Import Packages and Data

https://www.kaggle.com/datasets/raminhuseyn/airline-customer-satisfaction

In [89]:

```
import os
os.getcwd()

%cd "C:\Users\Angela\OneDrive\Desktop\ANA500"
```

```
C:\Users\Angela\OneDrive\Desktop\ANA500
```

In [137…

```
#import packages
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import classification_report
```

In [91]:

```
#load dataset
df = pd.read_csv('Airline_customer_satisfaction.csv')
```

Prepare Data

In [92]:

```
#Review data type of variables
```

```
df.dtypes
```

Out[92]:

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

In [93]:

```
#Review size of data
df.shape
```

Out[93]:

```
(129880, 22)
```

In [94]:

```
# check if there are null values
df.isnull().sum()
```

Out[94]:

```
satisfaction                           0
Customer Type                          0
Age                                    0
Type of Travel                         0
Class                                  0
Flight Distance                        0
Seat comfort                           0
Departure/Arrival time convenient      0
Food and drink                         0
Gate location                          0
Inflight wifi service                  0
Inflight entertainment                 0
Online support                         0
Ease of Online booking                 0
On-board service                       0
Leg room service                       0
Baggage handling                       0
Checkin service                        0
Cleanliness                            0
Online boarding                        0
Departure Delay in Minutes             0
Arrival Delay in Minutes             393
dtype: int64
```

Arrival Delay in Minutes has 393 missing values.

Handle Missing Values

In [66]:

```
#Take a closer look at Arrival Delay in Minutes
df['Arrival Delay in Minutes'].describe()
```

Out[66]:

```
count    129487.000000
mean         15.091129
std          38.465650
min           0.000000
25%           0.000000
50%           0.000000
75%          13.000000
max        1584.000000
Name: Arrival Delay in Minutes, dtype: float64
```

Arrival Delay in Minutes has values that range from 0 to 1584. The values have a mean of 15.091129

In [95]:

```
#Create new variable with copy of original data
Arrival_Delay = df['Arrival Delay in Minutes']

#Add new variable to dataframe
df2 = df.assign(Arrival_Delay_Minutes = Arrival_Delay)

#Check that new variable matches orginal variable
df2['Arrival_Delay_Minutes'].describe()
```

Out[95]:

```
count    129487.000000
mean         15.091129
std          38.465650
min           0.000000
25%           0.000000
50%           0.000000
75%          13.000000
max        1584.000000
Name: Arrival_Delay_Minutes, dtype: float64
```

I want to ensure that I keep the integrity of the original data. Therefore, I created a new variable to use in place of Arrival Delay in Minutes. I will be copying Arrival Delay in Minutes to the new variabl. I will be manipulating the new variable to handle the missing values.

In [68]:

```
#fill missing values on new variable with mean value
# filling missing value using fillna()
df2['Arrival_Delay_Minutes'].fillna(15.091129, inplace = True)
```

```
#check the new variable to see if the missing values are updated
df2['Arrival_Delay_Minutes'].isnull().sum()
```

Out[68]:

```
0
```

Because there are a small amount of missing values, I chose to replace them with the Mean value of the variable. This will help keep more data to work with rather than dropping the values.

In [96]:

```
#Drop the orginal variable in second dataframe
df2.drop('Arrival Delay in Minutes', axis=1, inplace=True)

#Check that the variable was dropped
df2.columns
```

Out[96]:

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

Since the new variable was assigned to a new dataframe, I chose to drop Arrival Delay in Minutes since the data is duplicative of the new variable Arrival_Delay_Minutes.

Data Visualizations

In [97]:

```
#Print the statistics of the variables
df2.describe()
```

Out[97]:

	Age	Flight Distance	Seat comfort	Departure/Arrival time convenient	Food and drink	Gate location	Inflight wifi service	e
count	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	1
mean	39.427957	1981.409055	2.838597	2.990645	2.851994	2.990422	3.249130	
std	15.119360	1027.115606	1.392983	1.527224	1.443729	1.305970	1.318818	
min	7.000000	50.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	27.000000	1359.000000	2.000000	2.000000	2.000000	2.000000	2.000000	
50%	40.000000	1925.000000	3.000000	3.000000	3.000000	3.000000	3.000000	
75%	51.000000	2544.000000	4.000000	4.000000	4.000000	4.000000	4.000000	
max	85.000000	6951.000000	5.000000	5.000000	5.000000	5.000000	5.000000	

Numeric Variables

In [98]:

```
#Create dataframe with only numeric variables
num_df = df2.select_dtypes(include = np.number)
num_df.columns = num_df.columns.str.replace(' ', '_')
num_df.columns = num_df.columns.str.replace('/', '_')
num_df.columns = num_df.columns.str.replace('-', '_')

#Check to see that the dataframe is correct
num_df.dtypes
```

Out[98]:

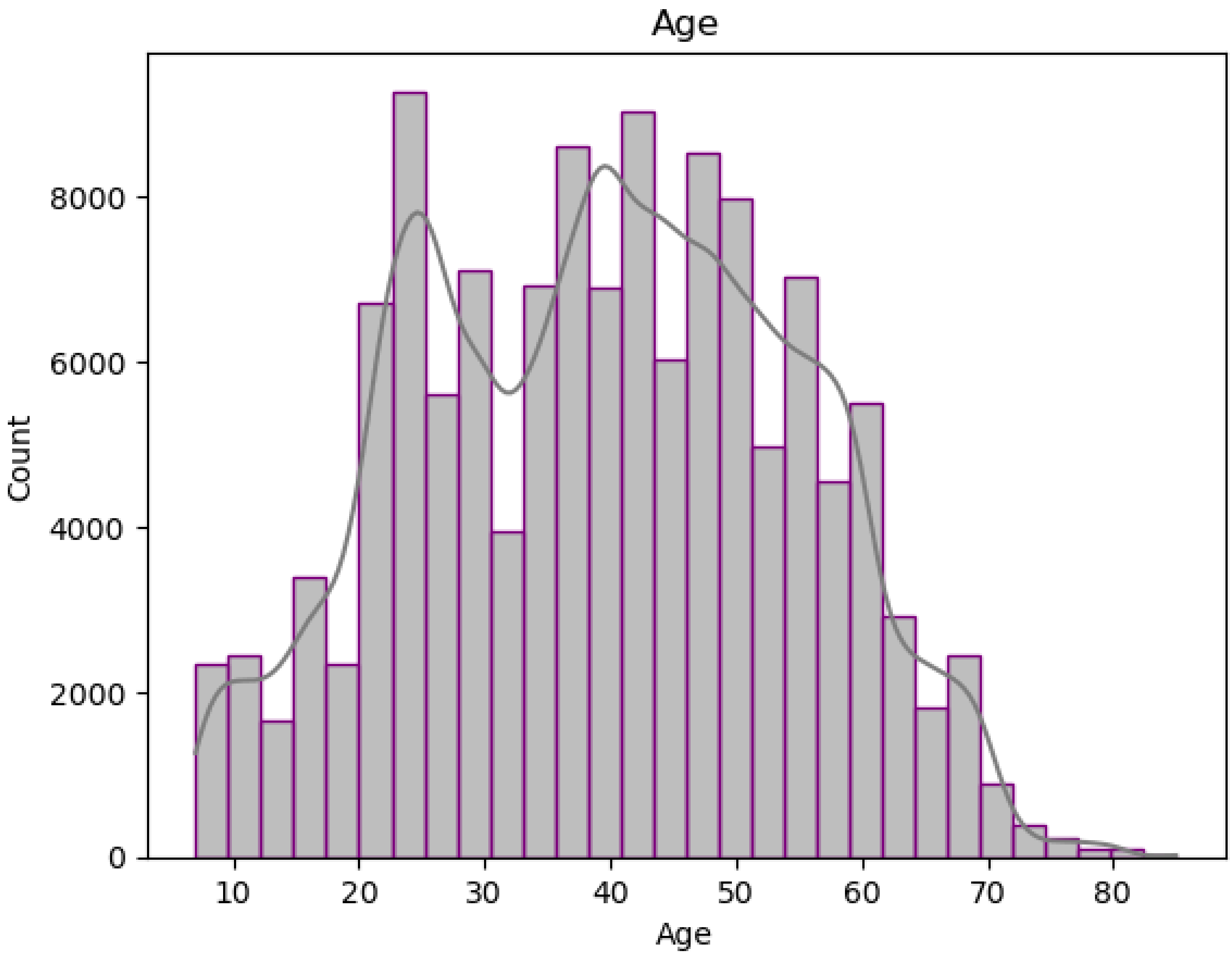
```
Age                                    int64
Flight_Distance                        int64
Seat_comfort                           int64
Departure_Arrival_time_convenient      int64
Food_and_drink                         int64
Gate_location                          int64
Inflight_wifi_service                  int64
Inflight_entertainment                 int64
Online_support                         int64
Ease_of_Online_booking                 int64
On_board_service                       int64
Leg_room_service                       int64
Baggage_handling                       int64
Checkin_service                        int64
Cleanliness                            int64
Online_boarding                        int64
Departure_Delay_in_Minutes             int64
Arrival_Delay_Minutes                float64
dtype: object
```

In [99]:

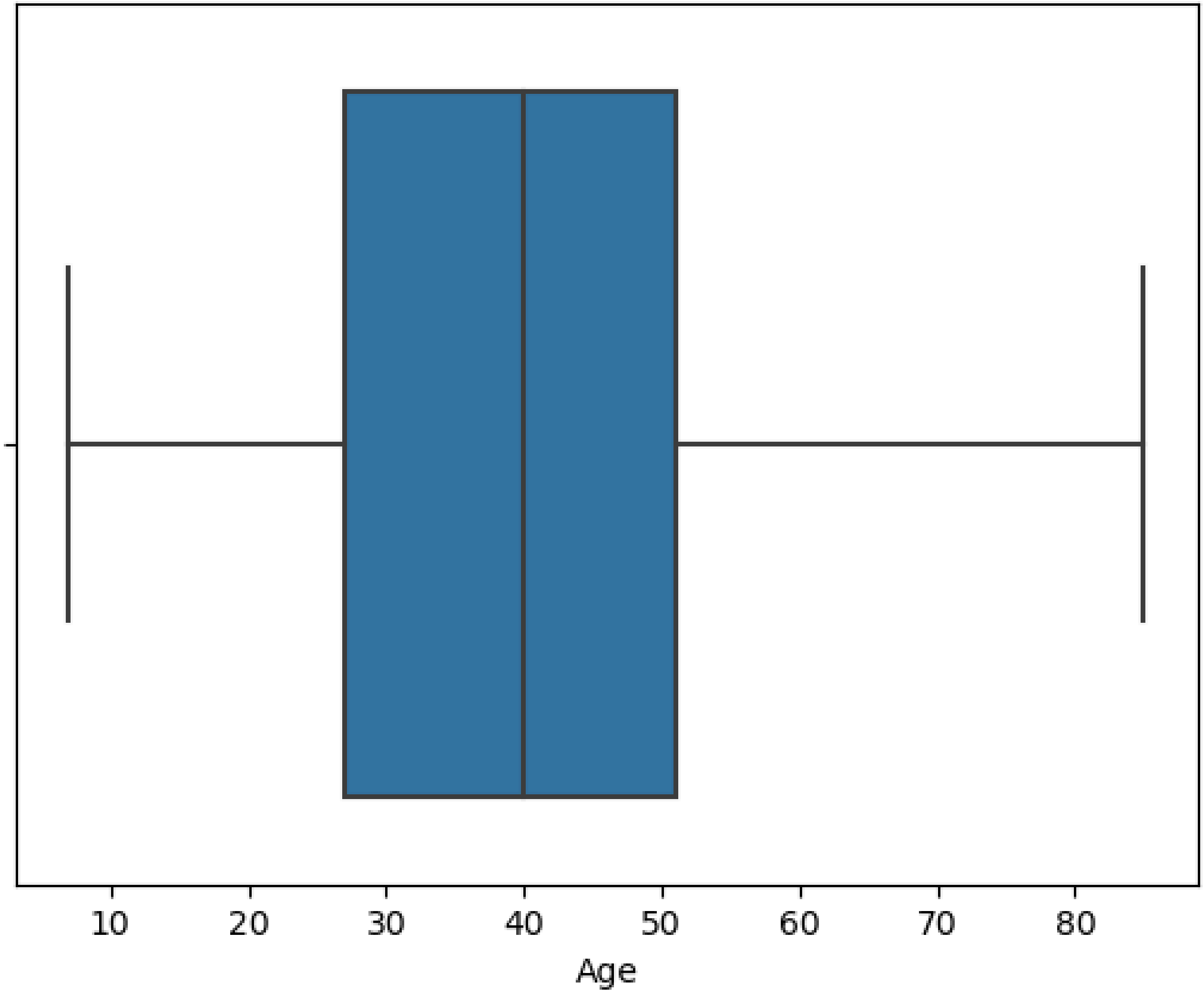
```
#Age Histogram
sns.histplot(num_df.Age, bins=30, kde=True, color='grey', edgecolor='purple')
plt.title('Age')
```

Out[99]:

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



```
In [100…  sns.boxplot(x= num_df.Age)

Out[100]: <Axes: xlabel='Age'>
```



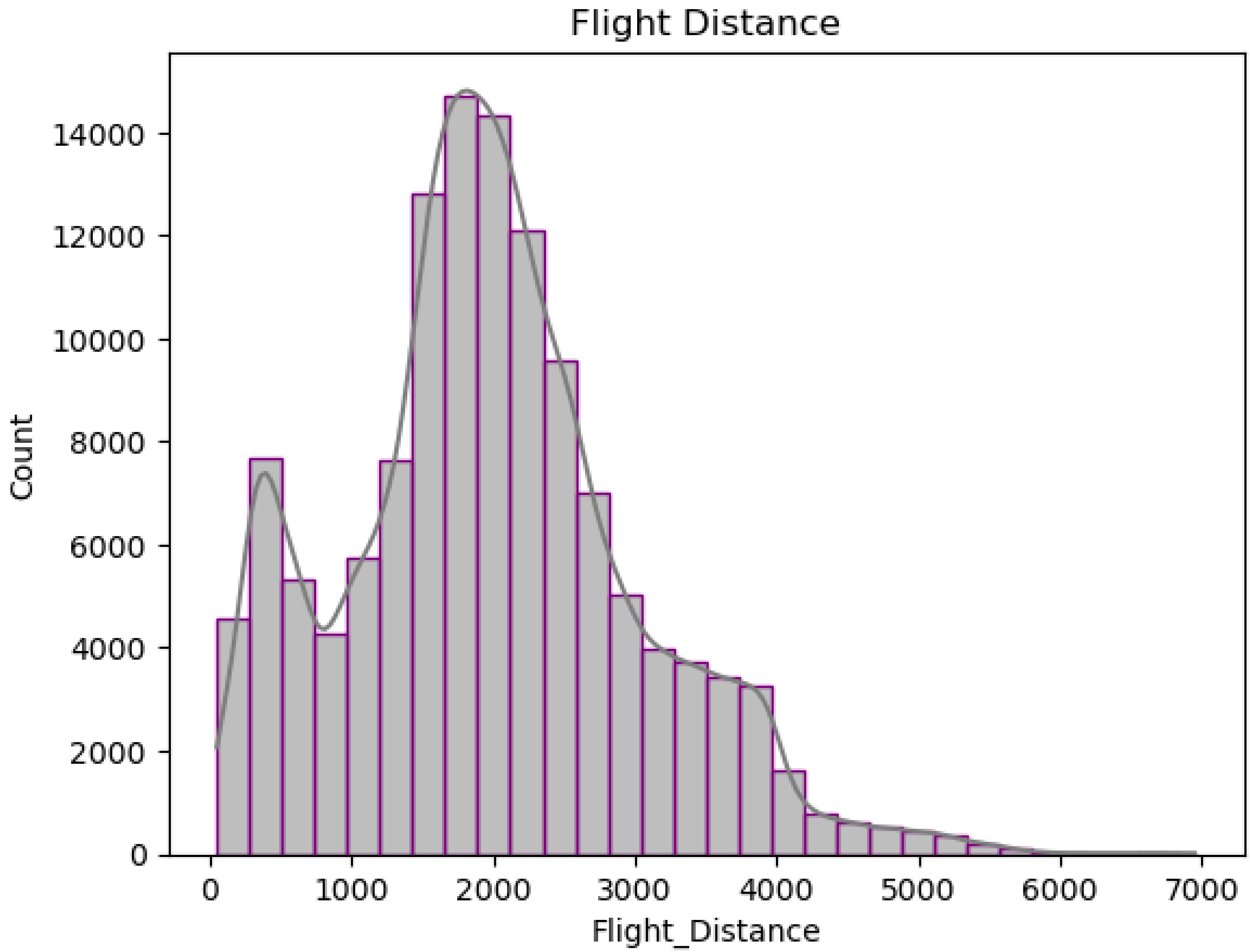
Customers age ranges from 10 years old to 80+ years old. Age has a normal distribution. Age does not have any outliers. The most ages that travel are between 20 and 50 years old.

In [101…

```
#Flight Distance Histogram
sns.histplot(num_df.Flight_Distance, bins=30, kde=True, color='grey', edgecolor='purple')
plt.title('Flight Distance')
```

Out[101]:

```
Text(0.5, 1.0, 'Flight Distance')
```



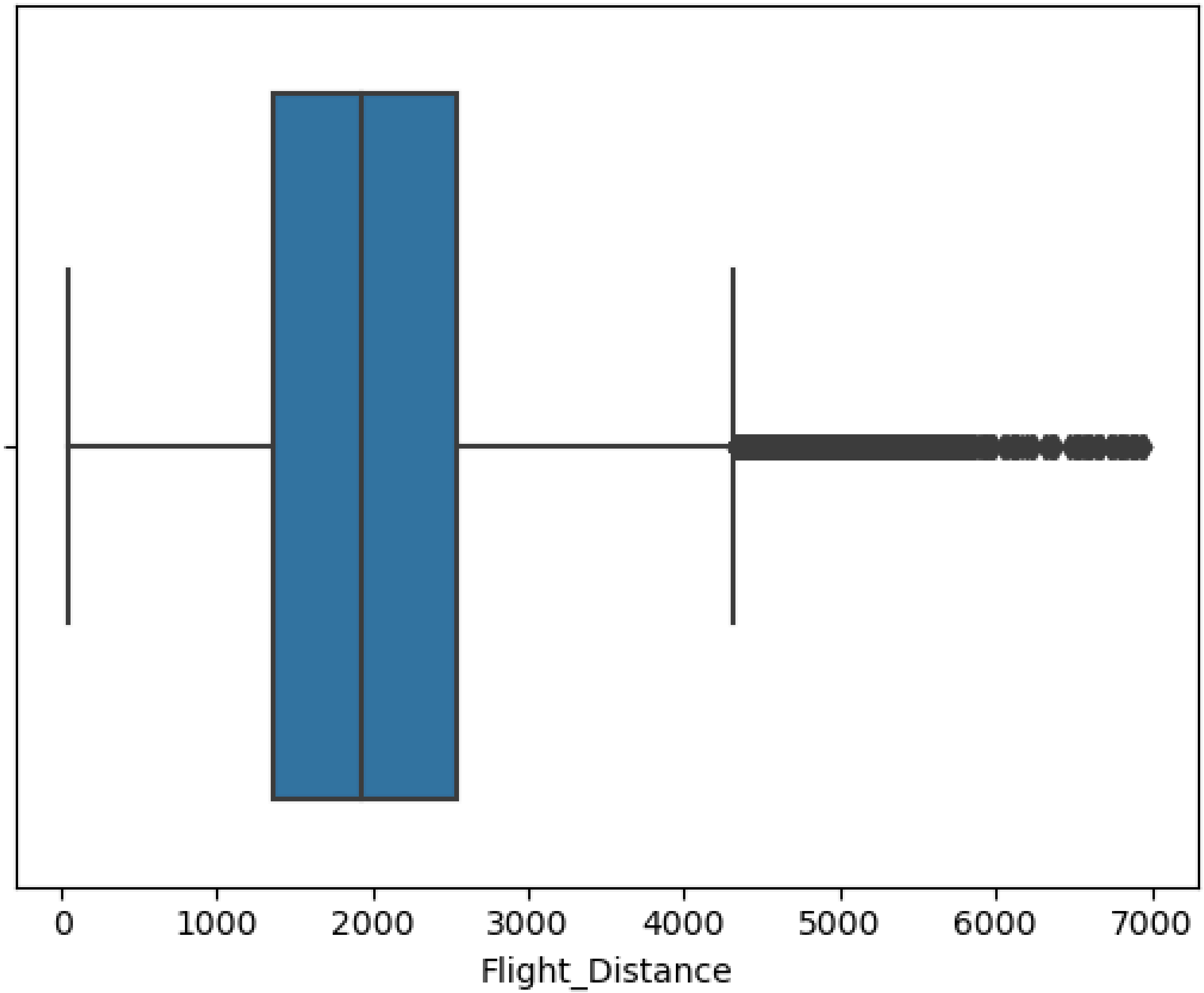
There appears to be a right skew for Flight Distance. The data appears to go flat at 6000. It also appears that there may be some outlier at 7000.

In [102…

```
#Flight Distance BoxPlot
sns.boxplot(x= num_df.Flight_Distance)
```

Out[102]:

```
<Axes: xlabel='Flight_Distance'>
```



Outliers are shown just after 4000.

In [103…

```
# How many values are outliers?

flight_outliers = num_df[(num_df['Flight_Distance'] > 4200)]
flight_outliers.Flight_Distance.describe()
```

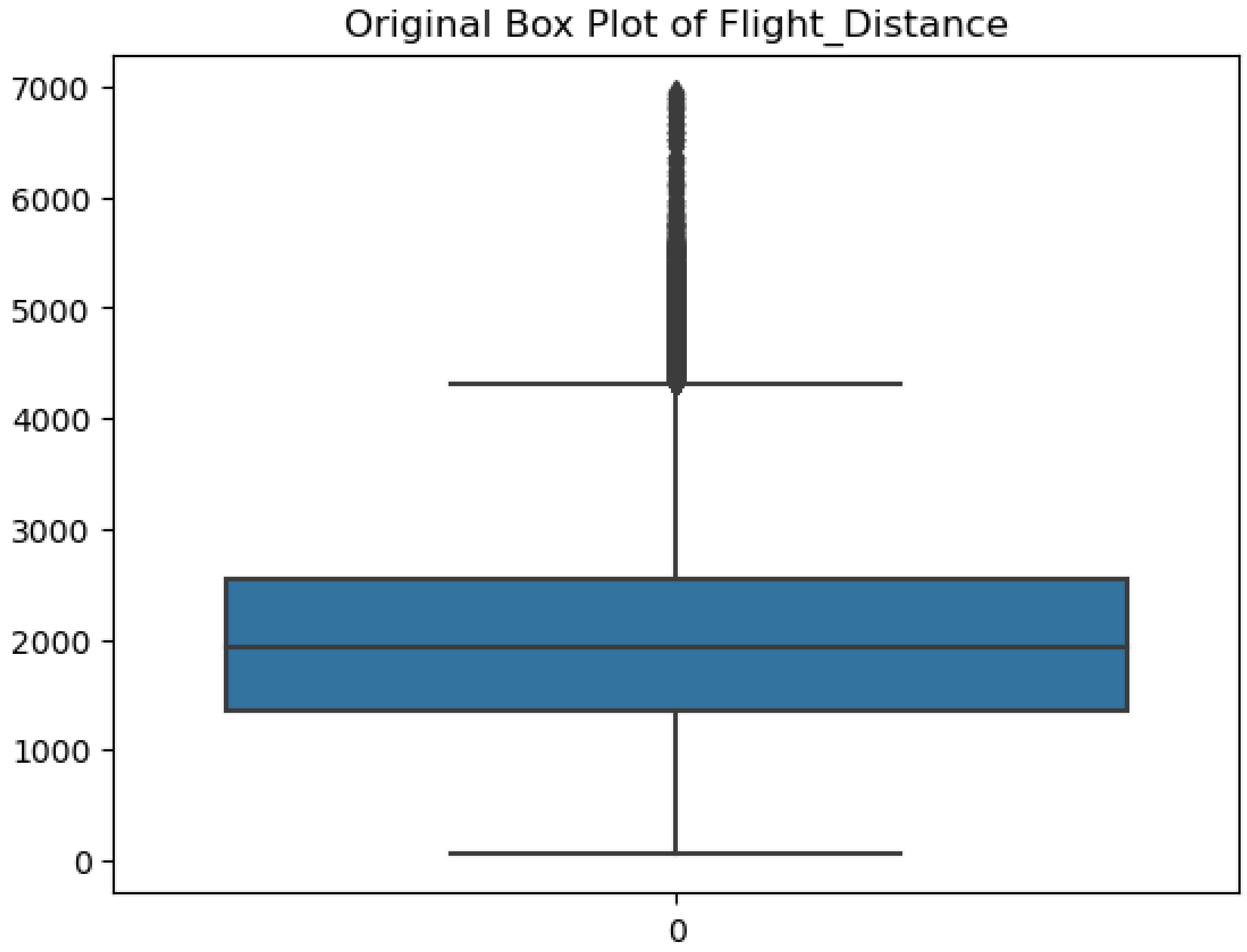
Out[103]:

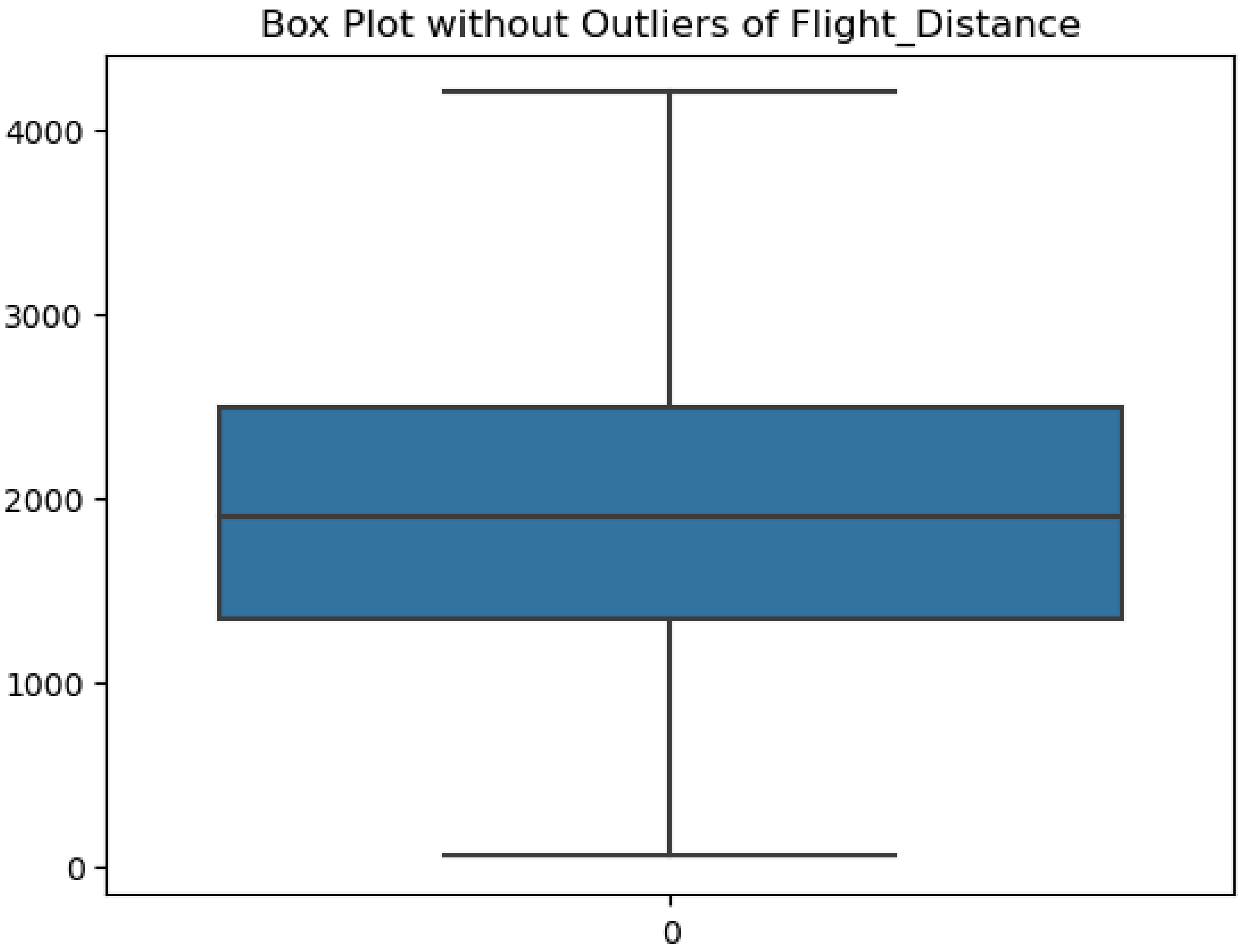
```
count    3011.000000
mean     4813.997675
std       482.818474
min      4201.000000
25%      4429.500000
50%      4729.000000
75%      5103.000000
max      6951.000000
Name: Flight_Distance, dtype: float64
```

There are only 3011 values that are greater than 4200 which are listed as outliers on the boxplot. These values will be dropped since they are such a small portion of the data.

In [104…

```
def removal_box_plot(df, column, threshold):
    sns.boxplot(df[column])
    plt.title(f'Original Box Plot of {column}')
    plt.show()

    removed_outliers = df[df[column] <= threshold]

    sns.boxplot(removed_outliers[column])
    plt.title(f'Box Plot without Outliers of {column}')
    plt.show()
    return removed_outliers


threshold_value = 4200

no_outliers = removal_box_plot(num_df, 'Flight_Distance', threshold_value)
```


Original Box Plot of Flight_Distance
7000
6000
5000
4000
3000
2000
1000
0
0



In [105…

```
# drop rows containing outliers
df3 = df2.drop(flight_outliers.index)
df3['Flight Distance'].describe()
```

Out[105]:

```
count    126869.000000
mean       1914.182826
std         937.831785
min          50.000000
25%        1337.000000
50%        1900.000000
75%        2488.000000
max        4200.000000
Name: Flight Distance, dtype: float64
```

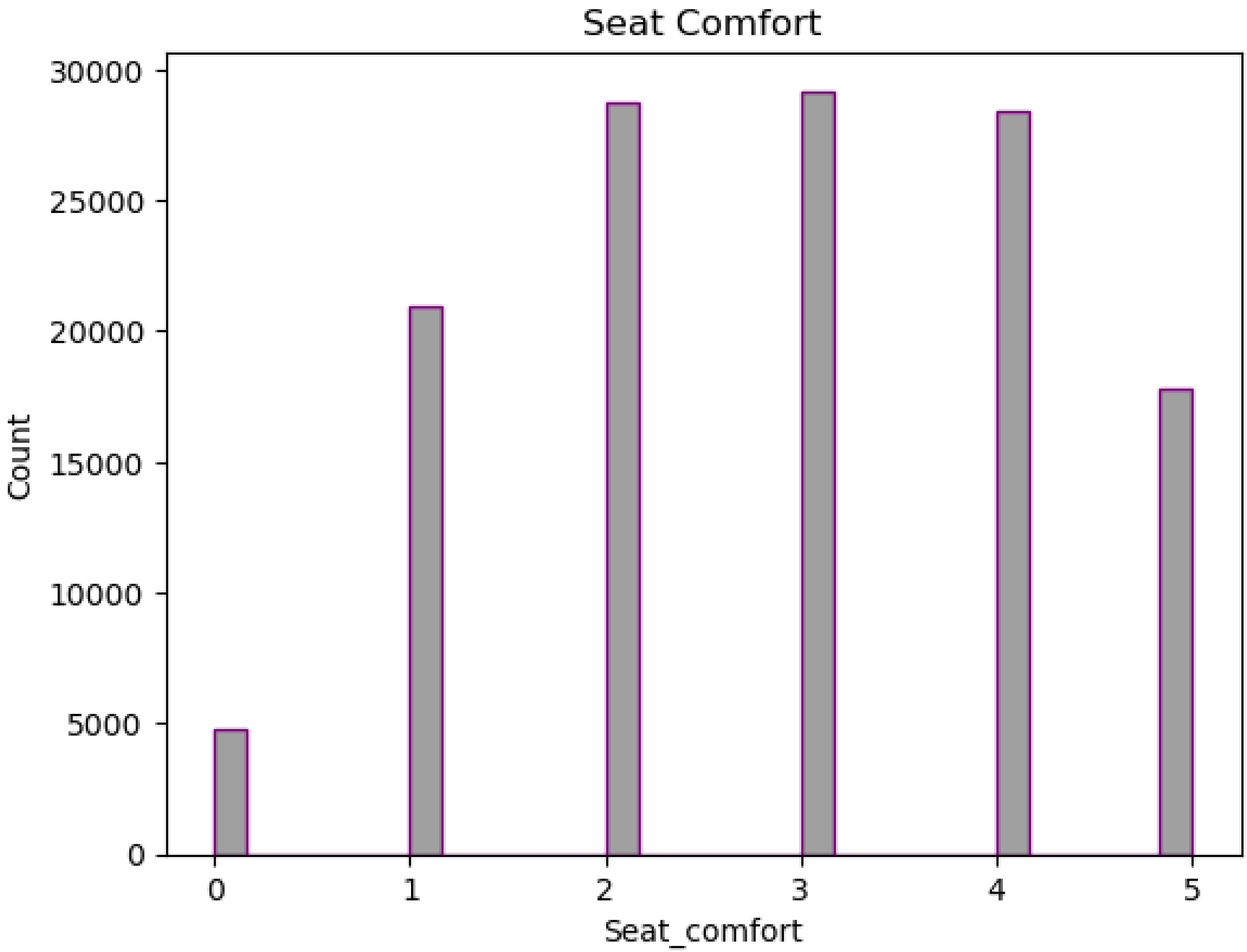
After Flight Distance outliers are removed, there are 126,869 values remaining. The values range from 50 to 4,200.

In [106…

```
# Histogram of Seat Comfort
sns.histplot(num_df.Seat_comfort, bins=30, color='grey', edgecolor='purple')
plt.title('Seat Comfort')
```

Out[106]:

```
Text(0.5, 1.0, 'Seat Comfort')
```



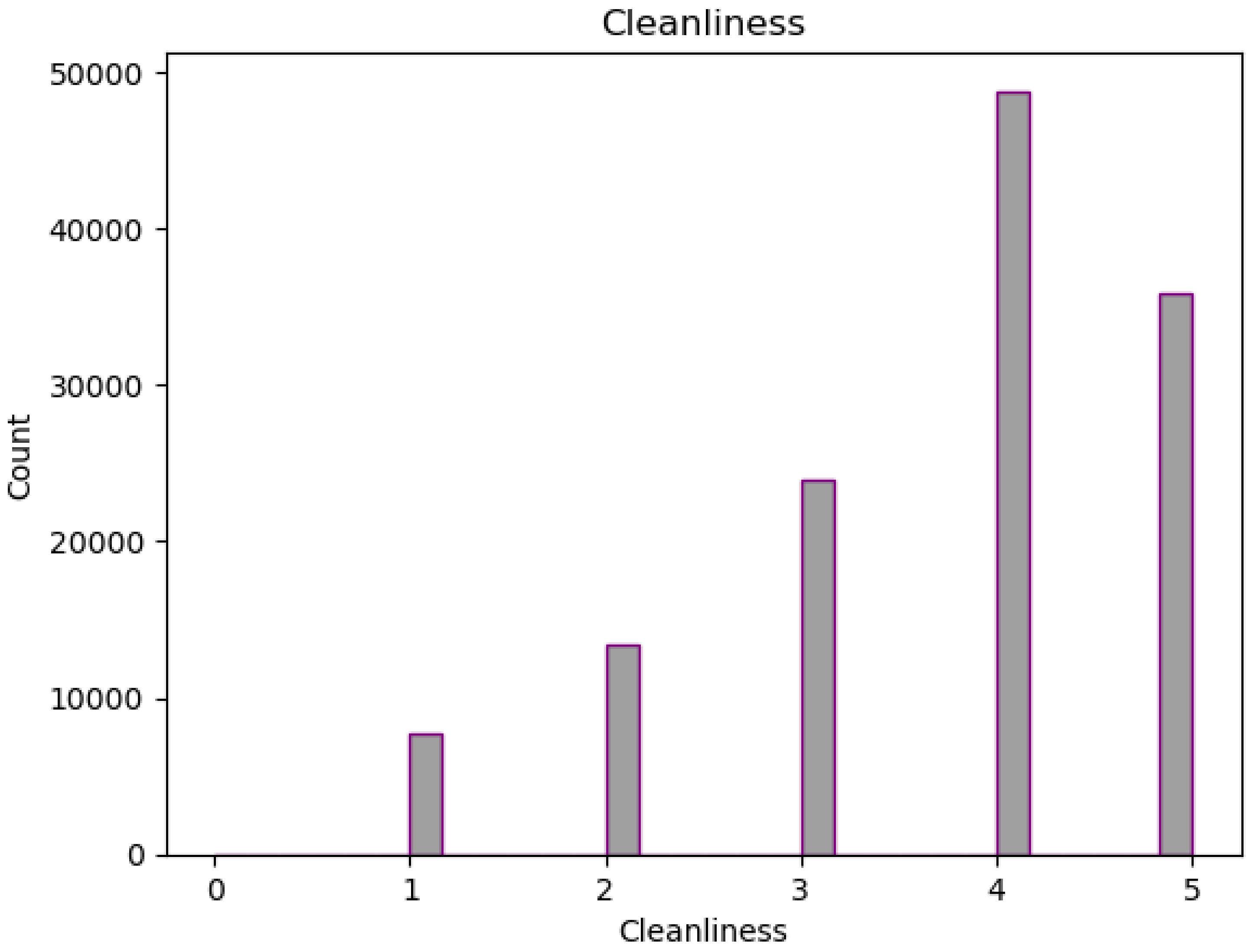
Customers age ranges from 10 years old to 80+ years old. Age has a normal distribution. Age does not have any outliers. The most ages that travel are between 20 and 50 years old.

In [211…

```
#Cleanliness Histogram
sns.histplot(num_df.Cleanliness, bins=30, color='grey', edgecolor='purple')
plt.title('Cleanliness')
```

Out[211]:

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



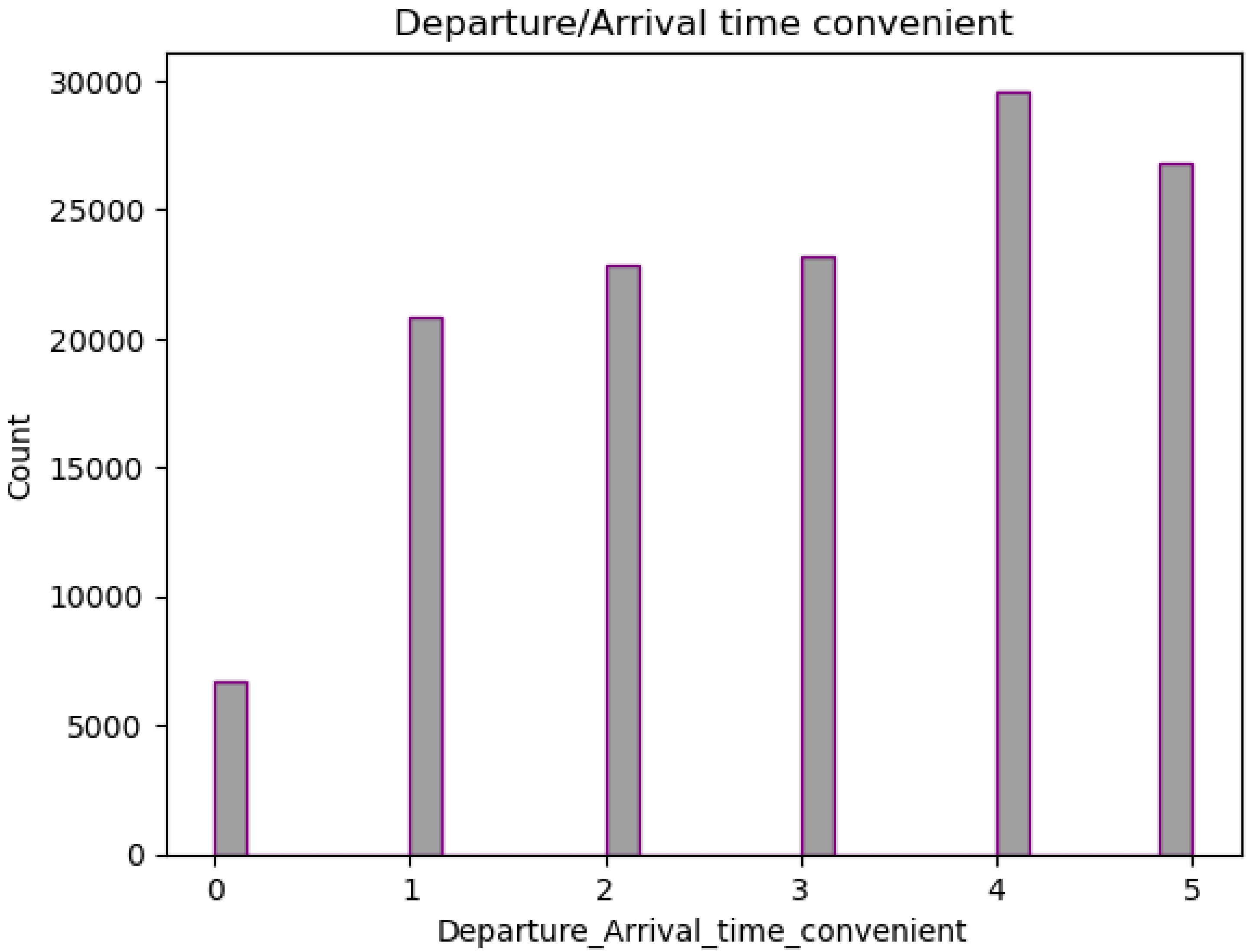
Cleanliness is a categorical variable that ranges from 0 to 5. The rating of 4 has the highest count of customers.

In [212…

```
#Departure/Arrival time convenient Histogram
sns.histplot(num_df.Departure_Arrival_time_convenient, bins=30, color='grey', edgecolor='purple')
plt.title('Departure/Arrival time convenient')
```

Out[212]:

```
Text(0.5, 1.0, 'Departure/Arrival time convenient')
```



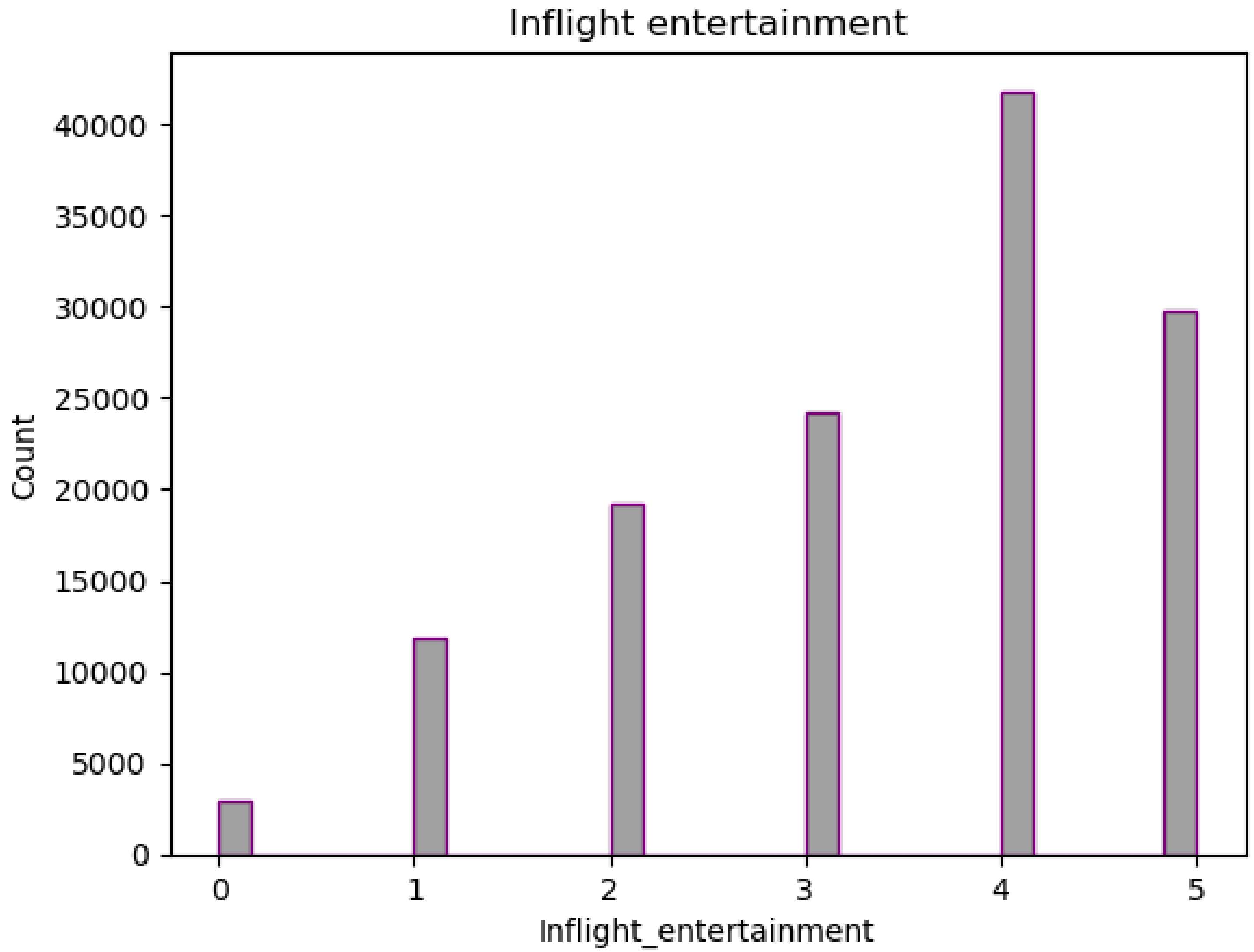
Departure/Arrival time convenient is a categorical variable. The rating scale ranges from 0 to 5. The rating 4 has the highest count of customers.

In [213…

```
#Inflight entertainment Histogram
sns.histplot(num_df.Inflight_entertainment, bins=30, color='grey', edgecolor='purple')
plt.title('Inflight entertainment')
```

Out[213]:

```
Text(0.5, 1.0, 'Inflight entertainment')
```



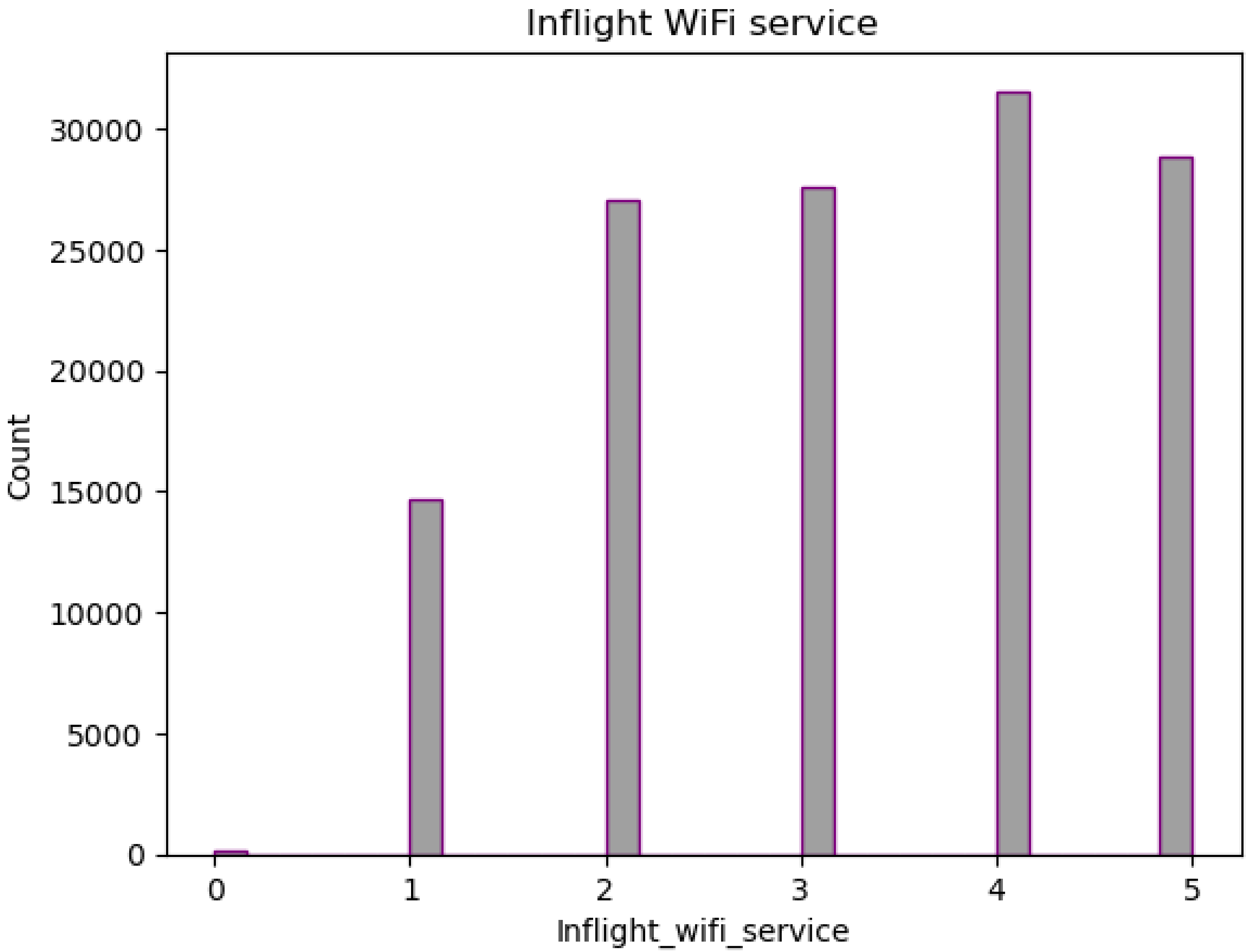
Inflight entertainment is a categorical variable. Inflight Entertainment is a rating scale from 0 to 5. The rating 4 has the highest count of customers.

In [215…

```
#Inflight WiFi service Histogram
sns.histplot(num_df.Inflight_wifi_service, bins=30, color='grey', edgecolor='purple')
plt.title('Inflight WiFi service')
```

Out[215]:

```
Text(0.5, 1.0, 'Inflight WiFi service')
```



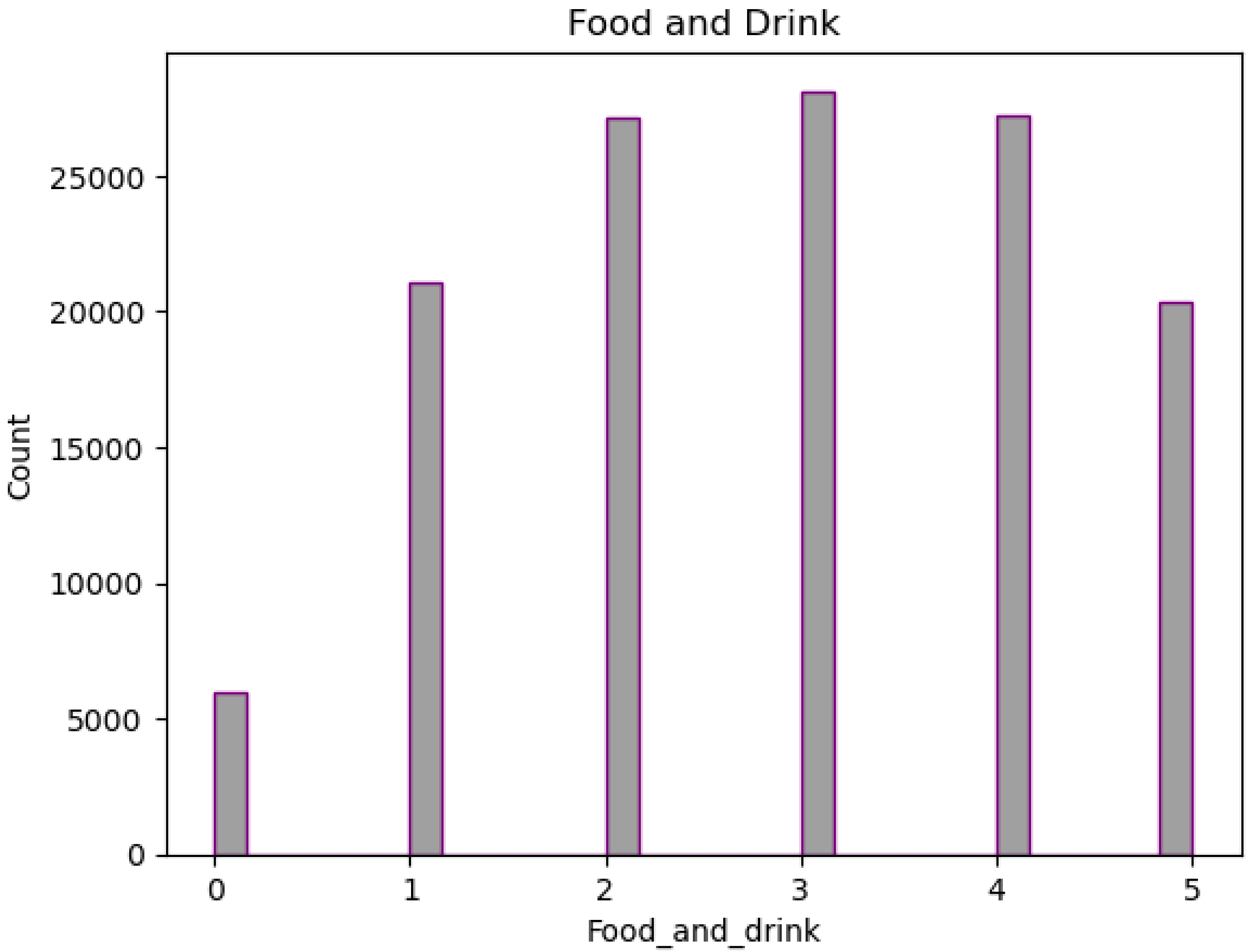
Inflight WiFi service is a categorical variable. Inflight WiFi service is a rating scale from 0 to 5. The rating 4 has the highest count of customers.

In [217…

```
#Food and Drink Histogram
sns.histplot(num_df.Food_and_drink, bins=30, color='grey', edgecolor='purple')
plt.title('Food and Drink')
```

Out[217]:

```
Text(0.5, 1.0, 'Food and Drink')
```



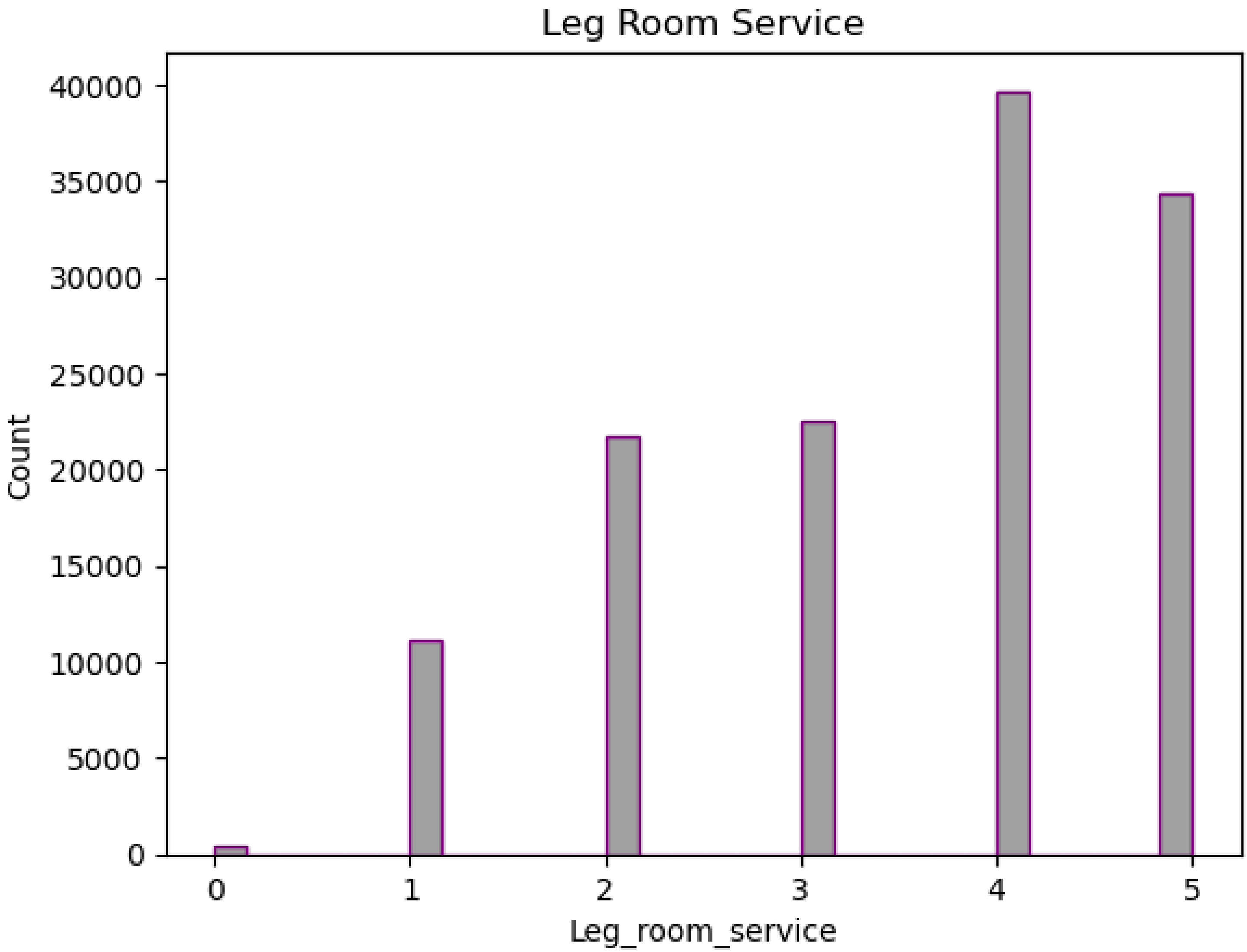
Food and Drink is a categorical variable. Food and Drink is a rating scale from 0 to 5. The rating 3 has the highest count of customers.

In [218…

```
#Leg room service Histogram
sns.histplot(num_df.Leg_room_service, bins=30, color='grey', edgecolor='purple')
plt.title('Leg Room Service')
```

Out[218]:

```
Text(0.5, 1.0, 'Leg Room Service')
```



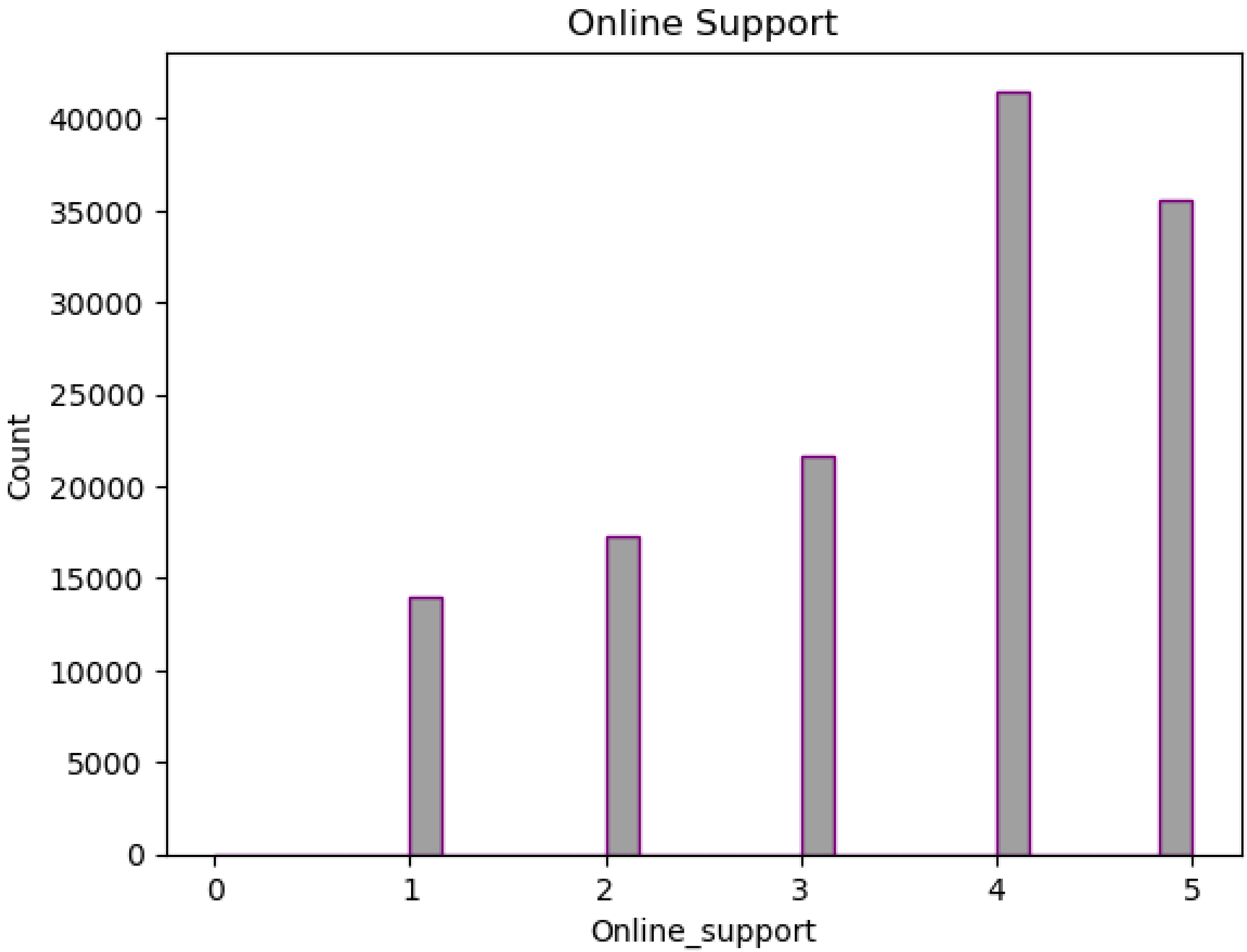
Leg Room Service is a categorical variable. Leg Room Service is a rating scale from 0 to 5. The rating 4 has the highest count of customers.

In [220…

```
#Online Support Histogram
sns.histplot(num_df.Online_support, bins=30, color='grey', edgecolor='purple')
plt.title('Online Support')
```

Out[220]:

```
Text(0.5, 1.0, 'Online Support')
```



Online Support is a categorical variable. Online Support is a rating scale ranging from 0 to 5. There are no customer ratings for Online Support. The rating 4 is the highest customer rating.

Categorical Data

In [107…

```
#Create dataframe with only categorical variables
cat_df = df2.select_dtypes(exclude = np.number)

cat_df.columns = cat_df.columns.str.replace(' ', '_')
```

```
#Review categories
cat_df.head()
```

Out[107]:

	satisfaction	Customer_Type	Type_of_Travel	Class
0	satisfied	Loyal Customer	Personal Travel	Eco
1	satisfied	Loyal Customer	Personal Travel	Business
2	satisfied	Loyal Customer	Personal Travel	Eco
3	satisfied	Loyal Customer	Personal Travel	Eco
4	satisfied	Loyal Customer	Personal Travel	Eco

In [108…

```
#Check Data types
cat_df.dtypes
```

Out[108]:

```
satisfaction      object
Customer_Type     object
Type_of_Travel    object
Class             object
dtype: object
```

In [109…

```
#Change Object to Category
cat_df.satisfaction = cat_df.satisfaction.astype('category')
cat_df.Customer_Type = cat_df.Customer_Type.astype('category')
cat_df.Type_of_Travel = cat_df.Type_of_Travel.astype('category')
cat_df.Class = cat_df.Class.astype('category')

#Check Data types again
cat_df.dtypes
```

Out[109]:

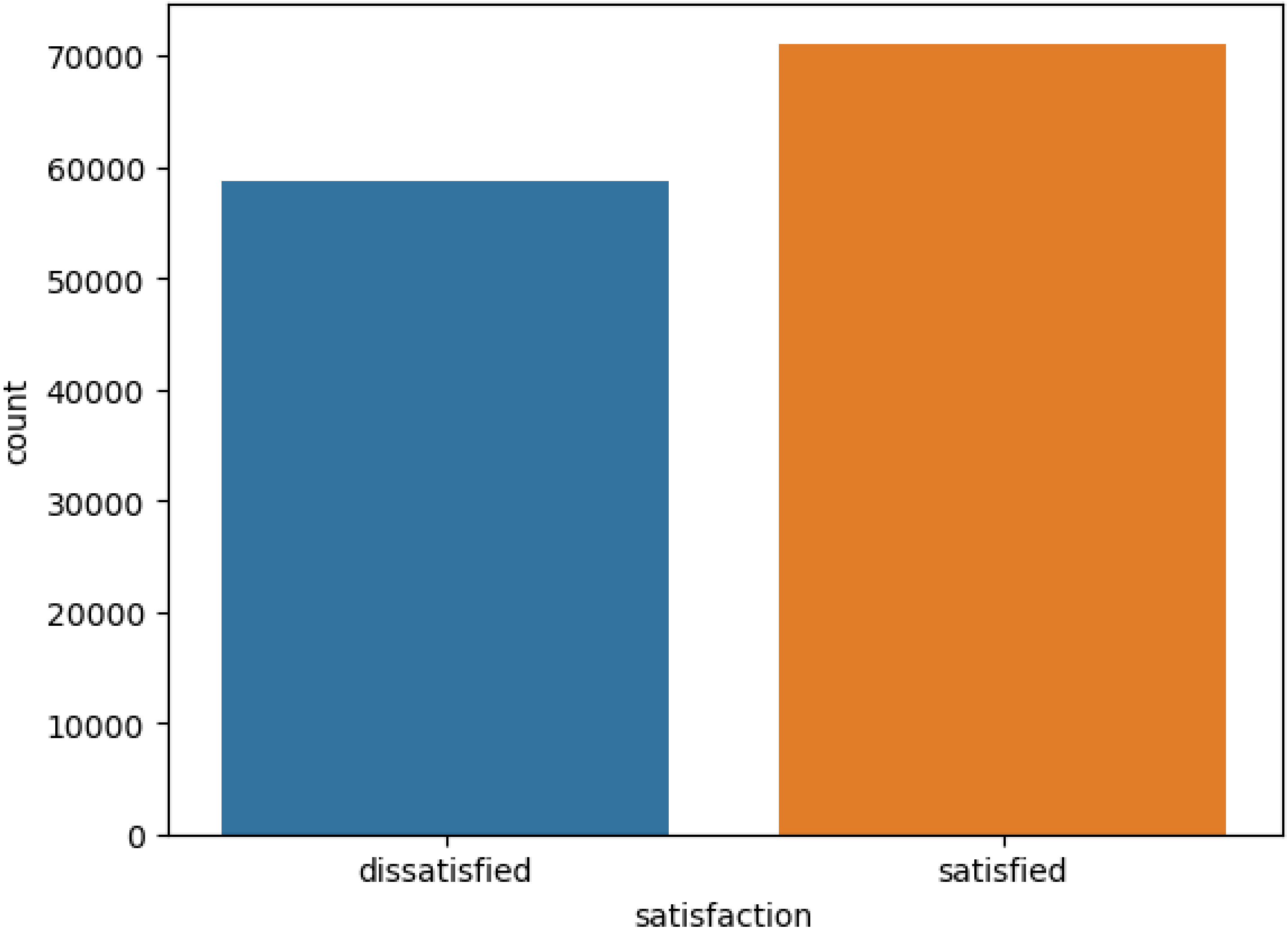
```
satisfaction      category
Customer_Type     category
Type_of_Travel    category
Class             category
dtype: object
```

In [110…

```
sns.countplot(cat_df, x="satisfaction")
```

Out[110]:

```
<Axes: xlabel='satisfaction', ylabel='count'>
```

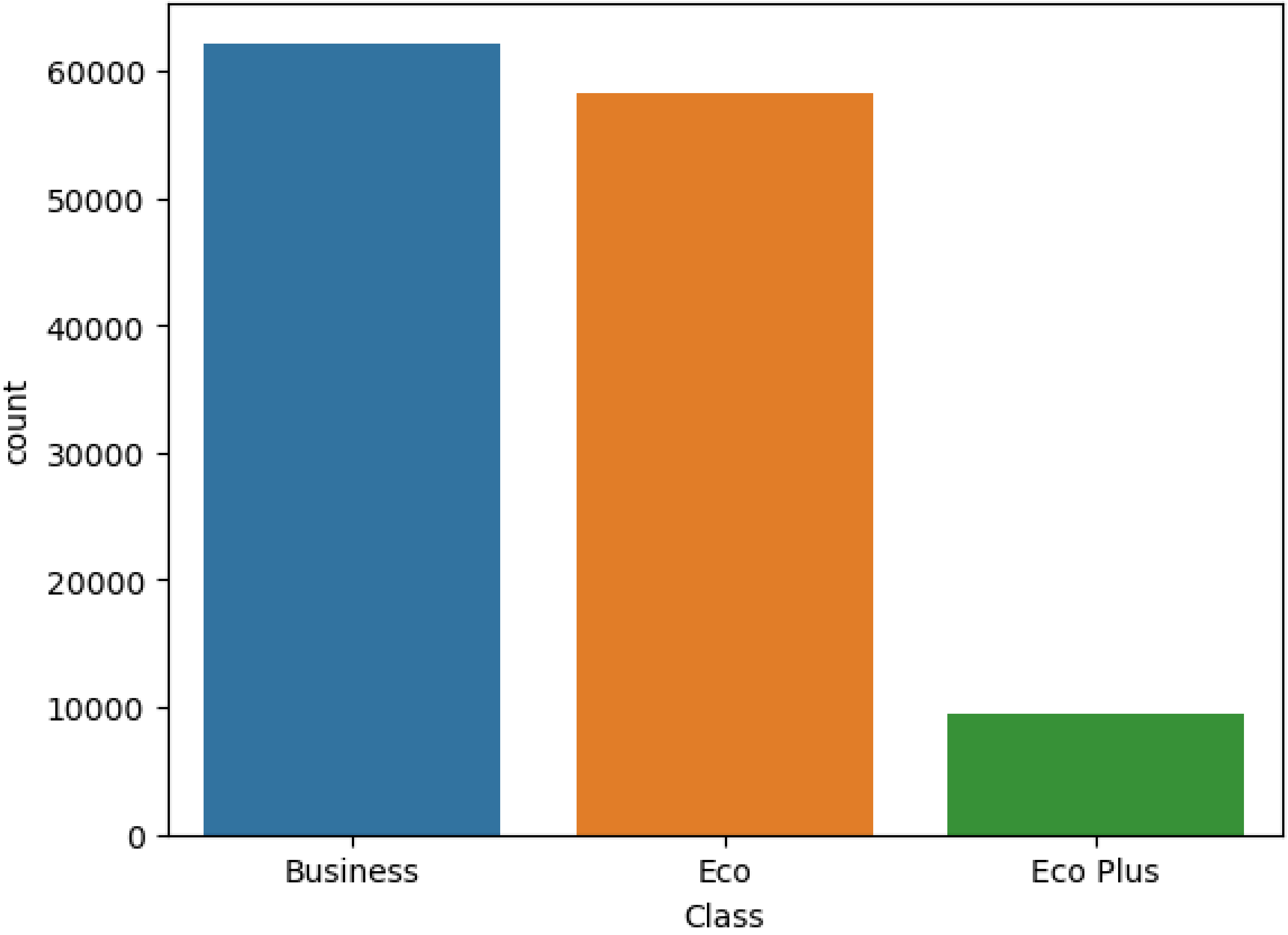


In [111…

```
sns.countplot(cat_df, x="Class")
```

Out[111]:

```
<Axes: xlabel='Class', ylabel='count'>
```



Logistic Regression

In [114…

```
df3.columns
```

Out[114]:

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

In [201…

```
#split data into features and target
X = df3[['Cleanliness', 'Seat comfort', 'Flight Distance', 'Age', 'Departure/Arrival time convenient
y = df3['satisfaction']
```

In [202…

```
#Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

In [203…

```
#Standardize
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [204…

```
# Initialize and train Linear Regression model
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
```

Out[204]:

```
▾ LogisticRegression
LogisticRegression()
```

In [205…

```
#Predict on test set
y_pred = lr_model.predict(X_test)
```

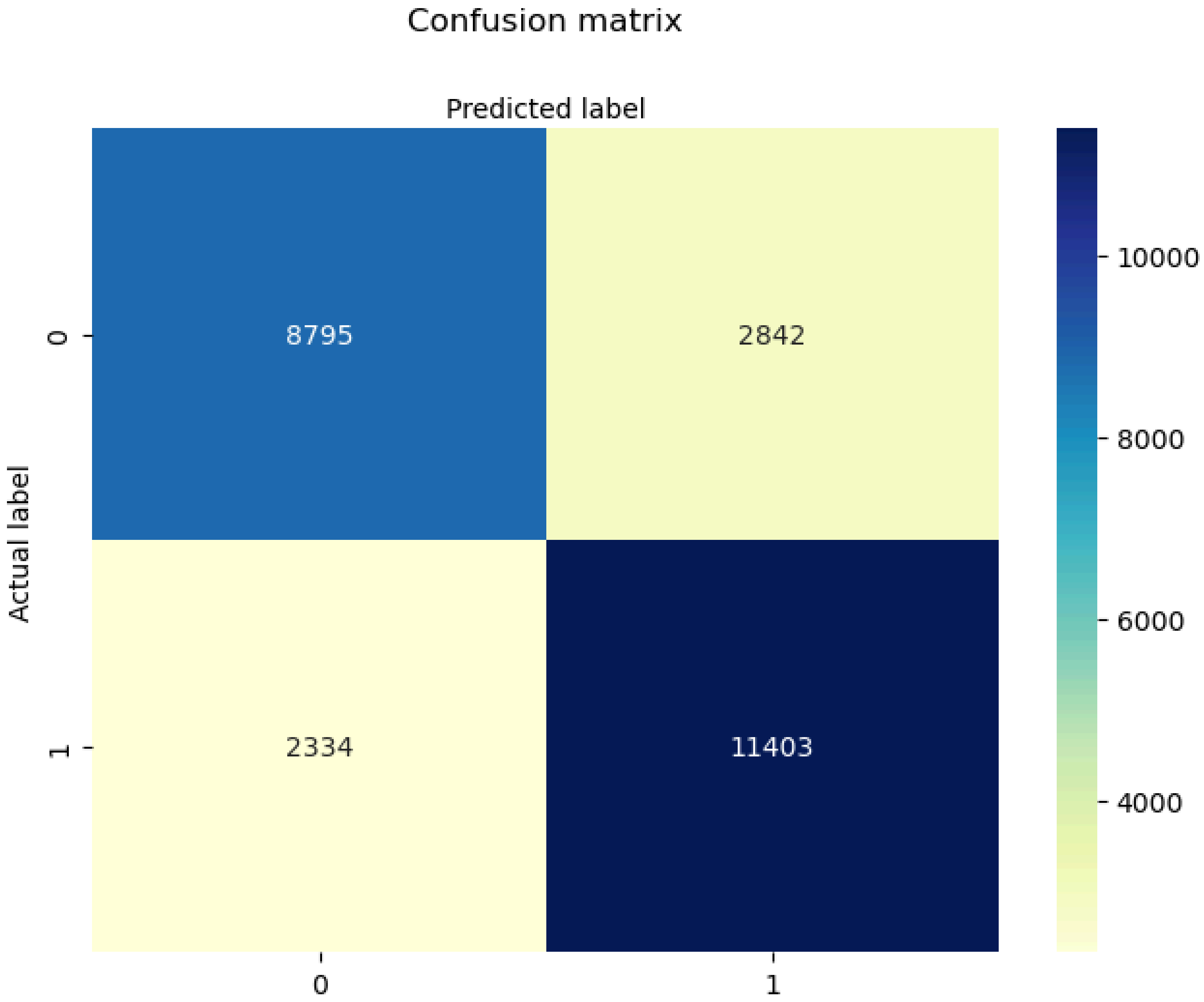
In [206…

```
#Evaluate Model (Confusion Matrix)
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[206]:
```
array([[ 8795,  2842],
       [ 2334, 11403]], dtype=int64)
```

In [207…
```
#Visualize predicted and actual values
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[207]:
```
Text(0.5, 427.9555555555555, 'Predicted label')
```



In [208…

```
#Confusion Matrix Metrics
target_names = ['Dissatisfied', 'Satisfied']
print(classification_report(y_test, y_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

Dissatisfied       0.79      0.76      0.77     11637
   Satisfied       0.80      0.83      0.82     13737

    accuracy                           0.80     25374
   macro avg       0.80      0.79      0.79     25374
weighted avg       0.80      0.80      0.80     25374
```

```
In [ ]:
```