CIND 820 Big Data Analytics Project

Name: Nelly Grillo

Student number: 501144764

Supervisor: Ceni Babaoglu, Ph.D

Level of Satisfaction of airline customers using Logistic Regression, Naïve Bayes, K-Nearest Neighbors, Decision Tree, Random Forest, and Extreme Gradient Boosting Machine Learning Algorithms

Library Imports

In [798…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, roc_auc_score, classification_report, plot_confusion_matrix, plot_roc_curve
#!pip install eli5
import eli5
from eli5.sklearn import PermutationImportance
from sklearn.model_selection import train_test_split
import time
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, plot_confusion_matrix,classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB,GaussianNB,BernoulliNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier as rf
import xgboost as xgb
from xgboost import XGBClassifier, plot_importance
from sklearn.model_selection import GridSearchCV, cross_val_score, StratifiedKFold, learning_curve
import warnings
warnings.filterwarnings('ignore')
```

In [799…

```
# Loading the dataset
```

```
df_train = pd.read_csv('train.csv', index_col='id')
df_test = pd.read_csv('test.csv')
```

Initial Analysis

Data Description

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 (Satisfied, neutral or dissatisfied) (Target variable)

In [800…
```
# Looking at first few instances in train dataset
df_train.head()
```

Out[800]:

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

5 rows × 24 columns

In [801…
```
# Looking at first few instances in test dataset
df_test.head()
```

Out[801]:

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

5 rows × 25 columns

In [802…

```
# Looking at column names in train dataset
df_train.columns
```

Out[802]:

```
Index(['Unnamed: 0', '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 [803…

```
# Looking at column names in test dataset
df_test.columns
```

Out[803]:
```
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 [804…
```
# Looking for the shape in train dataset
print("The data shape in train dataset is: {}".format(df_train.shape))
```

```
The data shape in train dataset is: (103904, 24)
```

In [805…
```
# Looking for the shape in test dataset
print("The data shape in test dataset is: {}".format(df_test.shape))
```

```
The data shape in test dataset is: (25976, 25)
```

The test dataset contains 25976 entries, and 25 columns.

Variables:

Categorical Variables: Gender, Customer Type, Type of Travel, Class, 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, Check-in service, Inflight service, Cleanliness, and Satisfaction.

Numerical Variables: Age, Flight Distance, Departure Delay in Minutes, and Arrival Delay in Minutes.

In [806…
```
categ_var = ['Gender', 'Customer Type',  'Type of Travel', 'Class', 'Inflight wifi service',
       'Departure/Arrival time convenient', 'Ease of Online booking', 'Gate location', 'Food and drink', 'Online boardi
       'Inflight entertainment', 'On-board service', 'Leg room',   'Baggage handling', 'Checkin service', 'Inflight ser
       'Cleanliness','Satisfaction']

numer_var = ['Age', 'Flight Distance', 'Departure Delay in Minutes', 'Arrival Delay in Minutes']
```

In [807…
```
# Checking data in train dataset
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 103904 entries, 70172 to 62567
Data columns (total 24 columns):
 #   Column                            Non-Null Count   Dtype  
---  ------                            --------------   -----  
 0   Unnamed: 0                        103904 non-null  int64  
 1   Gender                            103904 non-null  object 
 2   Customer Type                     103904 non-null  object 
 3   Age                               103904 non-null  int64  
 4   Type of Travel                    103904 non-null  object 
 5   Class                             103904 non-null  object 
 6   Flight Distance                   103904 non-null  int64  
 7   Inflight wifi service             103904 non-null  int64  
 8   Departure/Arrival time convenient  103904 non-null  int64  
 9   Ease of Online booking            103904 non-null  int64  
 10  Gate location                     103904 non-null  int64  
 11  Food and drink                    103904 non-null  int64  
 12  Online boarding                   103904 non-null  int64  
 13  Seat comfort                      103904 non-null  int64  
 14  Inflight entertainment            103904 non-null  int64  
 15  On-board service                  103904 non-null  int64  
 16  Leg room service                  103904 non-null  int64  
 17  Baggage handling                  103904 non-null  int64  
 18  Checkin service                   103904 non-null  int64  
 19  Inflight service                  103904 non-null  int64  
 20  Cleanliness                       103904 non-null  int64  
 21  Departure Delay in Minutes        103904 non-null  int64  
 22  Arrival Delay in Minutes          103594 non-null  float64
 23  satisfaction                      103904 non-null  object 
dtypes: float64(1), int64(18), object(5)
memory usage: 19.8+ MB
```

In [808…

```
# # Checking data in test dataset
df_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
```

Assigning the appropriate column names and dropping columns in train and test dataset

Elements renamed in 'Customer Type': 'disloyal customer' as 'First-time Customer'

Elements renamed in 'Class' column: 'Eco', and 'Eco Plus' to 'Economy'

Column 'satisfaction' corrected to 'Satisfaction'

Column 'Leg room service' renamed

Drop unnecessary 'Unnamed' column and sort by ascending 'id'

Column 'Departure Delay in Minutes' type changed to float as 'Arrival Delay in Minutes'

In [809…

```
# Rename elements in 'Customer type' column
df_train['Customer Type'] = df_train['Customer Type'].map({'Loyal Customer': 'Loyal Customer', 'disloyal Customer': 'Fi
df_test['Customer Type'] = df_test['Customer Type'].map({'Loyal Customer': 'Loyal Customer', 'disloyal Customer': 'Firs

# Rename elements in 'Class' column
df_train['Class'] = df_train['Class'].map({'Eco': 'Economy', 'Eco Plus': 'Economy', 'Business': 'Business'})
df_test['Class'] = df_test['Class'].map({'Eco': 'Economy', 'Eco Plus': 'Economy', 'Business': 'Business'})

# Column 'satisfaction'
df_train = df_train.rename(columns={'satisfaction': 'Satisfaction'})
df_test = df_test.rename(columns={'satisfaction': 'Satisfaction'})

# Column 'leg room service' renamed
df_train = df_train.rename(columns={'Leg room service': 'Leg room'})
df_test = df_test.rename(columns={'Leg room service': 'Leg room'})

# Drop "Unnamed" column and sort by 'ID'
df_train = df_train.drop('Unnamed: 0', axis=1)
df_train = df_train.sort_values('id', ascending = True)

df_test = df_test.drop('Unnamed: 0', axis=1)
df_test = df_test.drop('id', axis=1)

# Changing type of 'Departure Delay in Minutes' column to float64
df_train['Departure Delay in Minutes'] = df_train['Departure Delay in Minutes'].astype('float')
df_test['Departure Delay in Minutes'] = df_test['Departure Delay in Minutes'].astype('float')
```

In [810…

```
# Checking the changes in train dataset
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 103904 entries, 1 to 129880
Data columns (total 23 columns):
 #   Column                            Non-Null Count   Dtype  
---  ------                            --------------   -----  
 0   Gender                            103904 non-null  object 
 1   Customer Type                     103904 non-null  object 
 2   Age                               103904 non-null  int64  
 3   Type of Travel                    103904 non-null  object 
 4   Class                             103904 non-null  object 
 5   Flight Distance                   103904 non-null  int64  
 6   Inflight wifi service             103904 non-null  int64  
 7   Departure/Arrival time convenient  103904 non-null  int64  
 8   Ease of Online booking            103904 non-null  int64  
 9   Gate location                     103904 non-null  int64  
 10  Food and drink                    103904 non-null  int64  
 11  Online boarding                   103904 non-null  int64  
 12  Seat comfort                      103904 non-null  int64  
 13  Inflight entertainment            103904 non-null  int64  
 14  On-board service                  103904 non-null  int64  
 15  Leg room                          103904 non-null  int64  
 16  Baggage handling                  103904 non-null  int64  
 17  Checkin service                   103904 non-null  int64  
 18  Inflight service                  103904 non-null  int64  
 19  Cleanliness                       103904 non-null  int64  
 20  Departure Delay in Minutes        103904 non-null  float64
 21  Arrival Delay in Minutes          103594 non-null  float64
 22  Satisfaction                      103904 non-null  object 
dtypes: float64(2), int64(16), object(5)
memory usage: 19.0+ MB
```

In [811…

```
# Checking the changes in test dataset
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25976 entries, 0 to 25975
Data columns (total 23 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Gender                            25976 non-null  object 
 1   Customer Type                     25976 non-null  object 
 2   Age                               25976 non-null  int64  
 3   Type of Travel                    25976 non-null  object 
 4   Class                             25976 non-null  object 
 5   Flight Distance                   25976 non-null  int64  
 6   Inflight wifi service             25976 non-null  int64  
 7   Departure/Arrival time convenient  25976 non-null  int64  
 8   Ease of Online booking            25976 non-null  int64  
 9   Gate location                     25976 non-null  int64  
 10  Food and drink                    25976 non-null  int64  
 11  Online boarding                   25976 non-null  int64  
 12  Seat comfort                      25976 non-null  int64  
 13  Inflight entertainment            25976 non-null  int64  
 14  On-board service                  25976 non-null  int64  
 15  Leg room                          25976 non-null  int64  
 16  Baggage handling                  25976 non-null  int64  
 17  Checkin service                   25976 non-null  int64  
 18  Inflight service                  25976 non-null  int64  
 19  Cleanliness                       25976 non-null  int64  
 20  Departure Delay in Minutes        25976 non-null  float64
 21  Arrival Delay in Minutes          25893 non-null  float64
 22  Satisfaction                      25976 non-null  object 
dtypes: float64(2), int64(16), object(5)
memory usage: 4.6+ MB
```

In [812…

```
# Checking the shape of train and test data set after the changes
print("The data shape in train dataset is: {}".format(df_train.shape))
print("The data shape in test dataset is: {}".format(df_test.shape))
```

```
The data shape in train dataset is: (103904, 23)
The data shape in test dataset is: (25976, 23)
```

Exploratory Data Analysis

1. Univariate Analysis

Categorical Variables

Number of unique categorical values

In [813…
```
df_train.nunique()[:23].sort_values(ascending=False)
```

Out[813]:
```
Flight Distance                     3802
Arrival Delay in Minutes             455
Departure Delay in Minutes           446
Age                                   75
Online boarding                        6
Cleanliness                            6
Leg room                               6
On-board service                       6
Inflight entertainment                 6
Seat comfort                           6
Gate location                          6
Food and drink                         6
Inflight service                       6
Ease of Online booking                 6
Departure/Arrival time convenient      6
Inflight wifi service                  6
Checkin service                        6
Baggage handling                       5
Gender                                 2
Customer Type                          2
Class                                  2
Type of Travel                         2
Satisfaction                           2
dtype: int64
```

In [814…
```
# Wrapping the labels for the visualizations
import textwrap
def wrap_labels(ax, width, break_long_words=False):
    labels = []
    for label in ax.get_xticklabels():
        text = label.get_text()
        labels.append(textwrap.fill(text, width=width,
                      break_long_words=break_long_words))
    ax.set_xticklabels(labels, rotation=0)
```

Frequency table and Frequency Distribution of Categorical variable

Satisfaction - Categorical Variable: Ordinal - **Target variable**

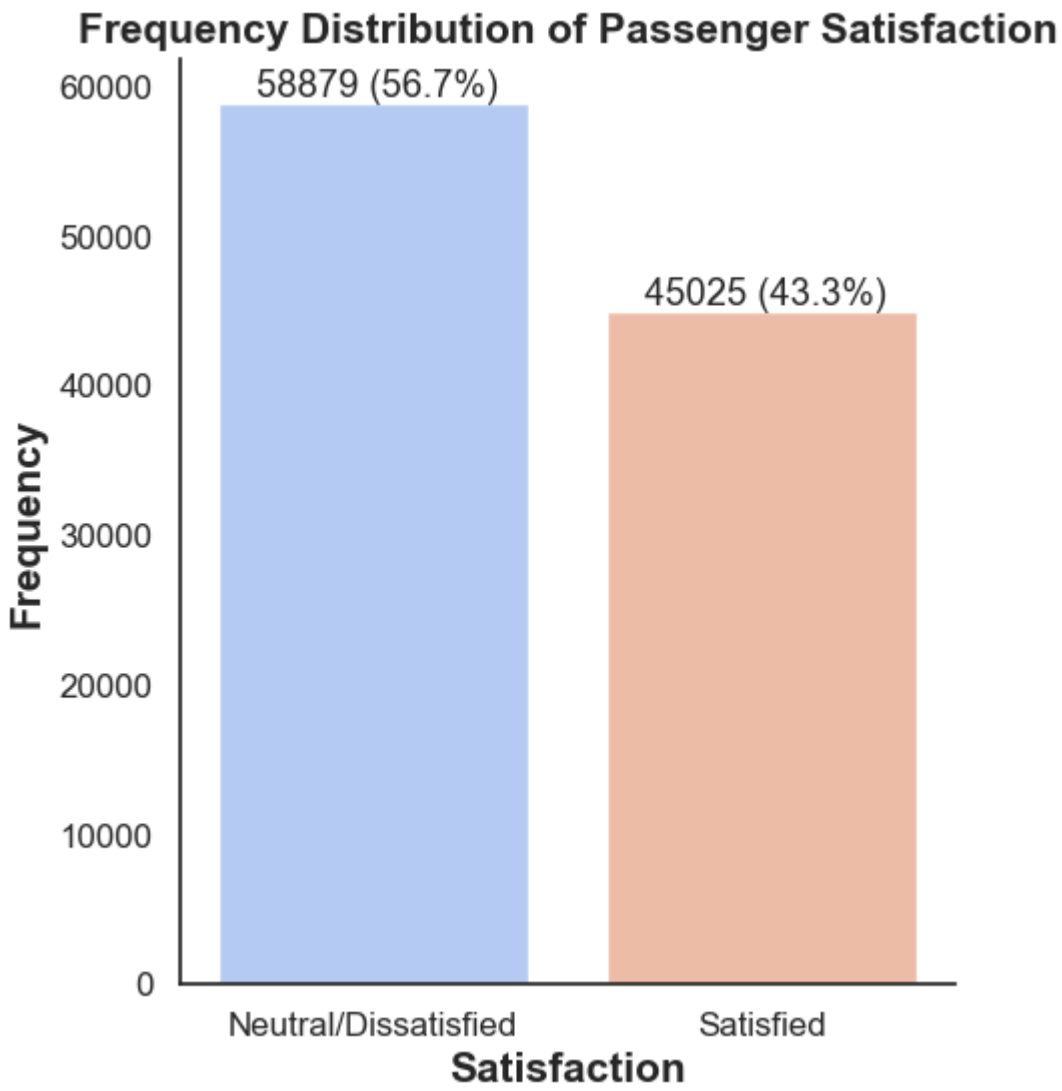
Checking for Imbalance

In [815…
```
# Table comparing each class in the target variable
s = df_train['Satisfaction']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[815]:

	Counts	Percent	%
neutral or dissatisfied	58879	0.566667	56.7%
satisfied	45025	0.433333	43.3%

In [816…
```
# Visualization of the Safisfaction distribution
sns.set(style='white',font_scale=1.1)
fig = plt.figure(figsize=[5,6])
ax = sns.countplot(data= df_train,x='Satisfaction',palette='coolwarm')
plt.title('Frequency Distribution of Passenger Satisfaction',weight='bold',fontsize='15')
ax.set_xticklabels(['Neutral/Dissatisfied','Satisfied'])
for p in ax.patches:
        ax.annotate(str(p.get_height())+' ('+str((p.get_height()/len(df_train)*100).round(1))+'%)', (p.get_x()+0.1, p.g
plt.xlabel('Satisfaction',weight='bold',fontsize='15')
plt.ylabel('Frequency',weight='bold',fontsize='15')
sns.despine()
plt.savefig('targetplot1.png',transparent=True, bbox_inches='tight')
```



The above plot shows that the data is relatively evenly distributed between the Neutral/Dissatisfied and the Satisfied passengers (57% and 43% respectively). It can be interpreted as a balanced data because there is no significant difference. The dataset will not need any special treatment/resampling.

Gender - Categorical Variable: Nominal

In [817…
```
# Gender ratio
s = df_train['Gender']
```

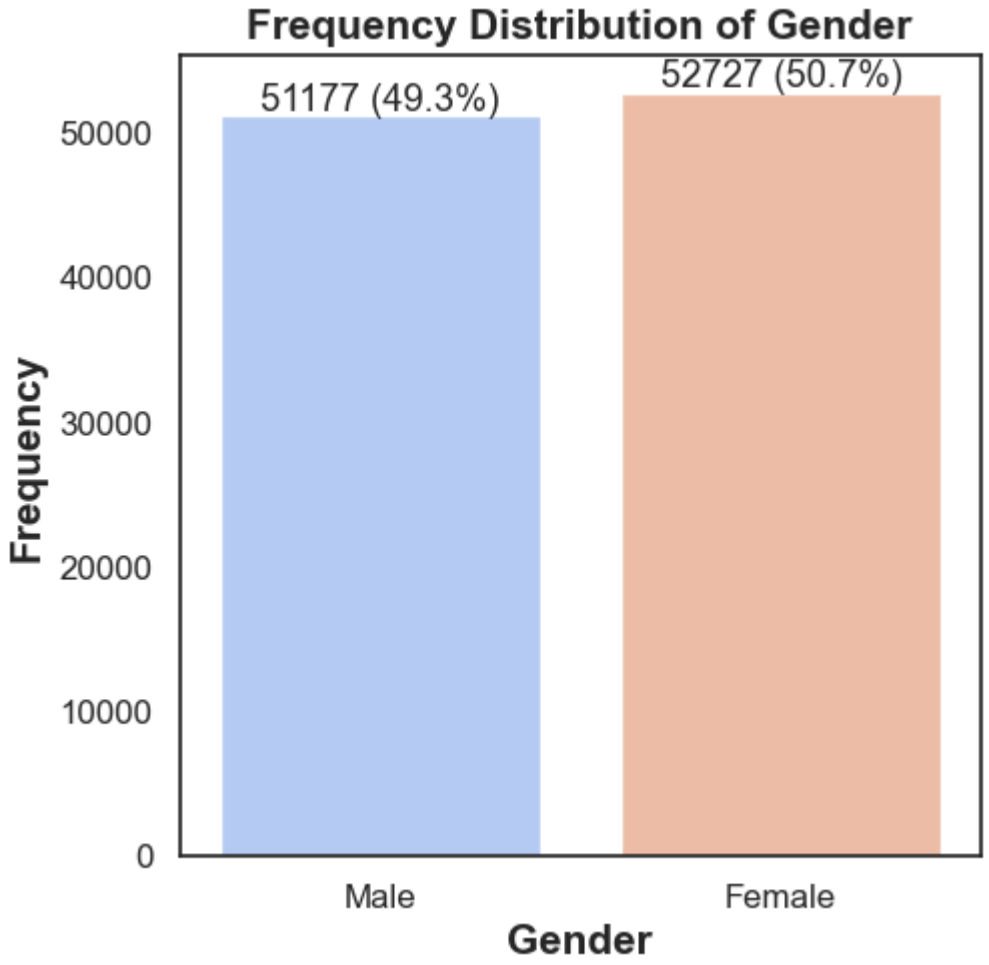
```
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[817]:

	Counts	Percent	%
Female	52727	0.507459	50.7%
Male	51177	0.492541	49.3%

In [818…

```
# Visualization of the Gender distribution
sns.set(style='white',font_scale=1.1)
fig = plt.figure(figsize=(4,4))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Gender', palette = 'coolwarm')
plt.title('Frequency Distribution of Gender',weight='bold',fontsize='15')
for p in ax.patches:
        ax.annotate(str(p.get_height())+' ('+str((p.get_height()/len(df_train)*100).round(1))+'%)', (p.get_x()+0.1, p.g
ax.set_xlabel('Gender', fontsize=15, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



This plot shows that there is no significant difference in the distribution between males and females.

Customer Type - Categorical Variable: Nominal

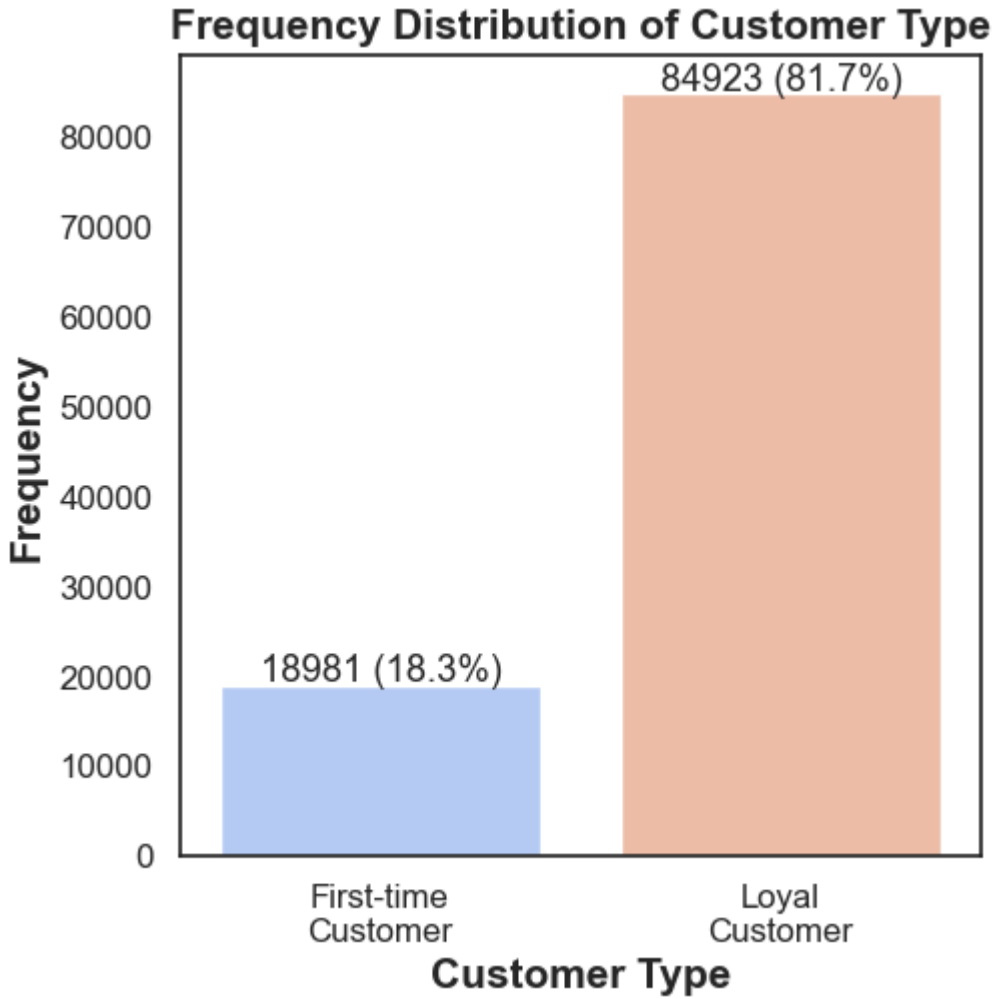
In [819…
```
# Customer Type ratio
s = df_train['Customer Type']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[819]:

	Counts	Percent	%
Loyal Customer	84923	0.817322	81.7%
First-time Customer	18981	0.182678	18.3%

In [820…

```
# Visualization of the Customer Type distribution
sns.set(style='white',font_scale=1.1)
fig = plt.figure(figsize=(4,4))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Customer Type', palette = 'coolwarm')
plt.title('Frequency Distribution of Customer Type',weight='bold',fontsize='15')
for p in ax.patches:
        ax.annotate(str(p.get_height())+' ('+str((p.get_height()/len(df_train)*100).round(1))+'%)', (p.get_x()+0.1, p.g
ax.set_xlabel('Customer Type', fontsize=15, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



The difference between First-time Customers and Loyal Customers is significant, with a large number of loyal or returning customers (around 82%).

Type of Travel - Categorical Variable: Nominal

In [821…

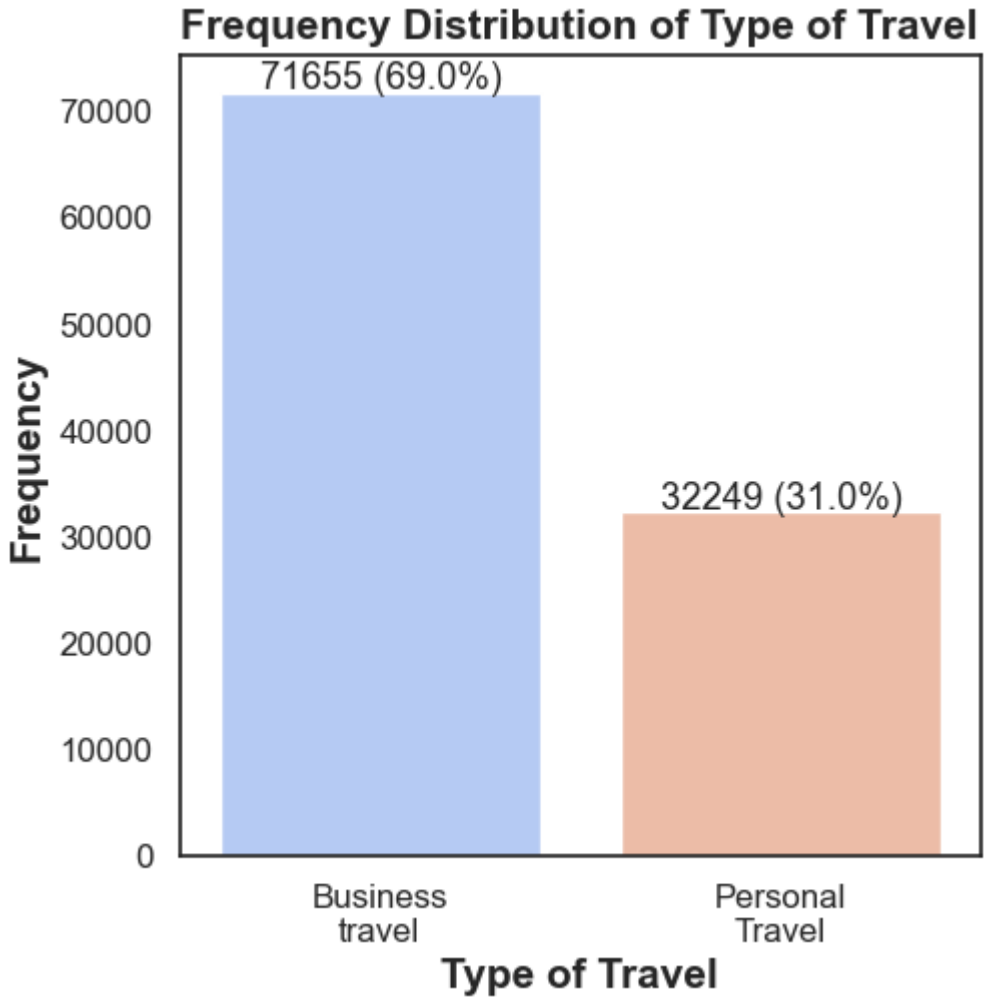
```
# Type of Travel ratio
s = df_train['Type of Travel']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[821]:

	Counts	Percent	%
Business travel	71655	0.689627	69.0%
Personal Travel	32249	0.310373	31.0%

In [822…

```
# Visualization of the Type of Travel distribution
sns.set(style='white',font_scale=1.1)
fig = plt.figure(figsize=(4,4))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Type of Travel', palette = 'coolwarm')
plt.title('Frequency Distribution of Type of Travel',weight='bold',fontsize='15')
for p in ax.patches:
        ax.annotate(str(p.get_height())+' ('+str((p.get_height()/len(df_train)*100).round(1))+'%)', (p.get_x()+0.1, p.g
ax.set_xlabel('Type of Travel', fontsize=15, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



This plot shows that there is a large number of Business customers, and the airline might focus more on this group.

Class - Categorical Variable: Nominal

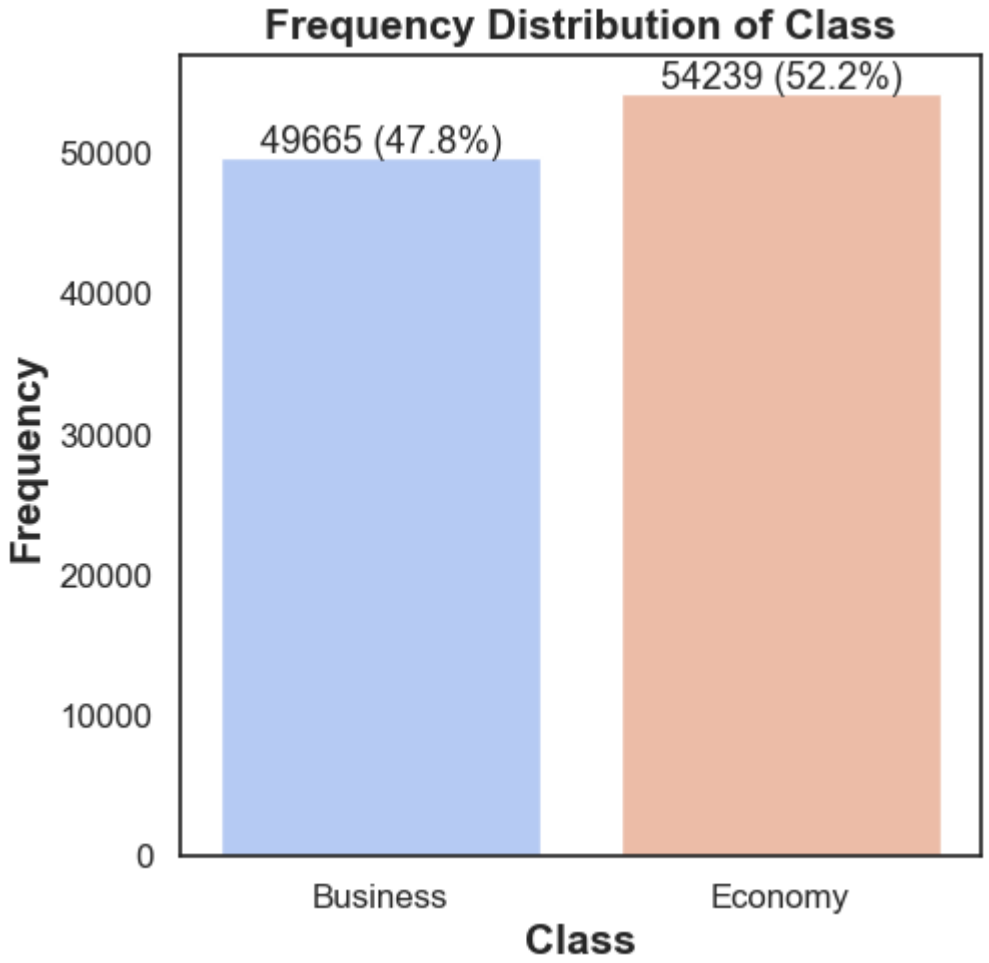
In [823…
```
# Class feature ratio
s = df_train['Class']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[823]:

	Counts	Percent	%
Economy	54239	0.522011	52.2%
Business	49665	0.477989	47.8%

In [824…

```
# Visualization of the Class distribution
sns.set(style='white',font_scale=1.1)
fig = plt.figure(figsize=(4,4))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Class', palette = 'coolwarm')
plt.title('Frequency Distribution of Class',weight='bold',fontsize='15')
for p in ax.patches:
        ax.annotate(str(p.get_height())+' ('+str((p.get_height()/len(df_train)*100).round(1))+'%)', (p.get_x()+0.1, p.g
ax.set_xlabel('Class', fontsize=15, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



This plot shows that the Economy class is the predominant after combining the Eco and Eco Plus clases, but the difference is small. Although, the Business Type of Travel was higher, there were lower customers in Business Class (approx. 48%).

Inflight wifi service - Categorical Variable: Ordinal

In [825…

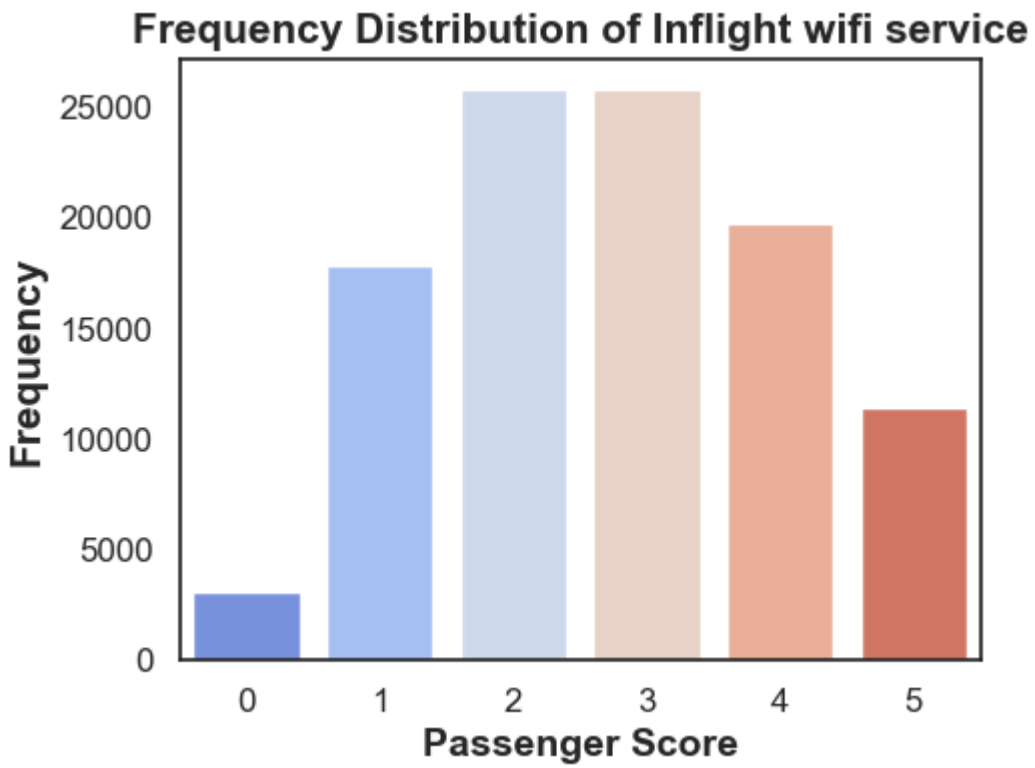
```
# Inflight wifi service ratio
s = df_train['Inflight wifi service']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[825]:

	Counts	Percent	%
3	25868	0.248961	24.9%
2	25830	0.248595	24.9%
4	19794	0.190503	19.1%
1	17840	0.171697	17.2%
5	11469	0.110381	11.0%
0	3103	0.029864	3.0%

In [826…

```
# Visualization of Inflight wifi service feature
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Inflight wifi service', palette = 'coolwarm')
plt.title('Frequency Distribution of Inflight wifi service',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=14, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



Departure/Arrival time convenient - Categorical Variable: Ordinal

In [827…

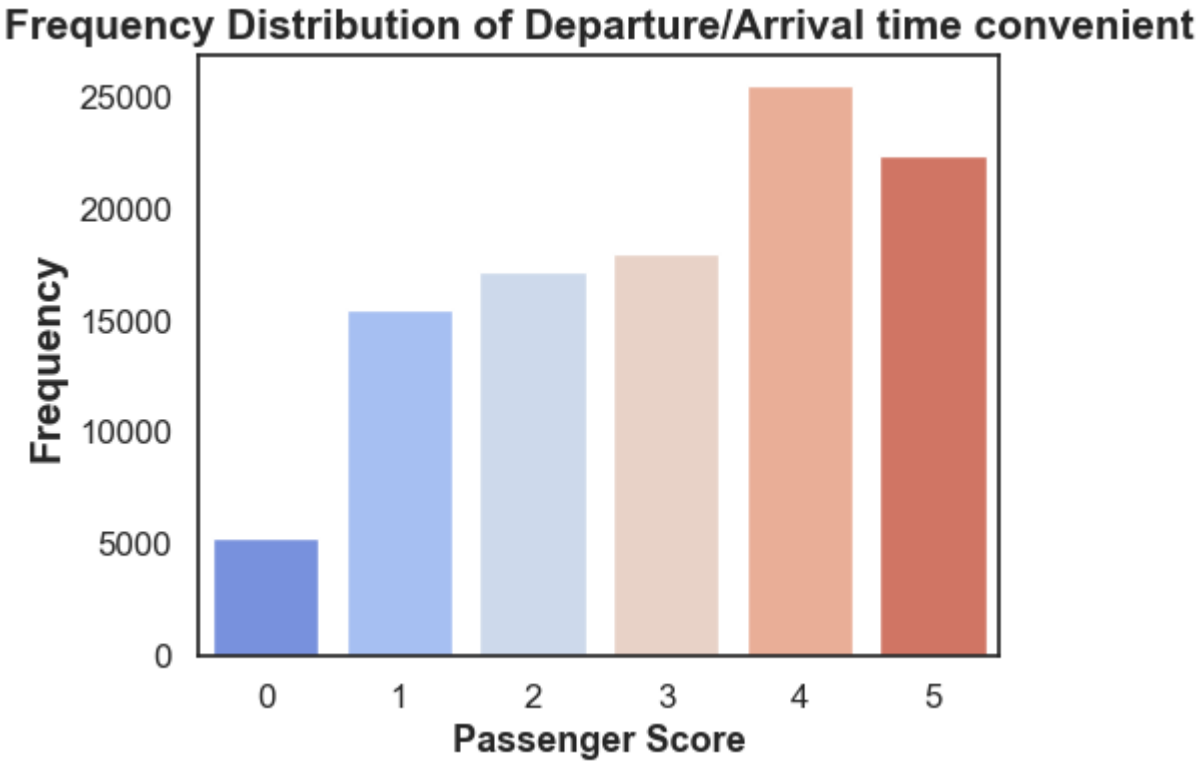
```
# Departure/Arrival time convenient ratio
s = df_train['Departure/Arrival time convenient']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[827]:

	Counts	Percent	%
4	25546	0.245862	24.6%
5	22403	0.215612	21.6%
3	17966	0.172910	17.3%
2	17191	0.165451	16.5%
1	15498	0.149157	14.9%
0	5300	0.051009	5.1%

In [828…

```
# Visualization of Departure/Arrival time convenient feature
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Departure/Arrival time convenient', palette = 'coolwarm')
plt.title('Frequency Distribution of Departure/Arrival time convenient',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=13, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



Ease of Online booking - Categorical Variable: Ordinal

In [829…

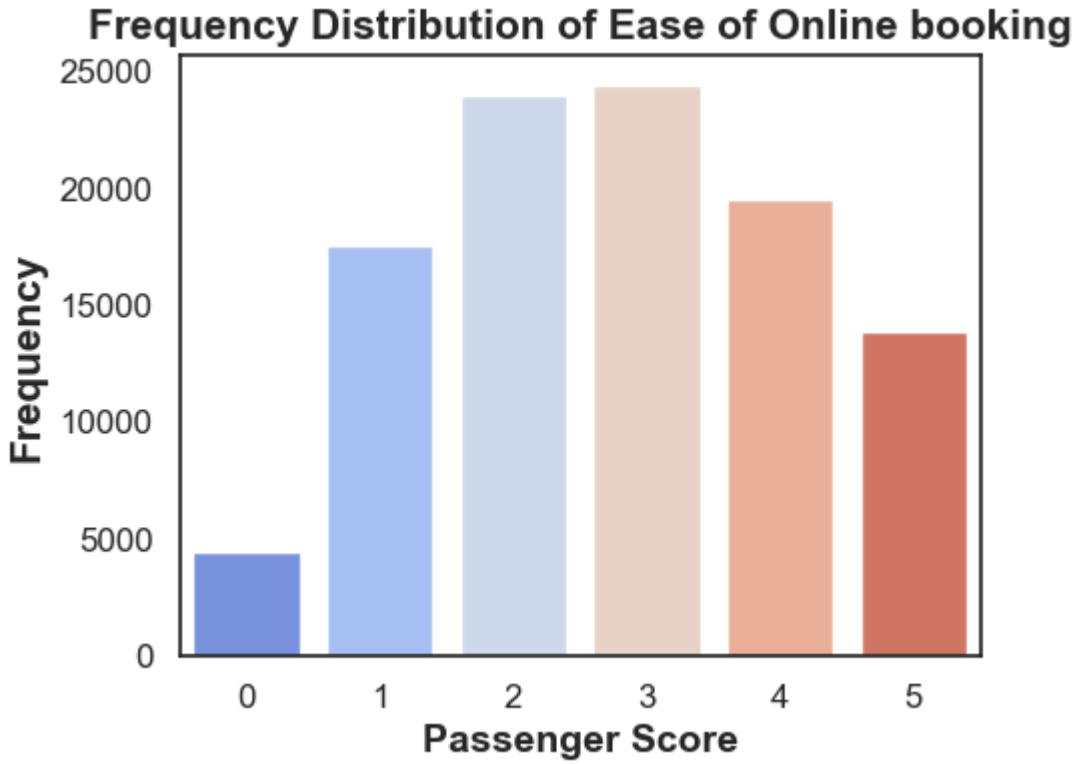
```
# Ease of Online booking ratio
s = df_train['Ease of Online booking']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[829]:

	Counts	Percent	%
3	24449	0.235304	23.5%
2	24021	0.231185	23.1%
4	19571	0.188357	18.8%
1	17525	0.168665	16.9%
5	13851	0.133306	13.3%
0	4487	0.043184	4.3%

In [830…

```
# Visualization of Ease of Online booking feature
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Ease of Online booking', palette = 'coolwarm')
plt.title('Frequency Distribution of Ease of Online booking',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=14, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



Gate location - Categorical Variable: Ordinal

In [831…

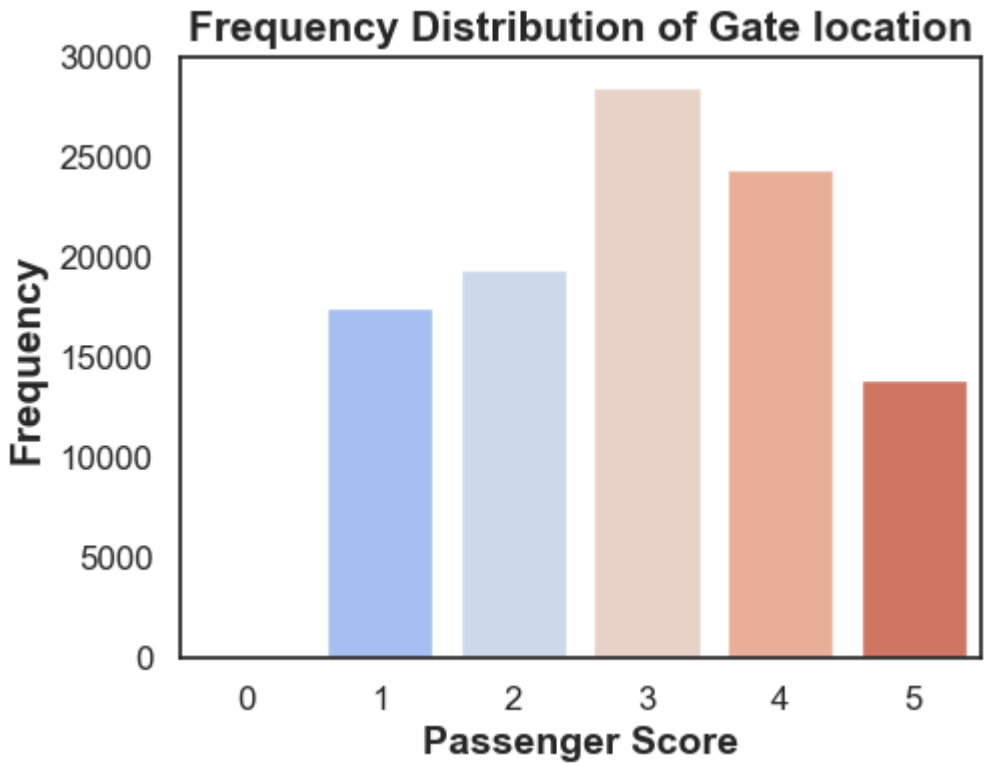
```
# Gate location ratio
s = df_train['Gate location']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[831]:

	Counts	Percent	%
3	28577	0.275033	27.5%
4	24426	0.235082	23.5%
2	19459	0.187279	18.7%
1	17562	0.169021	16.9%
5	13879	0.133575	13.4%
0	1	0.000010	0.0%

In [832…

```
# Visualization of Gate location feature
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Gate location', palette = 'coolwarm')
plt.title('Frequency Distribution of Gate location',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=14, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



Food and drink - Categorical Variable: Ordinal

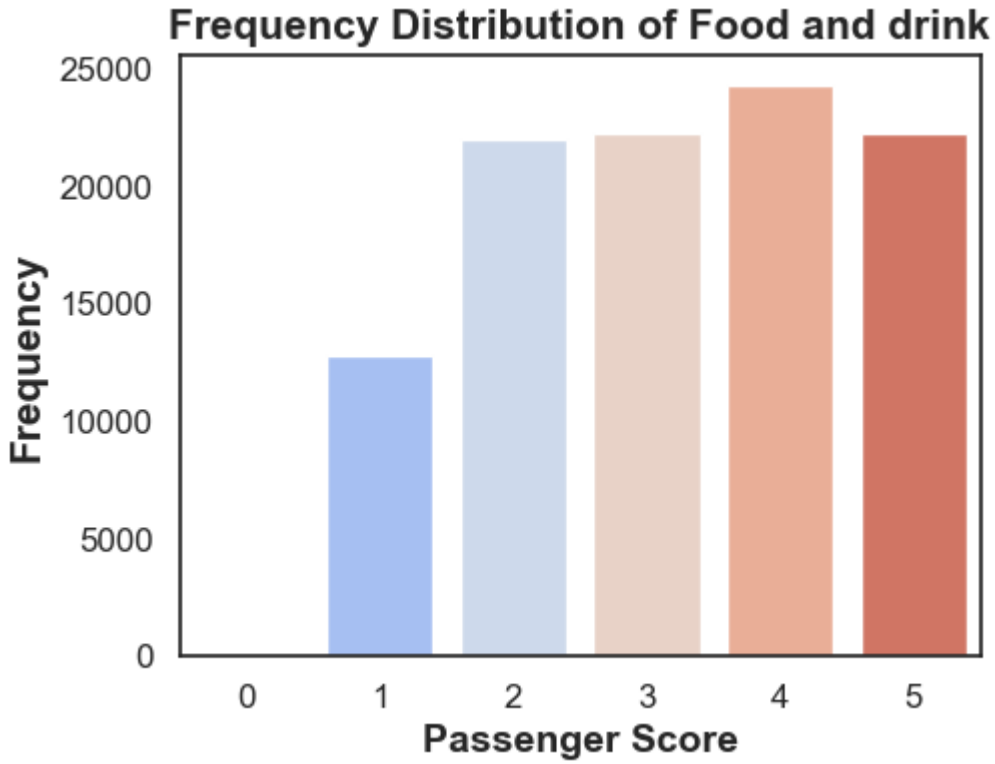
In [833…
```
# Food and drink ratio
s = df_train['Food and drink']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[833]:

	Counts	Percent	%
4	24359	0.234438	23.4%
5	22313	0.214746	21.5%
3	22300	0.214621	21.5%
2	21988	0.211618	21.2%
1	12837	0.123547	12.4%
0	107	0.001030	0.1%

In [834…

```
# Visualization of Food and drink feature
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Food and drink', palette = 'coolwarm')
plt.title('Frequency Distribution of Food and drink',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=14, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



Online boarding - Categorical Variable: Ordinal

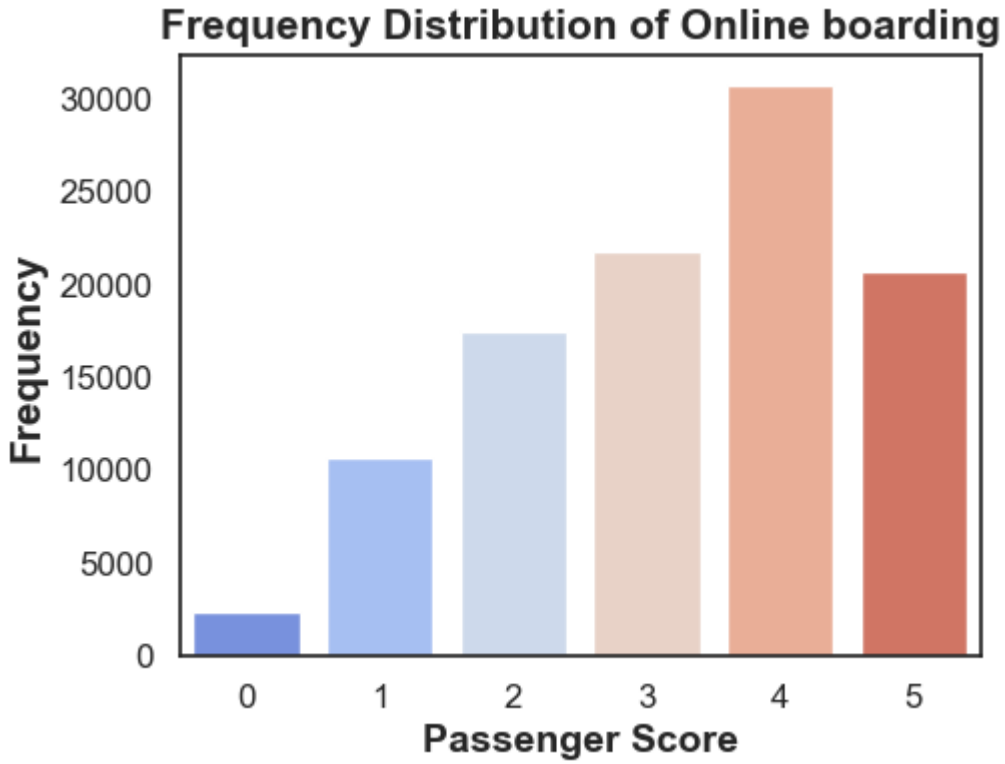
In [835…
```
# Online boarding ratio
s = df_train['Online boarding']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[835]:

	Counts	Percent	%
4	30762	0.296062	29.6%
3	21804	0.209848	21.0%
5	20713	0.199347	19.9%
2	17505	0.168473	16.8%
1	10692	0.102903	10.3%
0	2428	0.023368	2.3%

In [836…

```
# Visualization of Online boarding feature
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Online boarding', palette = 'coolwarm')
plt.title('Frequency Distribution of Online boarding',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=14, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



Seat comfort - Categorical Variable: Ordinal

In [837…

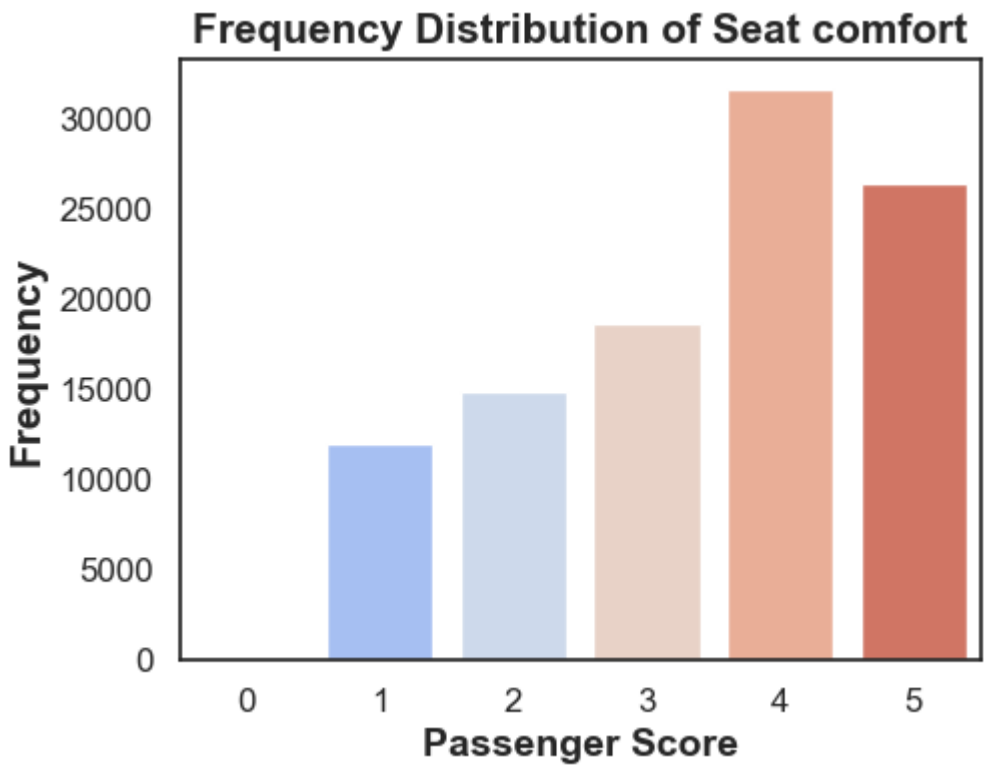
```
# Seat Comfort
s = df_train['Seat comfort']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[837]:

	Counts	Percent	%
4	31765	0.305715	30.6%
5	26470	0.254754	25.5%
3	18696	0.179935	18.0%
2	14897	0.143373	14.3%
1	12075	0.116213	11.6%
0	1	0.000010	0.0%

In [838…

```
# Visualization of Seat comfort feature
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Seat comfort', palette = 'coolwarm')
plt.title('Frequency Distribution of Seat comfort',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=14, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



Inflight entertainment - Categorical Variable: Ordinal

In [839…

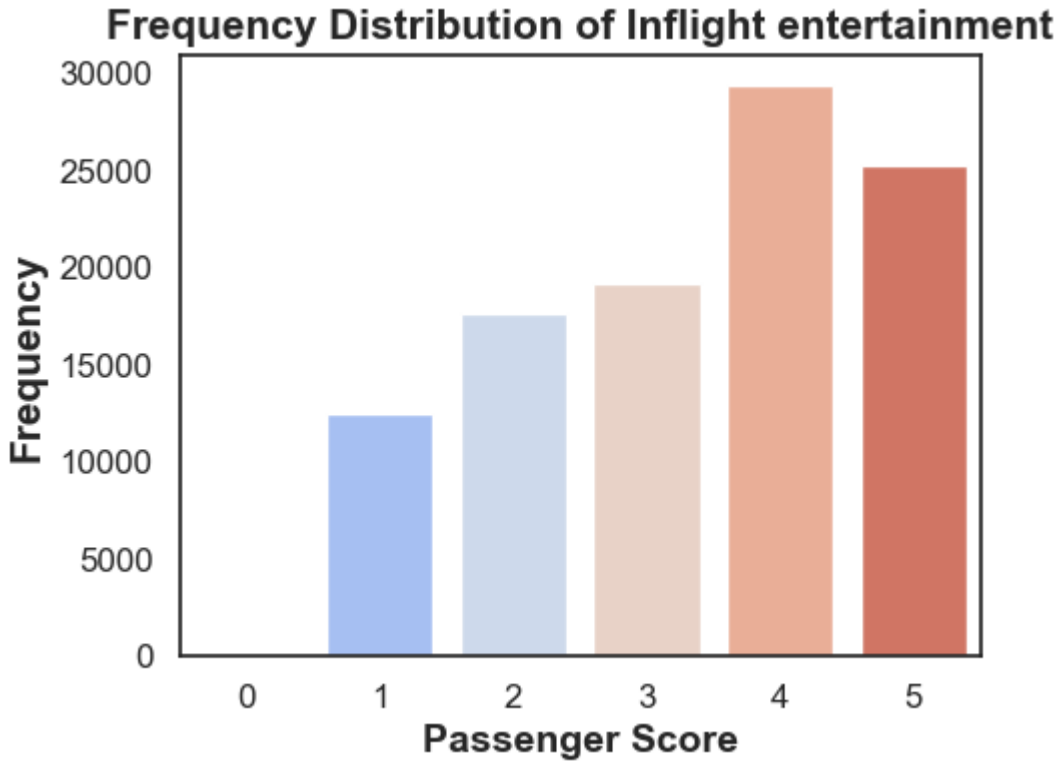
```
# Inflight entertainment ratio
s = df_train['Inflight entertainment']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[839]:

	Counts	Percent	%
4	29423	0.283175	28.3%
5	25213	0.242657	24.3%
3	19139	0.184199	18.4%
2	17637	0.169743	17.0%
1	12478	0.120092	12.0%
0	14	0.000135	0.0%

In [840…

```
# Visualization of Inflight entertainment feature
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Inflight entertainment', palette = 'coolwarm')
plt.title('Frequency Distribution of Inflight entertainment',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=14, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



On-board service - Categorical Variable: Ordinal

In [841…

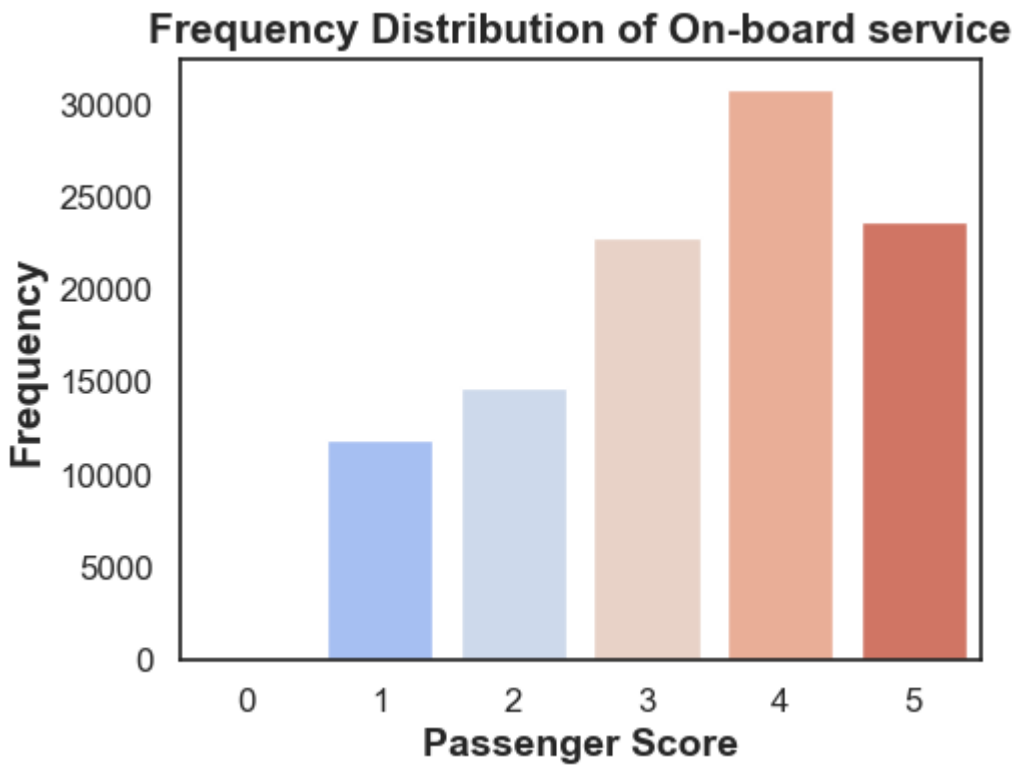
```
# On-board service ratio
s = df_train['On-board service']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[841]:

	Counts	Percent	%
4	30867	0.297072	29.7%
5	23648	0.227595	22.8%
3	22833	0.219751	22.0%
2	14681	0.141294	14.1%
1	11872	0.114259	11.4%
0	3	0.000029	0.0%

In [842…

```
# Visualization of On-board service feature
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'On-board service', palette = 'coolwarm')
plt.title('Frequency Distribution of On-board service',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=14, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



Leg room - Categorical Variable: Ordinal

In [843…

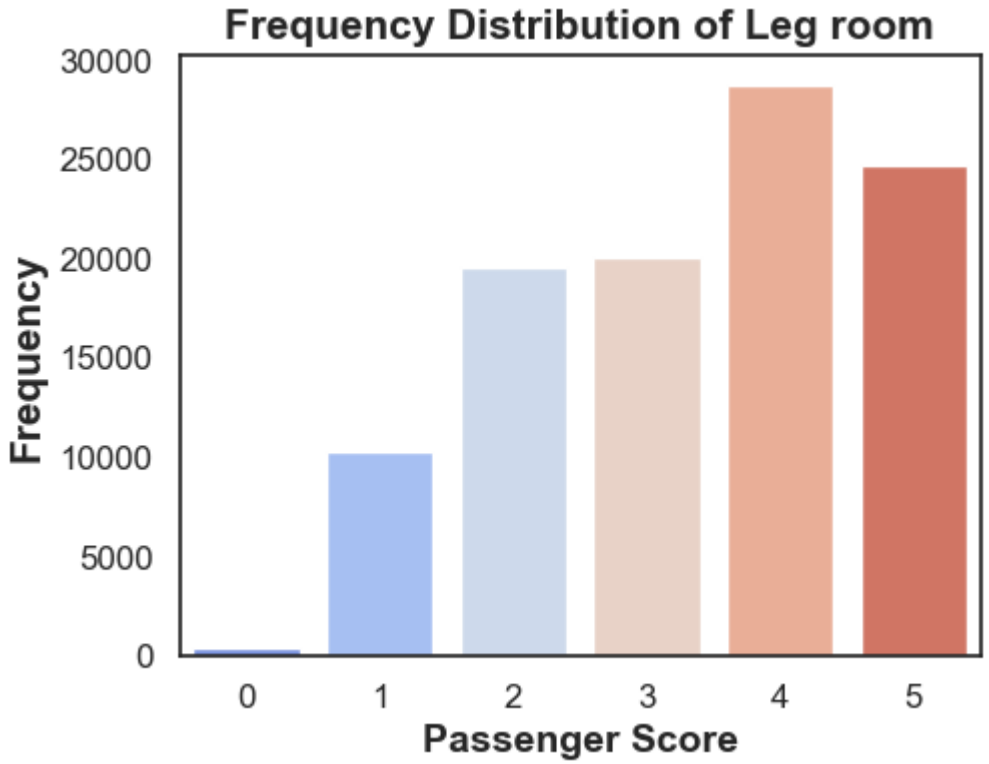
```
# leg room ratio
s = df_train['Leg room']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[843]:

	Counts	Percent	%
4	28789	0.277073	27.7%
5	24667	0.237402	23.7%
3	20098	0.193429	19.3%
2	19525	0.187914	18.8%
1	10353	0.099640	10.0%
0	472	0.004543	0.5%

In [844…

```
# Visualization of Leg room feature
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Leg room', palette = 'coolwarm')
plt.title('Frequency Distribution of Leg room',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=14, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



Baggage handling - Categorical Variable: Ordinal

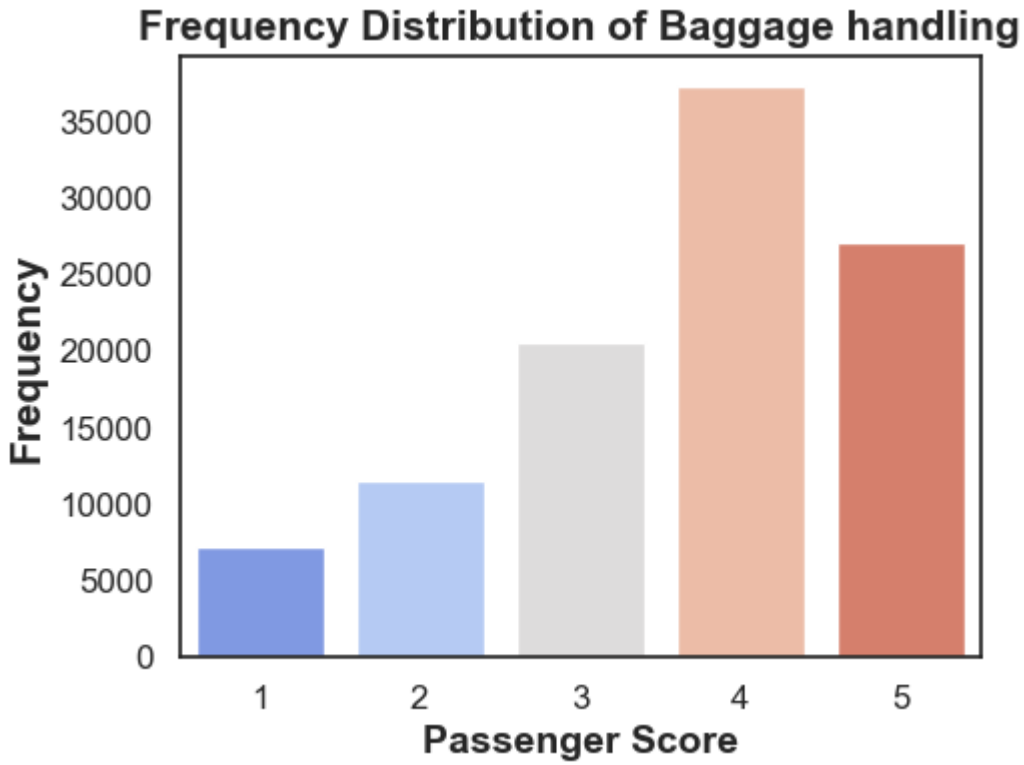
In [845…
```
# Baggage handling ratio
s = df_train['Baggage handling']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[845]:

	Counts	Percent	%
4	37383	0.359784	36.0%
5	27131	0.261116	26.1%
3	20632	0.198568	19.9%
2	11521	0.110881	11.1%
1	7237	0.069651	7.0%

In [846…

```
# Visualization of Baggage handling feature
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Baggage handling', palette = 'coolwarm')
plt.title('Frequency Distribution of Baggage handling',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=14, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



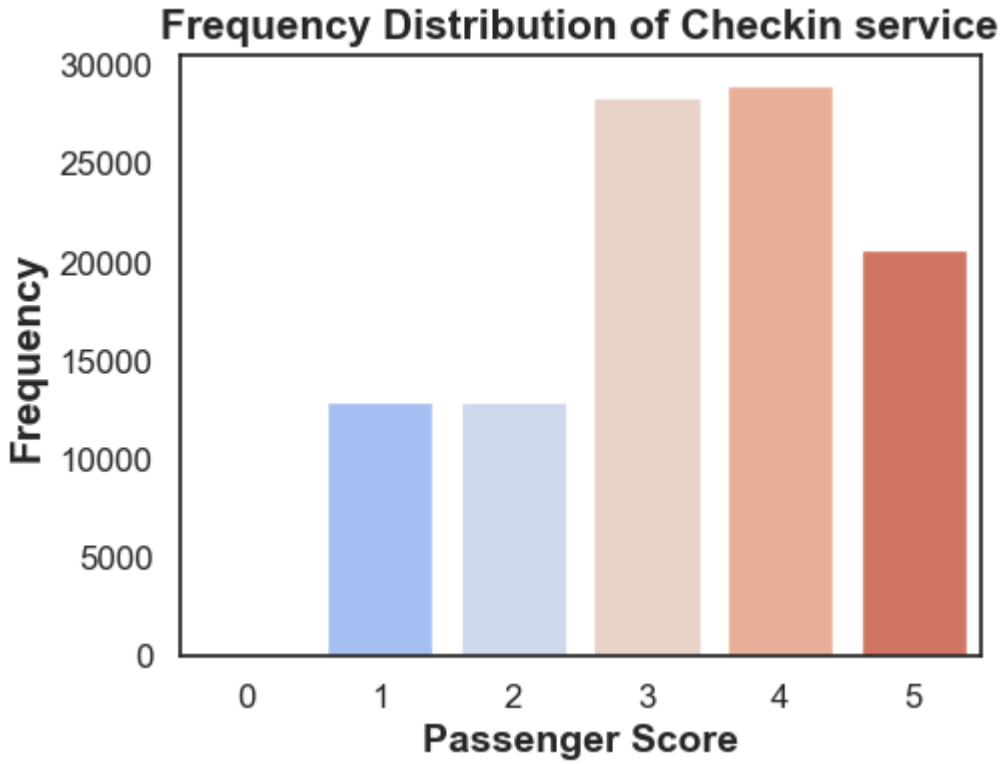
Check-in service - Categorical Variable: Ordinal

In [847…
```
# Check-in service ratio
s = df_train['Checkin service']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[847]:

	Counts	Percent	%
4	29055	0.279633	28.0%
3	28446	0.273772	27.4%
5	20619	0.198443	19.8%
2	12893	0.124086	12.4%
1	12890	0.124057	12.4%
0	1	0.000010	0.0%

In [848…
```
# Visualization of Checkin service feature
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Checkin service', palette = 'coolwarm')
plt.title('Frequency Distribution of Checkin service',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=14, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



Inflight service - Categorical Variable: Ordinal

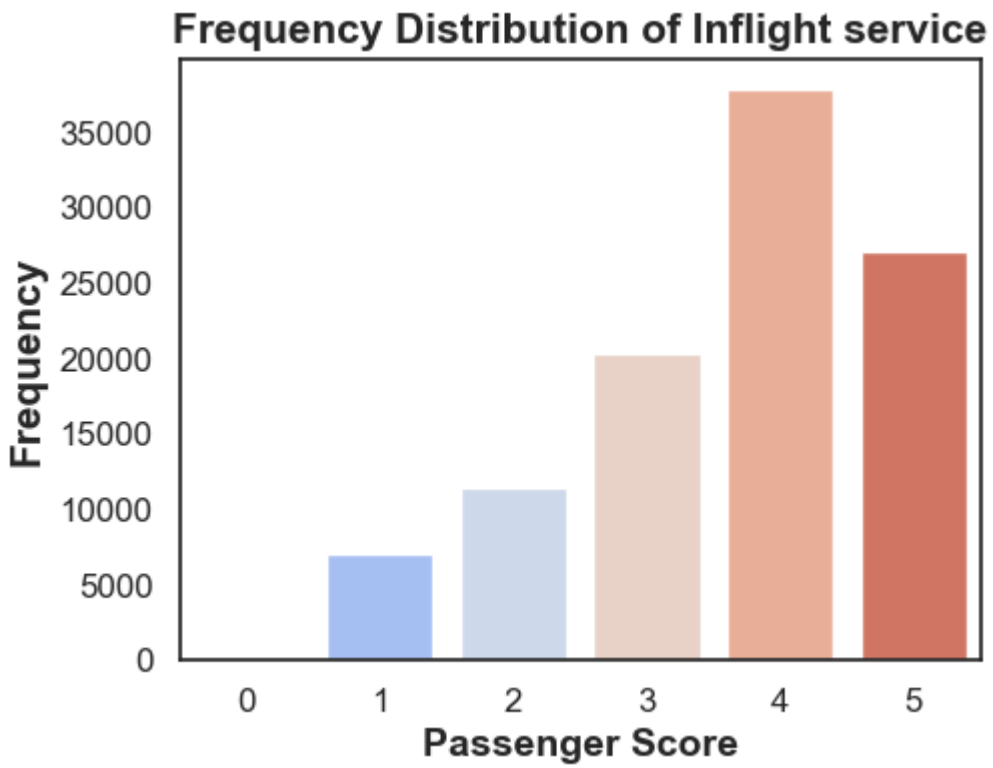
In [849…
```
# Inflight service ratio
s = df_train['Inflight service']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[849]:

	Counts	Percent	%
4	37945	0.365193	36.5%
5	27116	0.260972	26.1%
3	20299	0.195363	19.5%
2	11457	0.110265	11.0%
1	7084	0.068178	6.8%
0	3	0.000029	0.0%

In [850…

```
# Visualization of Inflight service
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Inflight service', palette = 'coolwarm')
plt.title('Frequency Distribution of Inflight service',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=14, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



Cleanliness - Categorical Variable: Ordinal

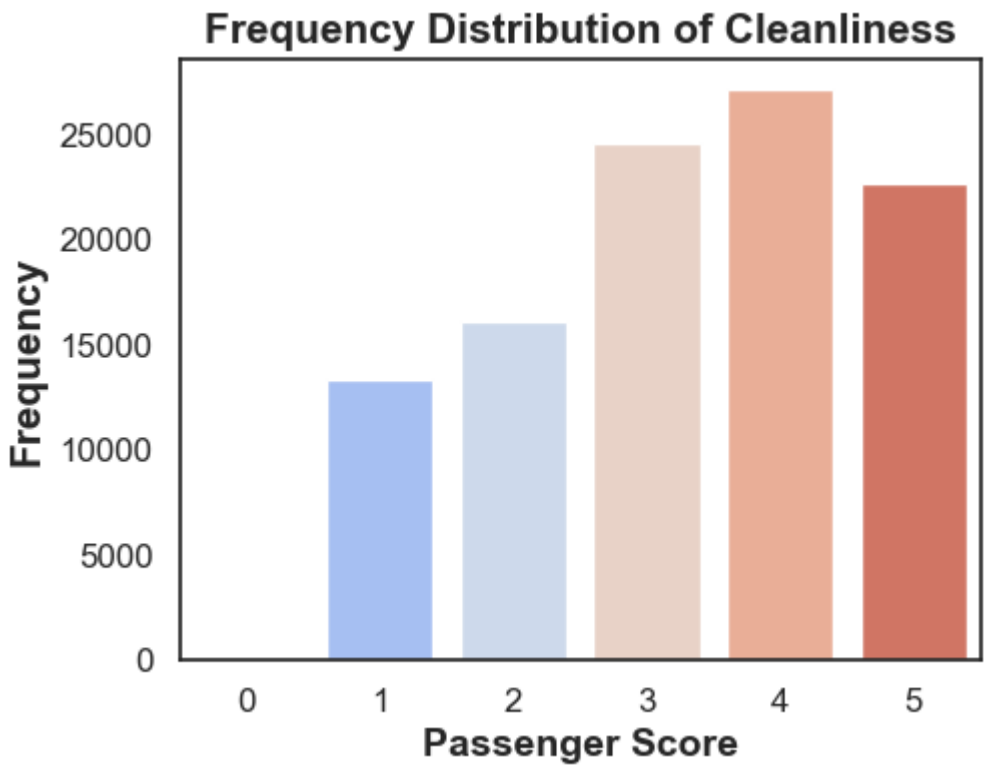
In [851…
```
# Cleanliness ratio
s = df_train['Cleanliness']
counts = s.value_counts()
percent = s.value_counts(normalize=True)
percent100 = s.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'Counts': counts, 'Percent': percent, '%': percent100})
```

Out[851]:

	Counts	Percent	%
4	27179	0.261578	26.2%
3	24574	0.236507	23.7%
5	22689	0.218365	21.8%
2	16132	0.155259	15.5%
1	13318	0.128176	12.8%
0	12	0.000115	0.0%

In [852…

```
# Visualization of Cleanliness feature
fig = plt.figure(figsize=(4,3))
ax = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Cleanliness', palette = 'coolwarm')
plt.title('Frequency Distribution of Cleanliness',weight='bold',fontsize='15')
ax.set_xlabel('Passenger Score', fontsize=14, weight='semibold')
ax.set_ylabel('Frequency', fontsize=15, weight='semibold')
wrap_labels(ax, 10)
```



Numerical Variables

Descriptive Statistics

In [853…
```
df_train[numer_var].describe(include='all').T
```

Out[853]:

	count	mean	std	min	25%	50%	75%	max
Age	103904.0	39.379706	15.114964	7.0	27.0	40.0	51.0	85.0
Flight Distance	103904.0	1189.448375	997.147281	31.0	414.0	843.0	1743.0	4983.0
Departure Delay in Minutes	103904.0	14.815618	38.230901	0.0	0.0	0.0	12.0	1592.0
Arrival Delay in Minutes	103594.0	15.178678	38.698682	0.0	0.0	0.0	13.0	1584.0

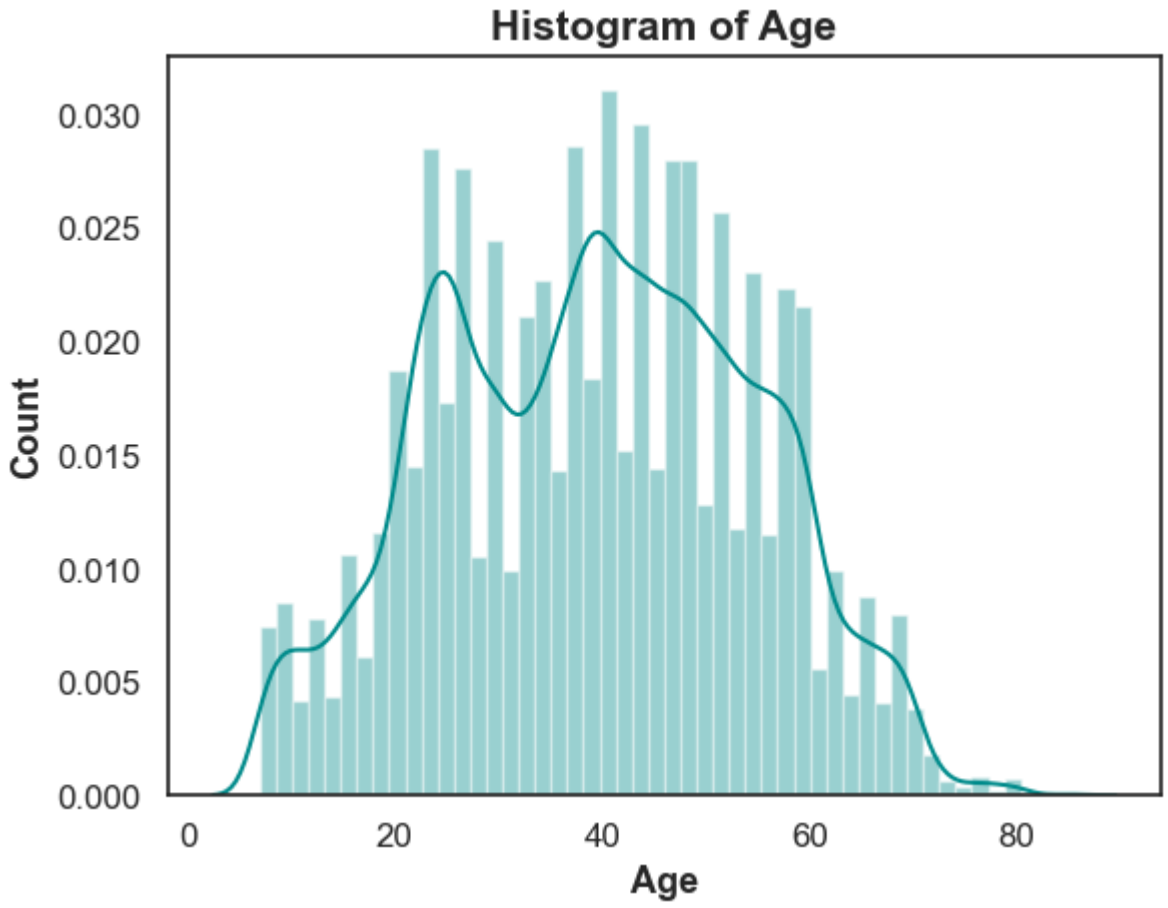
Departure and Arrival Delay features show large values: 1592 and 1584 respectively. This will be assessed when detecting and treating

Outliers in the dataset.

Histogram of Numerical Variables

Age - Numerical Variable

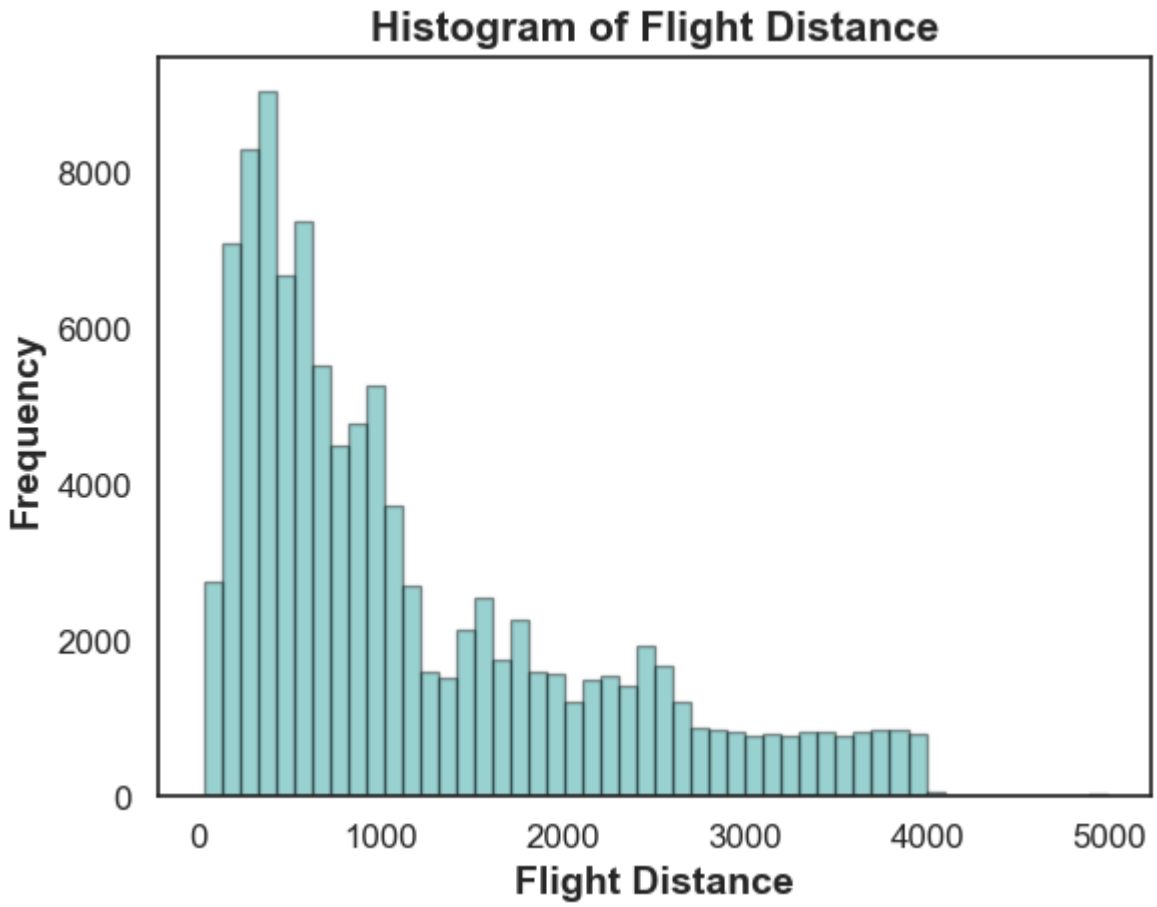
In [854…
```
# Histogram of Age feature
sns.distplot(df_train['Age'], hist = True, bins = 50, color = 'darkcyan')
plt.title('Histogram of Age', fontsize=15, weight='semibold')
plt.xlabel('Age', fontsize=13, weight='semibold')
plt.ylabel('Count', fontsize=13, weight='semibold')
plt.show()
```



Flight Distance - Numerical Variable

In [855…
```
# Histogram of Flight Distance feature
sns.distplot(df_train['Flight Distance'], hist=True, kde=False,
             bins=(50), color= 'darkcyan',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth: 4'})

plt.title('Histogram of Flight Distance', fontsize=15, weight='semibold')
plt.xlabel('Flight Distance', fontsize=14, weight='semibold')
plt.ylabel('Frequency', fontsize=14, weight='semibold')
```

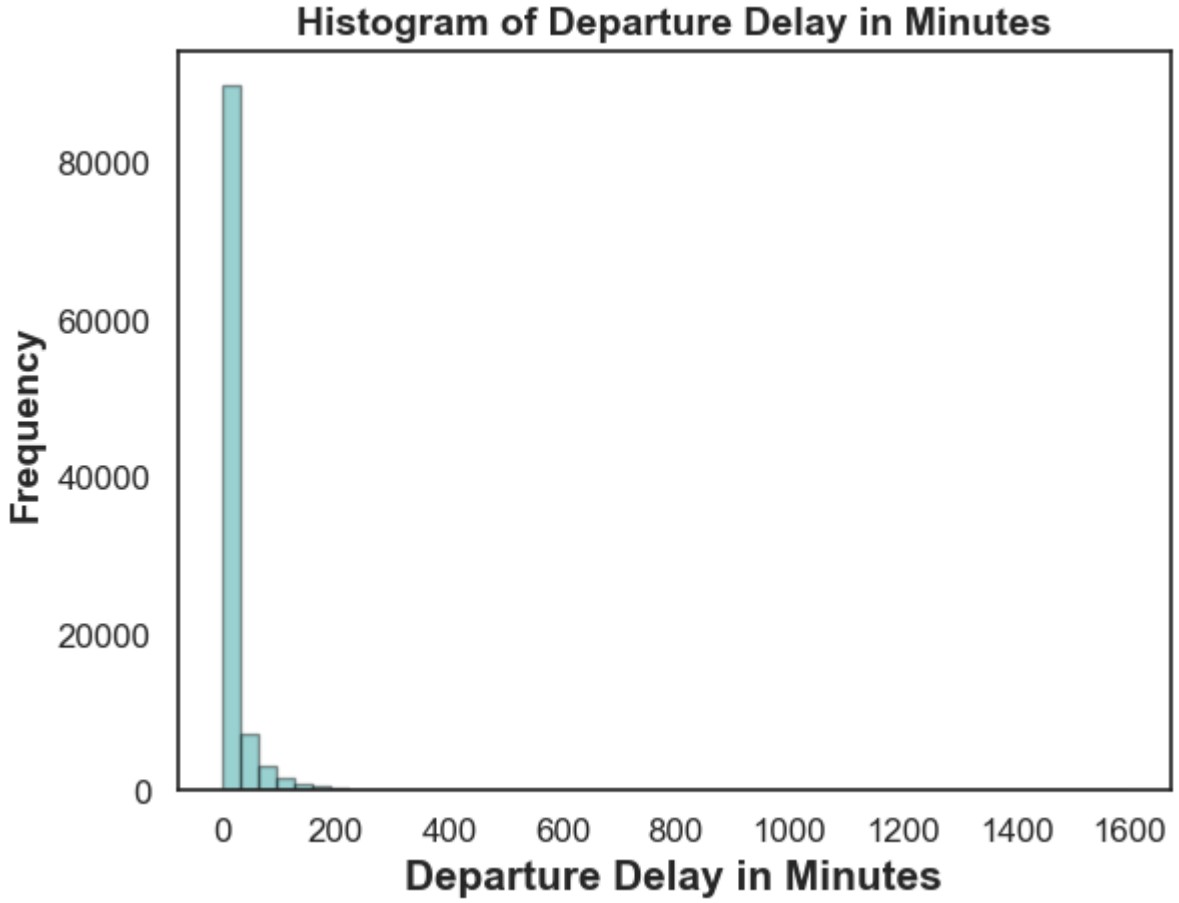
Out[855]:
```
Text(0, 0.5, 'Frequency')
```



This right-skewed histogram shows that most flights possibly are domestics with a flight distance under 1000 miles.

Departure Delay in Minutes - Numerical Variable

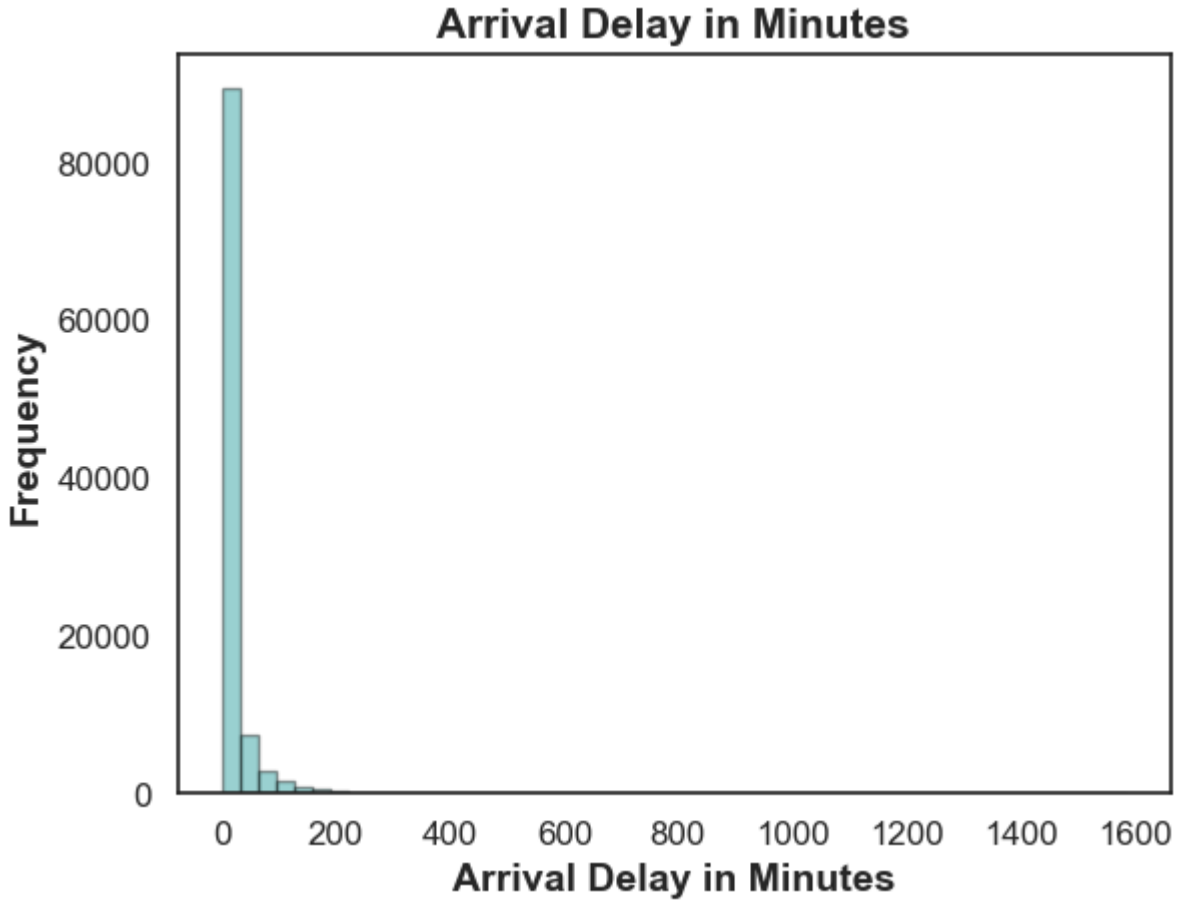
In [856…
```
# Histogram of Departure Delay in Minutes feature
sns.distplot(df_train['Departure Delay in Minutes'], hist=True, kde=False,
             bins=(50), color= 'darkcyan',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth: 4'})
axd = fig.add_axes([0,0,1,1])
plt.title('Histogram of Departure Delay in Minutes', fontsize=14, weight='semibold')
plt.xlabel('Departure Delay in Minutes', fontsize=15, weight='semibold')
plt.ylabel('Frequency', fontsize=14, weight='semibold')
wrap_labels(axd, 10)
```



Arrival Delay in Minutes - Numerical Variable

In [857…

```
# Histogram of Arrival Delay in Minutes
sns.distplot(df_train['Arrival Delay in Minutes'], hist=True, kde=False,
             bins=(50), color= 'darkcyan',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth: 4'})
ax_a = fig.add_axes([0,0,1,1])
plt.title('Arrival Delay in Minutes', fontsize=15, weight='semibold')
plt.xlabel('Arrival Delay in Minutes', fontsize=14, weight='semibold')
plt.ylabel('Frequency', fontsize=14, weight='semibold')
wrap_labels(ax_a, 10)
```



Data Cleaning

Missing Data - NaN Values

In [858…
```
# Check for missing values in train dataset
df_train.isnull().sum()
```

Out[858]:
```
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                               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
```

There are 310 missing values in the Arrival Delay in Minutes variable.

In [859…
```
# Check for missing values in test dataset
df_test.isnull().sum()
```

Out[859]:
```
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                              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
```

There are 83 missing values in the Arrival Delay in Minutes variable.

The missing values will be filled with the average Arrival Delay time.

In [860…
```
# Mean of 'Arrival Delay in Minutes' in train dataset
np.mean(df_train['Arrival Delay in Minutes'])
print("The average of Arrival Delay in Minutes in train dataset is: {}".format(np.mean(df_train['Arrival Delay in Minut
```

```
The average of Arrival Delay in Minutes in train dataset is: 15.178678301832152
```

In [861…
```
# Mean of 'Arrival Delay in Minutes' in test dataset
print("The average of Arrival Delay in Minutes in test dataset is: {}".format(np.mean(df_test['Arrival Delay in Minutes
```

```
The average of Arrival Delay in Minutes in test dataset is: 14.74085660217047
```

In [862…
```
# Fill the missing values with the average of the Arrival Delay in Minutes in the train dataset
df_train['Arrival Delay in Minutes'] = df_train['Arrival Delay in Minutes'].fillna(np.mean(df_train['Arrival Delay in M
```

In [863…
```
# Check for missing values in the train dataset
```

```
df_train[df_train['Arrival Delay in Minutes'].isnull()]
```

Out[863]:

id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	Inflight entertainment	On-board service	Leg room	Bagg handl

0 rows × 23 columns

In [864…
```
# Fill the missing values with the average of the Arrival Delay in Minutes in the test dataset
df_test['Arrival Delay in Minutes'] = df_test['Arrival Delay in Minutes'].fillna(np.mean(df_test['Arrival Delay in Minu
```

In [865…
```
# Check for missing values in the test dataset
df_test[df_test['Arrival Delay in Minutes'].isnull()]
```

Out[865]:

id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	Inflight entertainment	On-board service	Leg room	Baggag handlin

0 rows × 23 columns

In [866…
```
# Looking for the shape in train dataset
print("The data shape in train dataset is: {}".format(df_train.shape))
```

```
The data shape in train dataset is: (103904, 23)
```

In [867…
```
# Looking for the shape in test dataset
print("The data shape in test dataset is: {}".format(df_test.shape))
```

```
The data shape in test dataset is: (25976, 23)
```

Duplicate Values

In [868…
```
# Looking for duplicate values
df_train.duplicated().any()
```

Out[868]:
```
False
```

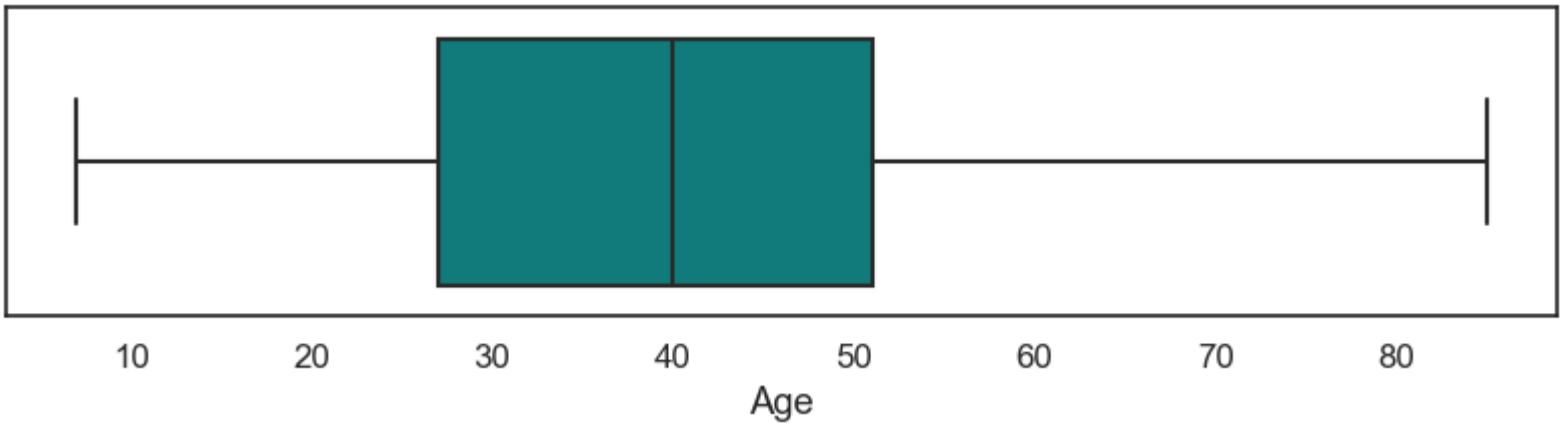
There are no duplicate values in the dataset.

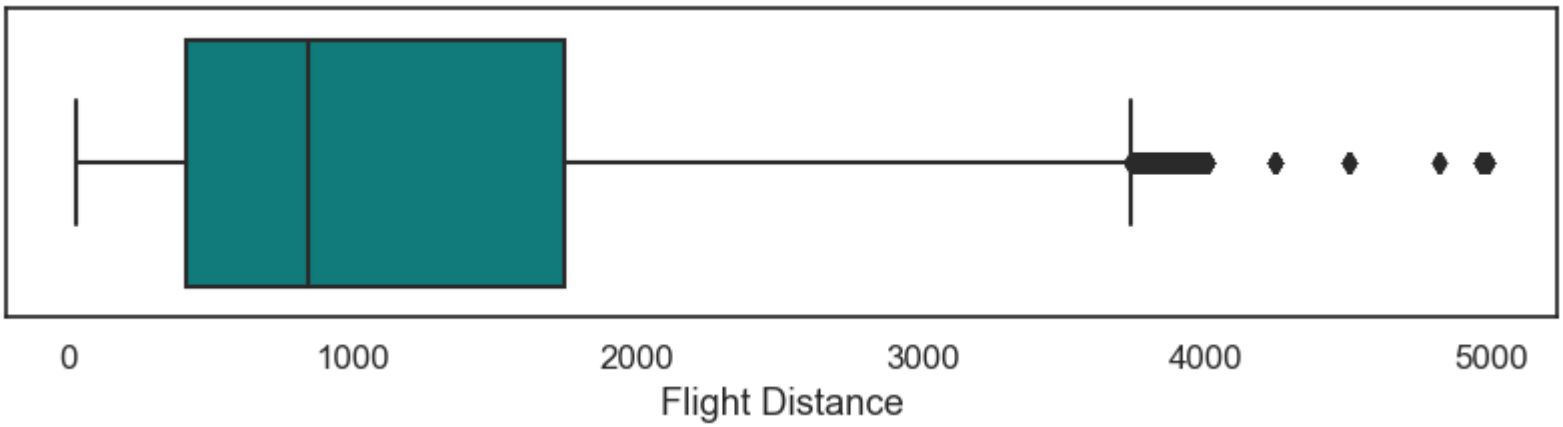
Outliers

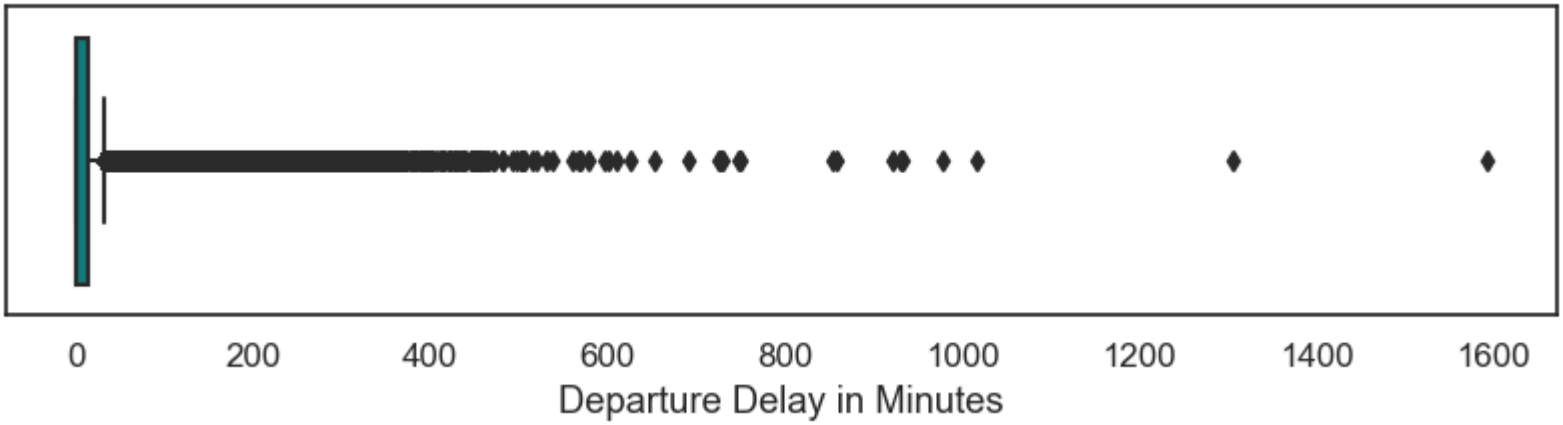
In [869…
```
# Looking for outliers
# numer_features = df_train.select_dtypes(exclude=['object'])
df_train[numer_var].columns
```

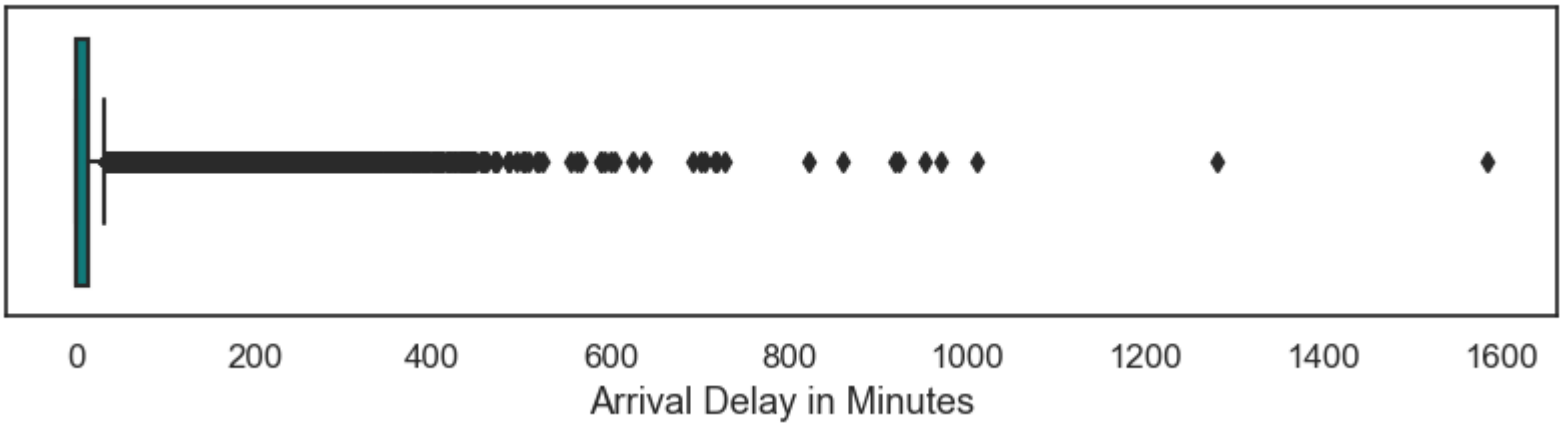
Out[869]:
```
Index(['Age', 'Flight Distance', 'Departure Delay in Minutes',
       'Arrival Delay in Minutes'],
      dtype='object')
```

In [870…
```
for column in numer_var:
    plt.figure(figsize=(10,2))
    sns.boxplot(data = df_train, x = column, color= 'darkcyan')
```





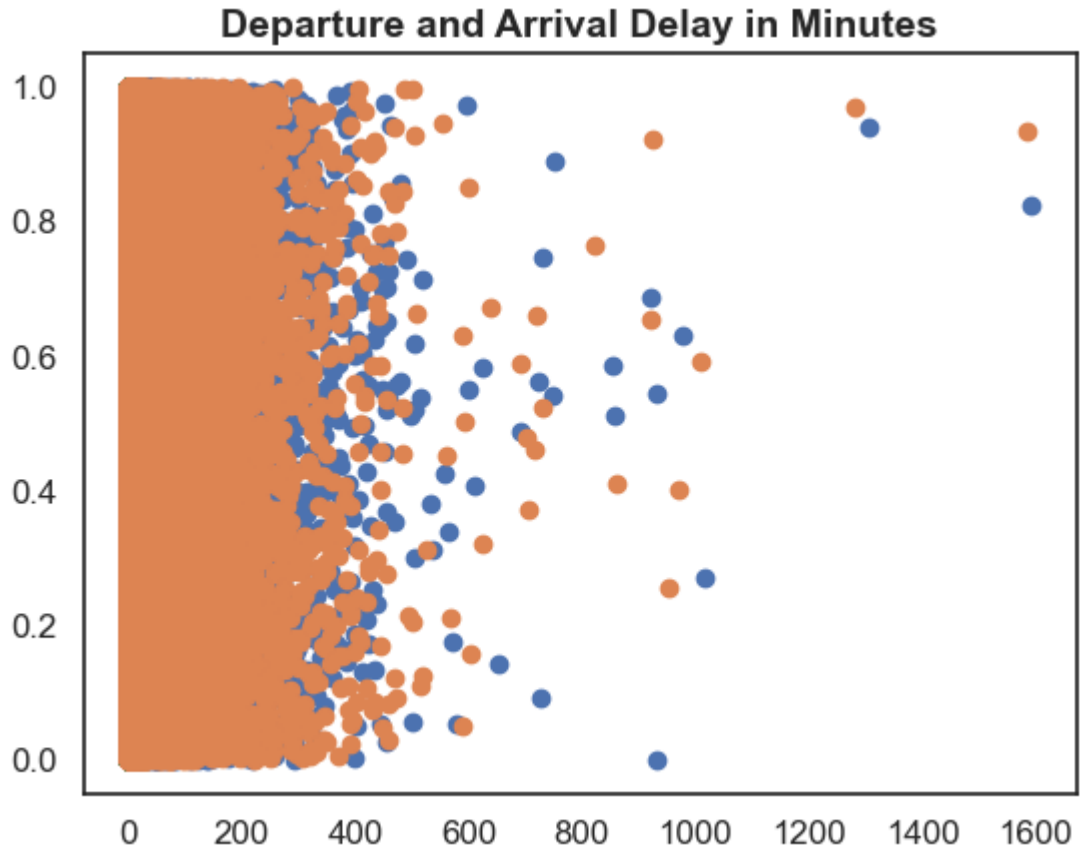




Here and from the descriptive statistics table, we can see that there are large values for the Departure Delay in Minutes and Arrival Delay in Minutes: 1592 and 1584 respectively.

In [871…
```
plt.scatter(df_train['Departure Delay in Minutes'], np.random.rand(df_train.shape[0]))
plt.scatter(df_train['Arrival Delay in Minutes'], np.random.rand(df_train.shape[0]))
plt.title('Departure and Arrival Delay in Minutes', fontsize=14, weight='semibold')
```

Out[871]:
```
Text(0.5, 1.0, 'Departure and Arrival Delay in Minutes')
```



Those extreme values are not errors, and they seem to be a natural part for the dataset. For this reason, the outliers will not be removed from the dataset.

2. Bivariate Analysis

Customer Satisfaction level in relation to the **Categorical Variables**

Customer Satisfaction level by Gender

In [872…

```
# Percentage of Customer Satisfaction in relation to Gender
pd.crosstab(df_train['Gender'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '%'
```

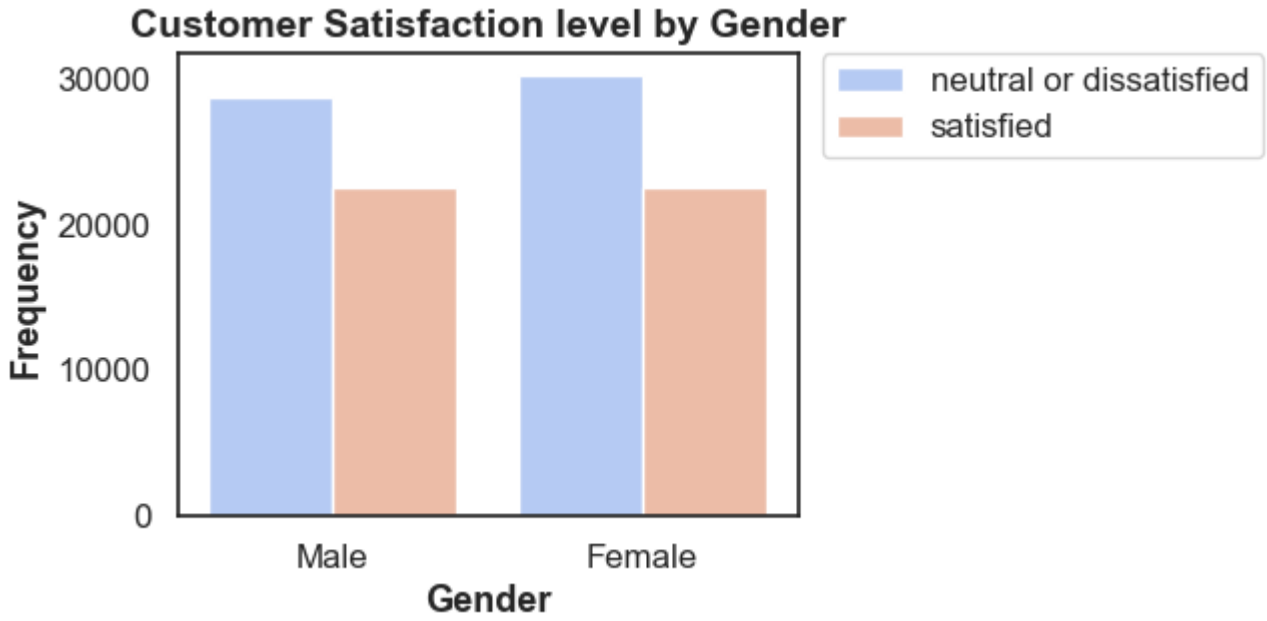
Out[872]:

Satisfaction	neutral or dissatisfied	satisfied
Gender		
Female	29.1%	21.7%
Male	27.6%	21.6%

Around 57% passengers are not satisfied with the service.

In [873…

```
# Visualization of the Customer Satisfaction in relation to Gender
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Gender', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Gender',weight='bold',fontsize='14')
plt.xlabel('Gender', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that there is a similar distribution of the neutral/dissatisfied and satisfied airline customers, with a high number of neutral/dissatisfied customers in both male and female.

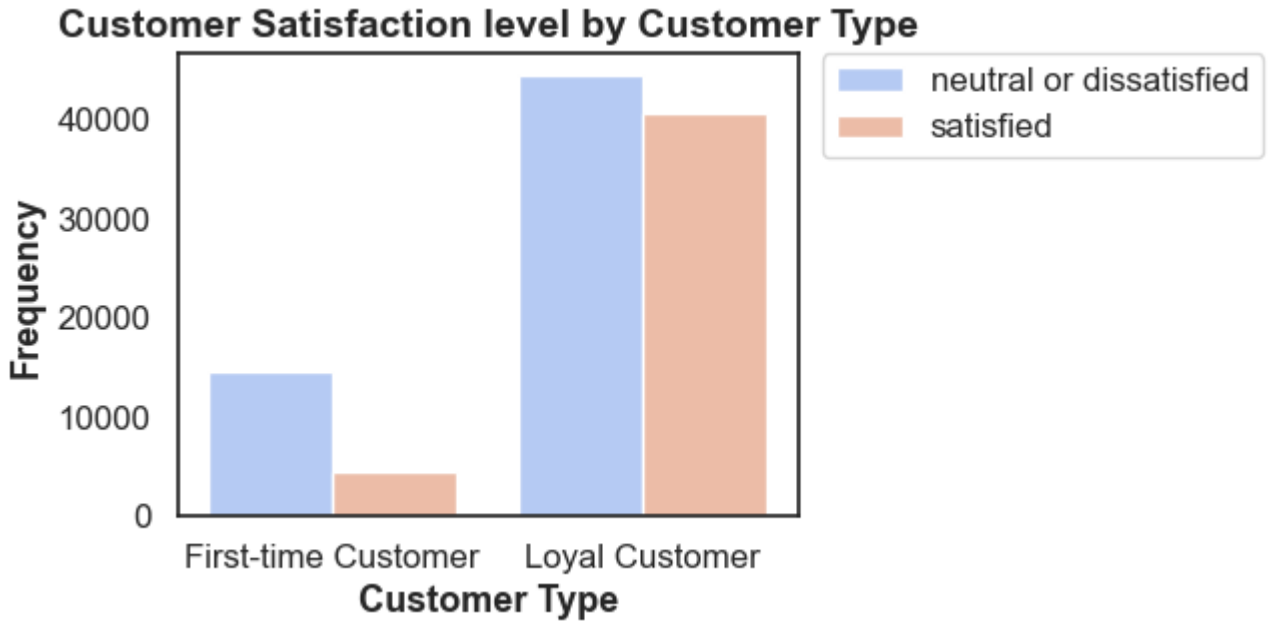
Customer Satisfaction level by Customer Type

In [874…
```
# Percentage of Customer Satisfaction in relation to Customer Type
pd.crosstab(df_train['Customer Type'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '%'
```

Out[874]:

Satisfaction	neutral or dissatisfied	satisfied
Customer Type		
First-time Customer	13.9%	4.3%
Loyal Customer	42.7%	39.0%

In [875…
```
# Visualization of the Customer Satisfaction in relation to Customer Type
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Customer Type', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Customer Type',weight='bold',fontsize='14')
plt.xlabel('Customer Type', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that among the Loyal customers, there is a similar ratio between neutral/dissatisfied (43%) and satisfied (39%) airline customers.

Customer Satisfaction level by Type of Travel

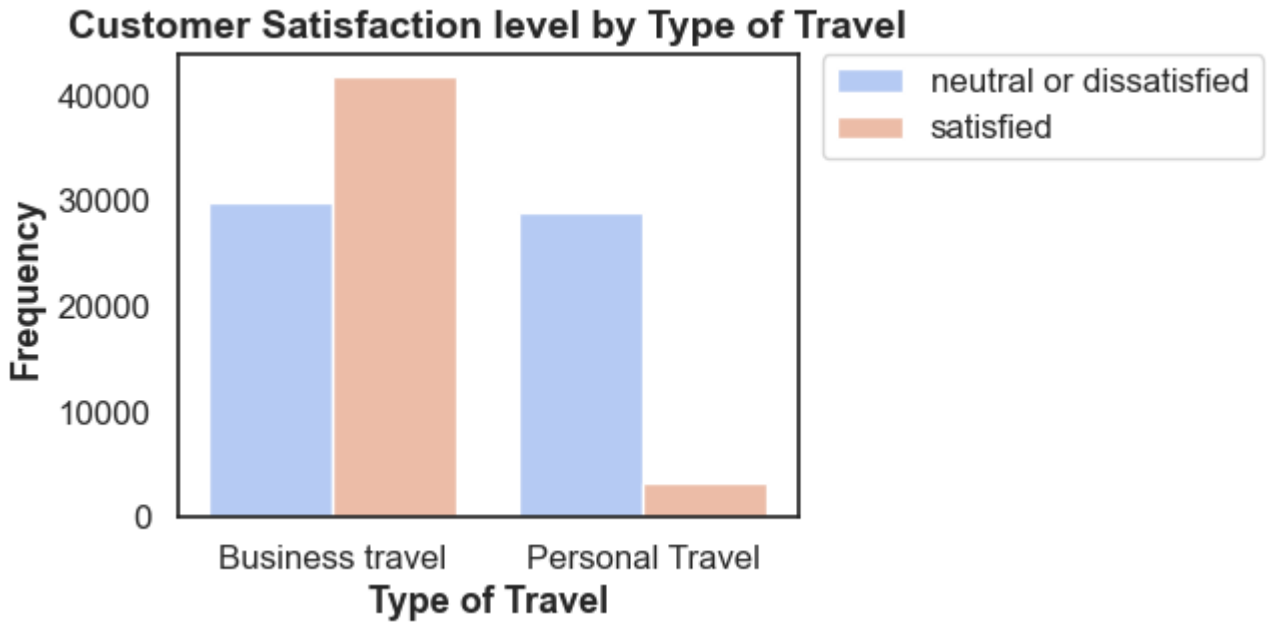
In [876…
```
# Percentage of Customer Satisfaction in relation to Type of Travel
pd.crosstab(df_train['Type of Travel'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '%'
```

Out[876]:

Satisfaction	neutral or dissatisfied	satisfied
Type of Travel		
Business travel	28.8%	40.2%
Personal Travel	27.9%	3.2%

In [877…
```
# Visualization of the Customer Satisfaction in relation to Type of Travel
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Type of Travel', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
```

```
plt.title('Customer Satisfaction level by Type of Travel',weight='bold',fontsize='14')
plt.xlabel('Type of Travel', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that among Business customers, there is a large amount of customers that are satisfied with the service provided. On the other hand, among customers with a Personal type of travel, there is a large amount of customers that found the service deficient.

Customer Satisfaction level by Class

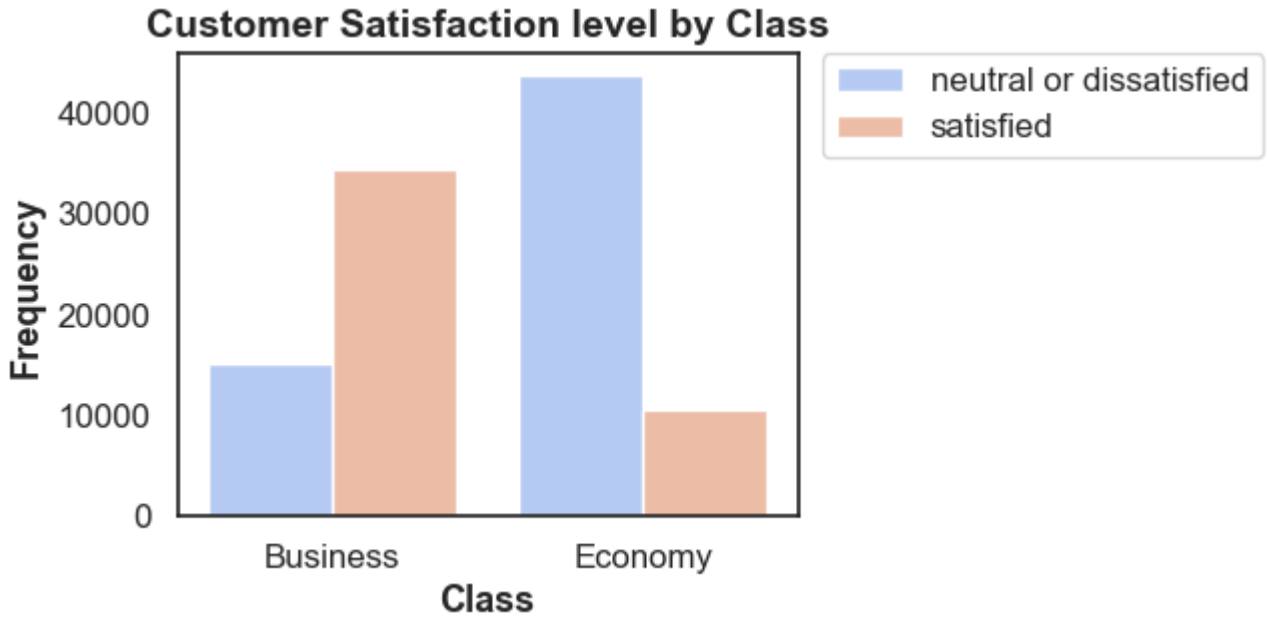
In [878…
```
# Percentage of Customer Satisfaction in relation to Class
pd.crosstab(df_train['Class'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '%'
```

Out[878]:

Satisfaction	neutral or dissatisfied	satisfied
Class		
Business	14.6%	33.2%
Economy	42.1%	10.1%

In [879…
```
# Visualization of the Customer Satisfaction in relation to Class
sns.set(style='white',font_scale=1.1)
```

```
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Class', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Class', weight='bold',fontsize='14')
plt.xlabel('Class', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows the predominance of customers satisfied in the Business class comparing with the large amount of neutral/dissatisfied customers in the Economy class.

Customer Satisfaction level by Inflight wifi service

In [880…

```
# Percentage of Customer Satisfaction in relation to Inflight wifi service
pd.crosstab(df_train['Inflight wifi service'], df_train['Satisfaction'])
```

Out[880]:

Satisfaction	neutral or dissatisfied	satisfied
Inflight wifi service		
0	8	3095
1	12034	5806
2	19407	6423
3	19386	6482
4	7938	11856
5	106	11363

In [881…

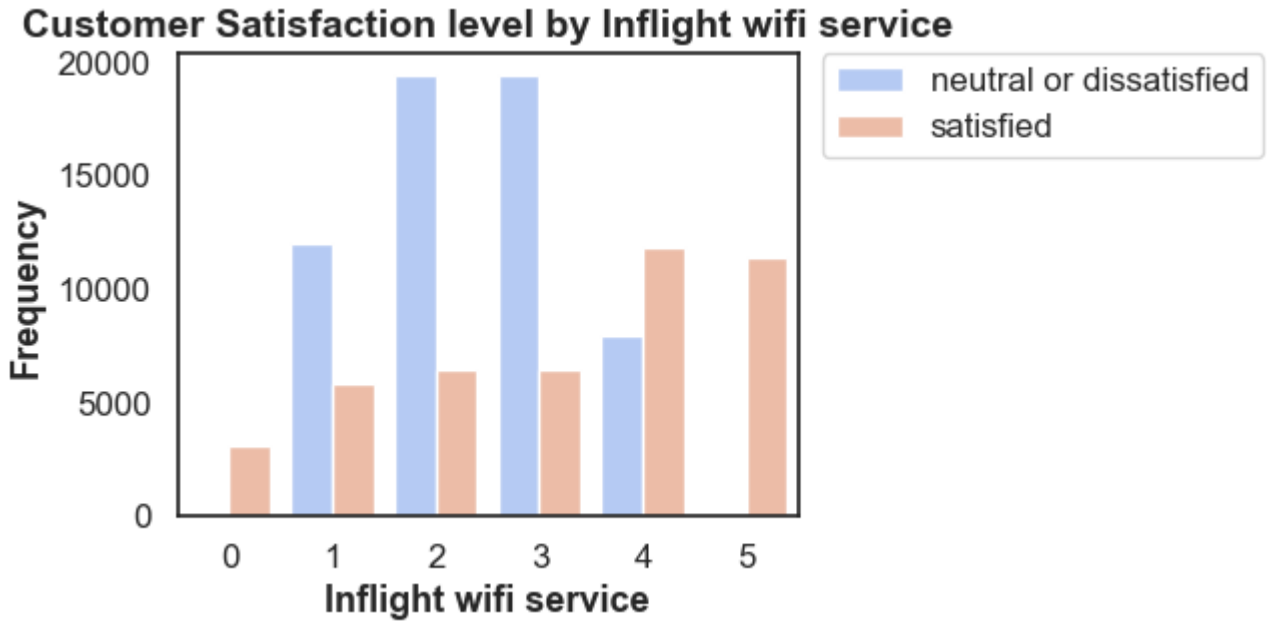
```
# Percentage of Customer Satisfaction in relation to Inflight wifi service
pd.crosstab(df_train['Inflight wifi service'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str
```

Out[881]:

Satisfaction	neutral or dissatisfied	satisfied
Inflight wifi service		
0	0.0%	3.0%
1	11.6%	5.6%
2	18.7%	6.2%
3	18.7%	6.2%
4	7.6%	11.4%
5	0.1%	10.9%

In [882…

```
# Visualization of the Customer Satisfaction in relation to Inflight wifi service
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Inflight wifi service', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Inflight wifi service',weight='bold',fontsize='14')
plt.xlabel('Inflight wifi service', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that there is a high level of customer dissatisfaction when the Inflight wifi service is rated as poor service (rate below 3). On the other hand, there is a high level of customer satisfaction when the service is rated good/excellent. This feature could be a major factor that the airlines should consider in order to improve their inflight services.

Customer Satisfaction level by Departure/Arrival time convenient

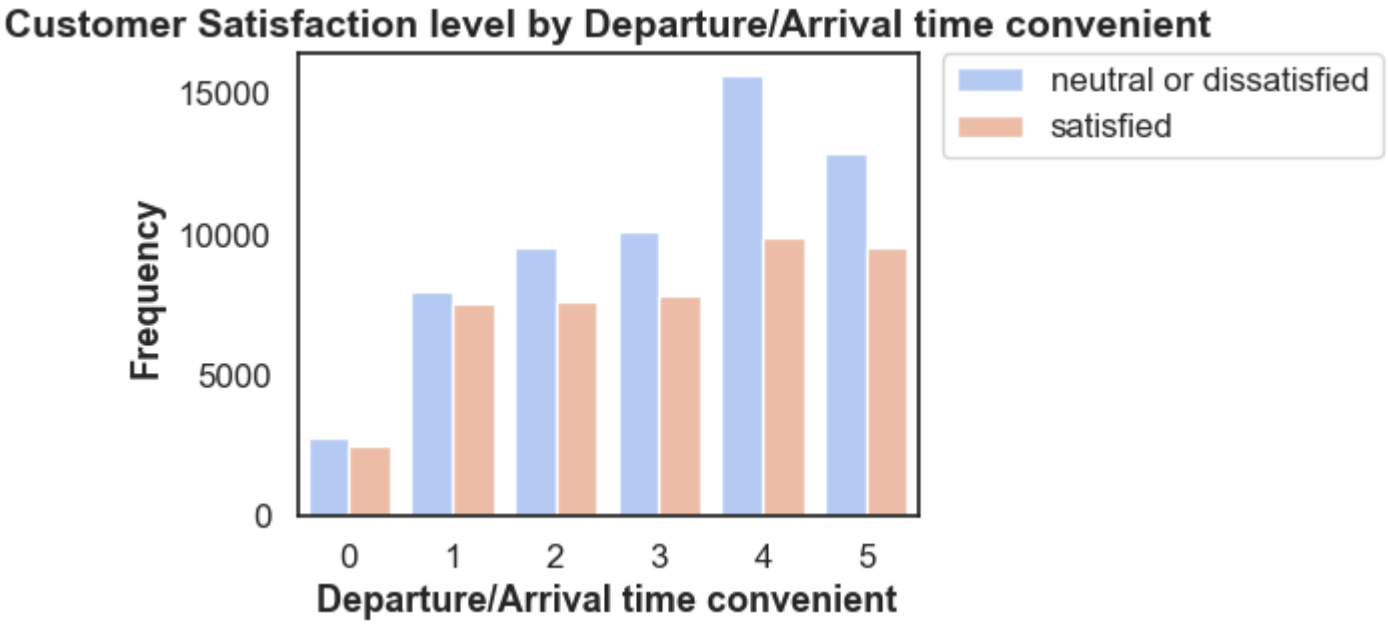
In [883…
```
# Percentage of Customer Satisfaction in relation to Departure/Arrival time convenient
pd.crosstab(df_train['Departure/Arrival time convenient'], df_train['Satisfaction'], normalize = True).mul(100).round(1
```

Out[883]:

Satisfaction	neutral or dissatisfied	satisfied
Departure/Arrival time convenient		
0	2.7%	2.4%
1	7.7%	7.3%
2	9.2%	7.4%
3	9.7%	7.6%
4	15.1%	9.5%
5	12.4%	9.2%

In [884…

```
# Visualization of the Customer Satisfaction in relation to Departure/Arrival time convenient
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Departure/Arrival time convenient', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Departure/Arrival time convenient',weight='bold',fontsize='14')
plt.xlabel('Departure/Arrival time convenient', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that, in general, there is a high level of customer dissatisfaction in all the rated values, even when the Departure/Arrival time is rated as extremely convenient. This feature may have a limited impact on Satisfaction.

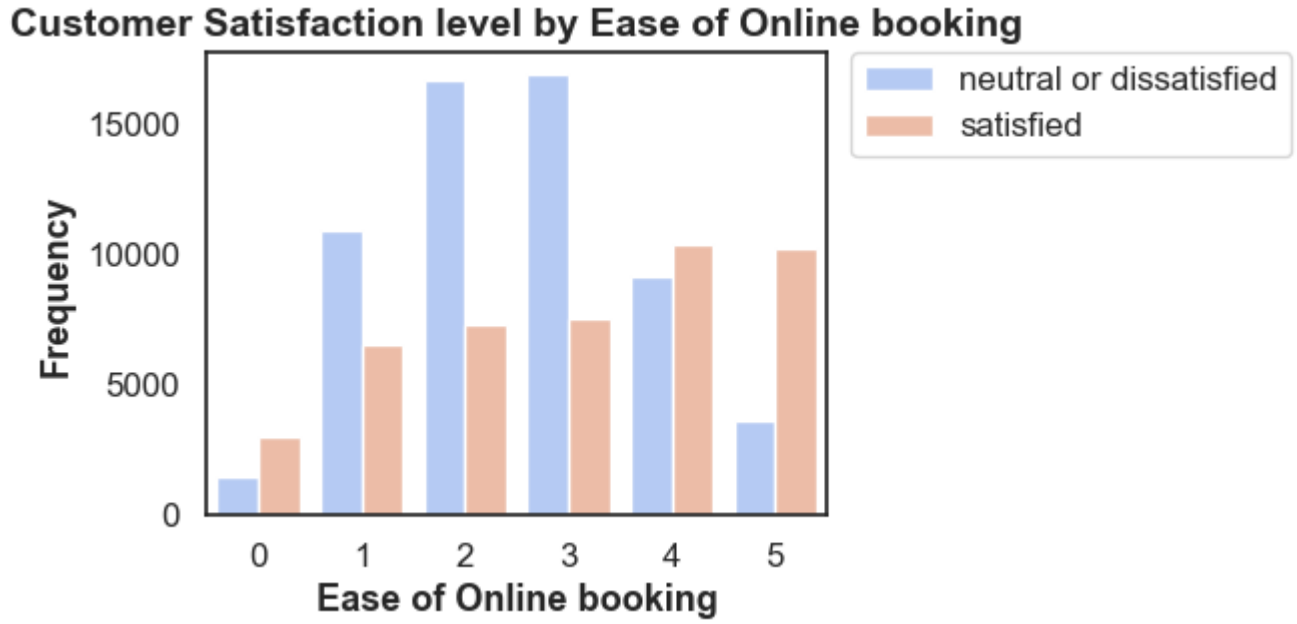
Customer Satisfaction level by Ease of Online booking

In [885…
```
# Percentage of Customer Satisfaction in relation to Ease of Online booking
pd.crosstab(df_train['Ease of Online booking'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(st
```

Out[885]:

Satisfaction	neutral or dissatisfied	satisfied
Ease of Online booking		
0	1.5%	2.9%
1	10.5%	6.3%
2	16.1%	7.0%
3	16.3%	7.3%
4	8.8%	10.0%
5	3.5%	9.8%

In [886…
```
# Visualization of the Customer Satisfaction in relation to Ease of Online booking
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Ease of Online booking', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Ease of Online booking',weight='bold',fontsize='14')
plt.xlabel('Ease of Online booking', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that the level of customer satisfaction is high when the Ease of Online booking rate is above 4.

Customer Satisfaction level by Gate location

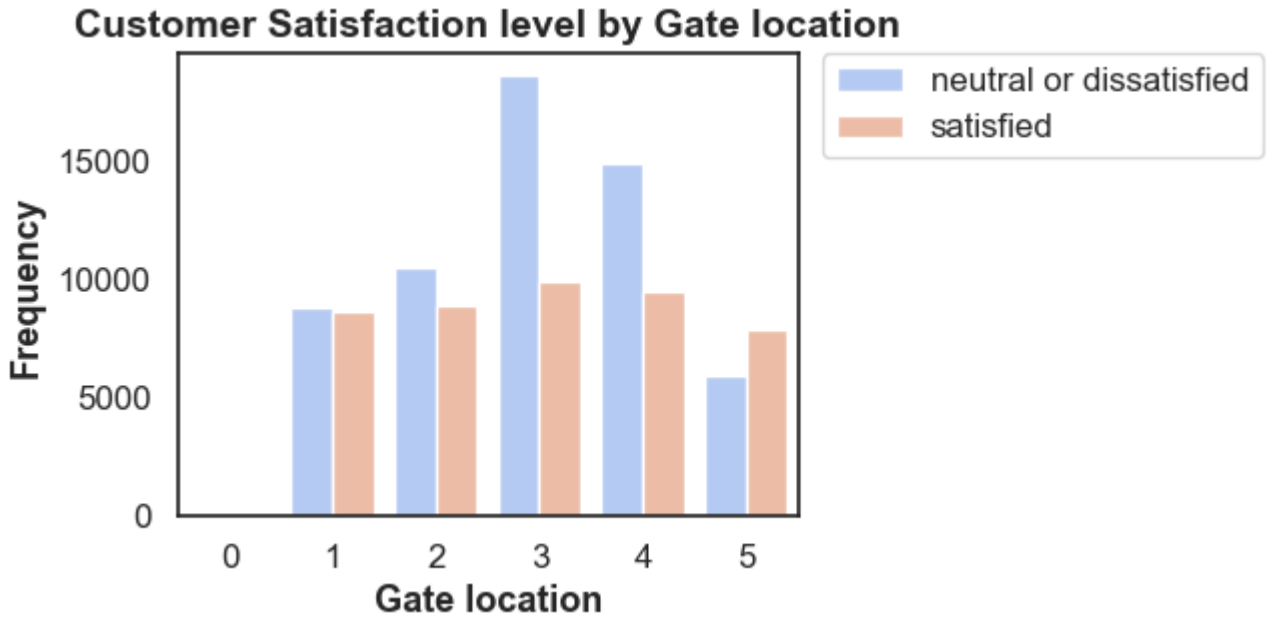
In [887…
```
# Percentage of Customer Satisfaction in relation to Gate location
pd.crosstab(df_train['Gate location'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '%'
```

Out[887]:

Satisfaction	neutral or dissatisfied	satisfied
Gate location		
0	0.0%	0.0%
1	8.5%	8.4%
2	10.1%	8.6%
3	18.0%	9.5%
4	14.4%	9.1%
5	5.7%	7.6%

In [888…

```
# Visualization of the Customer Satisfaction in relation to Gate location
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Gate location', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Gate location',weight='bold',fontsize='14')
plt.xlabel('Gate location', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that the level of customer satisfaction is low even when the Gate location is rated as extremely convenient (score 5). This feature may have a limited impact on Satisfaction.

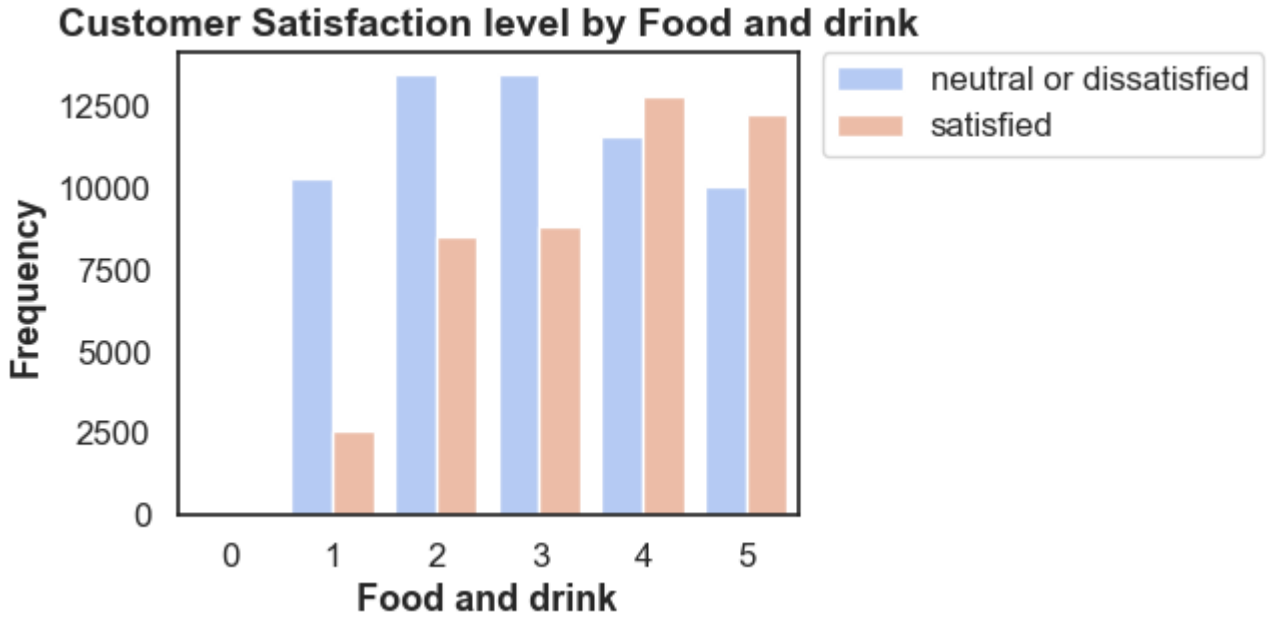
Customer Satisfaction level by Food and drink

In [889…
```
# Percentage of Customer Satisfaction in relation to Food and drink
pd.crosstab(df_train['Food and drink'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '%'
```

Out[889]:

Satisfaction	neutral or dissatisfied	satisfied
Food and drink		
0	0.1%	0.0%
1	9.9%	2.5%
2	13.0%	8.2%
3	13.0%	8.5%
4	11.1%	12.3%
5	9.7%	11.8%

In [890…
```
# Visualization of the Customer Satisfaction in relation to Food and drink
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Food and drink', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Food and drink',weight='bold',fontsize='14')
plt.xlabel('Food and drink', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that there is a high level of customer satisfaction when the Food and drink rate is above 4.

Customer Satisfaction level by Online boarding

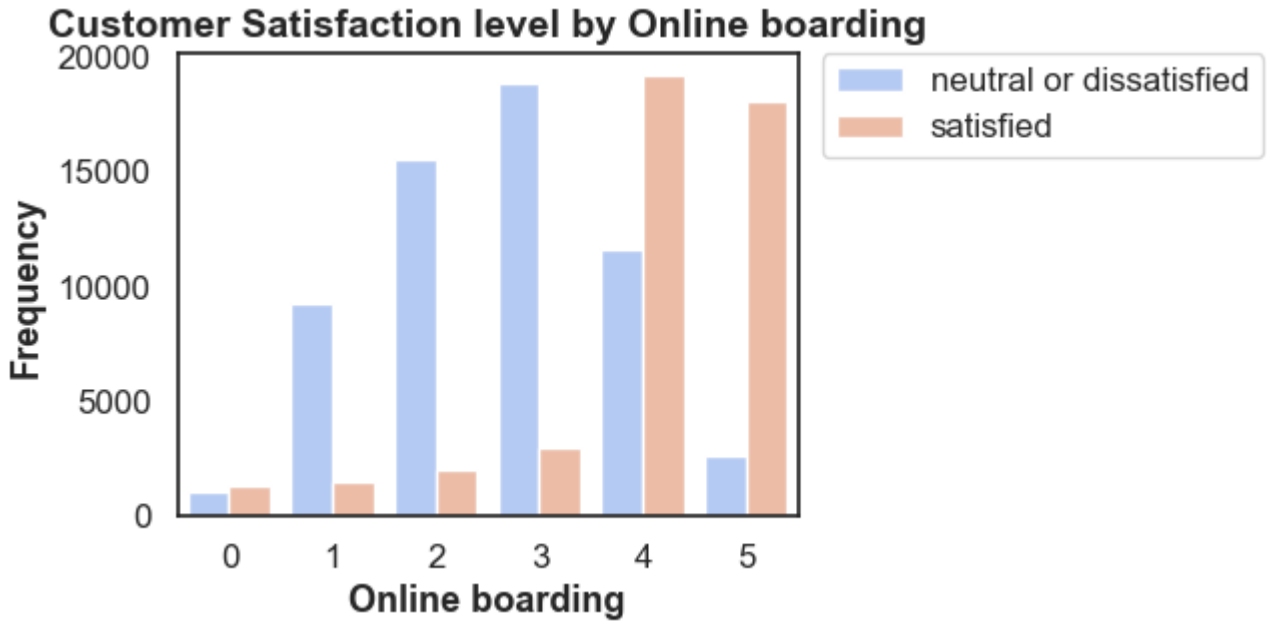
In [891…
```
# Percentage of Customer Satisfaction in relation to Online boarding
pd.crosstab(df_train['Online boarding'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '%
```

Out[891]:

Satisfaction	neutral or dissatisfied	satisfied
Online boarding		
0	1.0%	1.3%
1	8.9%	1.4%
2	14.9%	1.9%
3	18.1%	2.8%
4	11.2%	18.4%
5	2.6%	17.4%

In [892…

```
# Visualization of the Customer Satisfaction in relation to Online boarding
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Online boarding', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Online boarding',weight='bold',fontsize='14')
plt.xlabel('Online boarding', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that there is a high level of customer satisfaction when the Online boarding is considered extremely good (scores above 4).

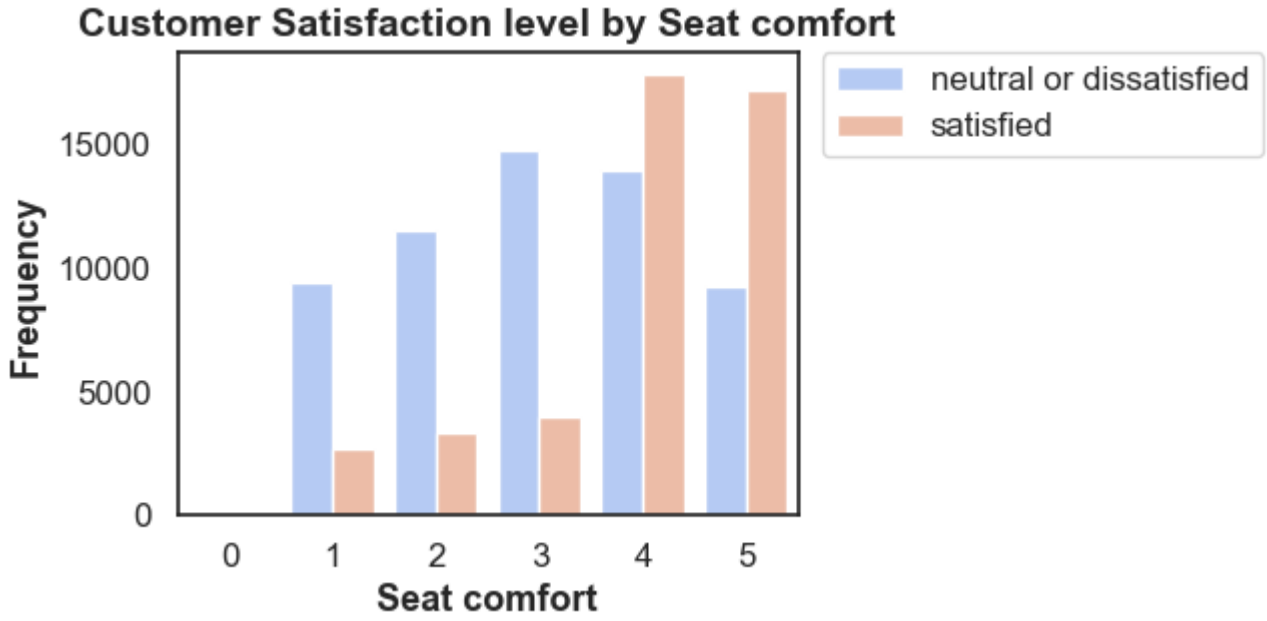
Customer Satisfaction level by Seat comfort

In [893…
```
# Percentage of Customer Satisfaction in relation to Seat comfort
pd.crosstab(df_train['Seat comfort'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '%'
```

Out[893]:

Satisfaction	neutral or dissatisfied	satisfied
Seat comfort		
0	0.0%	0.0%
1	9.0%	2.6%
2	11.1%	3.2%
3	14.2%	3.8%
4	13.4%	17.2%
5	8.9%	16.6%

In [894…
```
# Visualization of the Customer Satisfaction in relation to Seat comfort
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Seat comfort', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Seat comfort',weight='bold',fontsize='14')
plt.xlabel('Seat comfort', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that, in general, airline customers perceived a high level of customer satisfaction when rated the seat as extremely comfortable (score above 4).

Customer Satisfaction level by Inflight entertainment

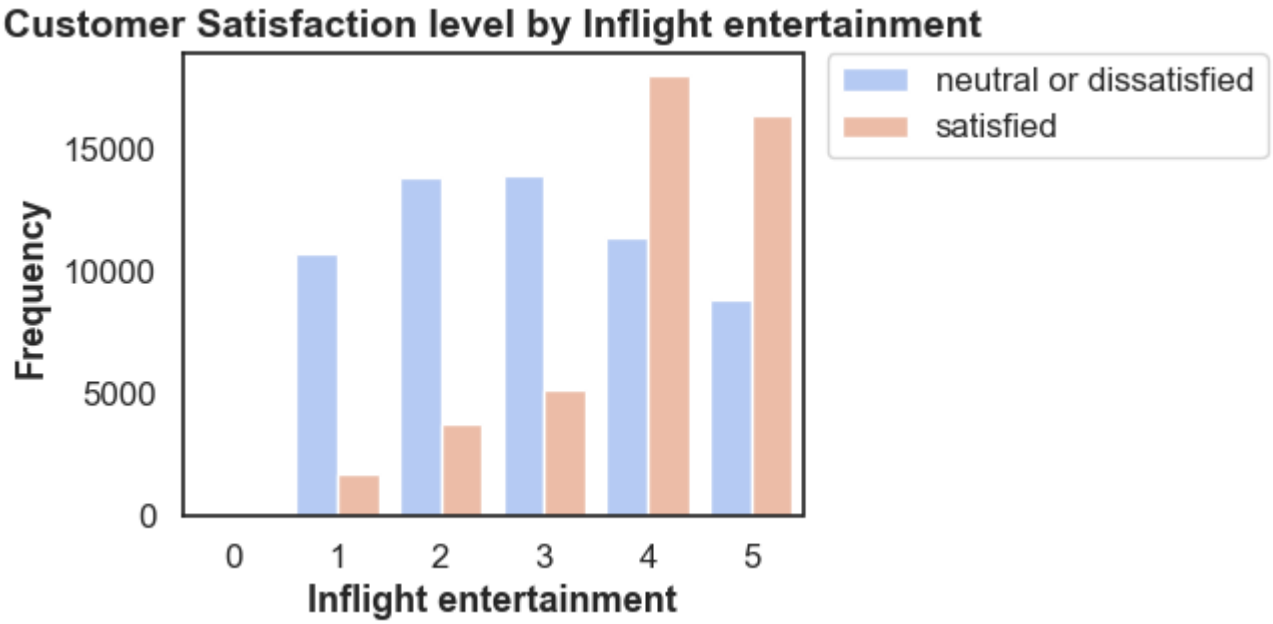
In [895…
```
# Percentage of Customer Satisfaction in relation to Inflight entertainment
pd.crosstab(df_train['Inflight entertainment'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(st
```

Out[895]:

Satisfaction	neutral or dissatisfied	satisfied
Inflight entertainment		
0	0.0%	0.0%
1	10.3%	1.7%
2	13.4%	3.6%
3	13.4%	5.0%
4	11.0%	17.3%
5	8.5%	15.7%

In [896…

```
# Visualization of the Customer Satisfaction in relation to Inflight entertainment
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Inflight entertainment', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Inflight entertainment',weight='bold',fontsize='14')
plt.xlabel('Inflight entertainment', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



Customer Satisfaction level by On-board service

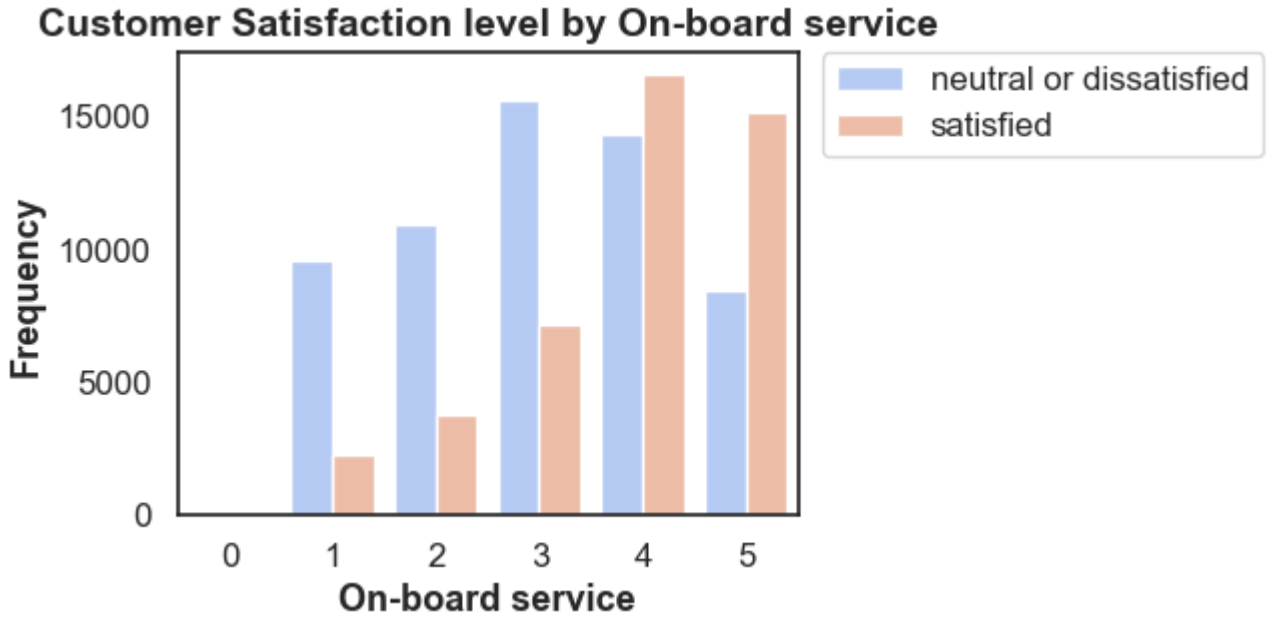
In [897…
```
# Percentage of Customer Satisfaction in relation to On-board service
pd.crosstab(df_train['On-board service'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '
```

Out[897]:

Satisfaction	neutral or dissatisfied	satisfied
On-board service		
0	0.0%	0.0%
1	9.2%	2.2%
2	10.5%	3.6%
3	15.0%	6.9%
4	13.8%	15.9%
5	8.1%	14.6%

In [898…
```
# Visualization of the Customer Satisfaction in relation to On-board service
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
```

```
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'On-board service', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by On-board service',weight='bold',fontsize='14')
plt.xlabel('On-board service', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that there is a high level of customer satisfaction when the On-board service is extremely good, with a rate above 4.

Customer Satisfaction level by Leg room

In [899…

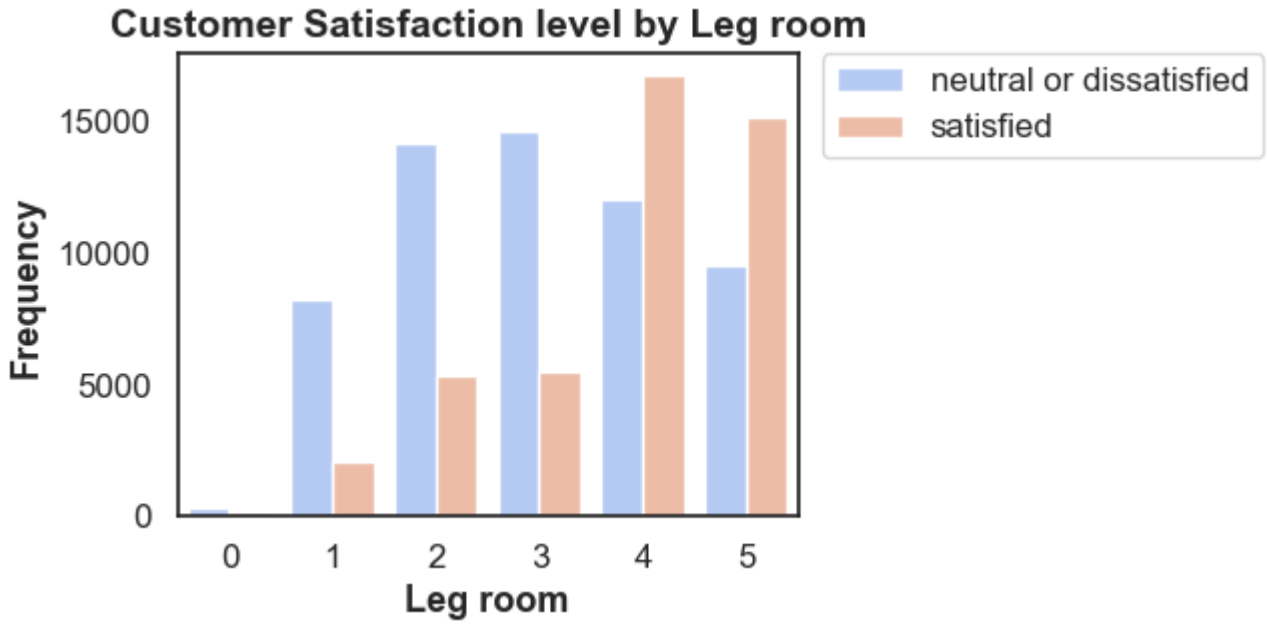
```
# Percentage of Customer Satisfaction in relation to leg room
pd.crosstab(df_train['Leg room'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '%'
```

Out[899]:

Satisfaction	neutral or dissatisfied	satisfied
Leg room		
0	0.3%	0.2%
1	7.9%	2.0%
2	13.6%	5.2%
3	14.1%	5.3%
4	11.6%	16.1%
5	9.2%	14.6%

In [900…

```
# Visualization of the Customer Satisfaction in relation to Leg room
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Leg room', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Leg room',weight='bold',fontsize='14')
plt.xlabel('Leg room', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that there is a high level of customer satisfaction when the Leg room service is extremely good, with a rate above 4.

Customer Satisfaction level by Baggage handling

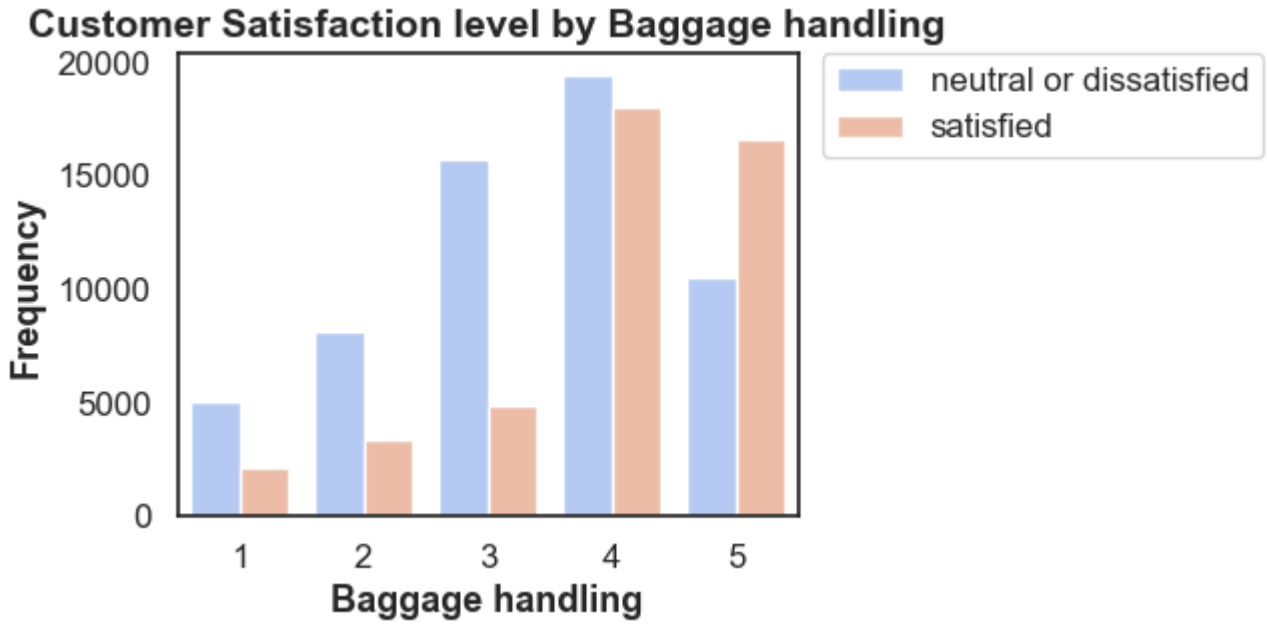
In [901…
```
# Percentage of Customer Satisfaction in relation to Baggage handling
pd.crosstab(df_train['Baggage handling'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '
```

Out[901]:

Satisfaction	neutral or dissatisfied	satisfied
Baggage handling		
1	4.9%	2.1%
2	7.8%	3.3%
3	15.2%	4.7%
4	18.7%	17.3%
5	10.1%	16.0%

In [902…
```
# Visualization of the Customer Satisfaction in relation to Baggage handling
sns.set(style='white',font_scale=1.1)
```

```
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Baggage handling', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Baggage handling',weight='bold',fontsize='14')
plt.xlabel('Baggage handling', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



Customer Satisfaction level by Checkin service

In [903…

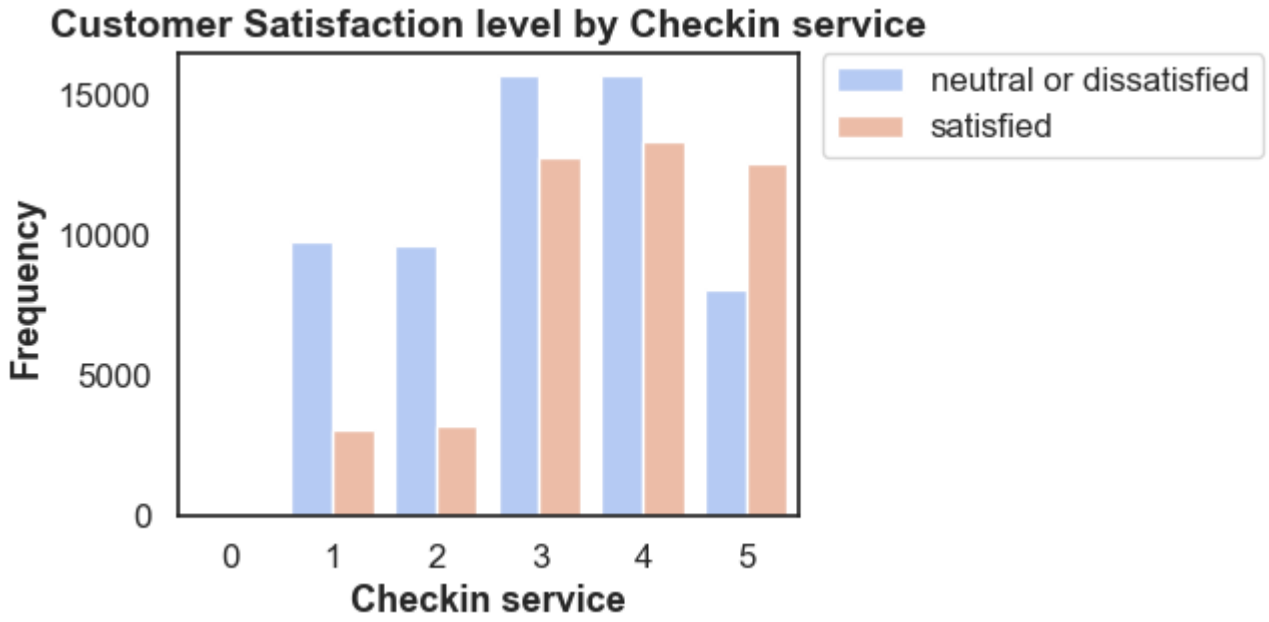
```
# Percentage of Customer Satisfaction in relation to Checkin service
pd.crosstab(df_train['Checkin service'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '%
```

Out[903]:

Satisfaction	neutral or dissatisfied	satisfied
Checkin service		
0	0.0%	0.0%
1	9.4%	3.0%
2	9.3%	3.1%
3	15.1%	12.3%
4	15.1%	12.9%
5	7.7%	12.1%

In [904…

```
# Visualization of the Customer Satisfaction in relation to Checkin service
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Checkin service', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Checkin service',weight='bold',fontsize='14')
plt.xlabel('Checkin service', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that, in general, there is a high level of customer dissatisfaction when the Check-in service is deficient (rate below 4).

Customer Satisfaction level by Inflight service

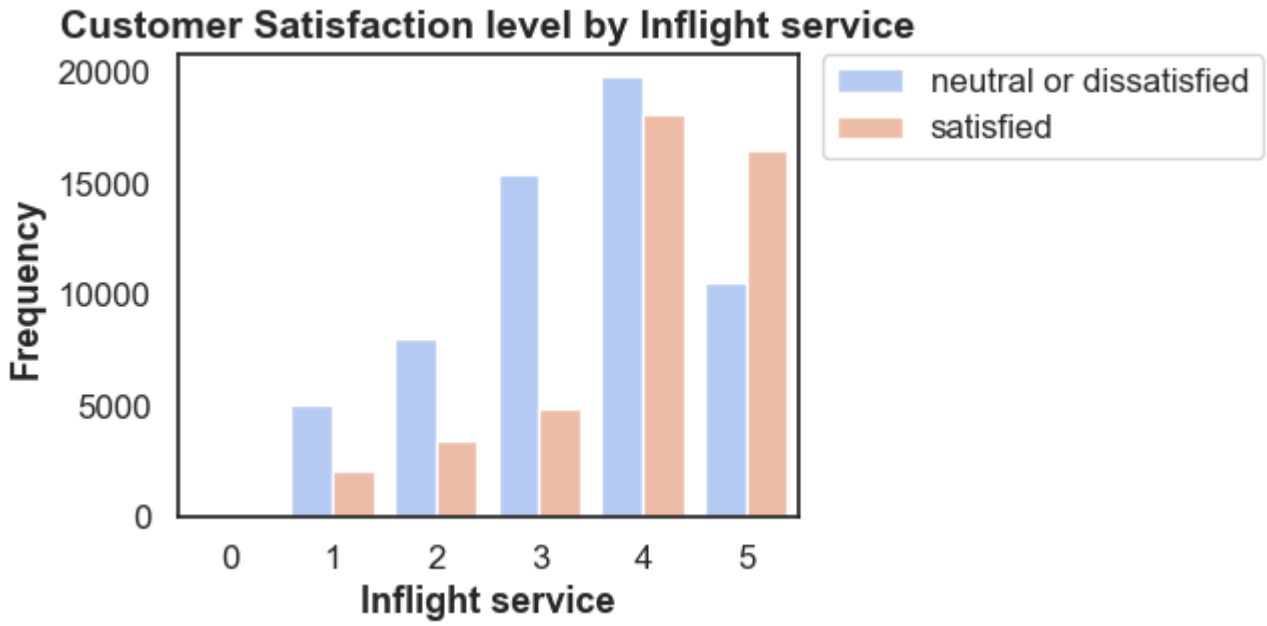
In [905…
```
# Percentage of Customer Satisfaction in relation to Inflight service
pd.crosstab(df_train['Inflight service'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '
```

Out[905]:

Satisfaction	neutral or dissatisfied	satisfied
Inflight service		
0	0.0%	0.0%
1	4.8%	2.0%
2	7.7%	3.3%
3	14.9%	4.7%
4	19.1%	17.4%
5	10.2%	15.9%

In [906…
```
# Visualization of the Customer Satisfaction in relation to Inflight service
```

```
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Inflight service', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Inflight service',weight='bold',fontsize='14')
plt.xlabel('Inflight service', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that there is a high level of customer satisfaction when the Inflight service is extremely good, with a rate above 4.

Customer Satisfaction level by Cleanliness

In [907…

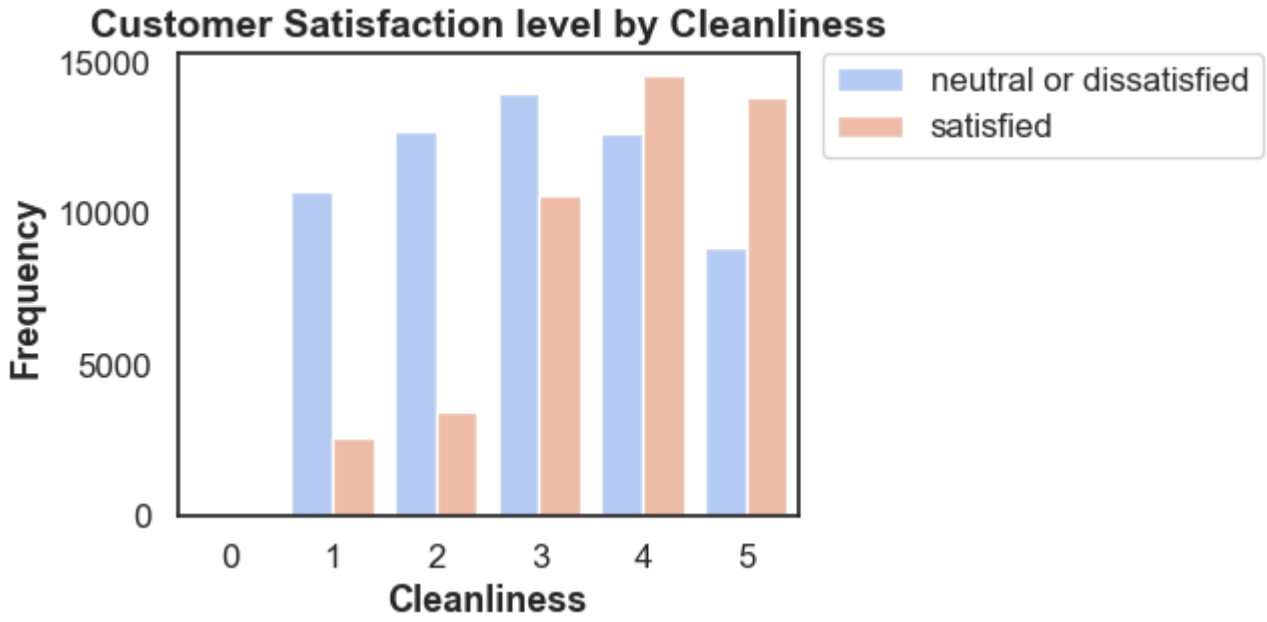
```
# Percentage of Customer Satisfaction in relation to Cleanliness
pd.crosstab(df_train['Cleanliness'], df_train['Satisfaction'], normalize = True).mul(100).round(1).astype(str) + '%'
```

Out[907]:

Satisfaction	neutral or dissatisfied	satisfied
Cleanliness		
0	0.0%	0.0%
1	10.3%	2.5%
2	12.2%	3.3%
3	13.4%	10.2%
4	12.2%	14.0%
5	8.5%	13.3%

In [908…

```
# Visualization of the Customer Satisfaction in relation to Cleanliness
sns.set(style='white',font_scale=1.1)
fig_st = plt.figure(figsize=(4,3))
axe = fig.add_axes([0,0,1,1])
sns.countplot(data = df_train, x = 'Cleanliness', hue = 'Satisfaction', palette = 'coolwarm')
plt.legend(bbox_to_anchor = (1.04, 1), loc = 'upper left', borderaxespad = 0)
plt.title('Customer Satisfaction level by Cleanliness',weight='bold',fontsize='14')
plt.xlabel('Cleanliness', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
wrap_labels(axe, 10)
```



This plot shows that the level of customer satisfaction is high when the Cleanliness score is above 4.

Customer Satisfaction level in relation to the **Numerical Variables**

Customer Satisfaction level by Age

In [909…

```
# Visualization of the Customer Satisfaction in relation to Age
sns.set(font_scale = 1.2)
sns.set_style('ticks')
sns.displot(data = df_train, x = 'Age', hue = 'Satisfaction', kind = 'kde', bw_adjust = 0.75, fill = True)
plt.axvline(df_train['Age'].mean(), ls = '--', color = 'black')
plt.title('Customer Satisfaction level by Age', fontsize=15, weight='semibold')
plt.xlabel('Age', fontsize=13, weight='semibold')
plt.ylabel('Density', fontsize=13, weight='semibold')
sns.despine()
```

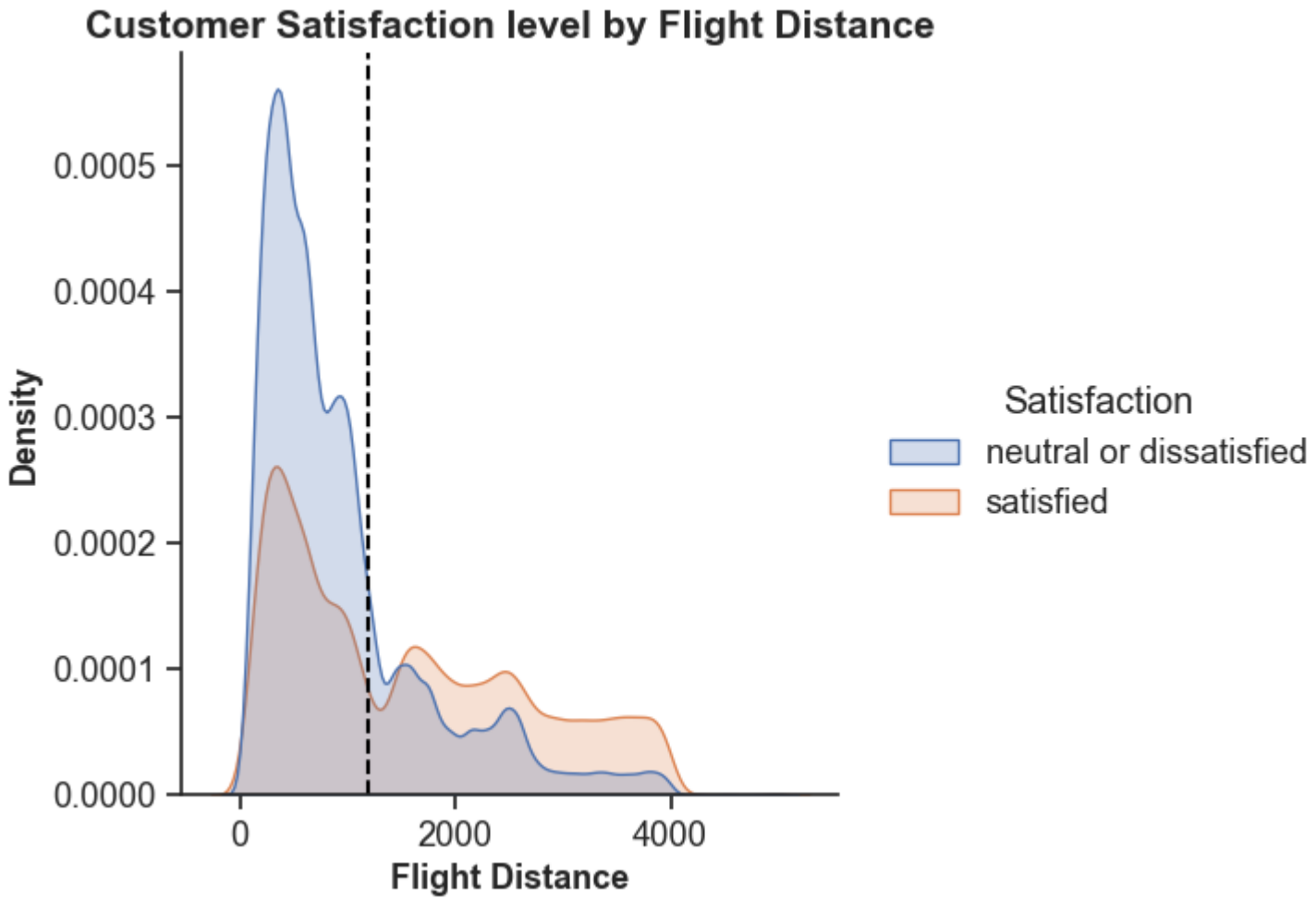
This density plot shows that in the age range between 40 and 60, the number of satisfied customers is higher than the neutral/dissatisfied customers.

Customer Satisfaction level by Flight Distance

In [910…

```
# Visualization of the Customer Satisfaction in relation to Flight Distance
sns.set(font_scale = 1.2)
sns.set_style('ticks')
sns.displot(data = df_train, x = 'Flight Distance', hue = 'Satisfaction', kind = 'kde', bw_adjust = 0.75, fill = True)
plt.axvline(df_train['Flight Distance'].mean(), ls = '--', color = 'black')
plt.title('Customer Satisfaction level by Flight Distance', fontsize=15, weight='semibold')
plt.xlabel('Flight Distance', fontsize=13, weight='semibold')
```

```
plt.ylabel('Density', fontsize=13, weight='semibold')
sns.despine()
```



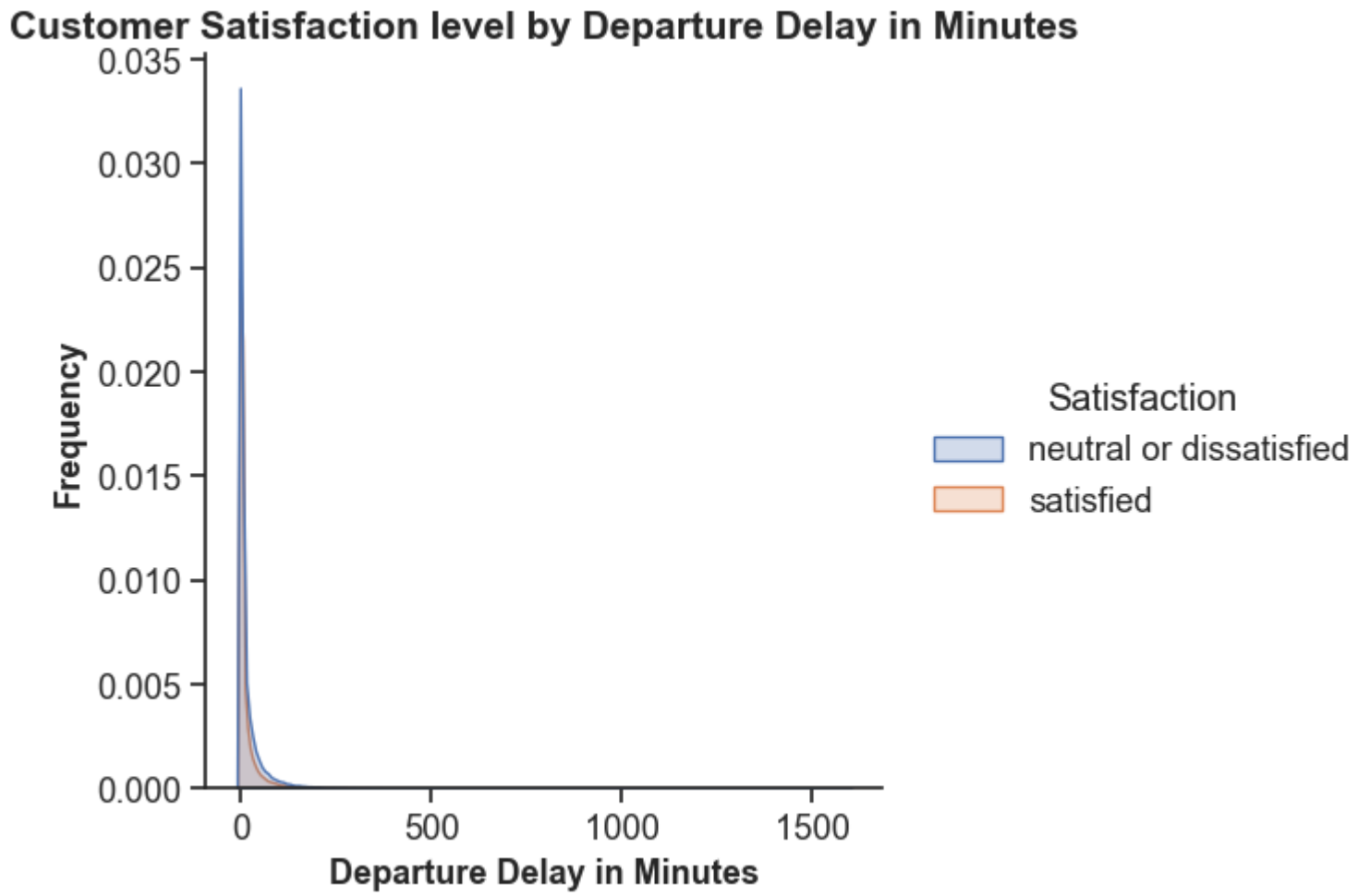
This plot shows that the level of customer dissatisfaction is high in customers that travelled approximately less than 1100 miles (approx. mean). On the other hand, there is a significant level of customer satisfaction in long distance flight (more than approx. 1100 miles).

Customer Satisfaction level by Departure Delay in Minutes

In [911…

```
# Visualization of the Customer Satisfaction in relation to Departure Delay in Minutes
sns.set(font_scale = 1.2)
sns.set_style('ticks')
```

```
sns.displot(data = df_train, x = 'Departure Delay in Minutes', hue = 'Satisfaction', kind = 'kde', bw_adjust = 0.75, fi
plt.title('Customer Satisfaction level by Departure Delay in Minutes', fontsize=15, weight='semibold')
plt.xlabel('Departure Delay in Minutes', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
sns.despine()
```

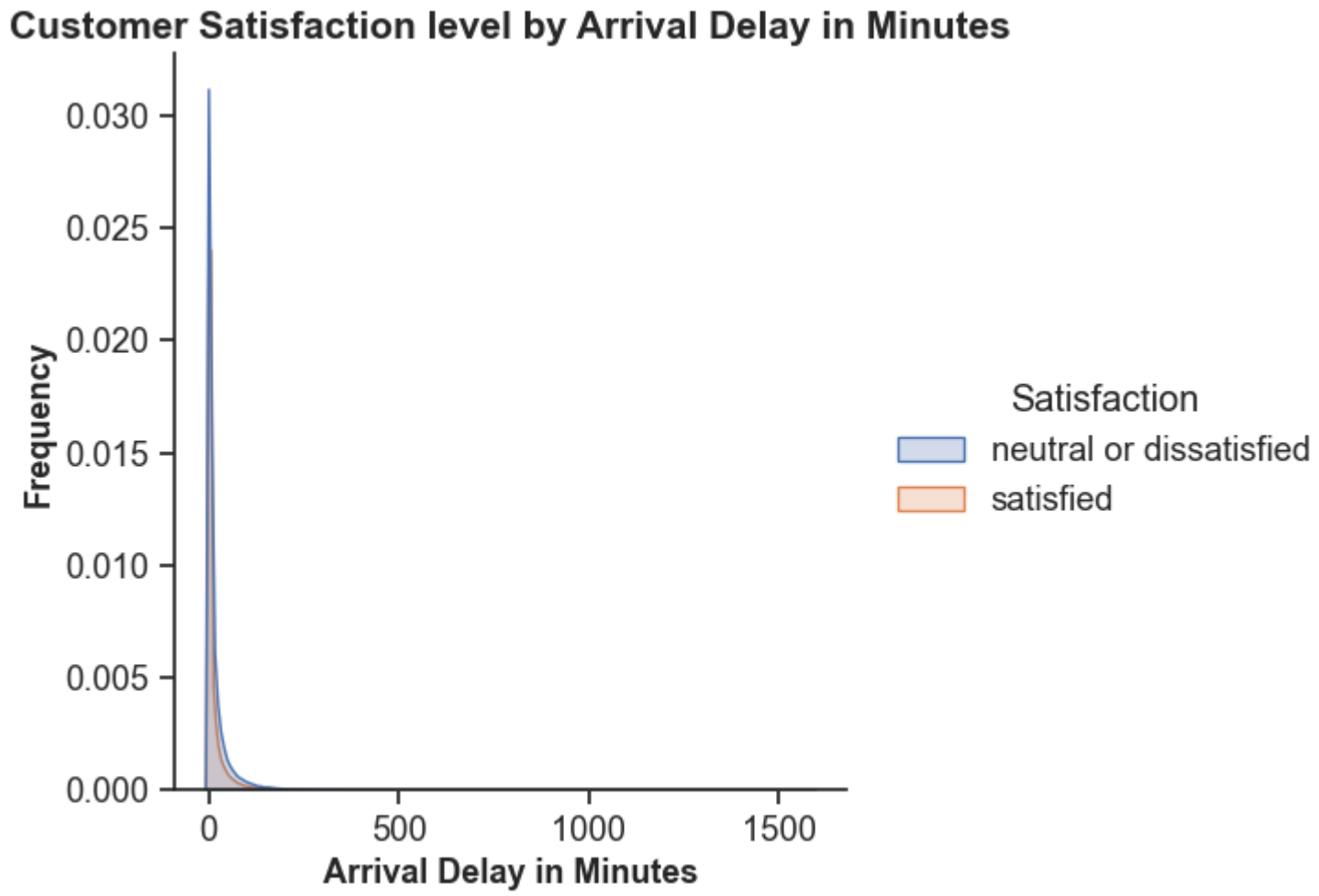


Customer Satisfaction level by Arrival Delay in Minutes

In [912…

```
# Visualization of the Customer Satisfaction in relation to Arrival Delay in Minutes
sns.set(font_scale = 1.2)
sns.set_style('ticks')
sns.displot(data = df_train, x = 'Arrival Delay in Minutes', hue = 'Satisfaction', kind = 'kde', bw_adjust = 0.75, fill
plt.title('Customer Satisfaction level by Arrival Delay in Minutes', fontsize=15, weight='semibold')
```

```
plt.xlabel('Arrival Delay in Minutes', fontsize=13, weight='semibold')
plt.ylabel('Frequency', fontsize=13, weight='semibold')
sns.despine()
```



3. Multivariate Analysis

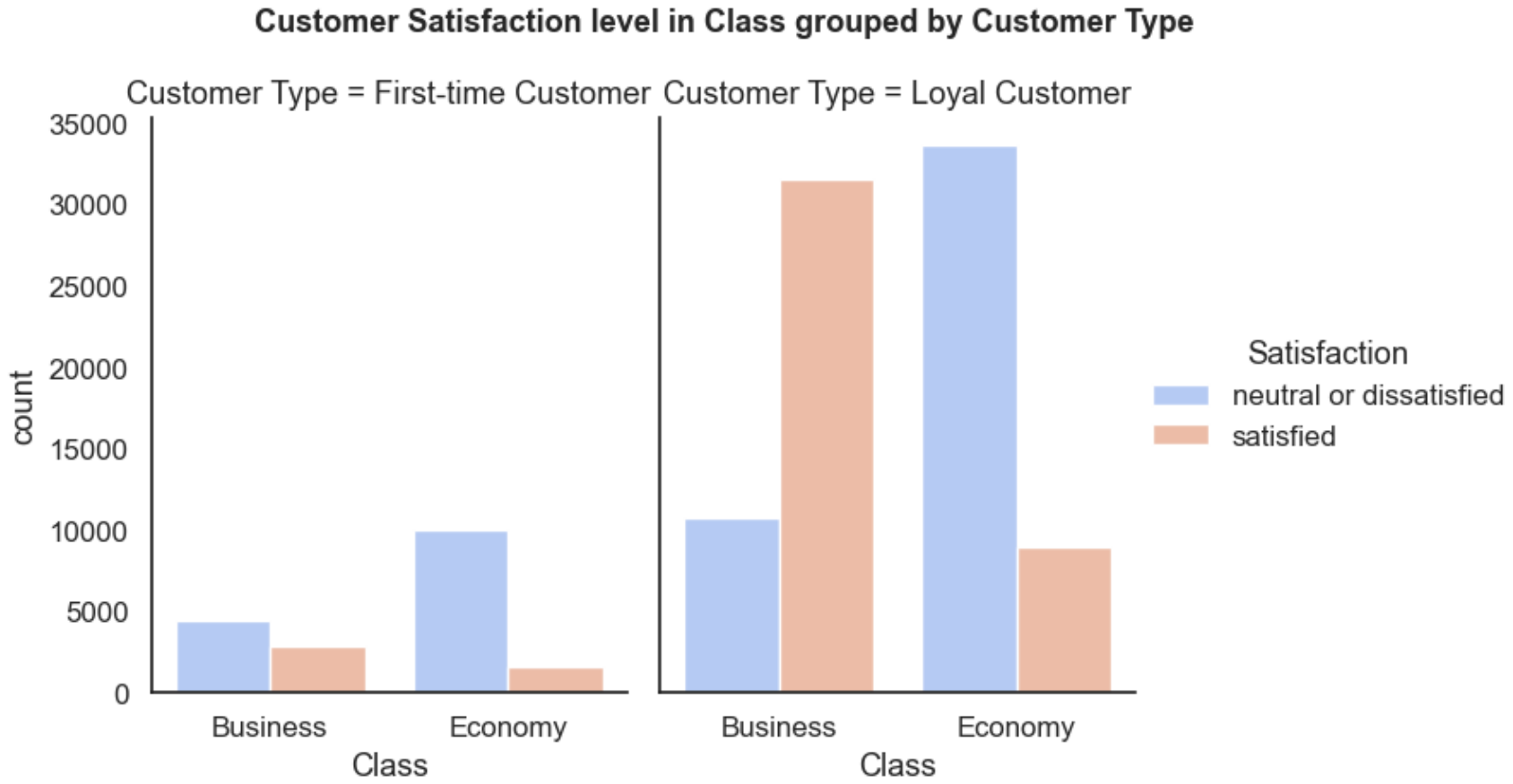
Customer Satisfaction level in relation to Class by Customer Type

In [913…

```
# Visualization of the Customer Satisfaction in relation to Class by Customer Type
sns.set(style='white',font_scale=1.1)
fig_ct = plt.figure(figsize=(4,3))
axt = fig.add_axes([1,1,2,2])
```

```
sns.catplot(data = df_train, x = 'Class', hue = 'Satisfaction', kind = 'count', col = 'Customer Type', palette = 'coolw
plt.suptitle('Customer Satisfaction level in Class grouped by Customer Type', y = 1.05, weight='bold', fontsize = 13)
wrap_labels(axt, 10)

plt.show()
```

```
<Figure size 400x300 with 0 Axes>
```



This plot confirms that Loyal Customers, who travel in Business Class, are likely more satisfied than the First-time customers who go in Business Class.

Customer Satisfaction level in relation to Class by Type of Travel

In [914…

```
# Visualization of the Customer Satisfaction in relation to Class by Type of Travel
sns.set(style='white',font_scale=1.1)
fig_ct = plt.figure(figsize=(4,3))
```

```
axt = fig.add_axes([1,1,2,2])
sns.catplot(data = df_train, x = 'Class', hue = 'Satisfaction', kind = 'count', col = 'Type of Travel', palette = 'cool
plt.suptitle('Customer Satisfaction level in Class grouped by Type of Travel', y = 1.05, weight='bold', fontsize = 13)
wrap_labels(axt, 10)

plt.show()
```

```
<Figure size 400x300 with 0 Axes>
```

This plot confirms that customers with Business Type of Travel, and who travel in Business Class, are likely more satisfied than the customers with Personal Type of Travel who go in Business Class.

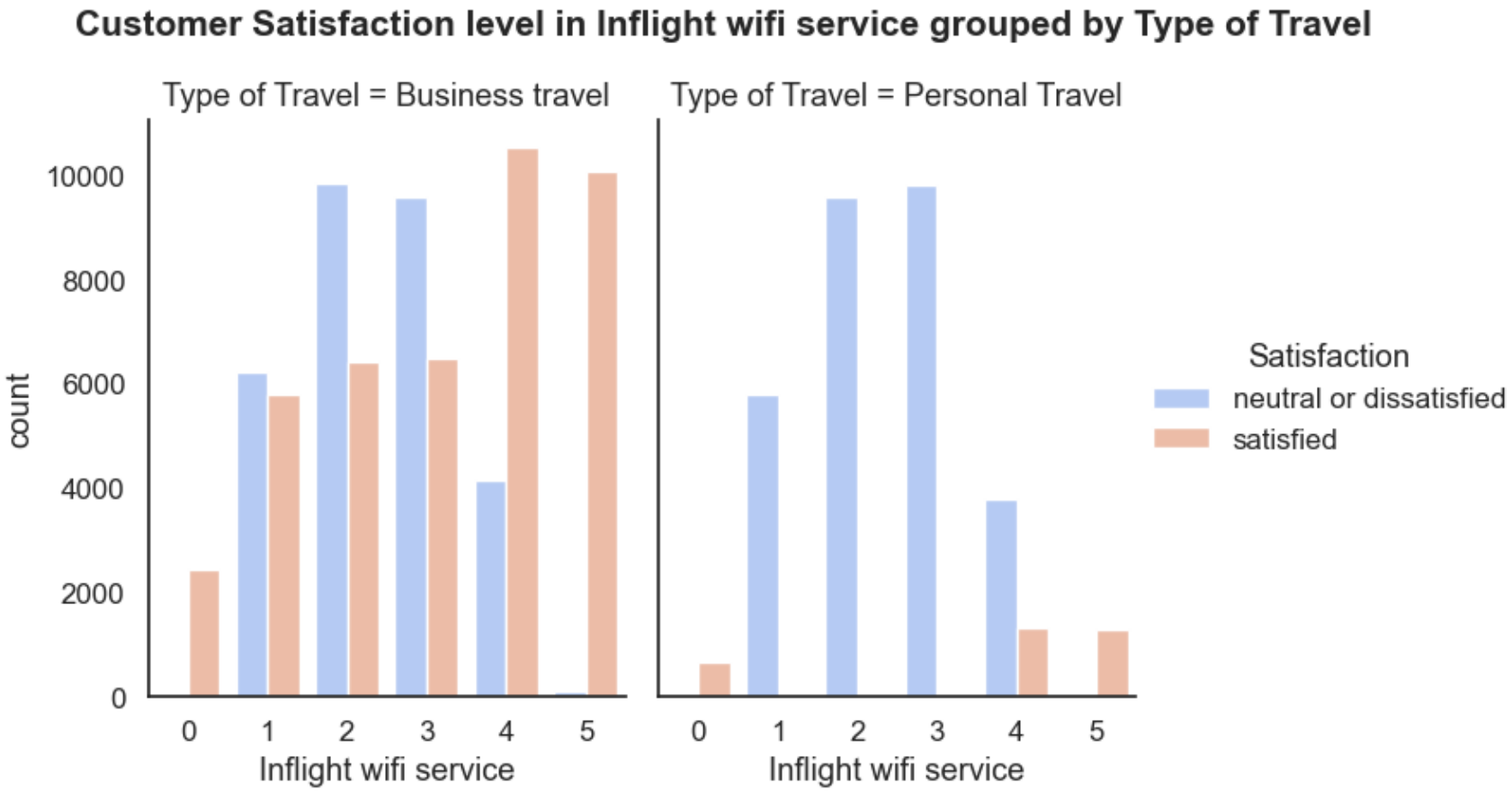
Customer Satisfaction level in relation to Inflight wifi service by Type of Travel

In [915…

```
# Visualization of the Customer Satisfaction in relation to Inflight wifi service by Type of Travel
sns.set(style='white',font_scale=1.1)
```

```
fig_ct = plt.figure(figsize=(4,3))
axt = fig.add_axes([1,1,2,2])
sns.catplot(data = df_train, x = 'Inflight wifi service', hue = 'Satisfaction', kind = 'count', col = 'Type of Travel',
plt.suptitle('Customer Satisfaction level in Inflight wifi service grouped by Type of Travel', y = 1.05, weight='bold',
wrap_labels(axt, 10)

plt.show()
```

```
<Figure size 400x300 with 0 Axes>
```



This plot shows that when customers with Business Type of Travel rates the Inflight wi-fi service between 1 and 3, the level of customer disatisfaction increases. Similarly, customers with Personal Type of Travel are more likely dissatisfied with rates between 1 and 3.

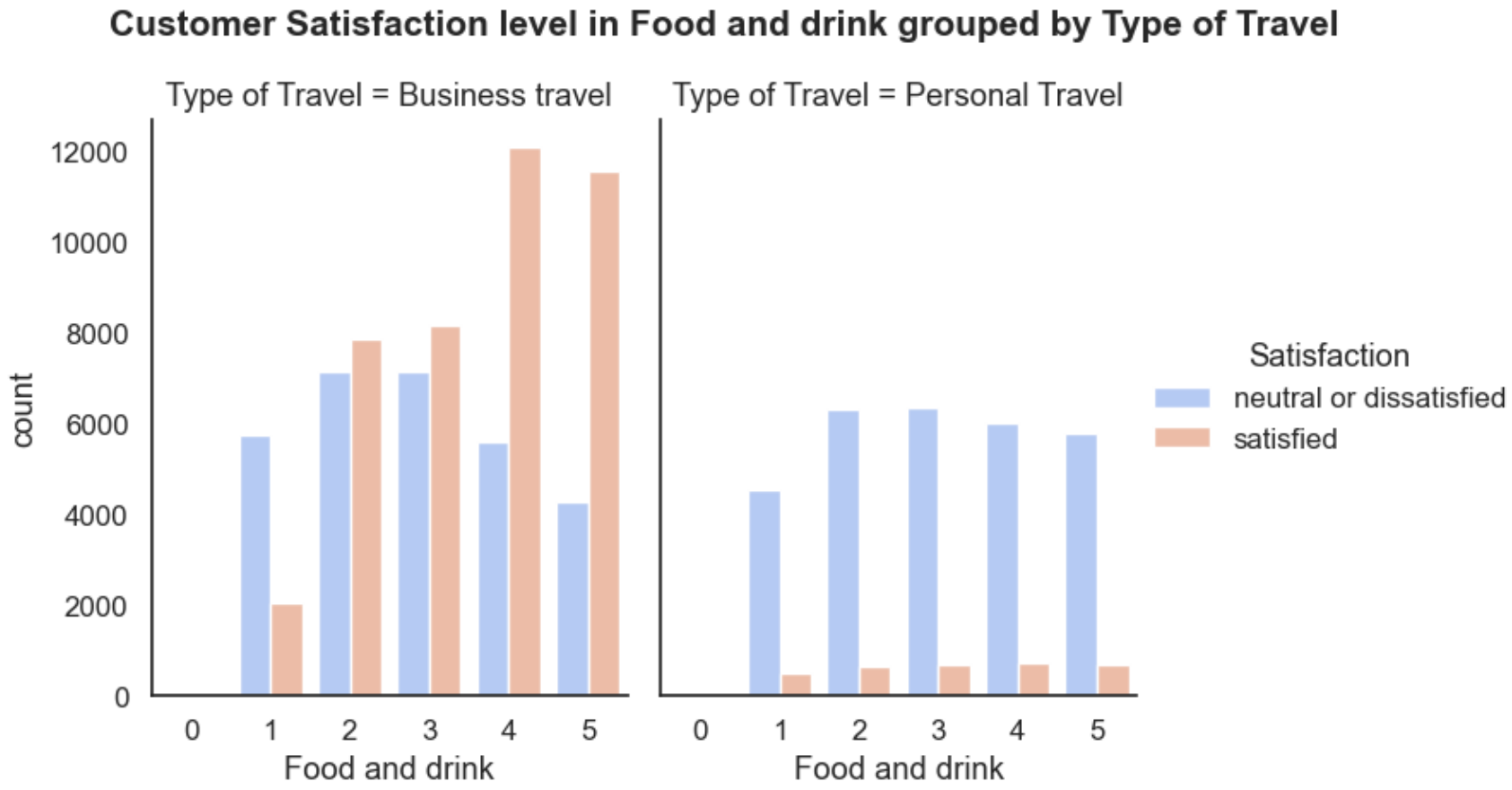
Customer Satisfaction level in relation to Food and drink by Type of Travel

In [916…

```
# Visualization of the Customer Satisfaction in relation to Food and drink by Type of Travel
```

```
sns.set(style='white',font_scale=1.1)
fig_ct = plt.figure(figsize=(4,3))
axt = fig.add_axes([1,1,2,2])
sns.catplot(data = df_train, x = 'Food and drink', hue = 'Satisfaction', kind = 'count', col = 'Type of Travel', palett
plt.suptitle('Customer Satisfaction level in Food and drink grouped by Type of Travel', y = 1.05, weight='bold', fontsi
wrap_labels(axt, 10)

plt.show()
```

```
<Figure size 400x300 with 0 Axes>
```

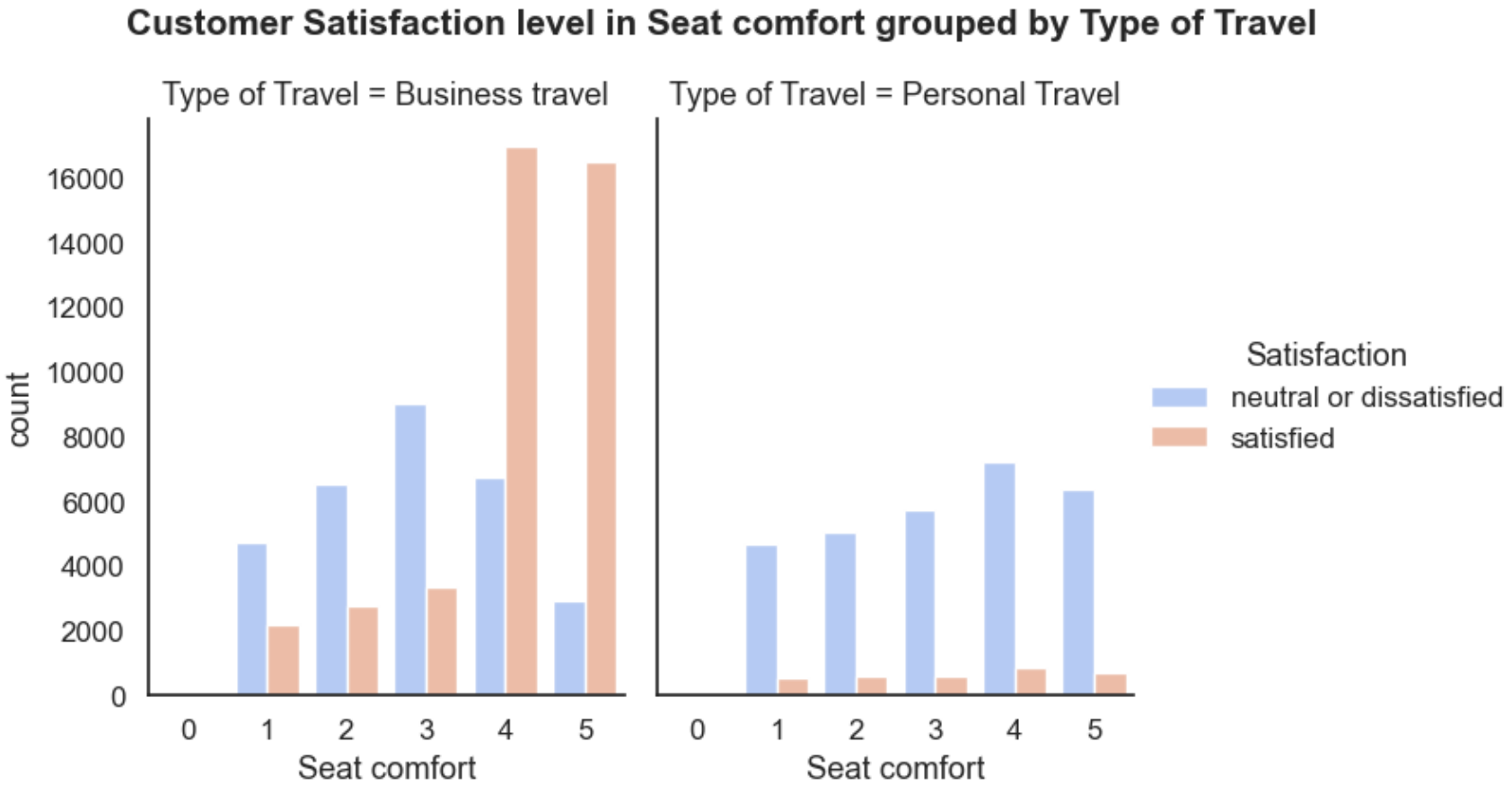


This plot shows that for Customer with Business Type of Travel, there is a high level of customer satisfaction when the Food and drink service is high rated, as should be expected.

Customer Satisfaction level in relation to Seat comfort by Type of Travel

In [917…
```
# Visualization of the Customer Satisfaction in relation to Seat comfort by Type of Travel
sns.set(style='white',font_scale=1.1)
fig_ct = plt.figure(figsize=(4,3))
axt = fig.add_axes([1,1,2,2])
sns.catplot(data = df_train, x = 'Seat comfort', hue = 'Satisfaction', kind = 'count', col = 'Type of Travel', palette
plt.suptitle('Customer Satisfaction level in Seat comfort grouped by Type of Travel', y = 1.05, weight='bold', fontsize
wrap_labels(axt, 10)

plt.show()
```

```
<Figure size 400x300 with 0 Axes>
```

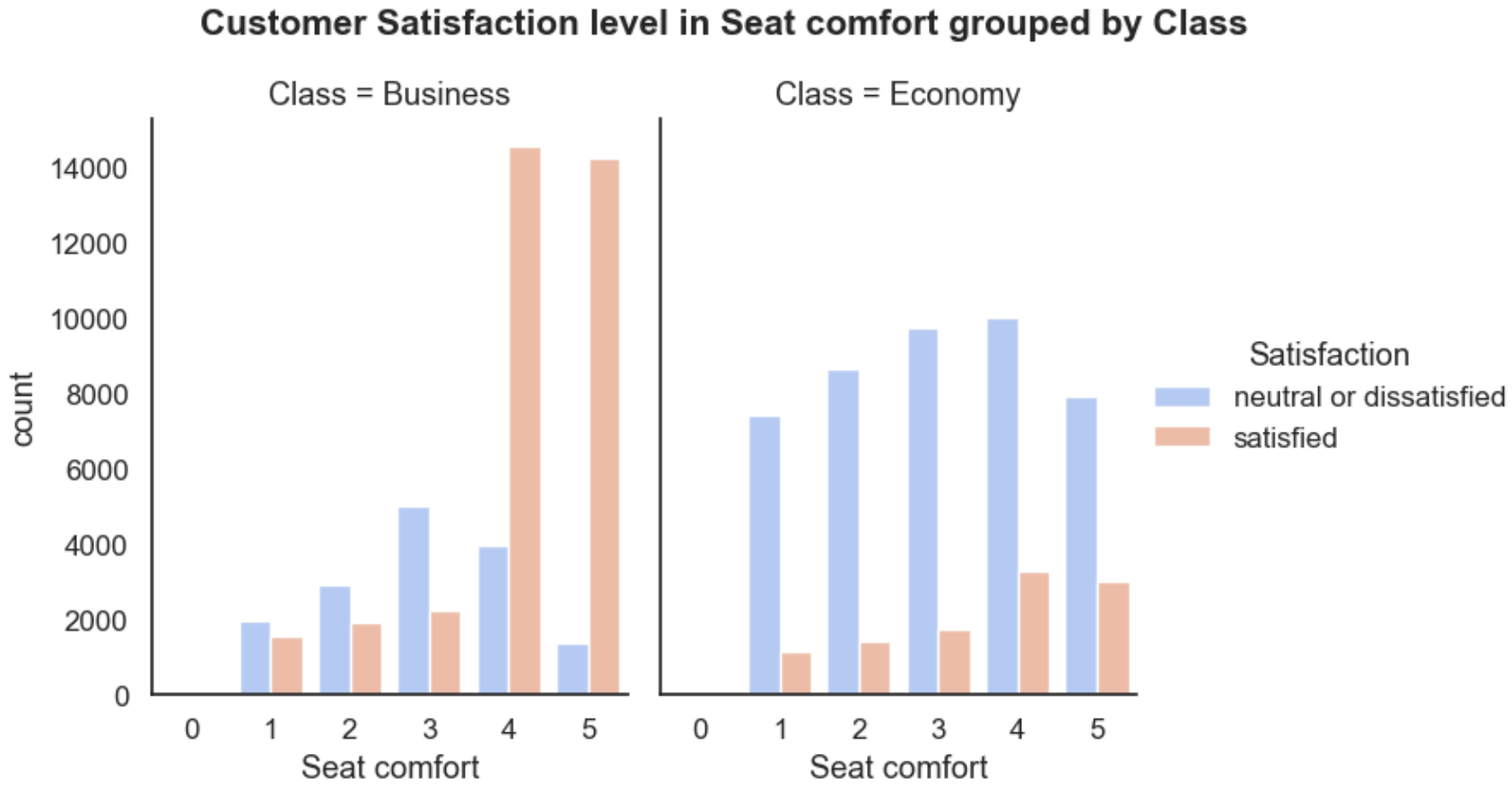


Customer Satisfaction level in relation to Seat comfort by Class

In [918…
```
# Visualization of the Customer Satisfaction in relation to Seat comfort by Class
sns.set(style='white',font_scale=1.1)
fig_ct = plt.figure(figsize=(4,3))
```

```
axt = fig.add_axes([1,1,2,2])
sns.catplot(data = df_train, x = 'Seat comfort', hue = 'Satisfaction', kind = 'count', col = 'Class', palette = 'coolwa
plt.suptitle('Customer Satisfaction level in Seat comfort grouped by Class', y = 1.05, weight='bold', fontsize = 15)
wrap_labels(axt, 10)

plt.show()
```

```
<Figure size 400x300 with 0 Axes>
```



This plot confirms that for Business Class, the level of customer stisfaction is high when the seat is rated as extremely comfortable.

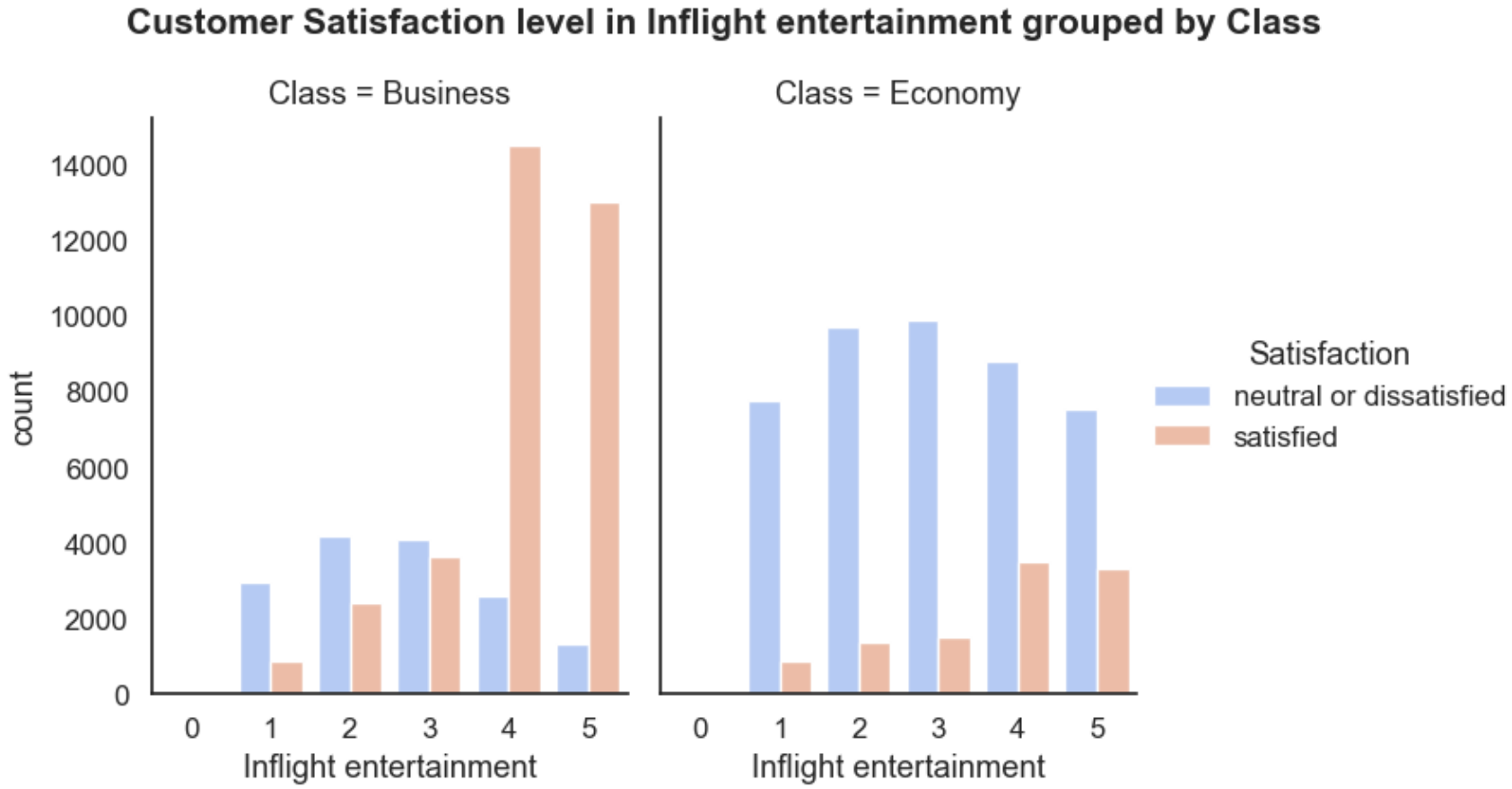
Customer Satisfaction level in relation to Inflight entertainment by Class

In [919…

```
# Visualization of the Customer Satisfaction in relation to Inflight entertainment by Class
sns.set(style='white',font_scale=1.1)
fig_ct = plt.figure(figsize=(4,3))
axt = fig.add_axes([1,1,2,2])
```

```
sns.catplot(data = df_train, x = 'Inflight entertainment', hue = 'Satisfaction', kind = 'count', col = 'Class', palette
plt.suptitle('Customer Satisfaction level in Inflight entertainment grouped by Class', y = 1.05, weight='bold', fontsiz
wrap_labels(axt, 10)

plt.show()
```

```
<Figure size 400x300 with 0 Axes>
```



This plot shows that the Inflight entertainment follows the same pattern as the Seat Comfort. For Business Class customers, there is a high level of customer satisfaction when the Inflight entertainment service is rated highly. On the other hand, Economy Class customers maintain a similar level of dissatisfaction regardless of their rating.

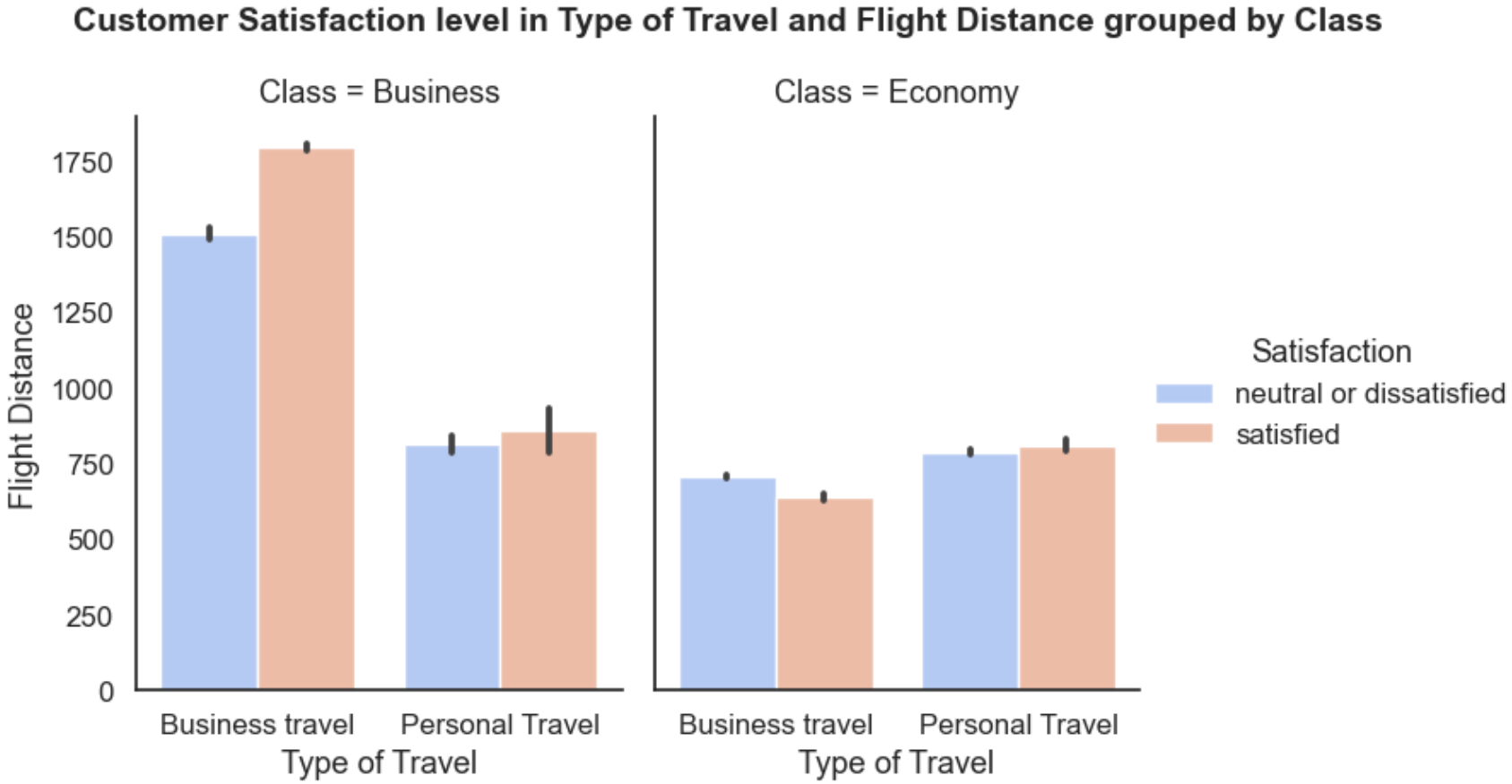
Customer Satisfaction level in relation to Type of Travel and Flight Distance by Class

In [920…

```
# Visualization of the Customer Satisfaction in relation to Type of Travel and Flight Distance by Class
sns.set(style='white',font_scale=1.1)
```

```
plt.figure(figsize=(4,3))
axt = fig.add_axes([1,1,2,2])
sns.catplot(data = df_train, x = 'Type of Travel', y = 'Flight Distance', hue = 'Satisfaction', col = 'Class', kind = '
plt.suptitle('Customer Satisfaction level in Type of Travel and Flight Distance grouped by Class', y = 1.05, weight='bo
wrap_labels(axt, 10)

plt.show()
```

```
<Figure size 400x300 with 0 Axes>
```



This plot shows that for customers in Business Type of Travel who travel in Business Class, there is a higher level of customer satisfaction for longer Flight Distance.

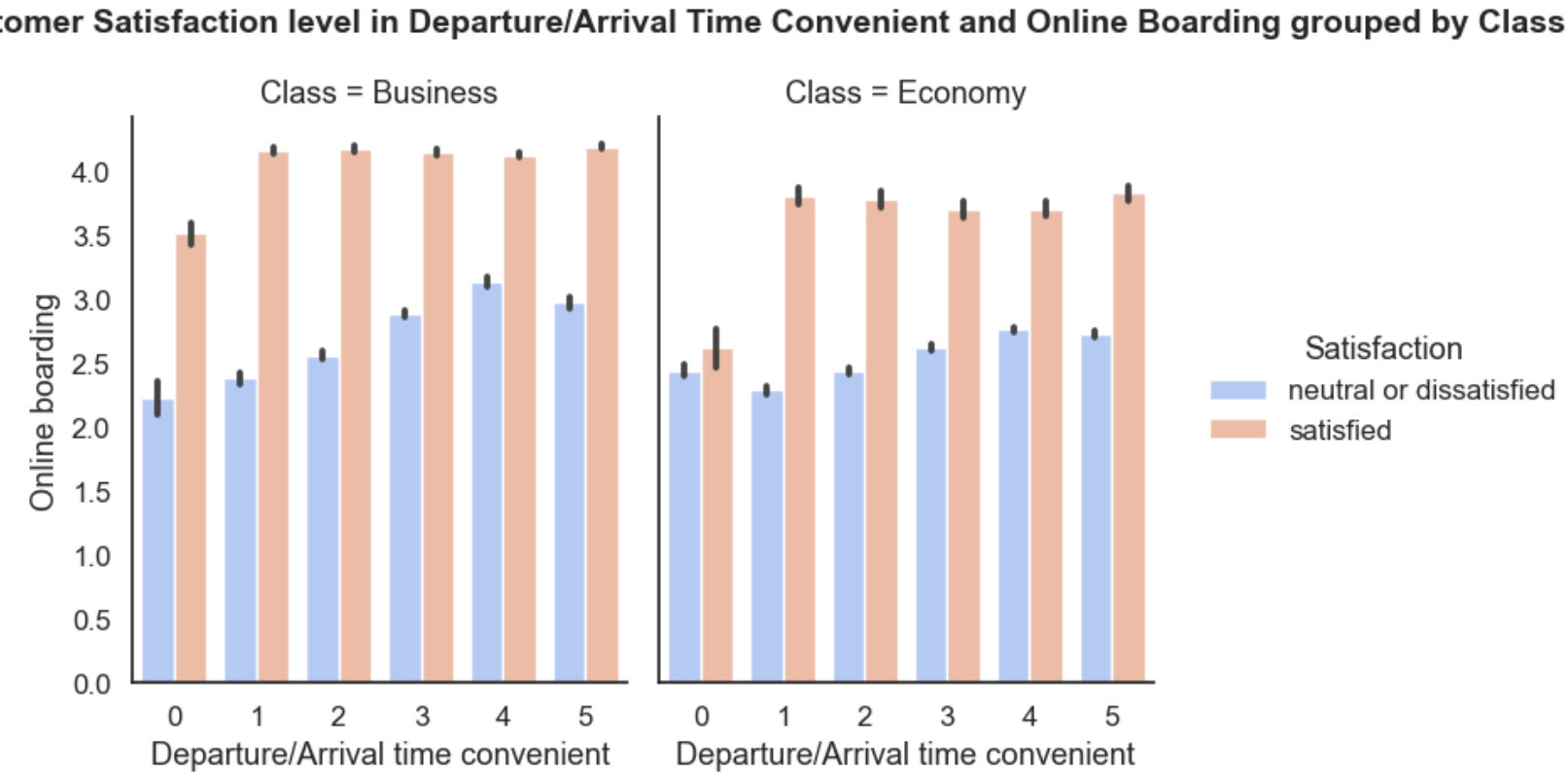
Customer Satisfaction level in relation to Departure/Arrival Time Convenient and Online Boarding by Class

In [921…

```
# Visualization of the Customer Satisfaction in relation to Departure/Arrival Time Convenient and Online Boarding by Cl
```

```
sns.set(style='white',font_scale=1.1)
plt.figure(figsize=(4,3))
axt = fig.add_axes([1,1,2,2])
sns.catplot(data = df_train, x = 'Departure/Arrival time convenient', y = 'Online boarding', hue = 'Satisfaction', col
plt.suptitle('Customer Satisfaction level in Departure/Arrival Time Convenient and Online Boarding grouped by Class', y
wrap_labels(axt, 10)

plt.show()
```

```
<Figure size 400x300 with 0 Axes>
```



This plot shows that, in general, for both Business and Economy Class, there is a high level of customer satisfaction when doing Online boarding even when the Departure/arrival time is inconvenient.

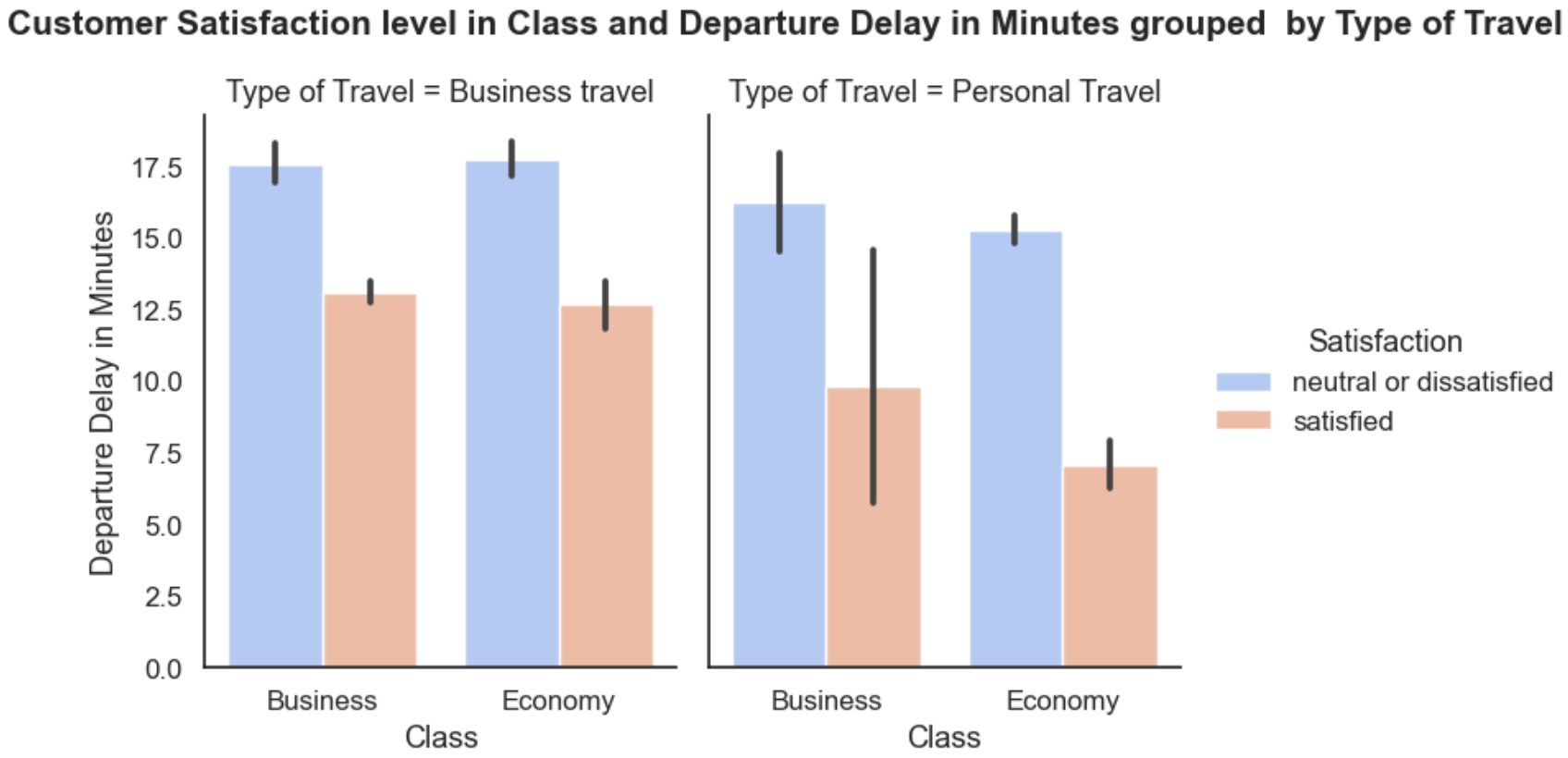
Customer Satisfaction level in relation to Class and Departure Delay in Minutes by Type of Travel

In [922…

```
# Visualization of the Customer Satisfaction level in relation to Class and Departure Delay in Minutes by Type of Trave
sns.set(style='white',font_scale=1.1)
fig_ct = plt.figure(figsize=(4,3))
```

```
axt = fig.add_axes([1,1,2,2])
sns.catplot(data = df_train, x = 'Class', y ='Departure Delay in Minutes', hue = 'Satisfaction', kind = 'bar', col = 'T
plt.suptitle('Customer Satisfaction level in Class and Departure Delay in Minutes grouped  by Type of Travel', y = 1.05
wrap_labels(axt, 10)

plt.show()
```

```
<Figure size 400x300 with 0 Axes>
```



This plot shows that for Personal Type of Travel, there is a high level of customer dissatisfaction when the Departure Delay in minutes is high.

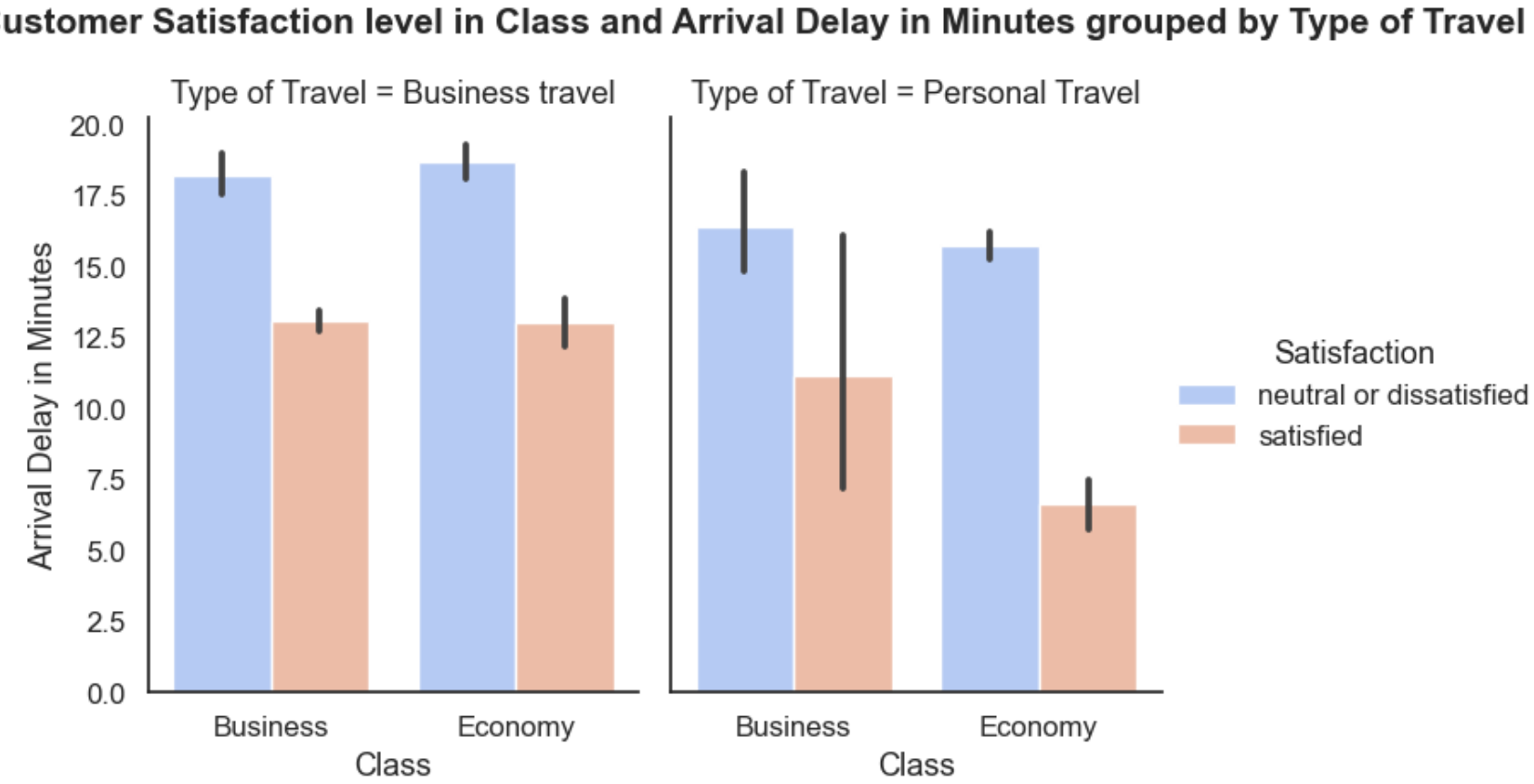
Customer Satisfaction level in relation to Class and Arrival Delay in Minutes by Type of Travel

In [923…

```
# Visualization of the Customer Satisfaction level in relation to Class and Arrival Delay in Minutes by Type of Travel
sns.set(style='white',font_scale=1.1)
fig_ct = plt.figure(figsize=(4,3))
axt = fig.add_axes([1,1,2,2])
```

```
sns.catplot(data = df_train, x = 'Class', y ='Arrival Delay in Minutes', hue = 'Satisfaction', kind = 'bar', col = 'Typ
plt.suptitle('Customer Satisfaction level in Class and Arrival Delay in Minutes grouped by Type of Travel', y = 1.05, w
wrap_labels(axt, 10)

plt.show()
```

```
<Figure size 400x300 with 0 Axes>
```



This plot shows the same pattern as for Departure Delay in Minutes feature, as is expected: For Personal Type of Travel, there is a high level of customer dissatisfaction when the Arrival Delay in minutes is high.

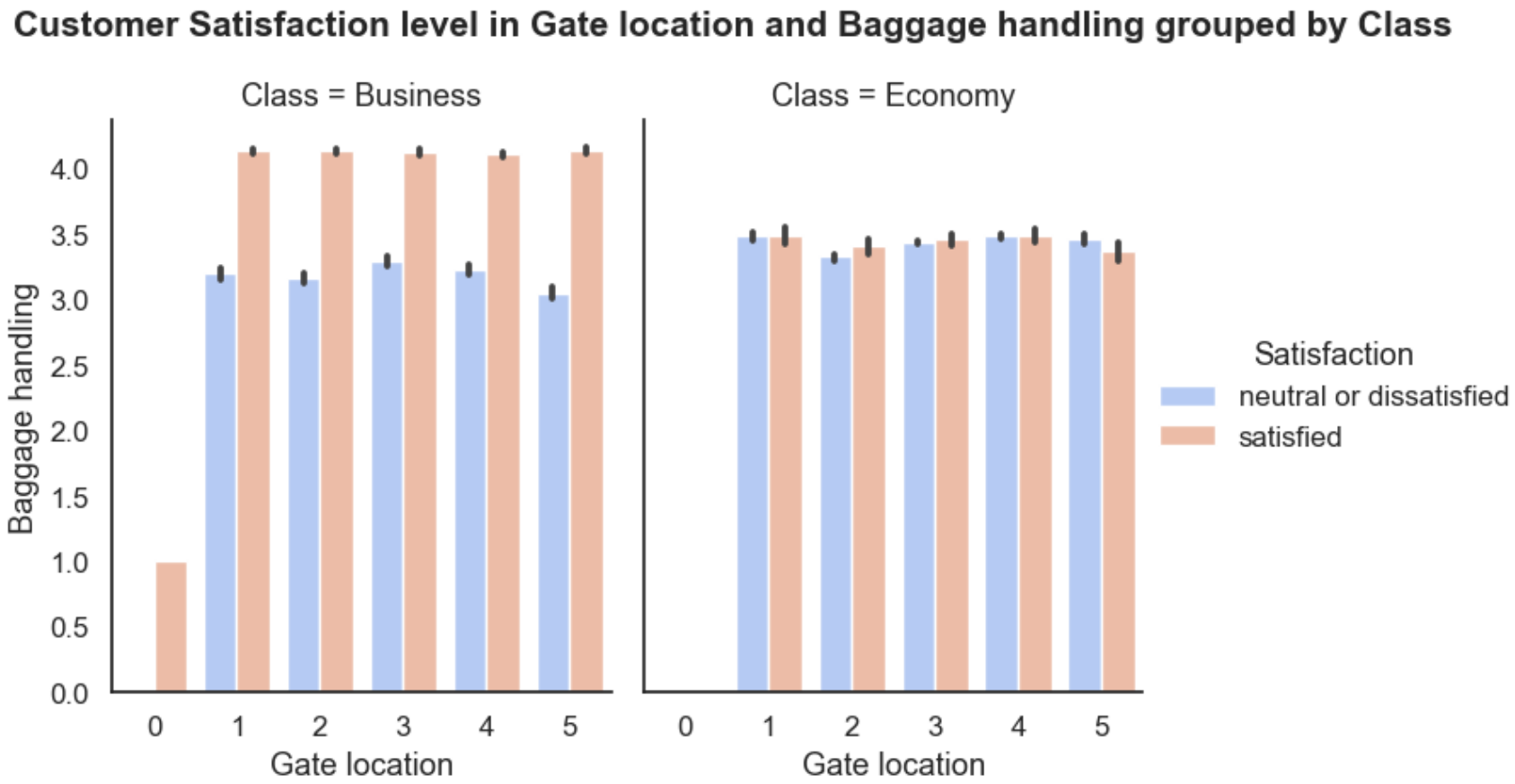
Customer Satisfaction level in relation to Gate location and Baggage handling by Class

In [924…

```
# Visualization of the Customer Satisfaction level in relation to  Gate location and Baggage handling by Class
sns.set(style='white',font_scale=1.1)
fig_ct = plt.figure(figsize=(4,3))
axt = fig.add_axes([1,1,2,2])
```

```
sns.catplot(data = df_train, x = 'Gate location', y ='Baggage handling', hue = 'Satisfaction', kind = 'bar', col = 'Cla
plt.suptitle('Customer Satisfaction level in Gate location and Baggage handling grouped by Class', y = 1.05, weight='bo
wrap_labels(axt, 10)

plt.show()
```

```
<Figure size 400x300 with 0 Axes>
```



This plot shows that for customers in Business Class there is a high customer satisfaction level in all Gate locations scores

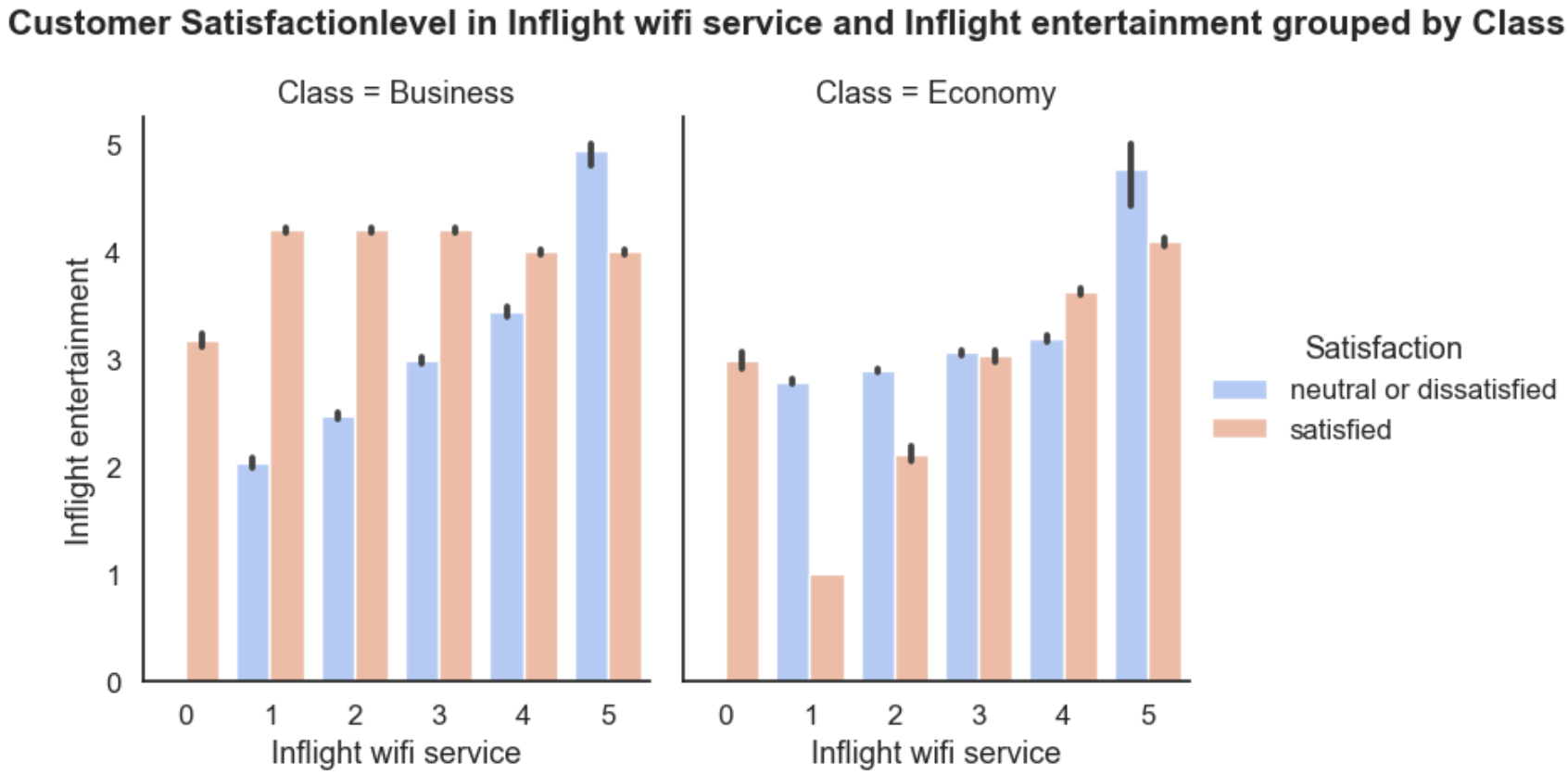
Customer Satisfaction level in relation to Inflight wifi service and Inflight entertainment by Class

In [925…

```
# Visualization of the Customer Satisfaction level in relation to Inflight wifi service and Inflight entertainment by C
sns.set(style='white',font_scale=1.1)
fig_ct = plt.figure(figsize=(4,3))
axt = fig.add_axes([1,1,2,2])
sns.catplot(data = df_train, x = 'Inflight wifi service', y ='Inflight entertainment', hue = 'Satisfaction', kind = 'ba
```

```
plt.suptitle('Customer Satisfactionlevel in Inflight wifi service and Inflight entertainment grouped by Class', y = 1.0
wrap_labels(axt, 10)

plt.show()
```

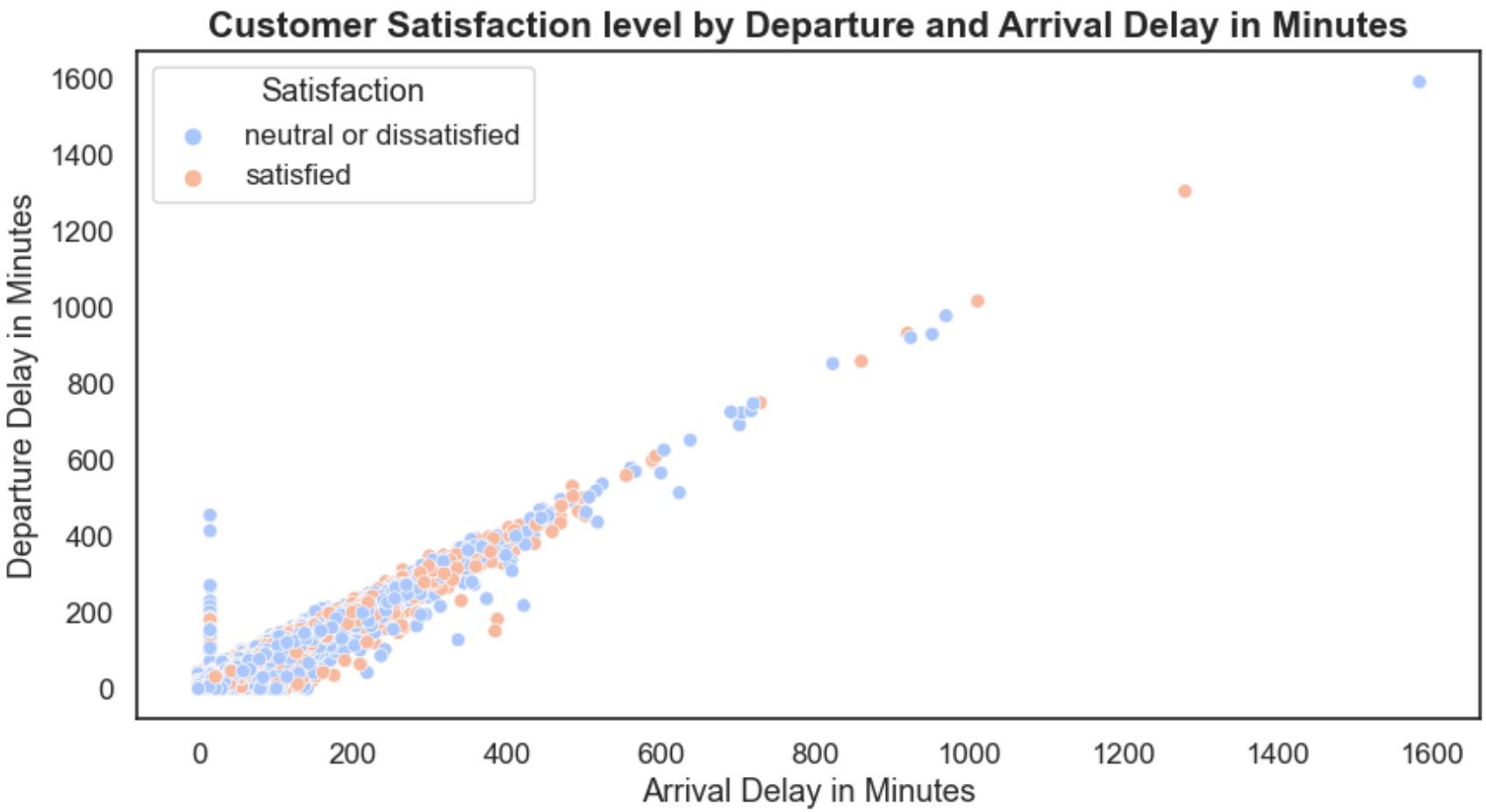
```
<Figure size 400x300 with 0 Axes>
```



Customer Satisfaction level by Departure and Arrival Delay in Minutes

In [926…

```
# Visualization of Customer Satisfaction in relation to Departure and Arrival Delay in Minutes
plt.figure(figsize=(10,5), dpi = 100)
ax_ad = fig.add_axes([0,0,1,1])
sns.scatterplot(data = df_train, x = 'Arrival Delay in Minutes', y = 'Departure Delay in Minutes', hue = 'Satisfaction'
plt.title('Customer Satisfaction level by Departure and Arrival Delay in Minutes', weight='bold',fontsize='15')
ax.set_xlabel('Arrival Delay in Minutes', fontsize=14, weight='semibold')
ax.set_ylabel('Departure Delay in Minutes', fontsize=15, weight='semibold')
wrap_labels(ax_ad, 10)
```

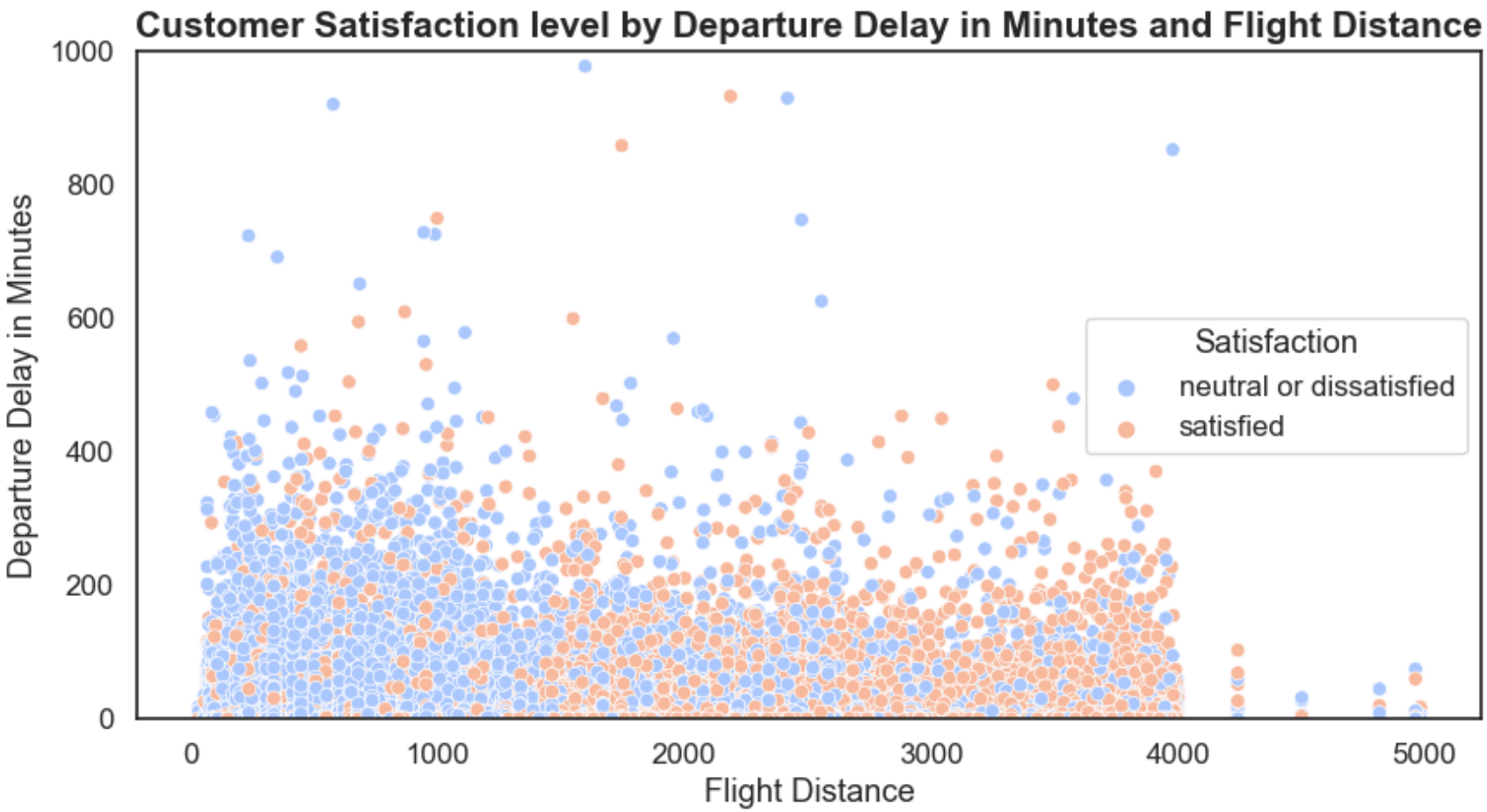


This scatterplot shows and confirms that the Departure and Arrival Delay in minutes have a linear relationship.

Customer Satisfaction level by Departure Delay in Minutes and Flight Distance

In [927…

```
# Visualization of Customer Satisfaction in relation to Flight Distance and Departure Delay in Minutes
plt.figure(figsize=(10,5), dpi = 100)
ax_fd = fig.add_axes([0,0,1,1])
sns.scatterplot(data = df_train, x = 'Flight Distance', y = 'Departure Delay in Minutes', hue = 'Satisfaction', palette
plt.title('Customer Satisfaction level by Departure Delay in Minutes and Flight Distance', weight='bold',fontsize='15')
ax.set_xlabel('Flight Distance', fontsize=14, weight='semibold')
ax.set_ylabel('Departure Delay in Minutes', fontsize=15, weight='semibold')
plt.ylim(0, 1000)
wrap_labels(ax_fd, 10)
```



This scatterplot shows that when the flight distance is small, the airline customers are more likely to be neutral/dissatisfied with a delay in the departure.

Dimensionality Reduction

Feature Selection

In [928…

```
# Encoding Categorical Variables
# Label Encoding in train dataset (transform categorical data)
from sklearn.preprocessing import LabelEncoder
lencoders = {}
for col in df_train.select_dtypes(include =['object']).columns:
```

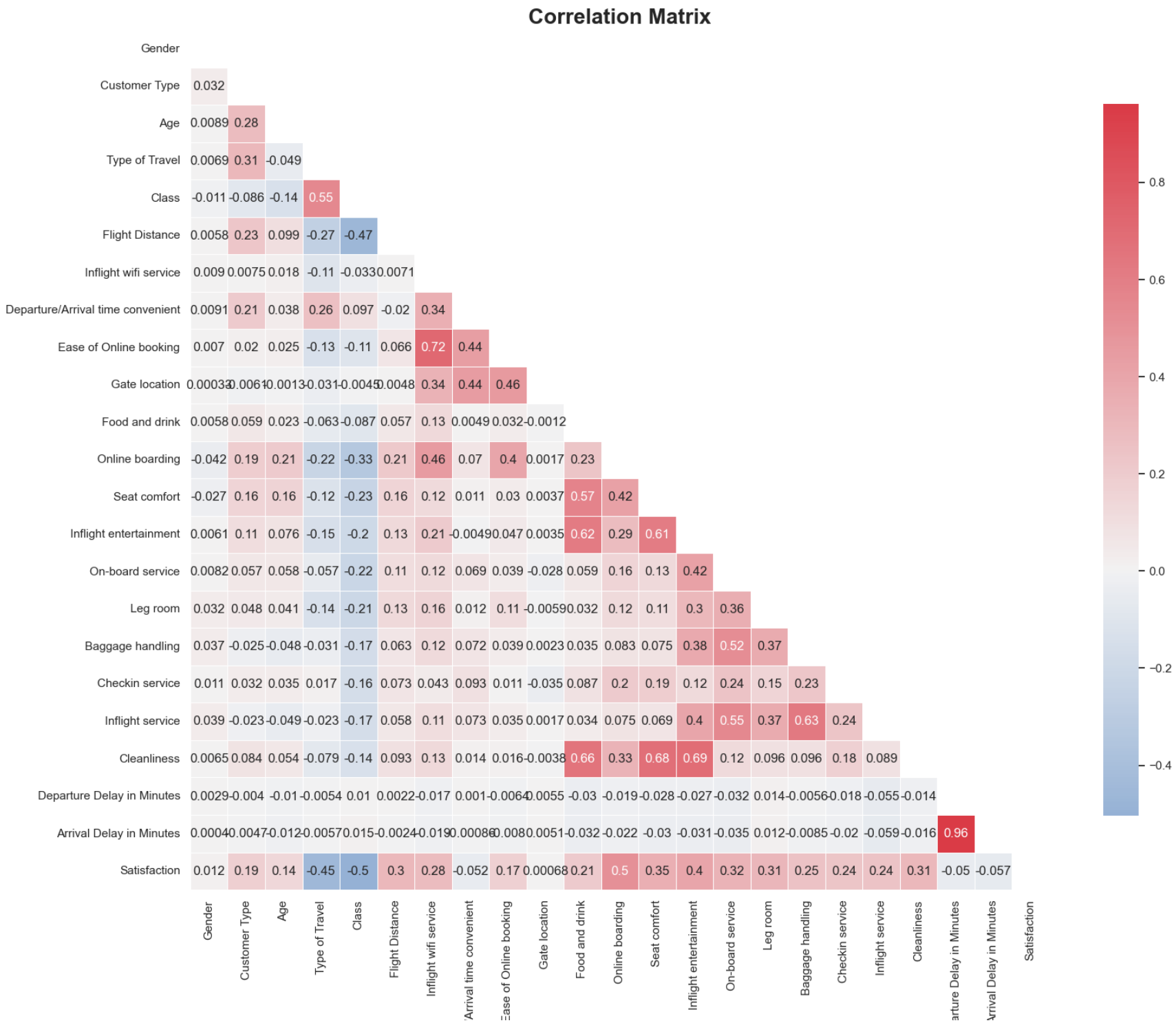
```
    lencoders[col] = LabelEncoder()
    df_train[col] = lencoders[col].fit_transform(df_train[col])
```

In [929…
```
# Label Encoding in test dataset
lencoders = {}
for col in df_test.select_dtypes(include =['object']).columns:
    lencoders[col] = LabelEncoder()
    df_test[col] = lencoders[col].fit_transform(df_test[col])
```

Correlation among features

In [930…
```
# Compute correlation
sns.set_theme(style="white")
corr = df_train.corr()
mask = np.triu(np.ones_like(corr, dtype=np.bool))
f, ax = plt.subplots(figsize=(18, 15))
plt.title('Correlation Matrix',weight='bold',fontsize='20')
cmap = sns.diverging_palette(250, 10, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=None, center=0,
            square=True, annot = True, linewidths=.5, cbar_kws={"shrink": .8})
```

Out[930]:
```
<AxesSubplot:title={'center':'Correlation Matrix'}>
```


Correlation Matrix
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
Baggage handling
Checkin service
Inflight service
Cleanliness
Departure Delay in Minutes
Arrival Delay in Minutes
Satisfaction
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

This correlation matrix shows that the Satisfaction target feature has some weak correlation with all other features, except for Gender and Gate location features. Moreover, it shows the following positive correlation between features: Ease of Online booking and Inflight wifi service (0.72), Cleanliness and Inflight entertainment (0.69), Cleanliness and Seat Comfort (0.68), and Departure and Arrival Delay in Minutes (0.96)

Feature Importance

In [931…
```
# Importance features from correlation matrix
df_train.corr().abs()['Satisfaction'].sort_values(ascending = False)
```

Out[931]:
```
Satisfaction                       1.000000
Class                              0.503848
Online boarding                    0.503557
Type of Travel                     0.449000
Inflight entertainment             0.398059
Seat comfort                       0.349459
On-board service                   0.322383
Leg room                           0.313131
Cleanliness                        0.305198
Flight Distance                    0.298780
Inflight wifi service              0.284245
Baggage handling                   0.247749
Inflight service                   0.244741
Checkin service                    0.236174
Food and drink                     0.209936
Customer Type                      0.187638
Ease of Online booking             0.171705
Age                                0.137167
Arrival Delay in Minutes           0.057497
Departure/Arrival time convenient  0.051601
Departure Delay in Minutes         0.050494
Gender                             0.012211
Gate location                      0.000682
Name: Satisfaction, dtype: float64
```

The features that have relatively high correlation with the Satisfaction target feature are: Class, Online boarding, Type of Travel, Inflight entertainment, Seat comfort, On-board service, Leg room, Cleanliness, Flight Distance.

In [932…
```
X = df_train.drop('Satisfaction', axis = 1)
```

```
y = df_train['Satisfaction']

# Feature Importance using Permutation Importance
perm = PermutationImportance(rf(n_estimators=100, random_state=0).fit(X,y),random_state=1).fit(X,y)
eli5.show_weights(perm, feature_names = X.columns.tolist())
```

Out[932]:

Weight	Feature
0.1498 ± 0.0015	Inflight wifi service
0.1394 ± 0.0016	Type of Travel
0.0537 ± 0.0008	Customer Type
0.0435 ± 0.0011	Online boarding
0.0288 ± 0.0006	Checkin service
0.0266 ± 0.0002	Class
0.0244 ± 0.0002	Baggage handling
0.0232 ± 0.0007	Seat comfort
0.0198 ± 0.0004	Inflight service
0.0153 ± 0.0003	Cleanliness
0.0124 ± 0.0005	On-board service
0.0124 ± 0.0004	Leg room
0.0120 ± 0.0003	Age
0.0107 ± 0.0002	Flight Distance
0.0079 ± 0.0003	Arrival Delay in Minutes
0.0078 ± 0.0005	Inflight entertainment
0.0054 ± 0.0001	Gate location
0.0047 ± 0.0003	Ease of Online booking
0.0046 ± 0.0001	Departure Delay in Minutes
0.0040 ± 0.0002	Departure/Arrival time convenient
… 2 more …	

Building Models

In [933…

```
features = ['Inflight wifi service', 'Type of Travel','Online boarding','Seat comfort','Flight Distance',
            'Inflight entertainment','On-board service','Leg room','Cleanliness','Checkin service', 'Customer Type', 'C
            'Inflight service', 'Baggage handling']
target = ['Satisfaction']

# Split into test and train
X_train = df_train[features]
y_train = df_train[target].to_numpy()
X_test = df_test[features]
y_test = df_test[target].to_numpy()
```

In [934…

```
# Normalize Features
scaler = StandardScaler()
```

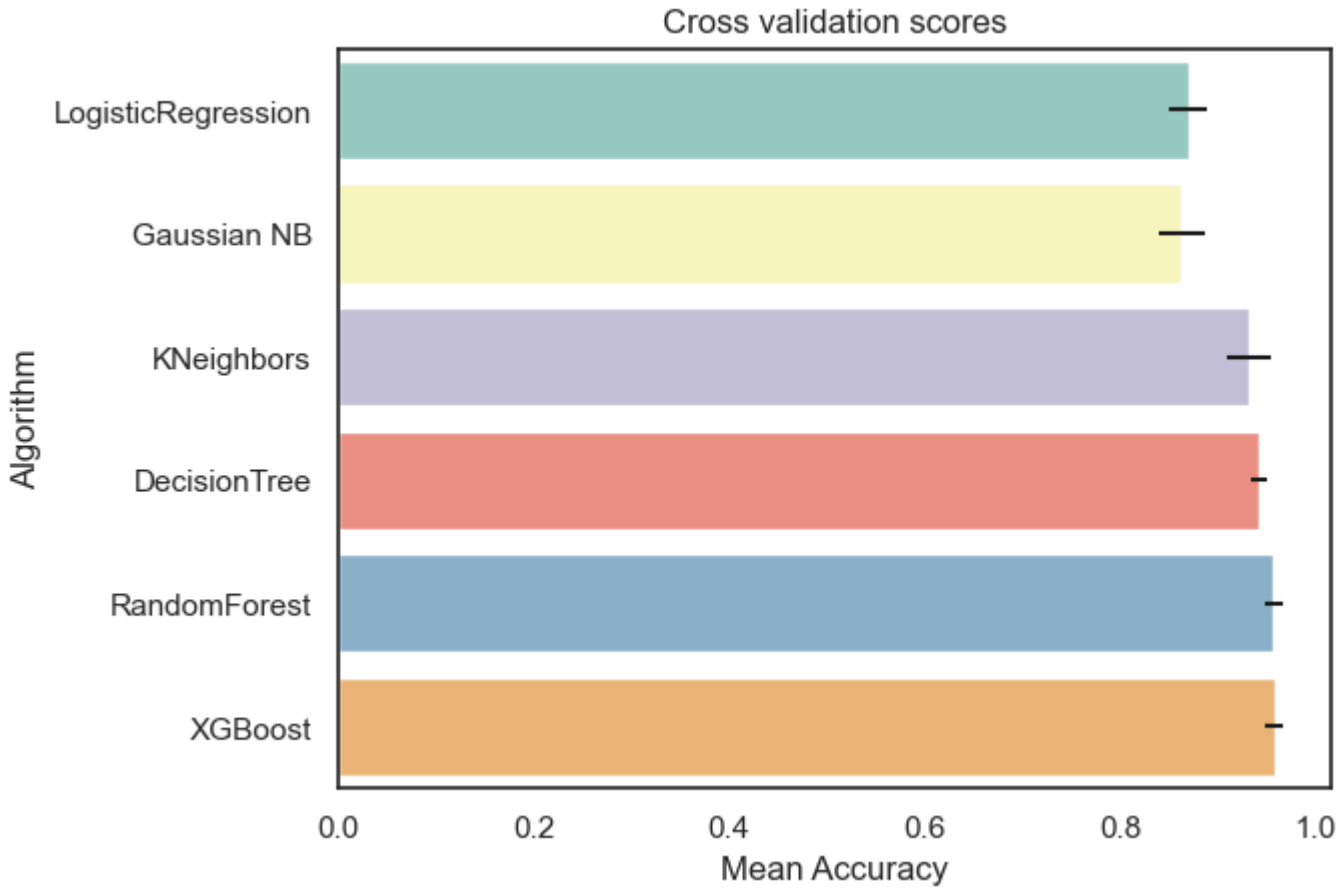
```
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

In [935…

```
# Stratified Cross validation
kfold = StratifiedKFold(n_splits = 10)
```

In [936…

```
# Modeling step to test differents algorithms
random_state = 2
classifiers = []
classifiers.append(LogisticRegression(random_state = random_state))
classifiers.append(GaussianNB())
classifiers.append(KNeighborsClassifier())
classifiers.append(DecisionTreeClassifier(random_state=random_state))
classifiers.append(RandomForestClassifier(random_state=random_state))
classifiers.append(XGBClassifier(random_state=random_state))

cv_results = []
for classifier in classifiers :
    kfold = StratifiedKFold(n_splits = 10)
    cv_results.append(cross_val_score(classifier, X_train, y = y_train, scoring = "accuracy", cv = kfold, n_jobs=4))

cv_means = []
cv_std = []
for cv_result in cv_results:
    cv_means.append(cv_result.mean())
    cv_std.append(cv_result.std())

cv_res = pd.DataFrame({"CrossValMeans":cv_means,"CrossValerrors": cv_std,"Algorithm":["LogisticRegression", "Gaussian N
"RandomForest","XGBoost"]})

g = sns.barplot("CrossValMeans","Algorithm",data = cv_res, palette="Set3",orient = "h",**{'xerr':cv_std})
g.set_xlabel("Mean Accuracy")
g = g.set_title("Cross validation scores")
```



In [937…

```
def run_model(model, X_train, y_train, X_test, y_test, verbose=True):
    t0=time.time()
    if verbose == False:
        model.fit(X_train,y_train.ravel(), verbose=0)
    else:
        model.fit(X_train,y_train.ravel())
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, y_pred)
    time_taken = time.time()-t0
    print("Accuracy = {}".format(accuracy))
    print("ROC Area under Curve = {}".format(roc_auc))
    print("Time taken = {}".format(time_taken))
    print(classification_report(y_test,y_pred,digits=5))
    plot_confusion_matrix(model, X_test, y_test,cmap=plt.cm.Blues, normalize = 'all')
    plot_roc_curve(model, X_test, y_test)
```

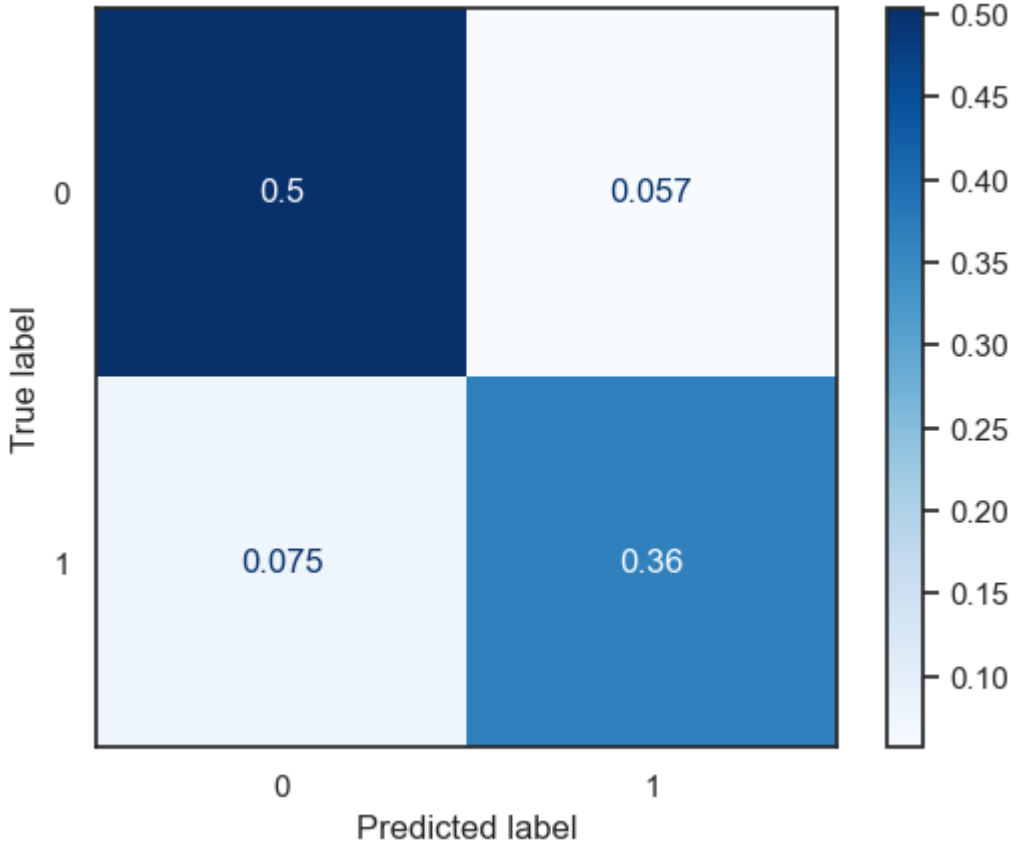
```
    return model, accuracy, roc_auc, time_taken
```

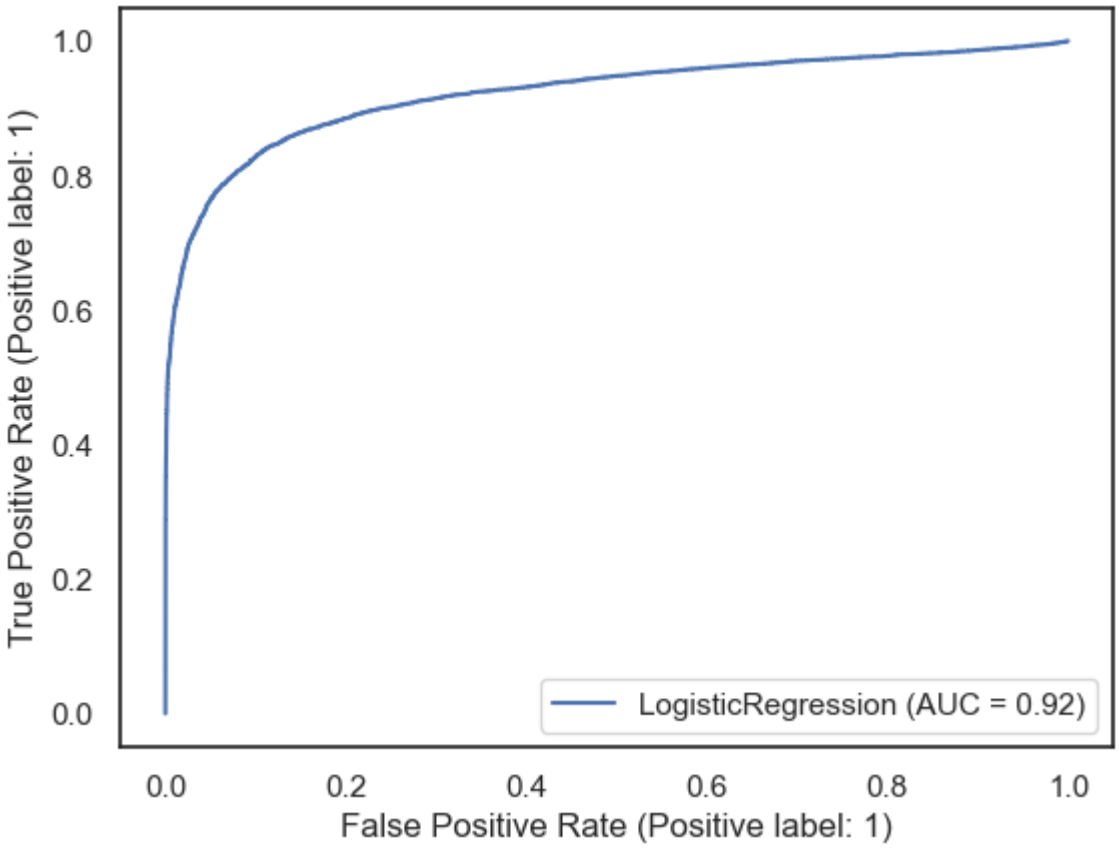
Logistic Regression

In [938…

```
# Logistic Regression

params_lr = {'penalty': 'elasticnet', 'l1_ratio':0.5, 'solver': 'saga'}

model_lr = LogisticRegression(**params_lr)
model_lr, accuracy_lr, roc_auc_lr, tt_lr = run_model(model_lr, X_train, y_train, X_test, y_test)
```

```
Accuracy = 0.8684170003079766
ROC Area under Curve = 0.864262601129349
Time taken = 0.8264930248260498
              precision    recall  f1-score   support

           0    0.87117   0.89831   0.88453     14573
           1    0.86465   0.83022   0.84708     11403

    accuracy                        0.86842     25976
   macro avg    0.86791   0.86426   0.86581     25976
weighted avg    0.86830   0.86842   0.86809     25976
```


0.5
0.057
0.075
0.36
0
1
True label
0
1
Predicted label
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10

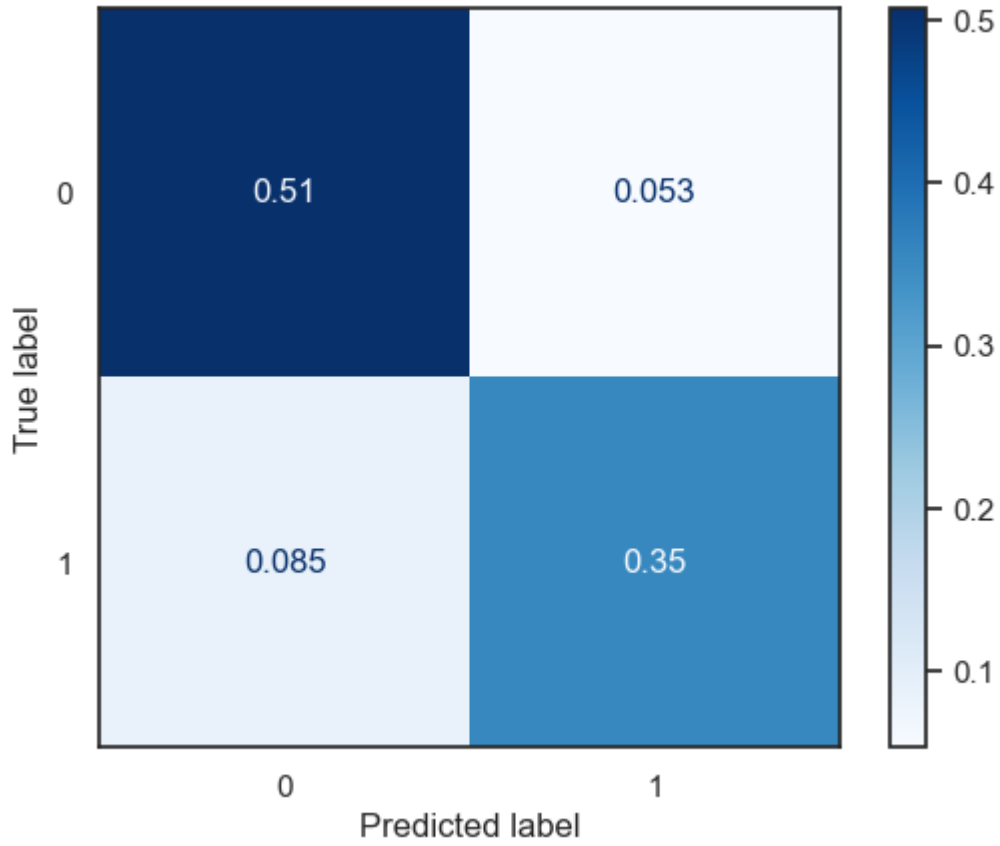


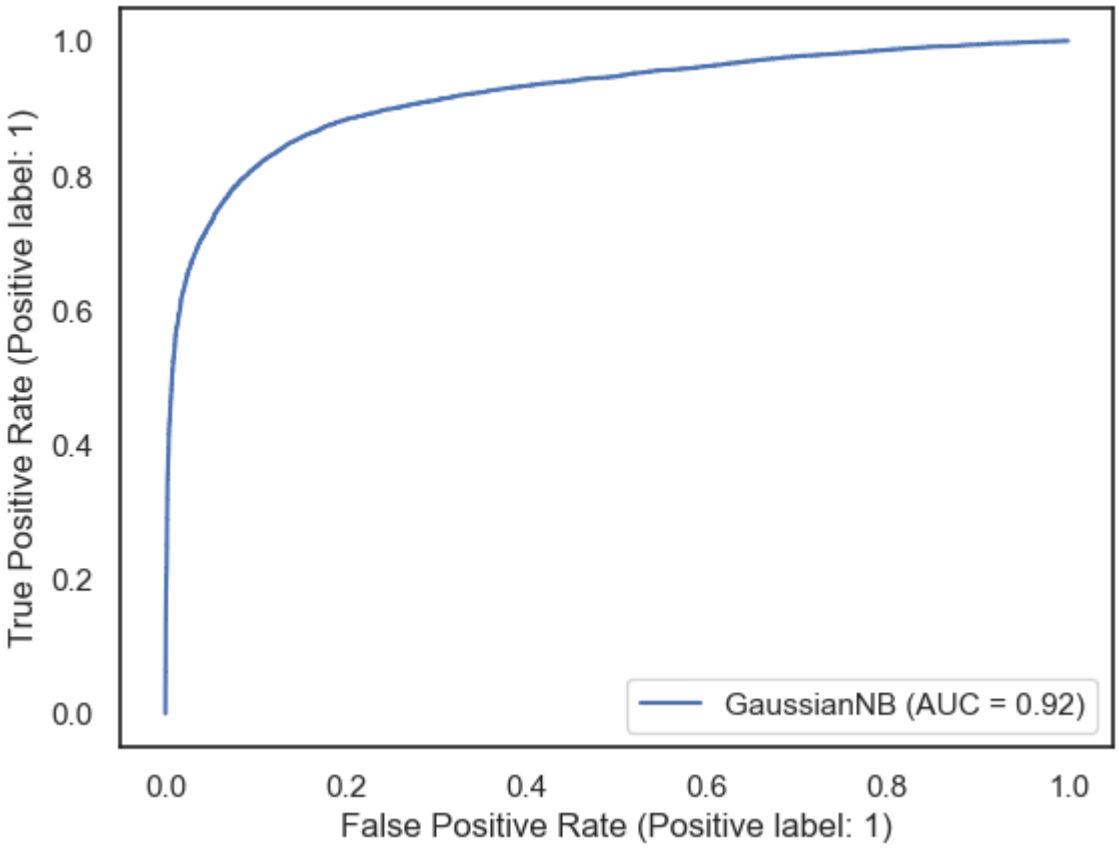
Naive Bayes

In [939…

```
params_nb = {}

model_nb = GaussianNB(**params_nb)
model_nb, accuracy_nb, roc_auc_nb, tt_nb = run_model(model_nb, X_train, y_train, X_test, y_test)
```

```
Accuracy = 0.8619494918386202
ROC Area under Curve = 0.8559804624880696
Time taken = 0.05875658988952637
              precision    recall  f1-score   support

           0    0.85702   0.90489   0.88031     14573
           1    0.86911   0.80707   0.83694     11403

    accuracy                        0.86195     25976
   macro avg    0.86307   0.85598   0.85862     25976
weighted avg    0.86233   0.86195   0.86127     25976
```



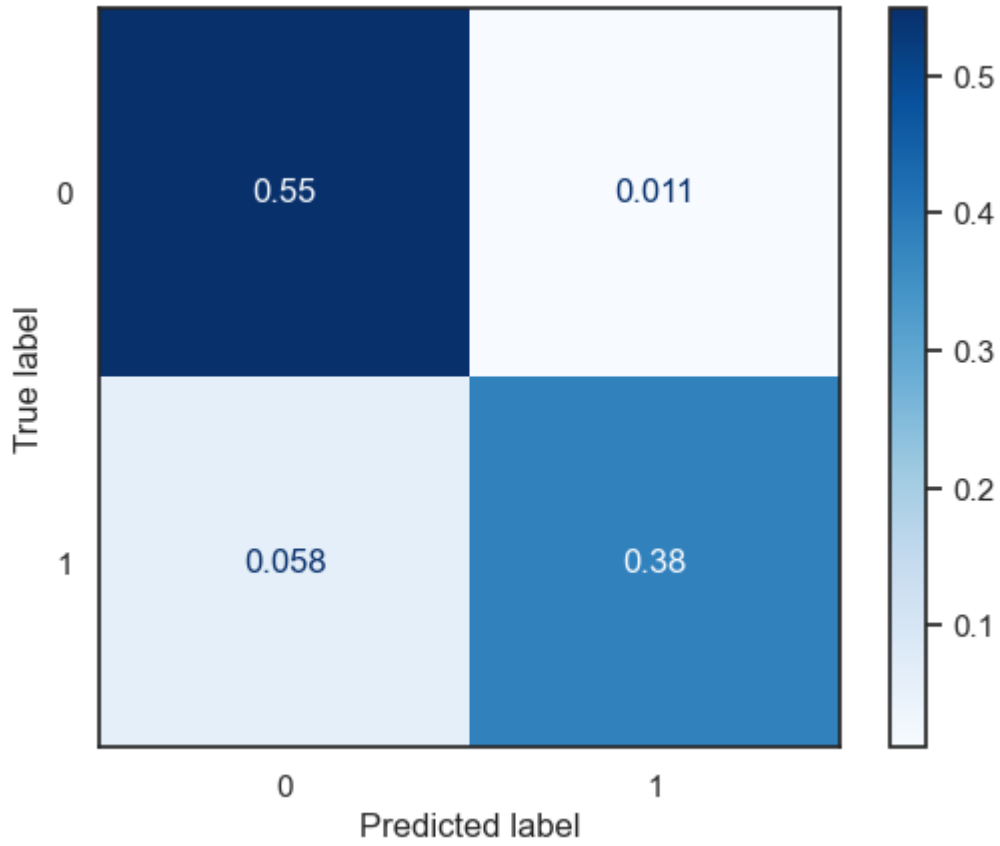


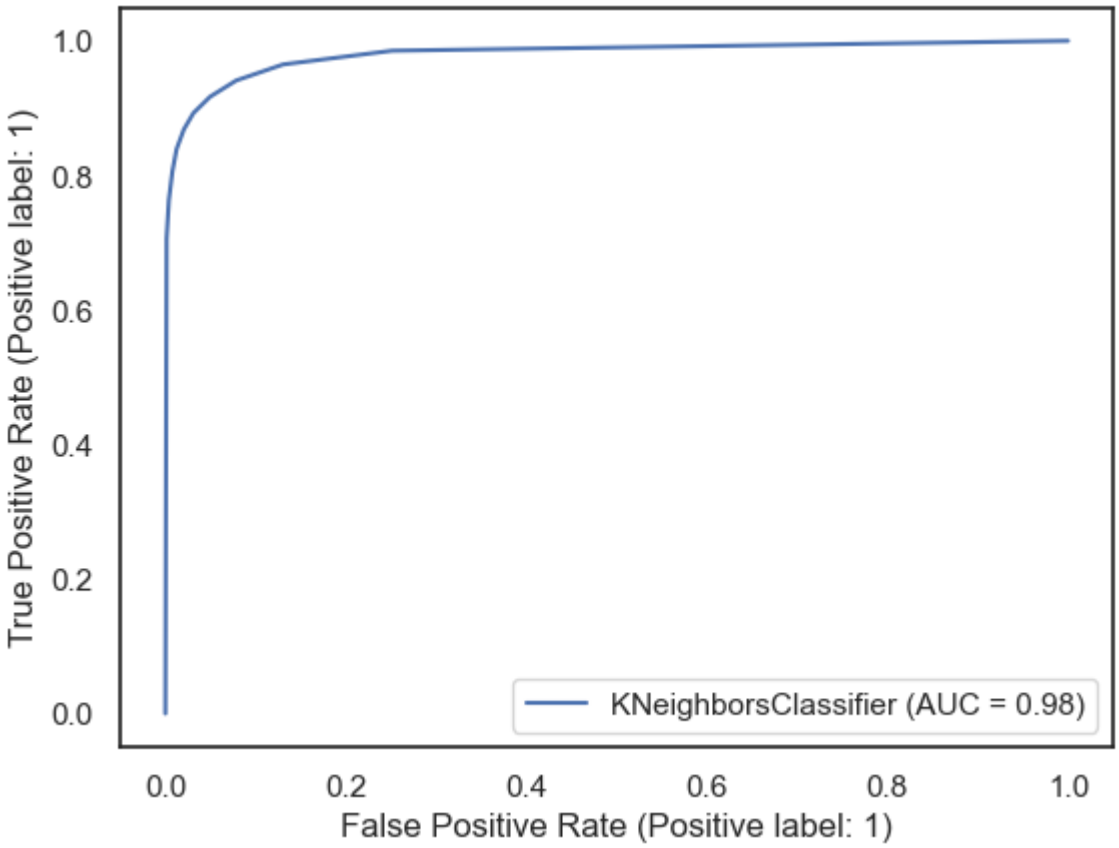
K-Nearest Neighbor

In [940…

```
params_kn = {'n_neighbors':10, 'algorithm': 'kd_tree', 'n_jobs':4}

model_kn = KNeighborsClassifier(**params_kn)
model_kn, accuracy_kn, roc_auc_kn, tt_kn = run_model(model_kn, X_train, y_train, X_test, y_test)
```

```
Accuracy = 0.9307052663997536
ROC Area under Curve = 0.9239157509939812
Time taken = 9.562151193618774
              precision    recall  f1-score   support

           0    0.90480   0.97955   0.94069     14573
           1    0.97078   0.86828   0.91667     11403

    accuracy                        0.93071     25976
   macro avg    0.93779   0.92392   0.92868     25976
weighted avg    0.93376   0.93071   0.93015     25976
```



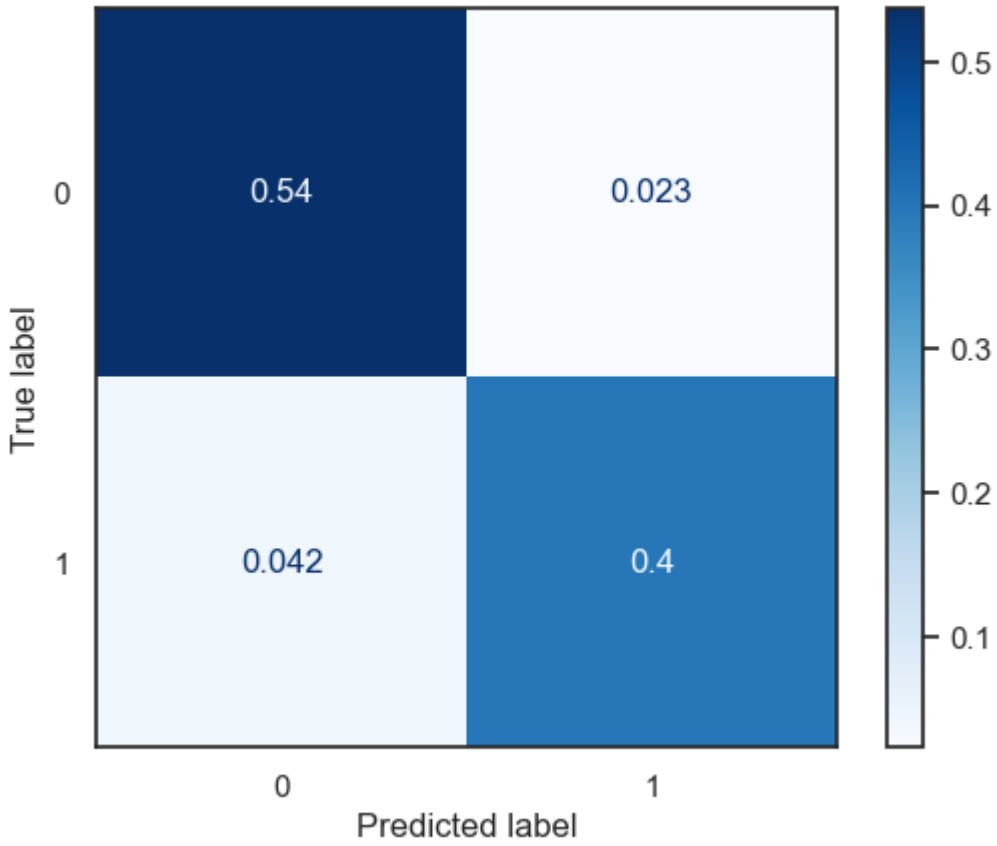


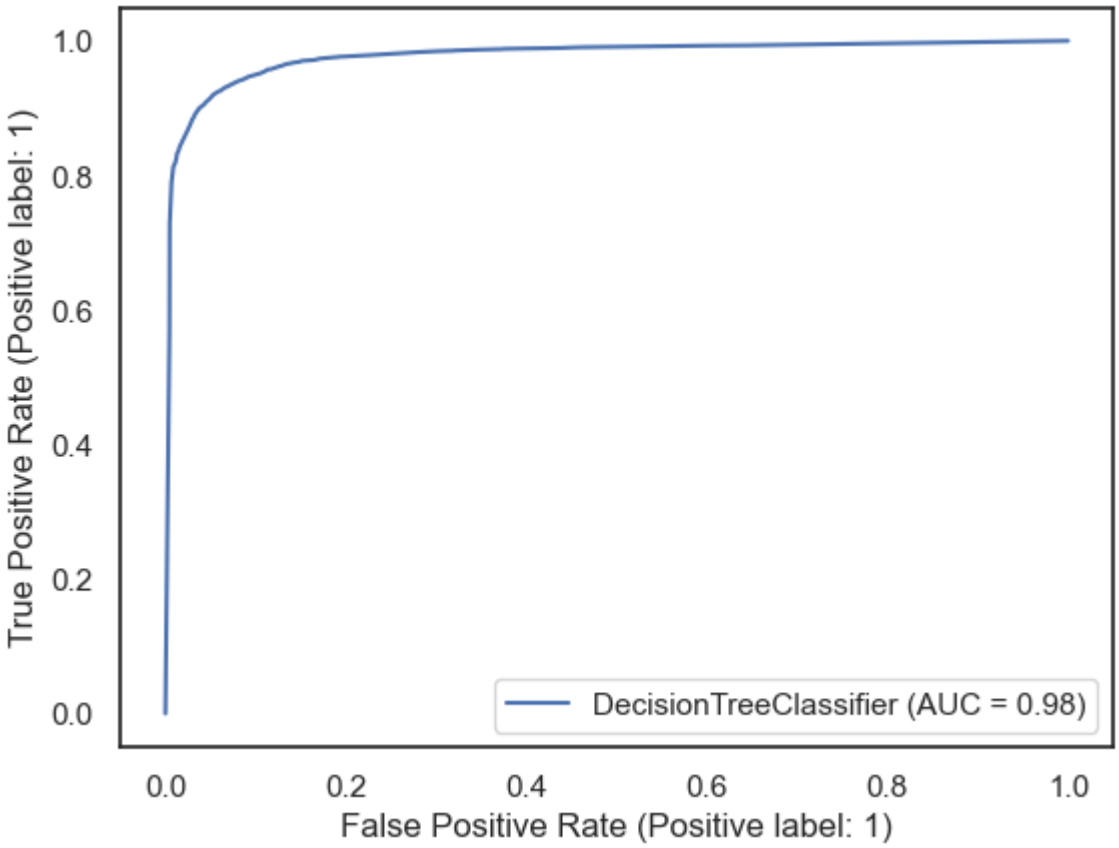
Decision Tree

In [941…

```
params_dt = {'max_depth': 12,
             'max_features': "sqrt"}

model_dt = DecisionTreeClassifier(**params_dt)
model_dt, accuracy_dt, roc_auc_dt, tt_dt = run_model(model_dt, X_train, y_train, X_test, y_test)
```

```
Accuracy = 0.9345549738219895
ROC Area under Curve = 0.9312669063680641
Time taken = 0.10897660255432129
              precision    recall  f1-score   support

           0    0.92753   0.95821   0.94262     14573
           1    0.94424   0.90432   0.92385     11403

    accuracy                        0.93455     25976
   macro avg    0.93588   0.93127   0.93324     25976
weighted avg    0.93486   0.93455   0.93438     25976
```



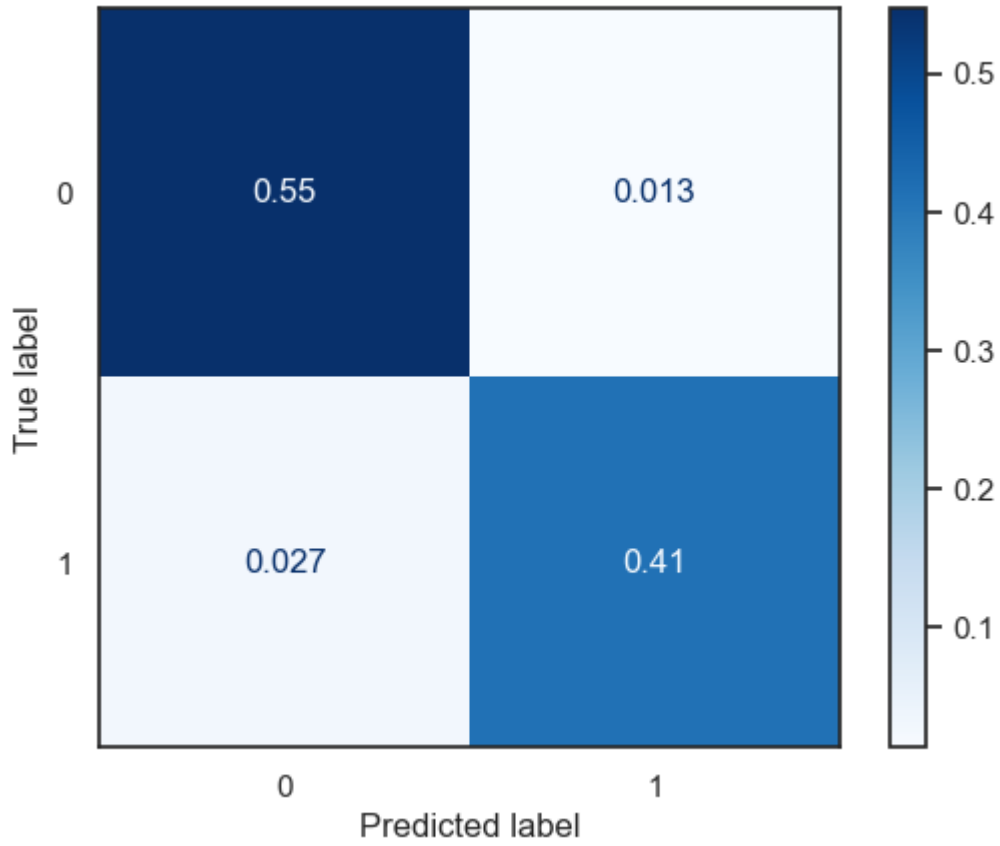


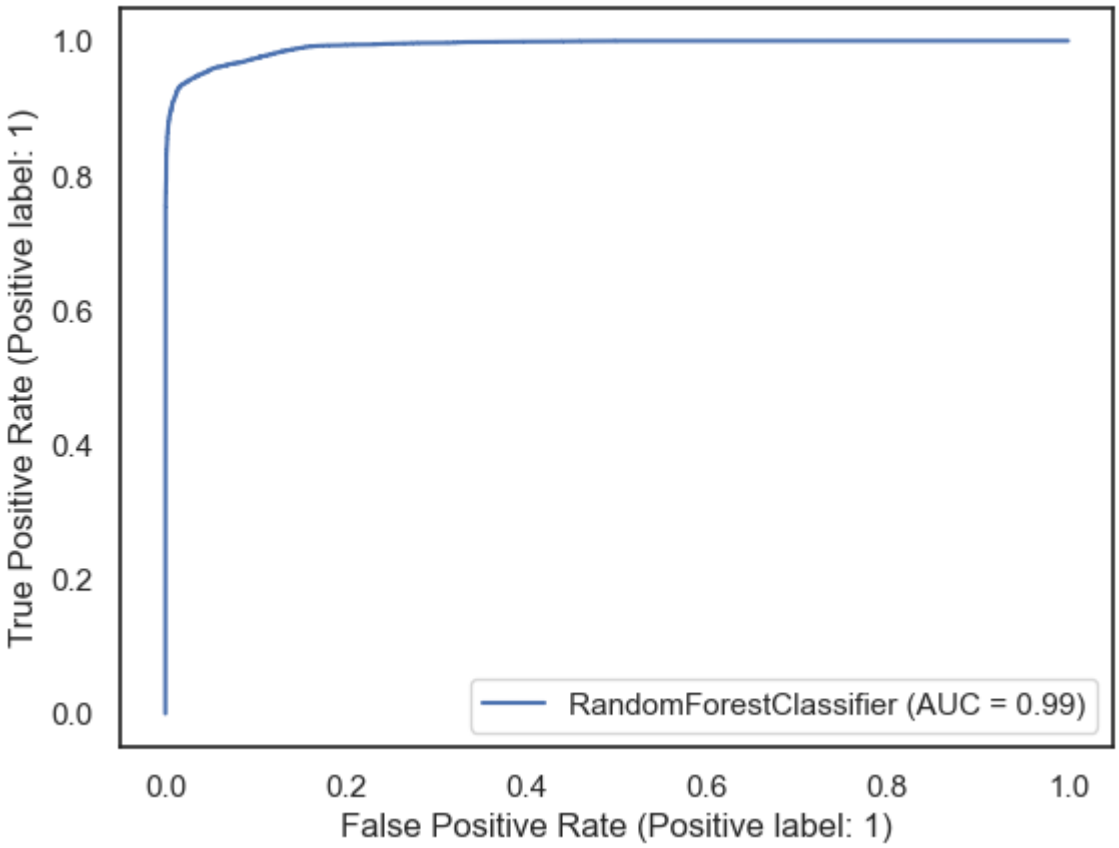
Random Forest

In [942…

```
params_rf = {'max_depth': 16,
             'min_samples_leaf': 1,
             'min_samples_split': 2,
             'n_estimators': 100,
             'random_state': 12345}

model_rf = RandomForestClassifier(**params_rf)
model_rf, accuracy_rf, roc_auc_rf, tt_rf = run_model(model_rf, X_train, y_train, X_test, y_test)
```

```
Accuracy = 0.9600400369571912
ROC Area under Curve = 0.9577953770786729
Time taken = 6.189152956008911
              precision    recall  f1-score   support

           0    0.95368   0.97619   0.96480     14573
           1    0.96862   0.93940   0.95379     11403

    accuracy                        0.96004     25976
   macro avg    0.96115   0.95780   0.95930     25976
weighted avg    0.96024   0.96004   0.95997     25976
```



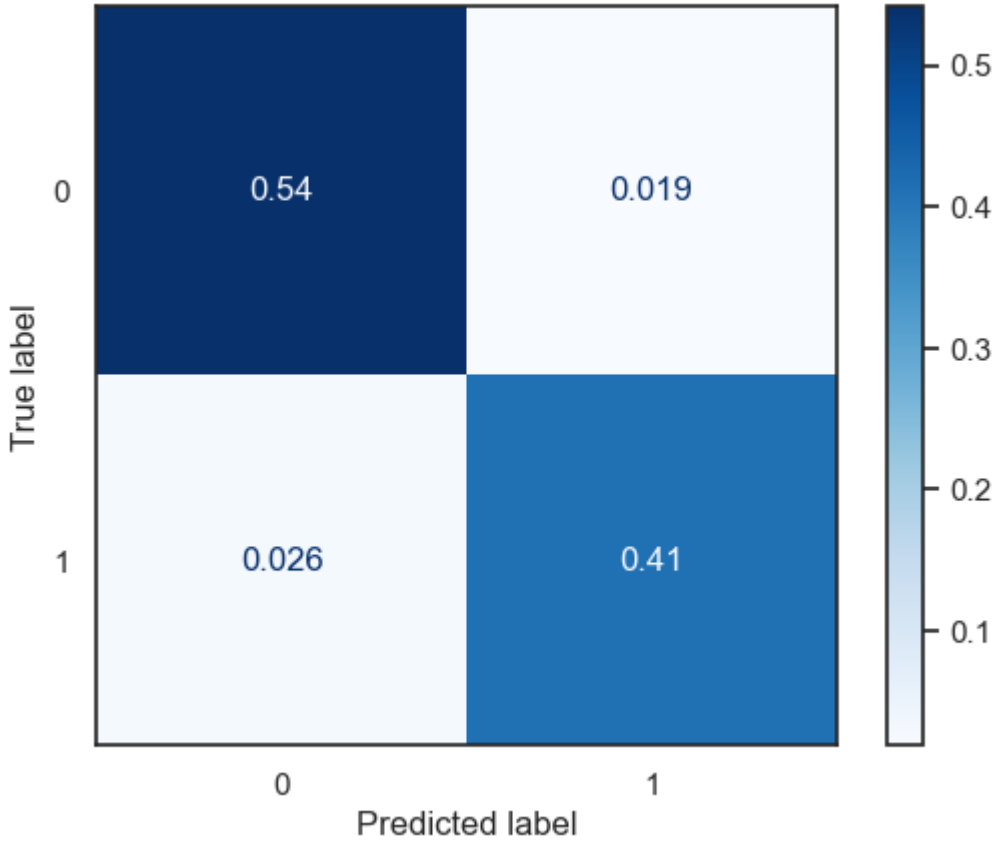


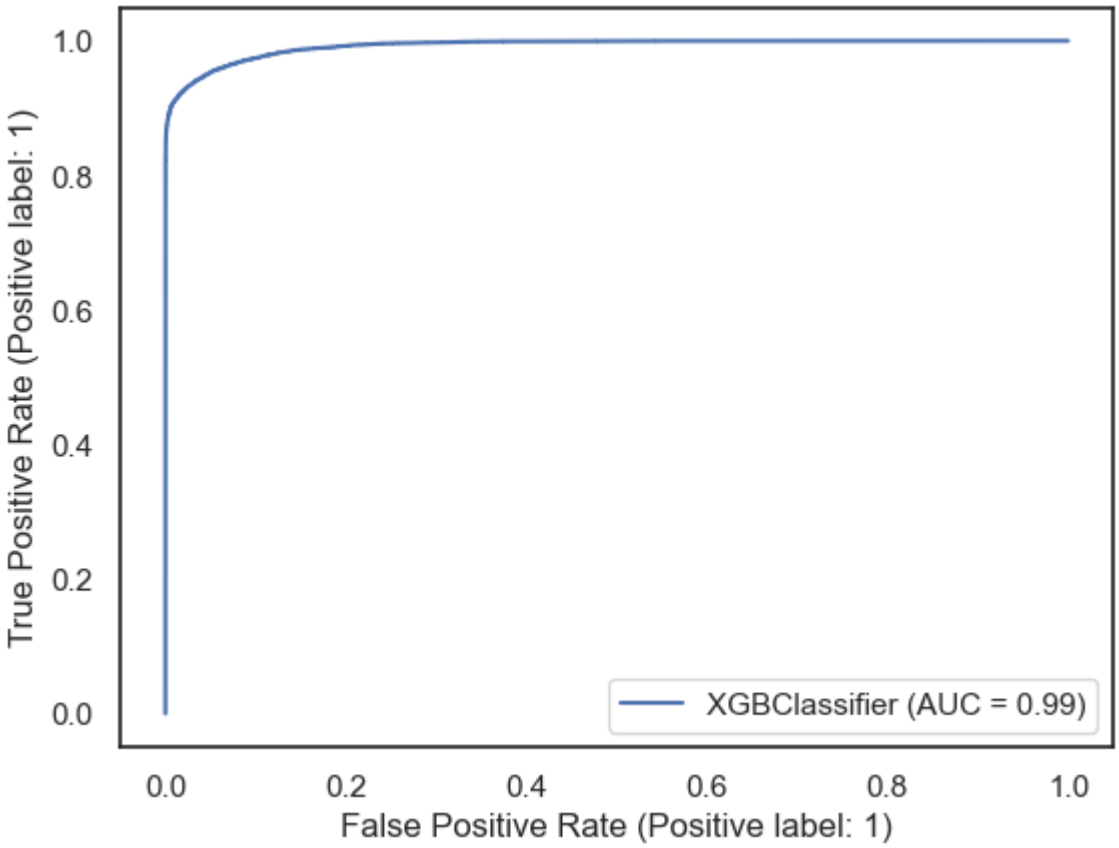
Extreme Gradient Boosting (XGBoost)

In [943…

```
params_xgb ={'n_estimators': 500,
             'max_depth': 16}

model_xgb = xgb.XGBClassifier(**params_xgb)
model_xgb, accuracy_xgb, roc_auc_xgb, tt_xgb = run_model(model_xgb, X_train, y_train, X_test, y_test)
```

```
[20:17:50] WARNING: C:\Windows\Temp\abs_557yfx631l\croots\recipe\xgboost-split_1659548953302\work\src\learner.cc:1115:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'er
ror' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
Accuracy = 0.9549969202340622
ROC Area under Curve = 0.9533389160916872
Time taken = 39.47240090370178
              precision    recall  f1-score   support

           0    0.95351   0.96693   0.96017     14573
           1    0.95696   0.93975   0.94828     11403

    accuracy                        0.95500     25976
   macro avg    0.95523   0.95334   0.95422     25976
weighted avg    0.95502   0.95500   0.95495     25976
```




1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
XGBClassifier (AUC = 0.99)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

In [ ]: