika perjalanan panjang.
- untuk type of travel, kalau melihat dari insight/informasi grafik diatas (EDA) menunjukkan bahwa pada Personal Travel memiliki ketidakpuasan yang sangat tinggi. mungkin bisa diperbaiki dari segi pelayanan yang diberikan kepada Personal Travel. seperti, ruang tunggu, pelayanan crew, pelayanan checkin pada personal travel.