

Institute of Technology of Cambodia

Department of Applied Mathematics and Statistic

Group: I3-AMS-03

Subject: Advance Probability

Topic: Analyzing the factors that affect customer satisfaction
in the airline industry using ordinal logistic regression

Name:

- Tang Piseth
- Kry Senghort
- Sao Samarth
- Hon Rathana
- Pheng Sothea
- Ek Vong Panharith

ID:

- e20201634
- e20200706
- e20200084
- e20201053
- e20201734
- e20200877



Table of contents:

- 1. Project Description and Objectives**
- 2. Dataset and Variables Description**
- 3. Data Preparation**
- 4. Exploratory Data Analysis(EDA)**
- 5. Feature Engineering**
- 6. Model Selection**
- 7. Decision Making**



I.) Project Description and Objectives

Airline Industry is the field of journey tourism or visiting to any places by using airline for travelling from one place to another one. On the other hand, We choose this topic because we want to use logistic linear regression to interpret and make prediction the result of satisfaction or dissatisfaction of passenger on airline industry. We will study on the variable that make effect to satisfaction or dissatisfaction of customer.



2). Dataset and Variables Description

There are two data sets such as : test.csv and train.csv. There exist 129879 rows with 24 columns. There exists 24 variables as following:

- Gender (object)
- Customer Type (object)
- Age (int64)
- Type of Travel (object)
- Class (object)
- Flight Distance (int64)
- Inflight wifi service (int64)
- Departure/Arrival time convenient (int64)
- Ease of Online booking (int64)
- Gate location (int64)
- Food and drink (int64)
- Online boarding (int64)
- Seat comfort (int64)
- Inflight entertainment (int64)
- On-board service (int64)
- Leg room service (int64)
- Baggage handling (int64)
- Checkin service (int64)
- Inflight service (int64)
- Cleanliness (int64)
- Departure Delay in Minutes (int64)
- Arrival Delay in Minutes (float)
- Satisfaction (object)(target variable)

3). Data Preparation

After checking to our dataset we observe that there exist some missing values and also we decide to handle it by substituting with the mean values for numerical variable and mode for the categorical variables. As the result, feature "Arrival_Delay_in_Minutes" existing 393 missing values.

<code>id</code>	0
<code>Gender</code>	0
<code>Customer_Type</code>	0
<code>Age</code>	0
<code>Type_of_Travel</code>	0
<code>Class</code>	0
<code>Flight_Distance</code>	0
<code>Inflight_wifi_service</code>	0
<code>Departure/Arrival_time_convenient</code>	0
<code>Ease_of_Online_booking</code>	0
<code>Gate_location</code>	0
<code>Food_and_drink</code>	0
<code>Online_boarding</code>	0
<code>Seat_comfort</code>	0
<code>Inflight_entertainment</code>	0
<code>On-board_service</code>	0
<code>Leg_room_service</code>	0
<code>Baggage_handling</code>	0
<code>Checkin_service</code>	0
<code>Inflight_service</code>	0
<code>Cleanliness</code>	0
<code>Departure_Delay_in_Minutes</code>	0
<code>Arrival_Delay_in_Minutes</code>	393
<code>satisfaction</code>	0
<code>dtype: int64</code>	

Descriptive Statistics Before Handle Missing

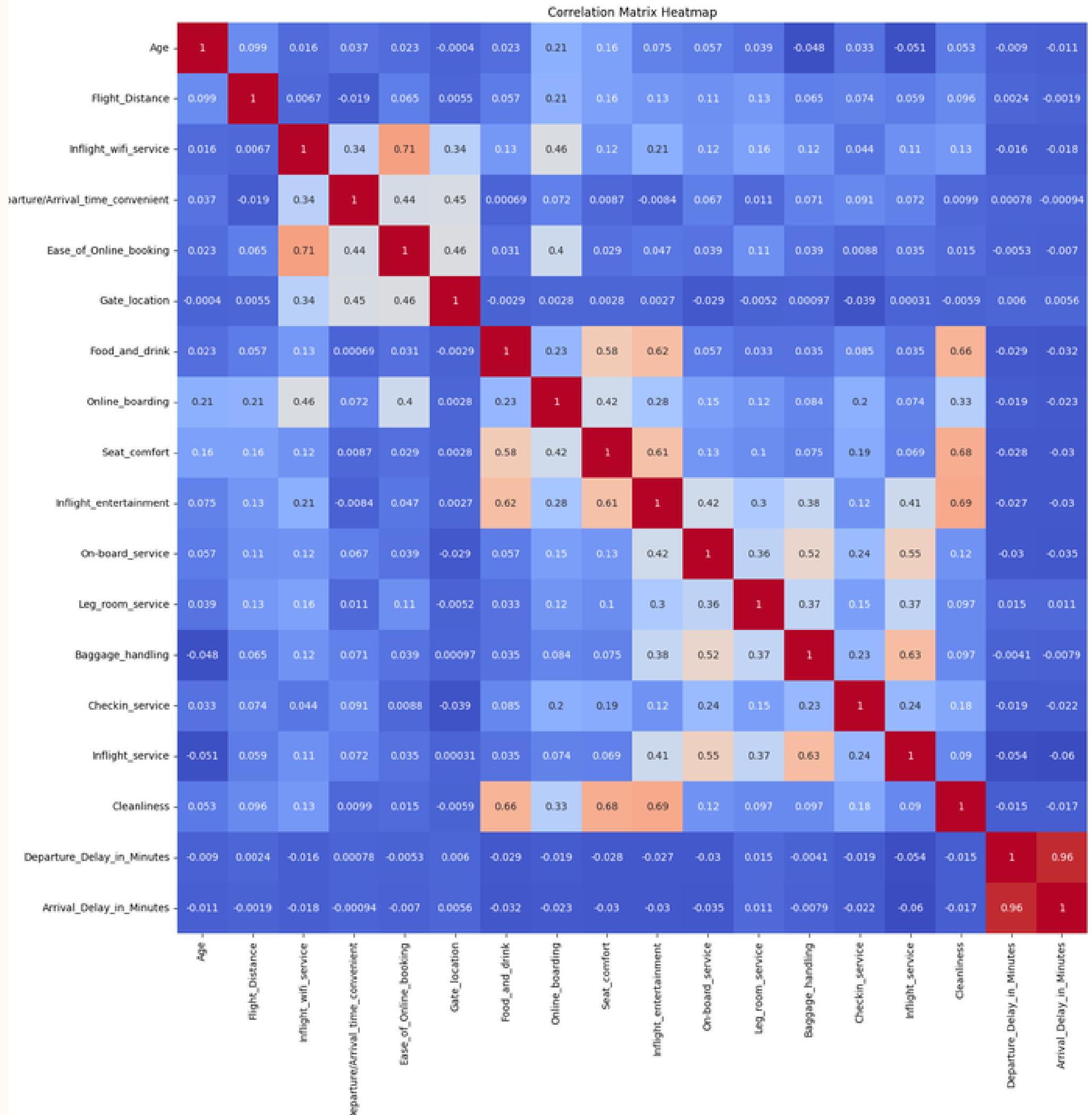
		count	mean	std	min	25%	50%	75%	max
	id	129880.0	64940.500000	37493.270818	1.0	32470.75	64940.5	97410.25	129880.0
	Age	129880.0	39.427957	15.119360	7.0	27.00	40.0	51.00	85.0
	Flight_Distance	129880.0	1190.316392	997.452477	31.0	414.00	844.0	1744.00	4983.0
	Inflight_wifi_service	129880.0	2.728696	1.329340	0.0	2.00	3.0	4.00	5.0
	Departure/Arrival_time_convenient	129880.0	3.057599	1.526741	0.0	2.00	3.0	4.00	5.0
	Ease_of_Online_booking	129880.0	2.756876	1.401740	0.0	2.00	3.0	4.00	5.0
	Gate_location	129880.0	2.976925	1.278520	0.0	2.00	3.0	4.00	5.0
	Food_and_drink	129880.0	3.204774	1.329933	0.0	2.00	3.0	4.00	5.0
	Online_boarding	129880.0	3.252633	1.350719	0.0	2.00	3.0	4.00	5.0
	Seat_comfort	129880.0	3.441361	1.319289	0.0	2.00	4.0	5.00	5.0
	Inflight_entertainment	129880.0	3.358077	1.334049	0.0	2.00	4.0	4.00	5.0
	On-board_service	129880.0	3.383023	1.287099	0.0	2.00	4.0	4.00	5.0
	Leg_room_service	129880.0	3.350878	1.316252	0.0	2.00	4.0	4.00	5.0
	Baggage_handling	129880.0	3.632114	1.180025	1.0	3.00	4.0	5.00	5.0
	Checkin_service	129880.0	3.306267	1.266185	0.0	3.00	3.0	4.00	5.0
	Inflight_service	129880.0	3.642193	1.176669	0.0	3.00	4.0	5.00	5.0
	Cleanliness	129880.0	3.286326	1.313682	0.0	2.00	3.0	4.00	5.0
	Departure_Delay_in_Minutes	129880.0	14.713713	38.071126	0.0	0.00	0.0	12.00	1592.0
	Arrival_Delay_in_Minutes	129487.0	15.091129	38.465650	0.0	0.00	0.0	13.00	1584.0

4). Exploratory Data Analysis (EDA)

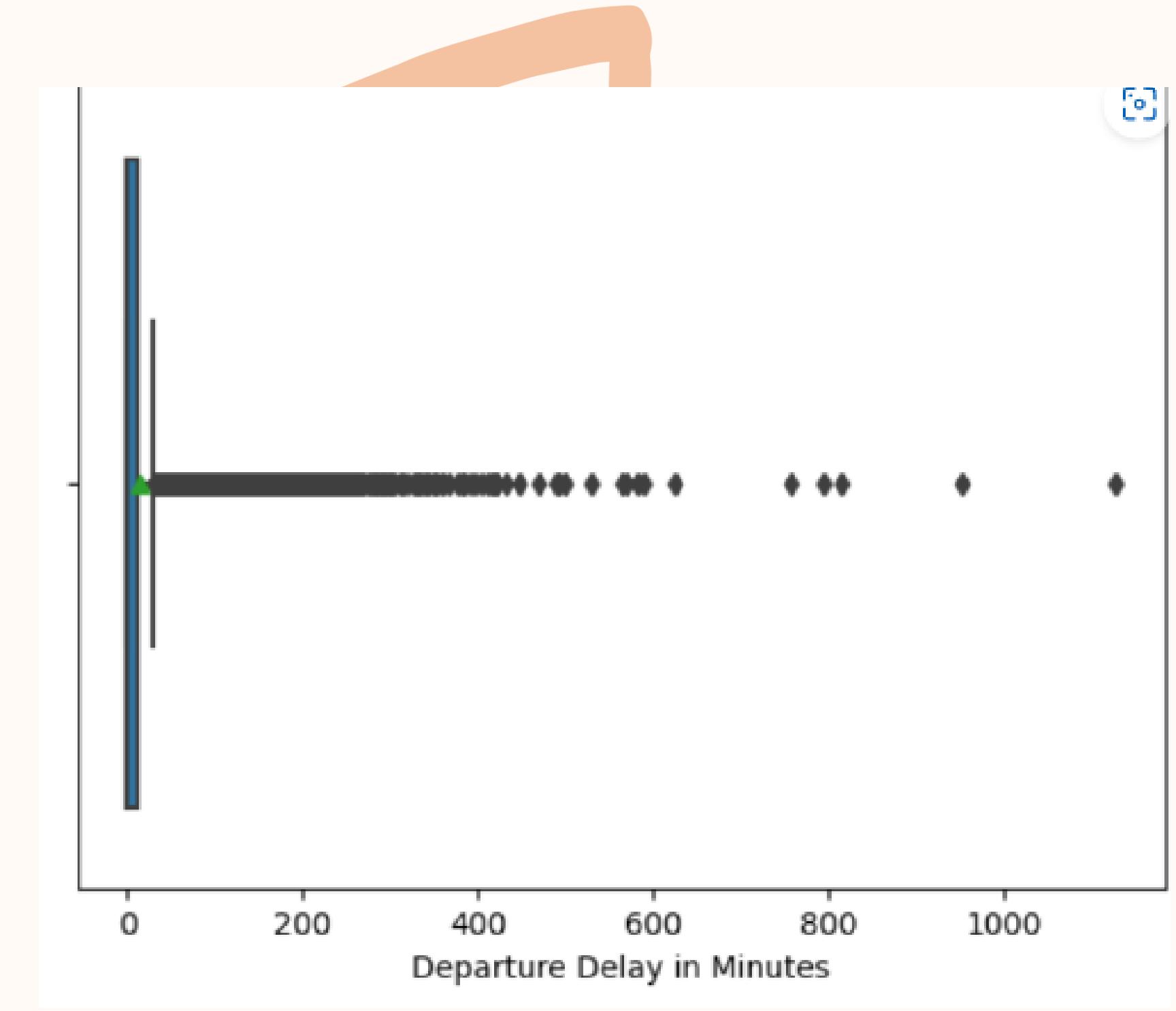
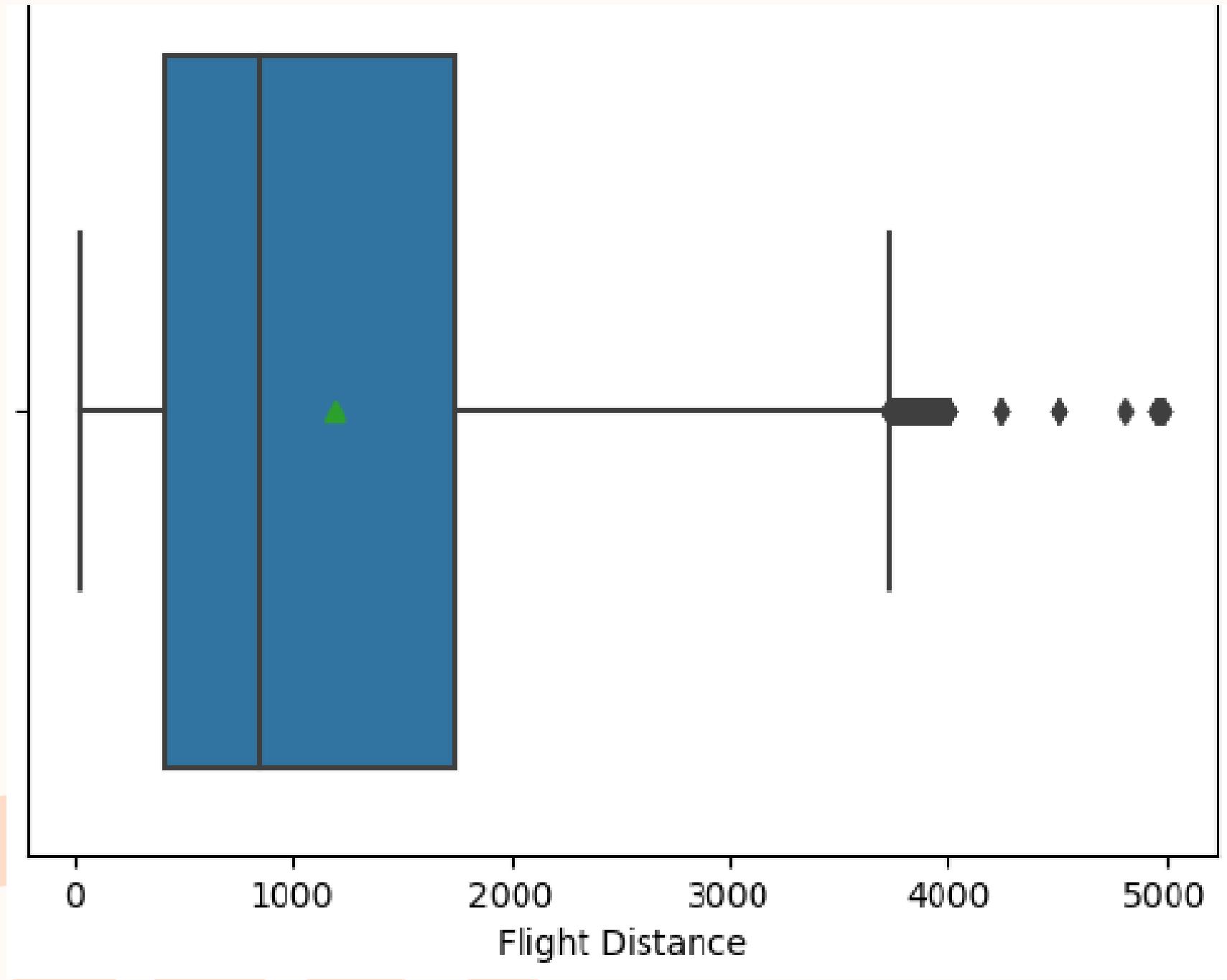
Statistic Table

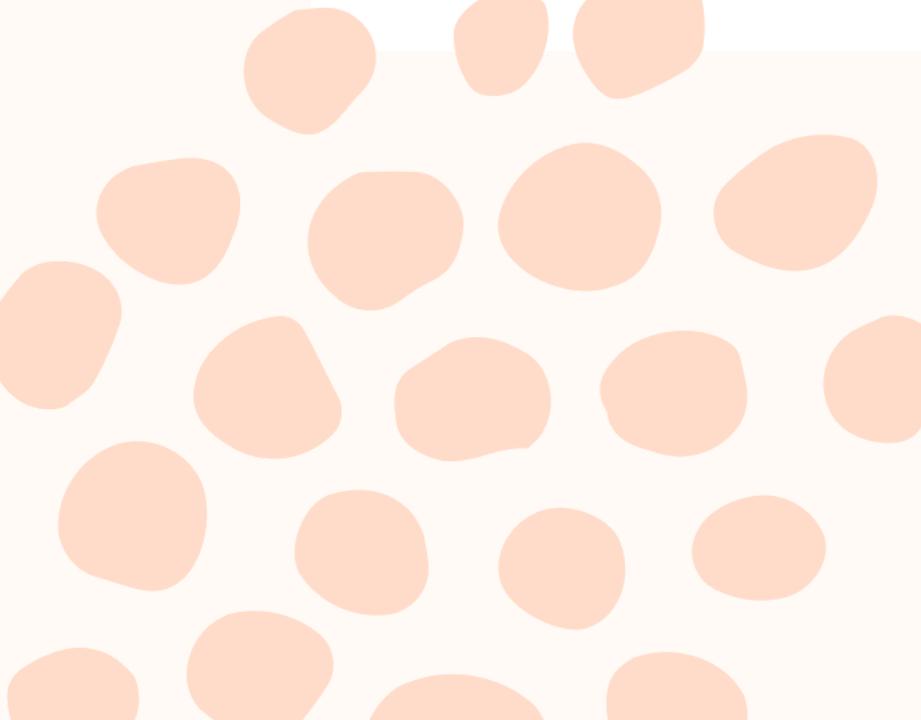
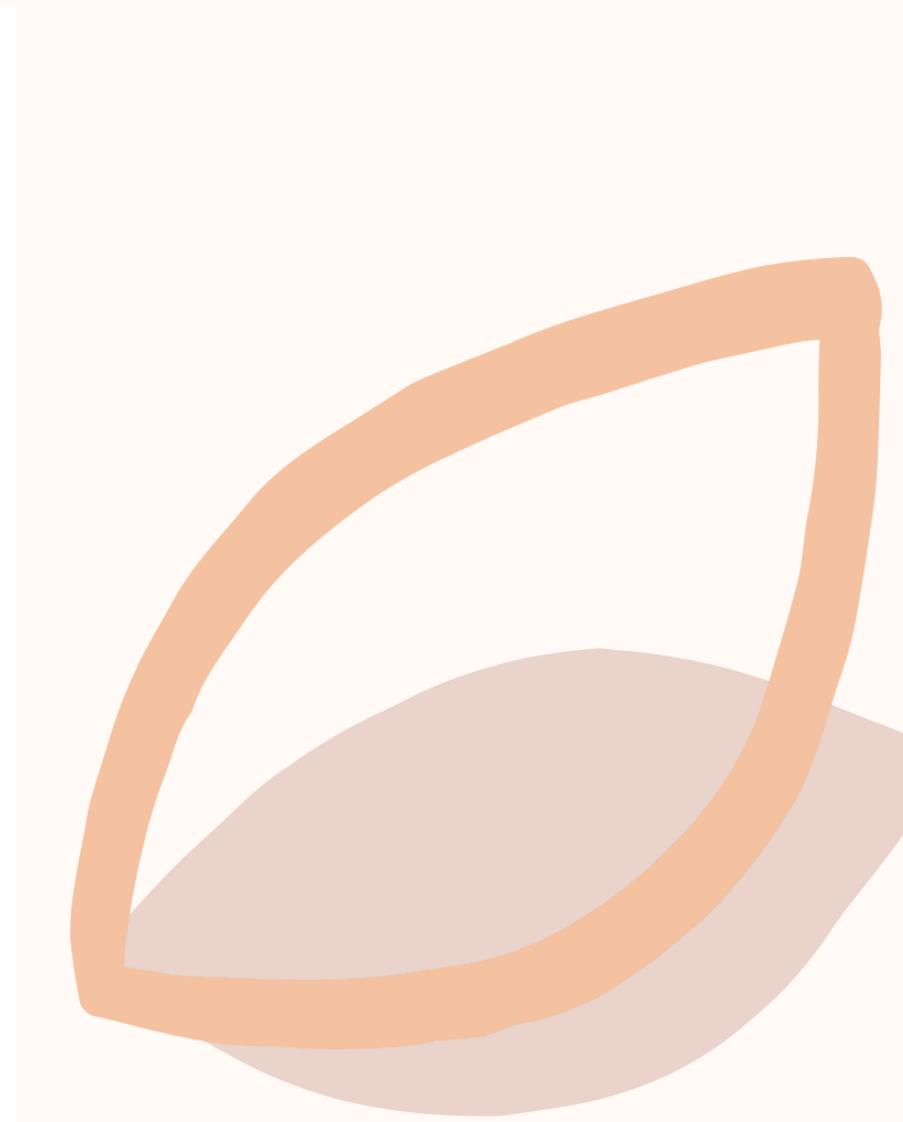
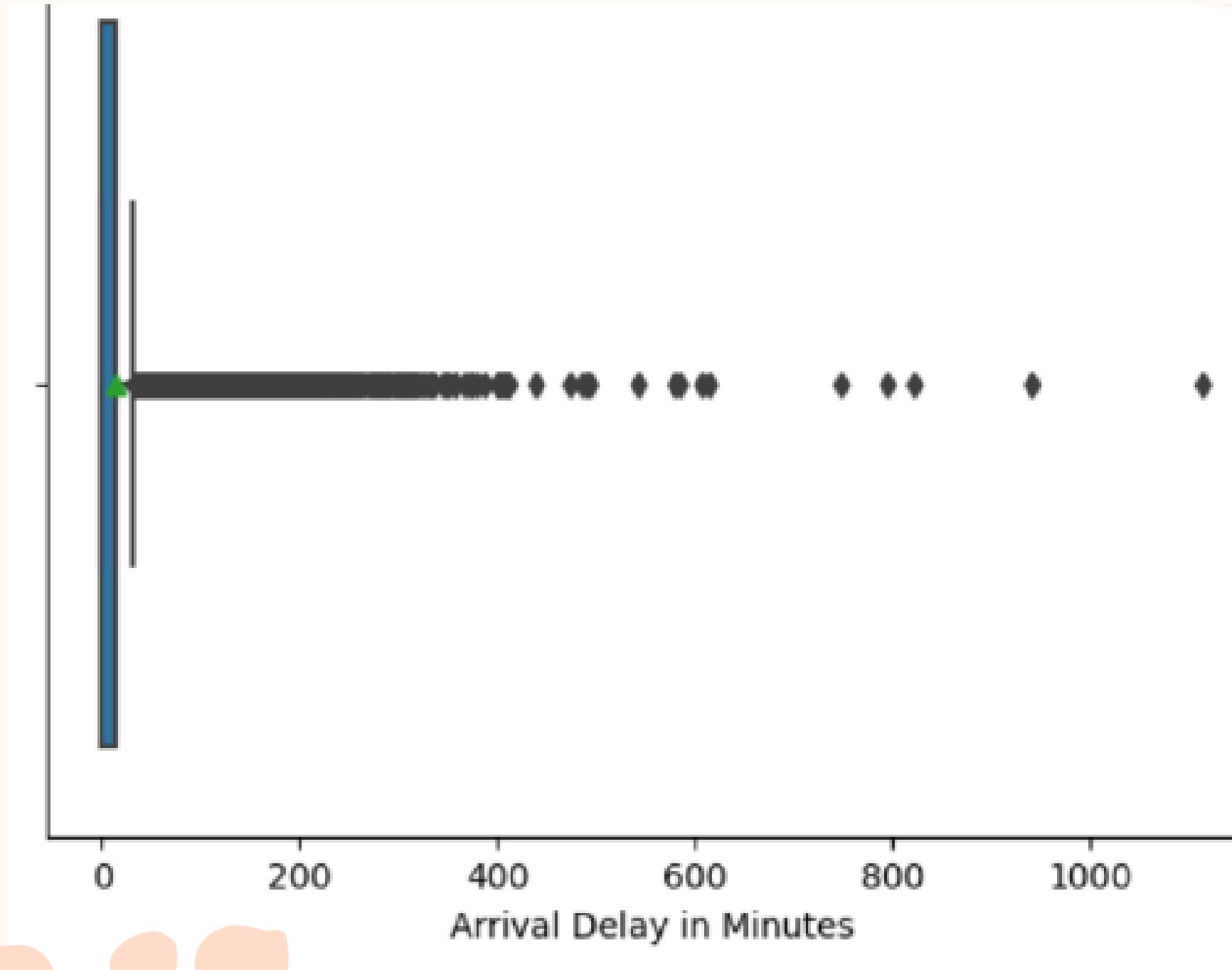
	id	Age	Flight_Distance	Inflight_wifi_service	Departure/Arrival_time_convenient	Ease_of_Online_booking	Gate_location	Food_and_drink	
count	129570.000000	129570.000000	129570.000000	129570.000000	129570.000000	129570.000000	129570.000000	129570.000000	
mean	64955.104754	39.428679	1190.219989	2.728749	3.057421	2.756942	2.977039	3.204777	
std	37490.942449	15.117899	997.573150	1.329373	1.526871	1.401750	1.278603	1.329829	
min	1.000000	7.000000	31.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	32488.250000	27.000000	414.000000	2.000000	2.000000	2.000000	2.000000	2.000000	
50%	64966.500000	40.000000	844.000000	3.000000	3.000000	3.000000	3.000000	3.000000	
75%	97411.750000	51.000000	1744.000000	4.000000	4.000000	4.000000	4.000000	4.000000	
max	129880.000000	85.000000	4983.000000	5.000000	5.000000	5.000000	5.000000	5.000000	
Baggage_handling	Checkin_service	Inflight_service	Cleanliness	Departure_Delay_in_Minutes	Online_boarding	Seat_comfort	Inflight_entertainment	On-board_service	Leg_room_service
129570.000000	129570.000000	129570.000000	129570.000000	129570.000000	129570.000000	129570.000000	129570.000000	129570.000000	129570.000000
3.631998	3.306298	3.642464	3.286363	14.659358	3.252736	3.441661	3.358223	3.383221	3.351154
1.180141	1.266187	1.176623	1.313623	37.978973	1.350661	1.319136	1.334083	1.287040	1.316097
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3.000000	3.000000	3.000000	2.000000	0.000000	2.000000	2.000000	2.000000	2.000000	2.000000
4.000000	3.000000	4.000000	3.000000	0.000000	3.000000	4.000000	4.000000	4.000000	4.000000
5.000000	4.000000	5.000000	4.000000	12.000000	4.000000	5.000000	4.000000	4.000000	4.000000
5.000000	5.000000	5.000000	5.000000	1592.000000	5.000000	5.000000	5.000000	5.000000	5.000000

Heat map

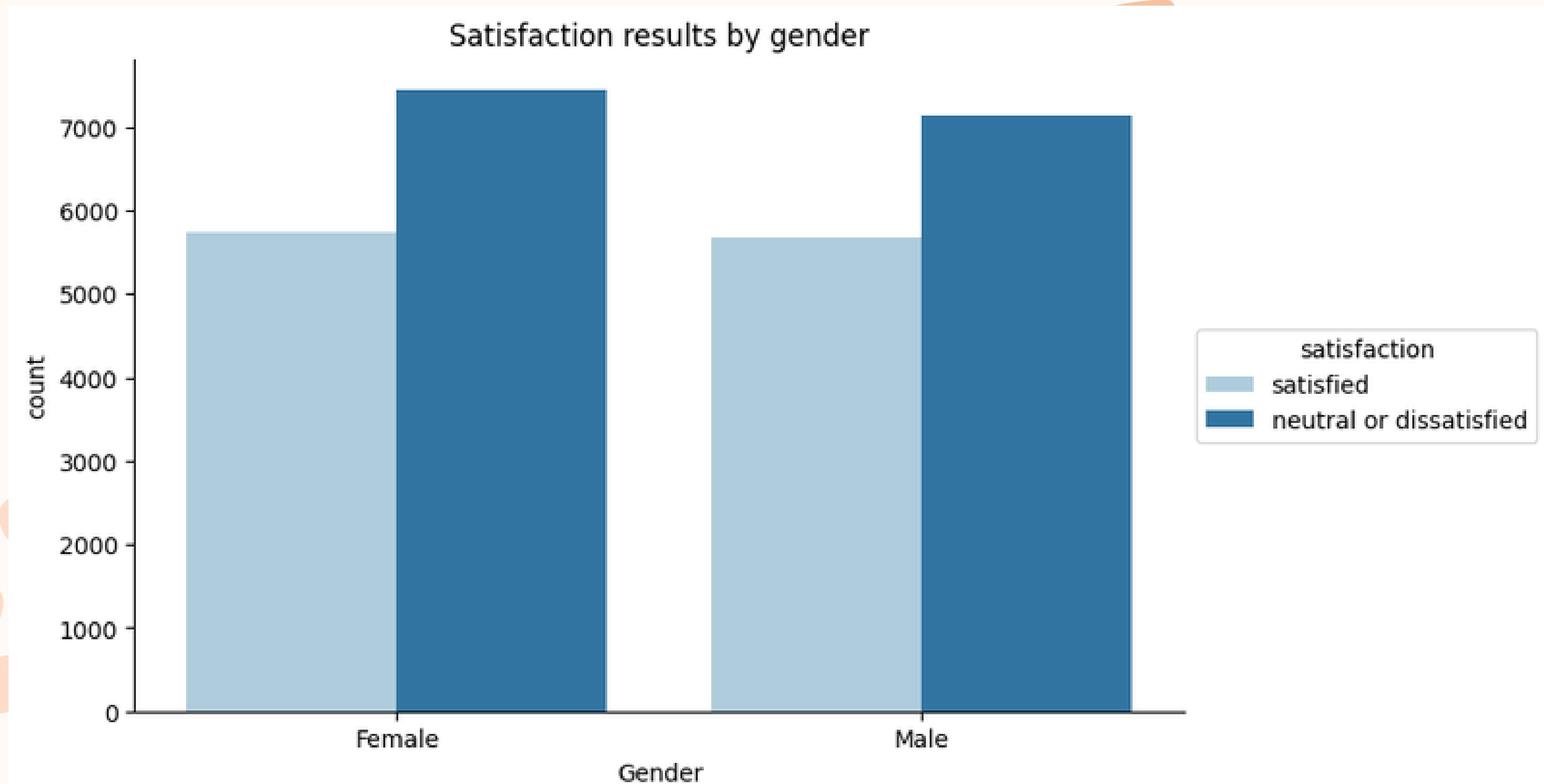


Box plot

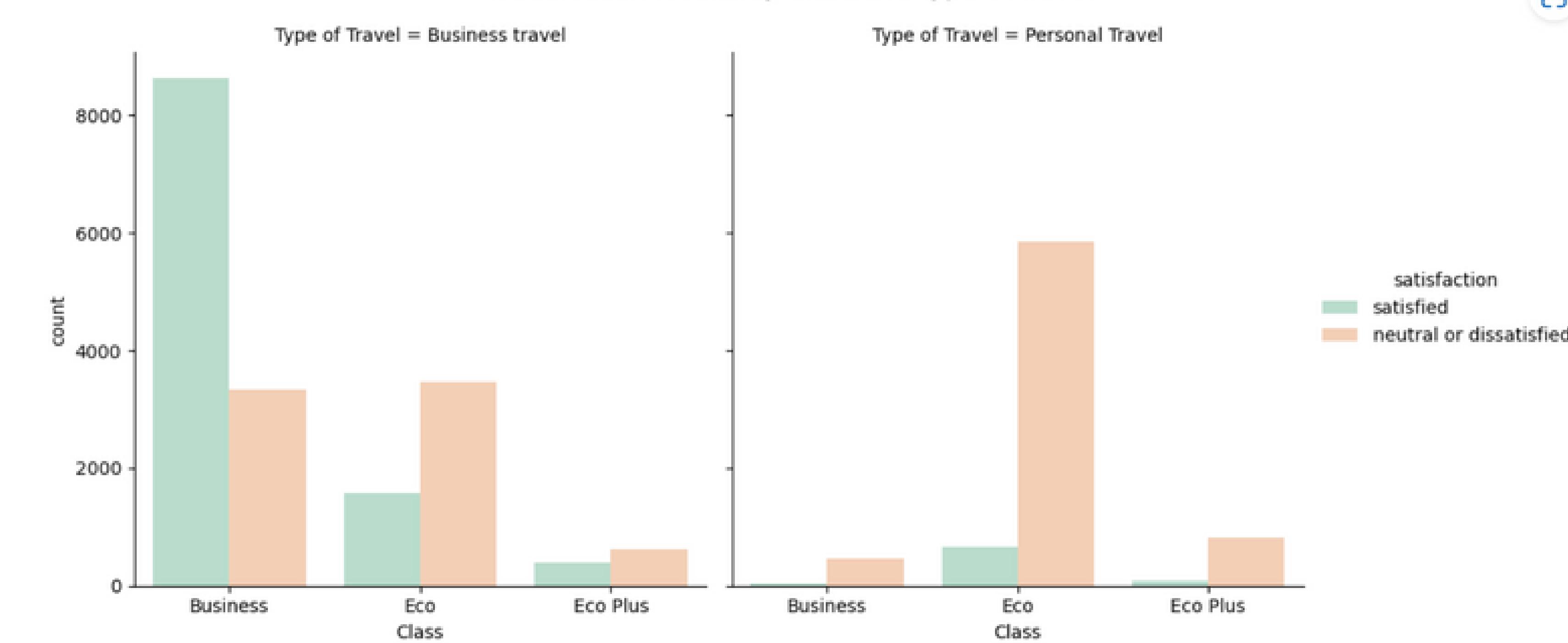




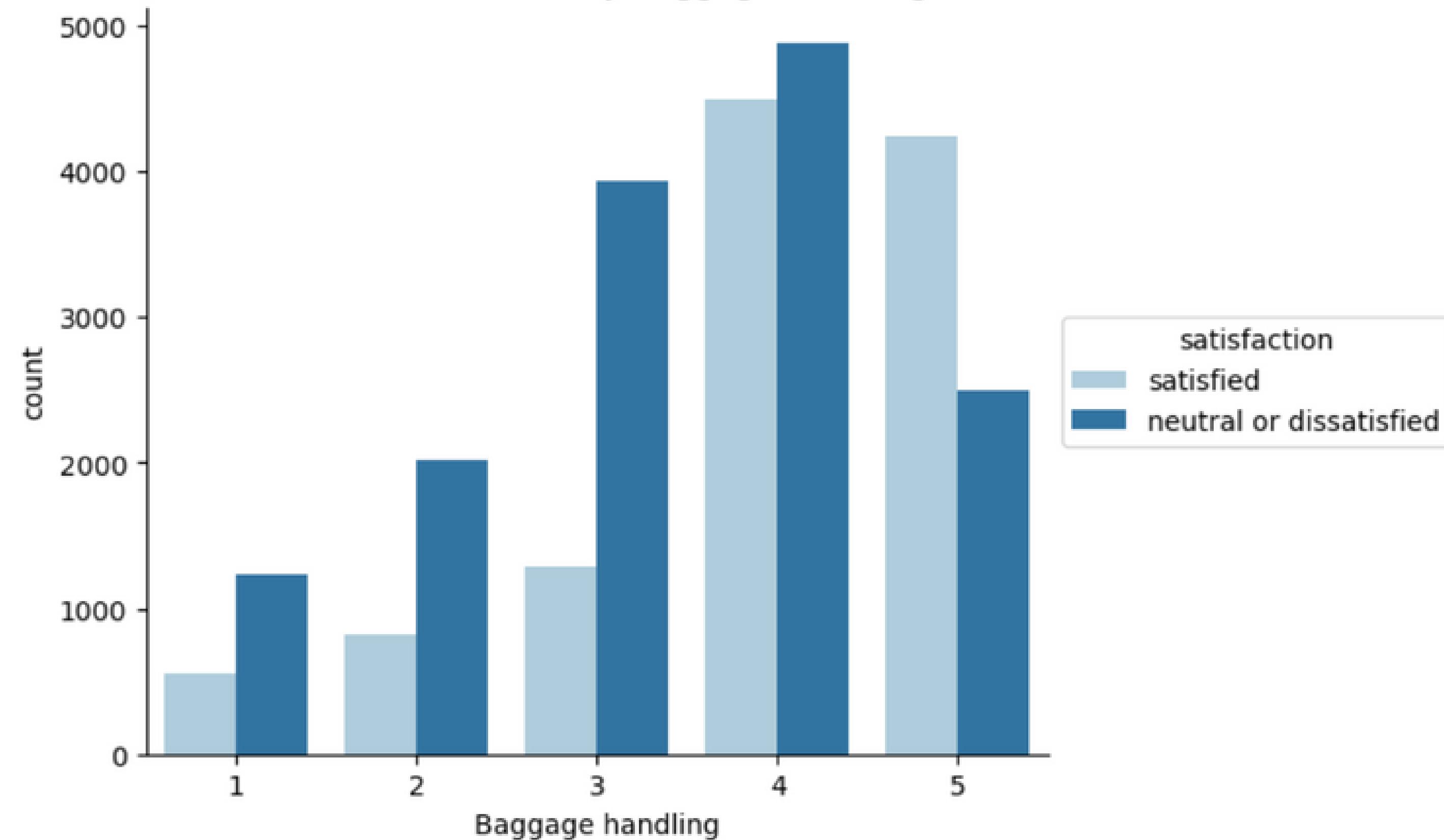
Bar graph



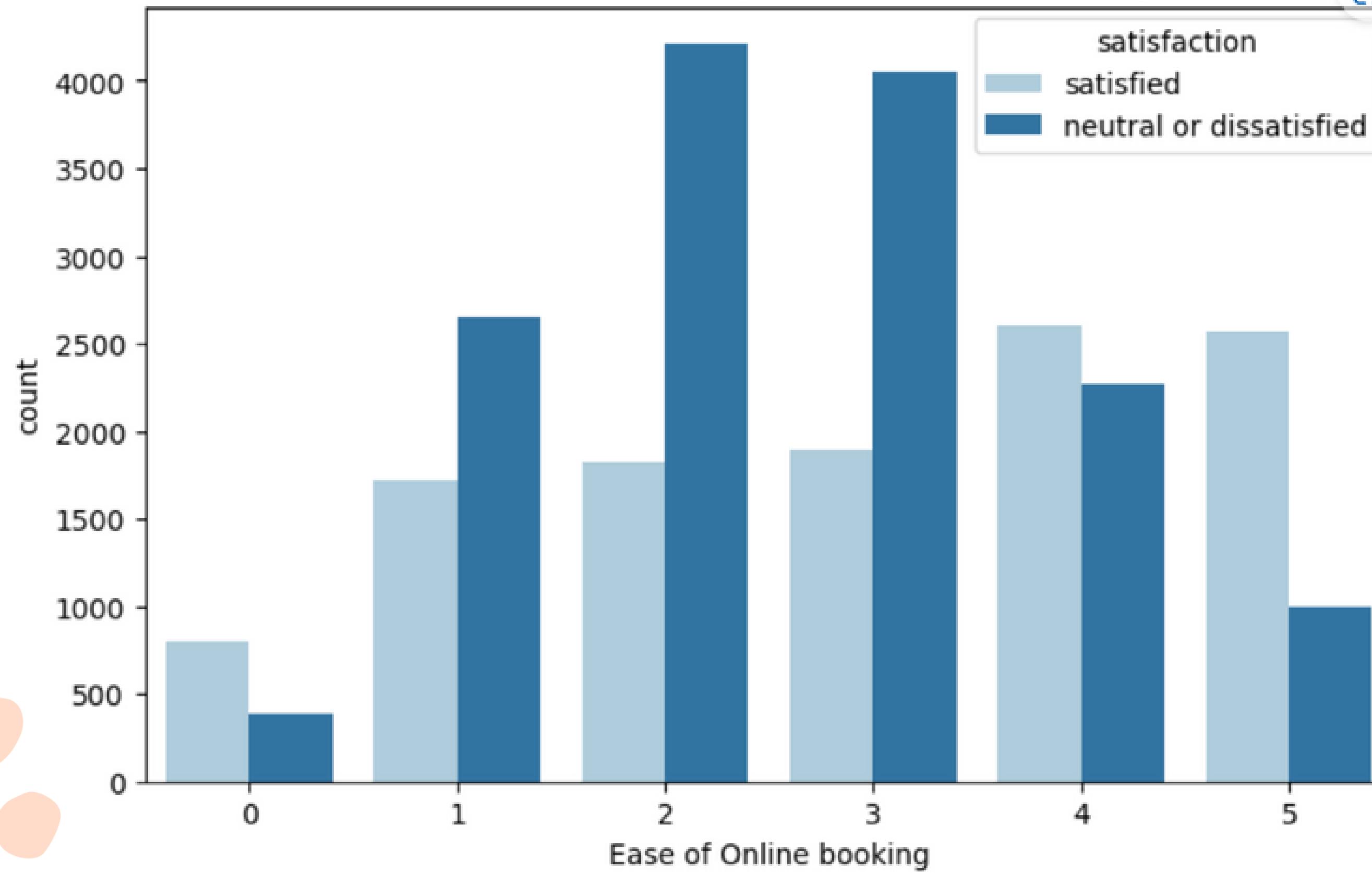
Satisfaction results by Class and Type of Travel



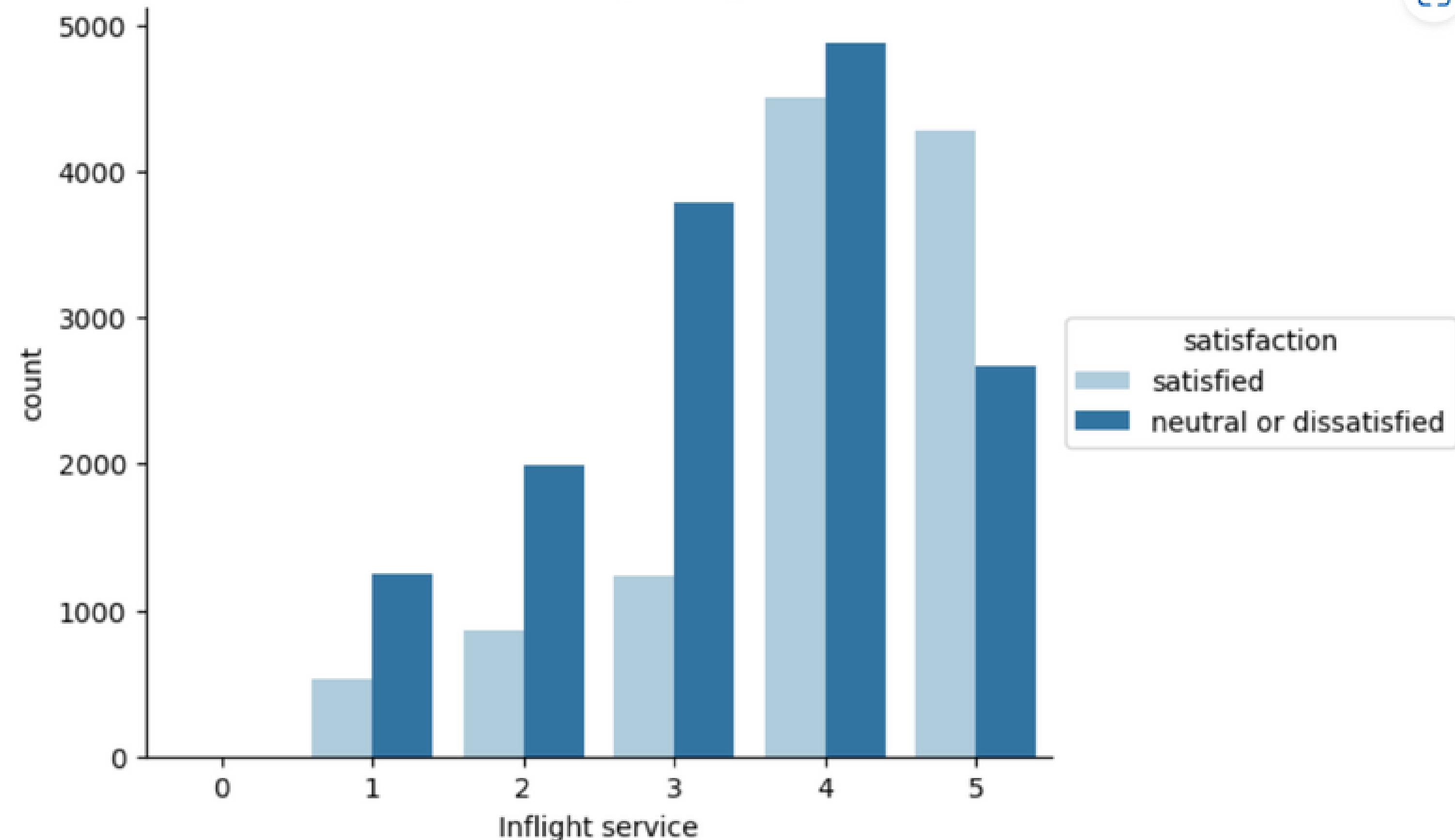
Satisfaction results by Baggage Handling



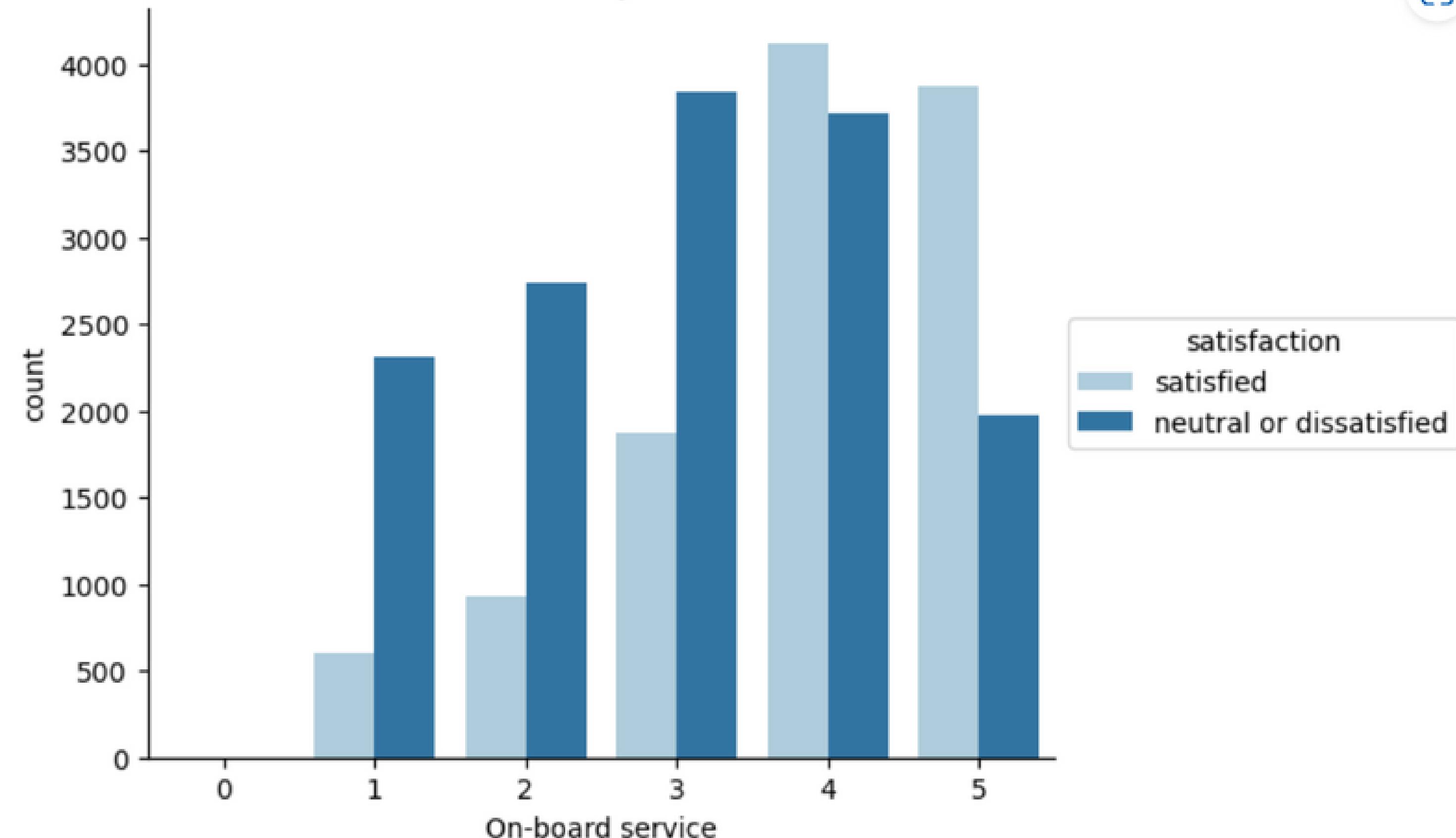
Satisfaction results by Ease of Online Booking



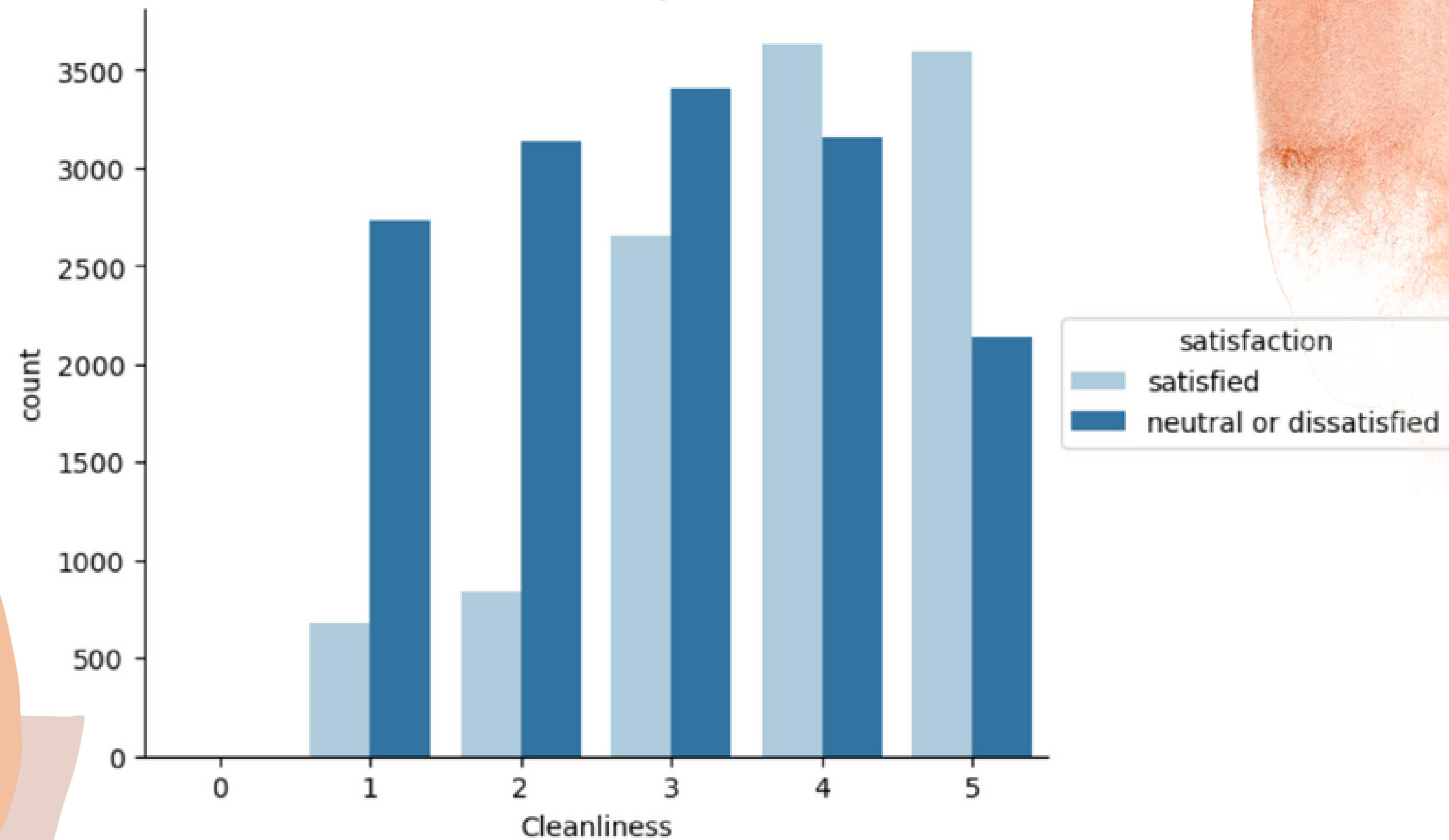
Satisfaction results by In-flight Service



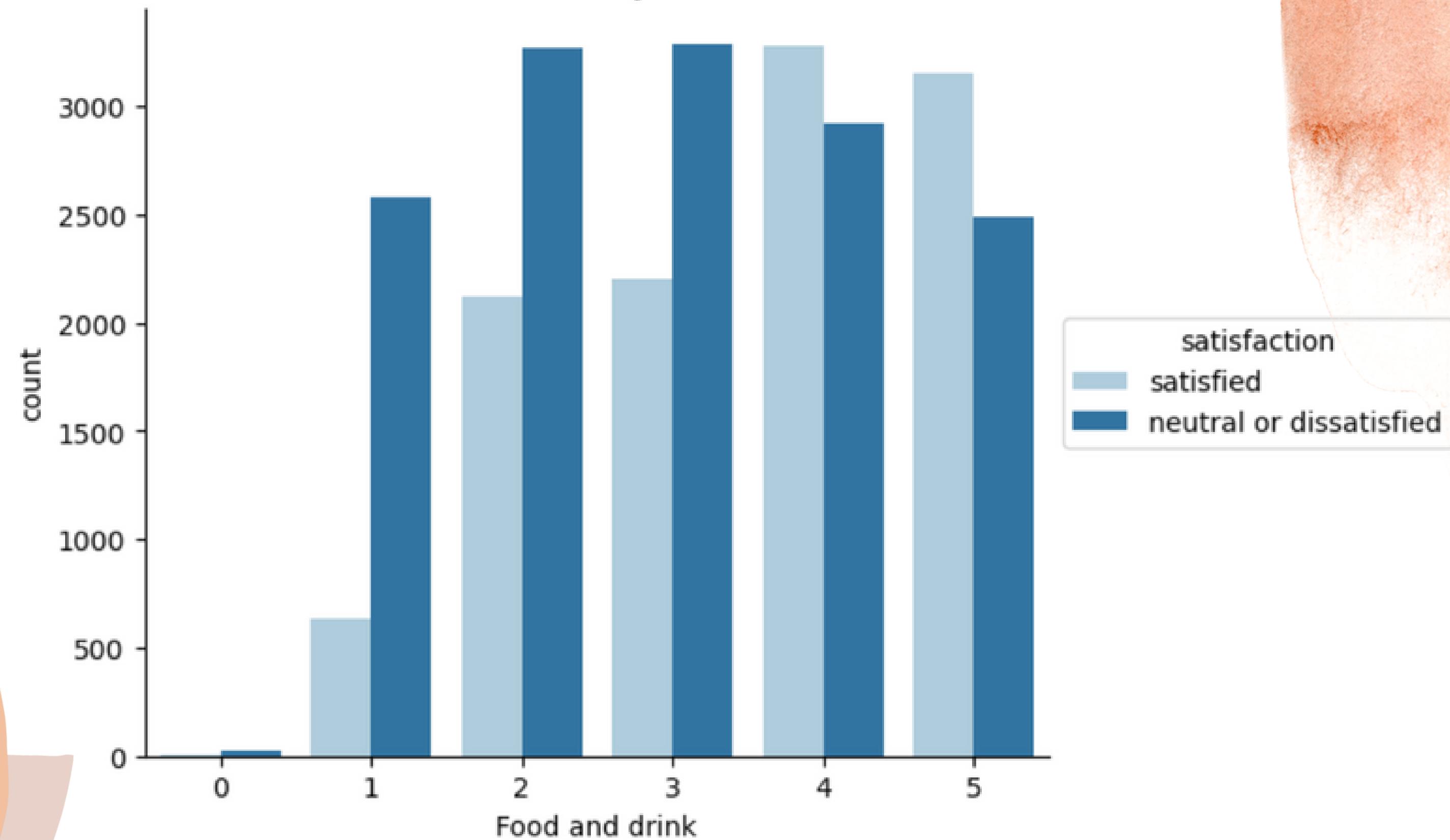
Satisfaction results by On-board Service



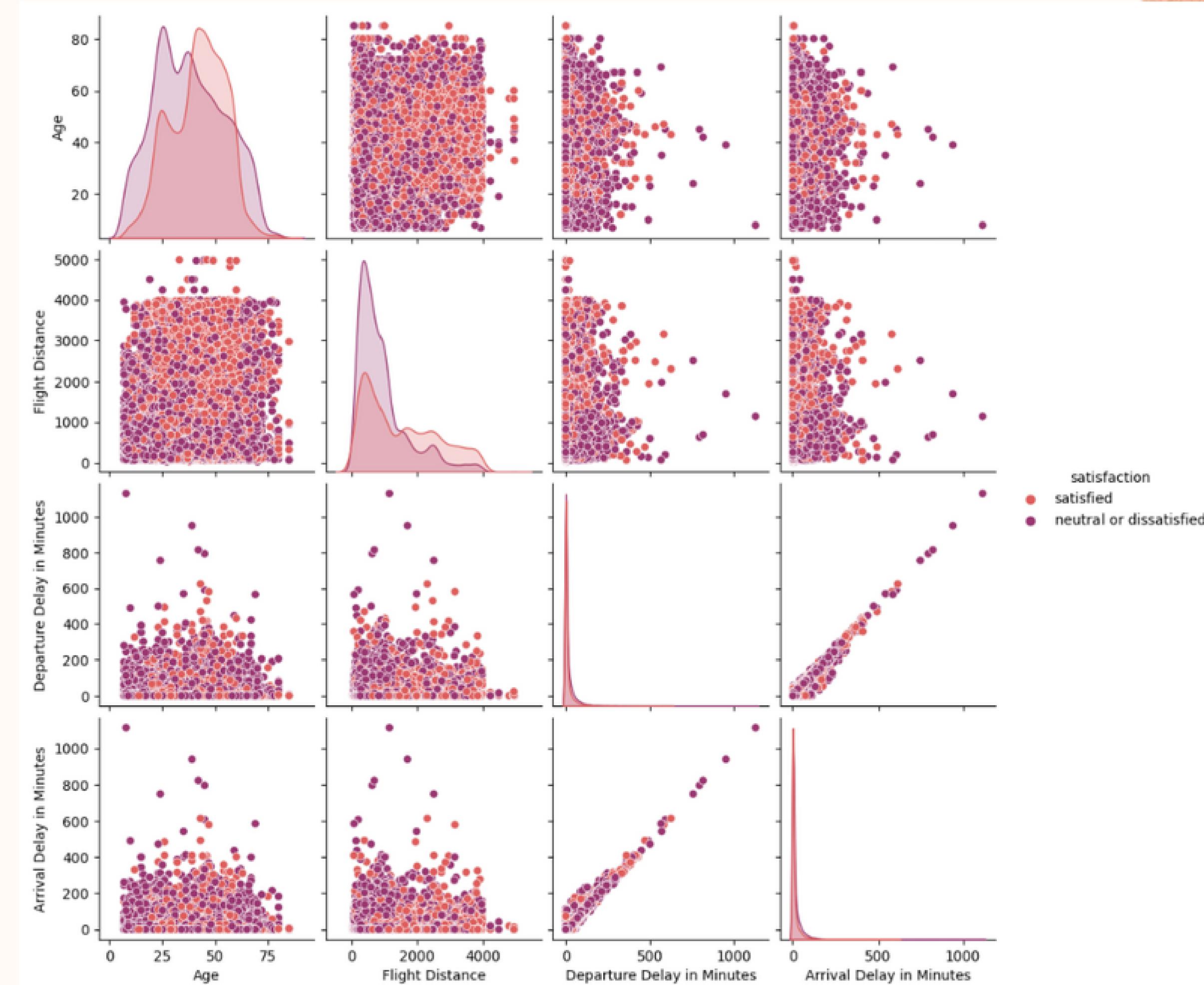
Satisfaction results by Cleanliness



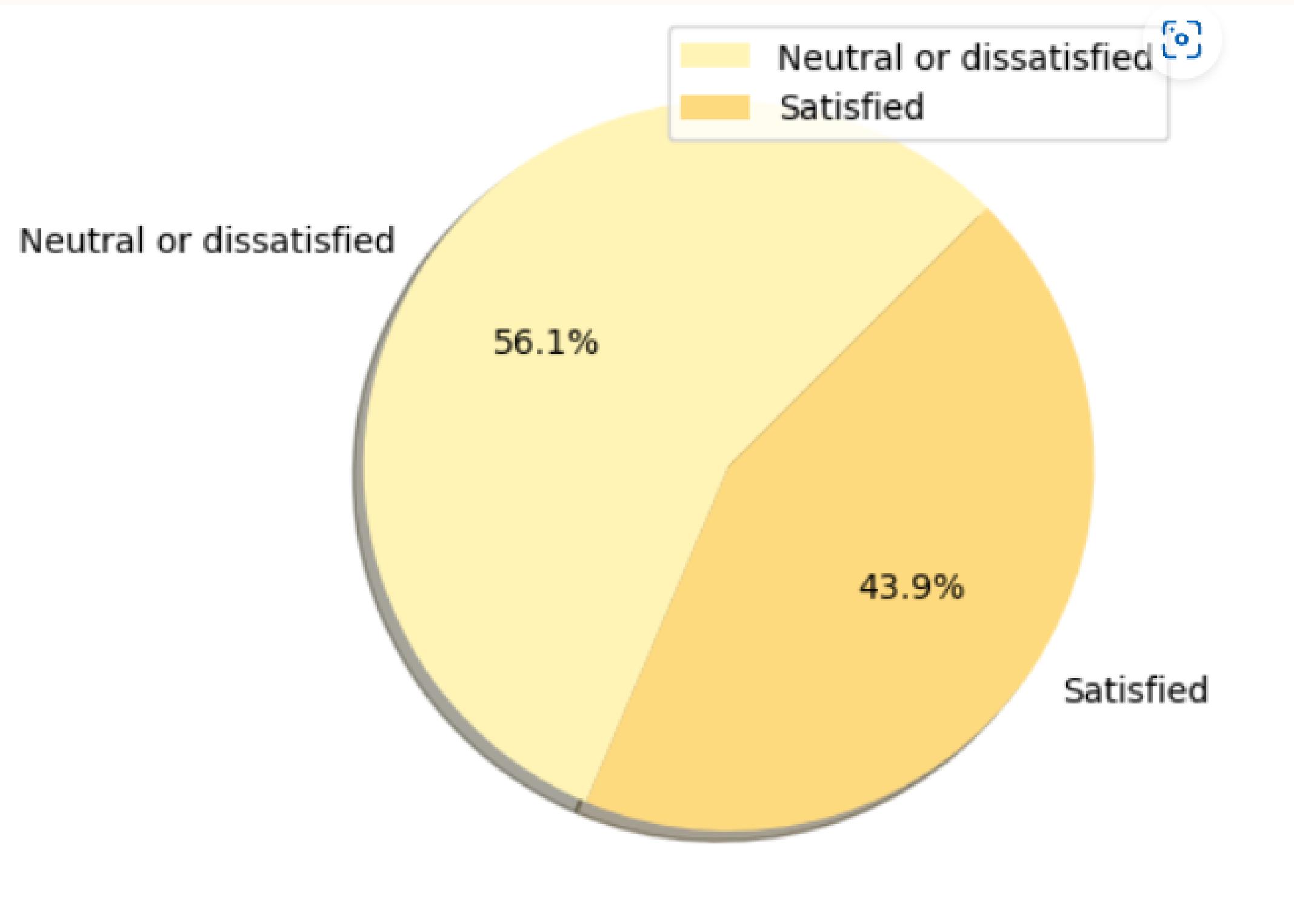
Satisfaction results by Food and drink

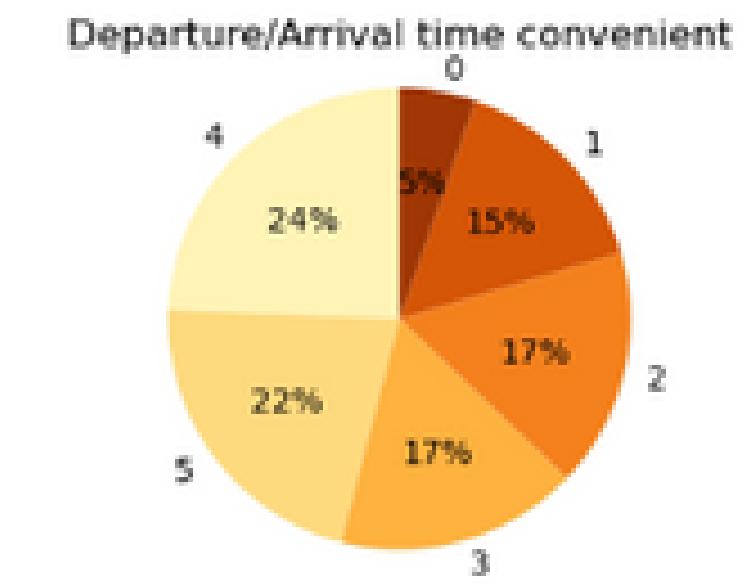
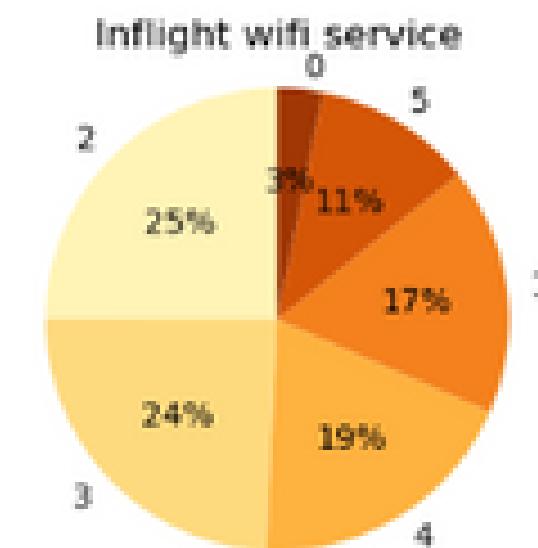
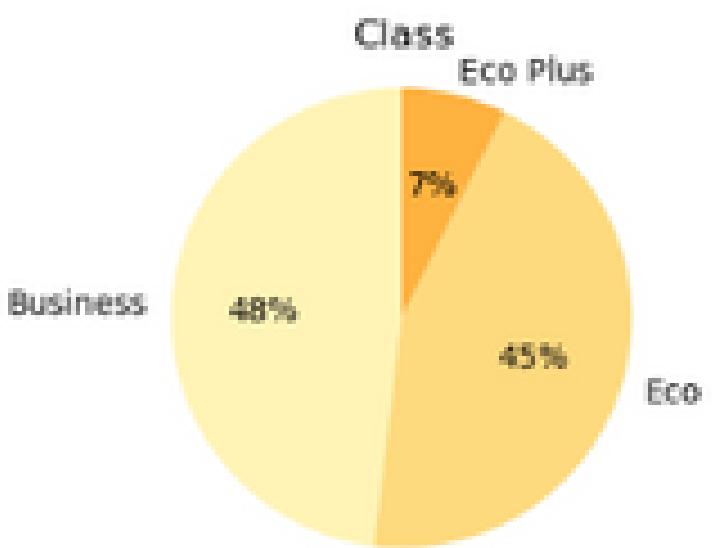
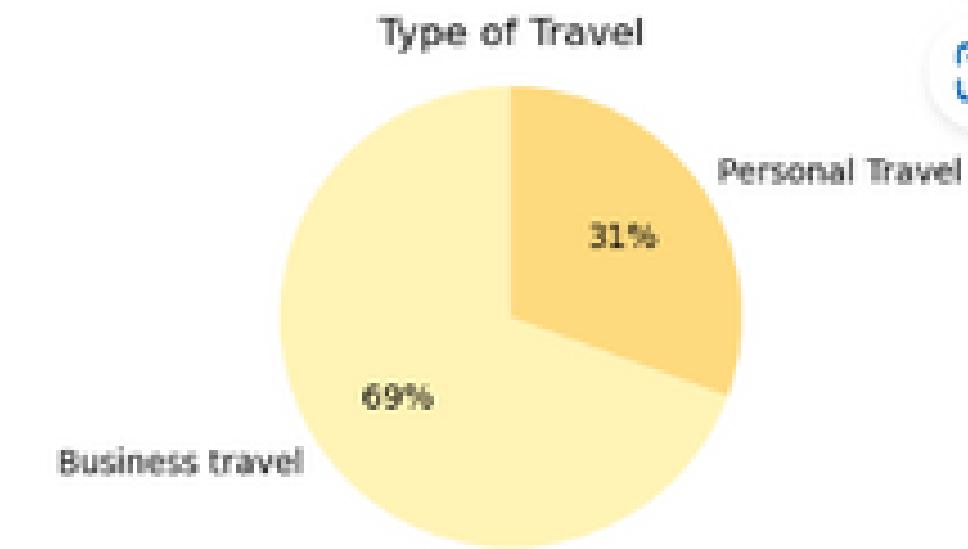
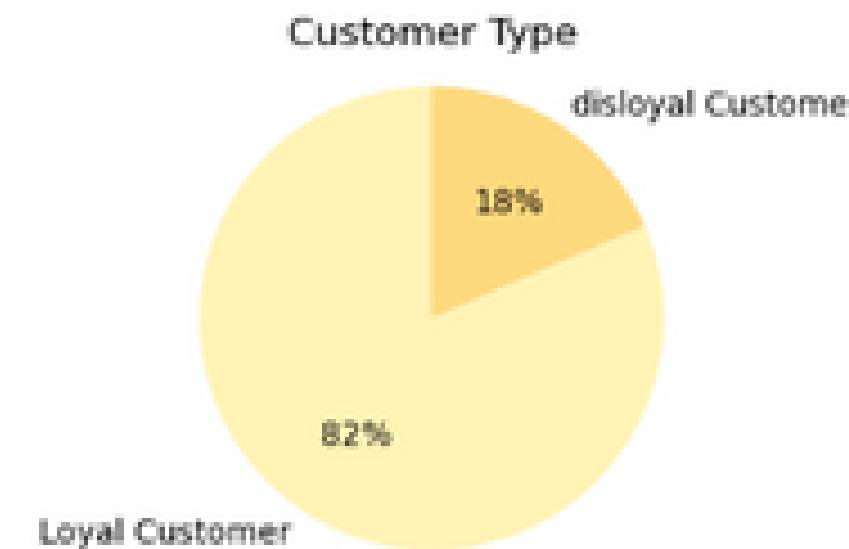
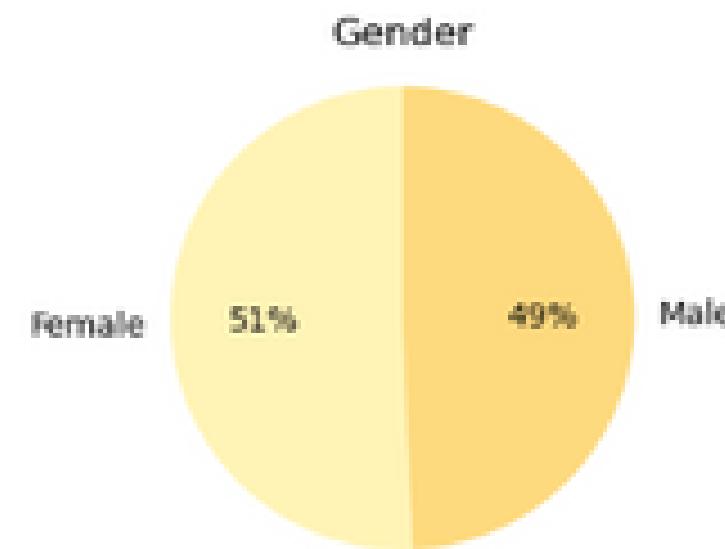


Pair pot

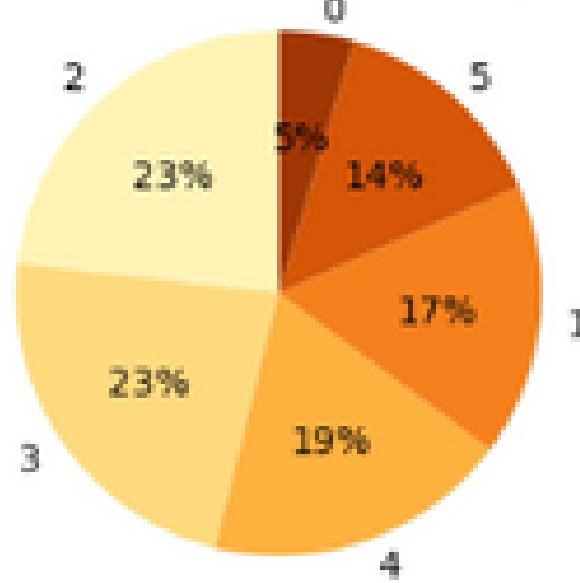


Pie chat

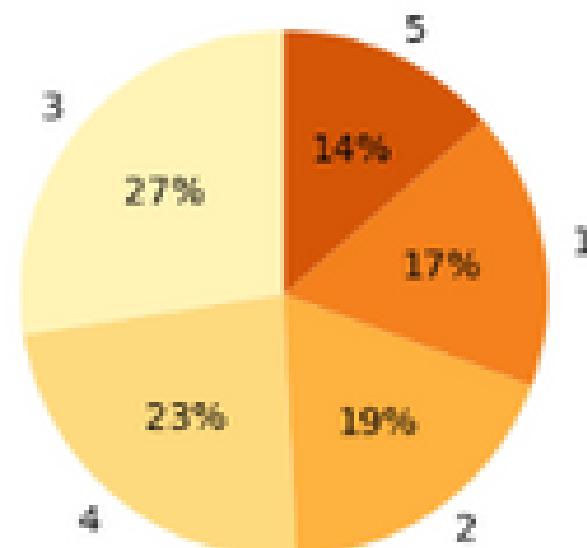




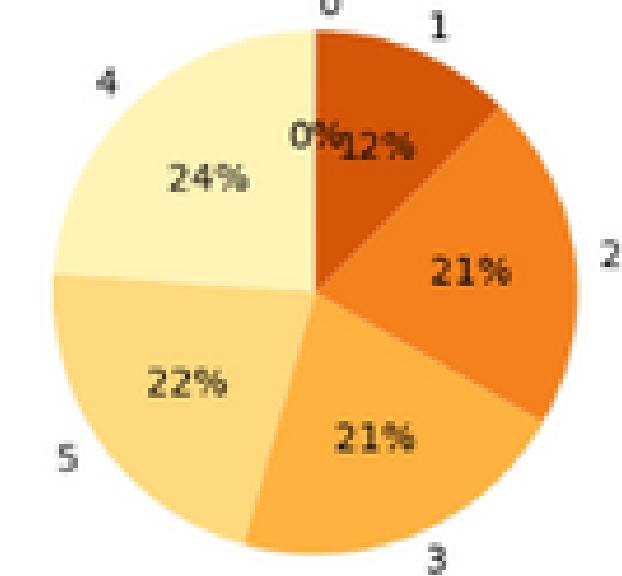
Ease of Online booking



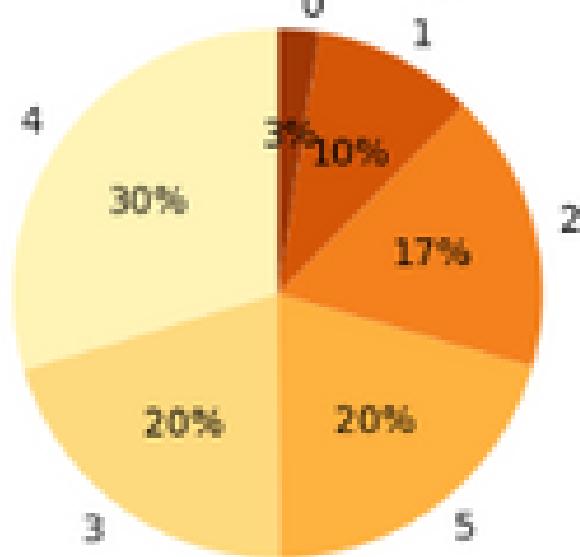
Gate location



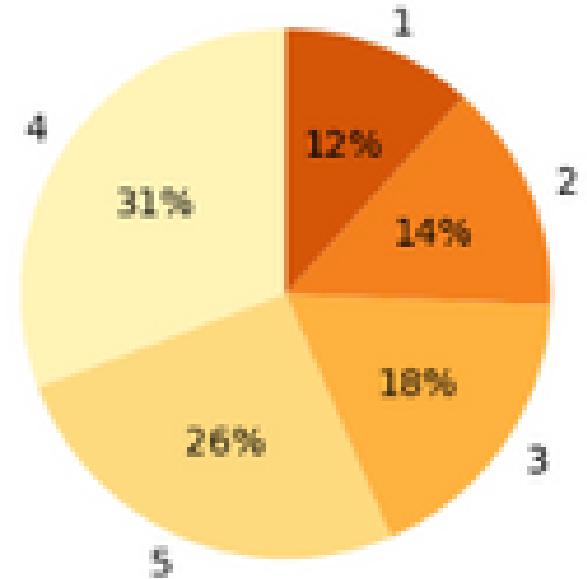
Food and drink



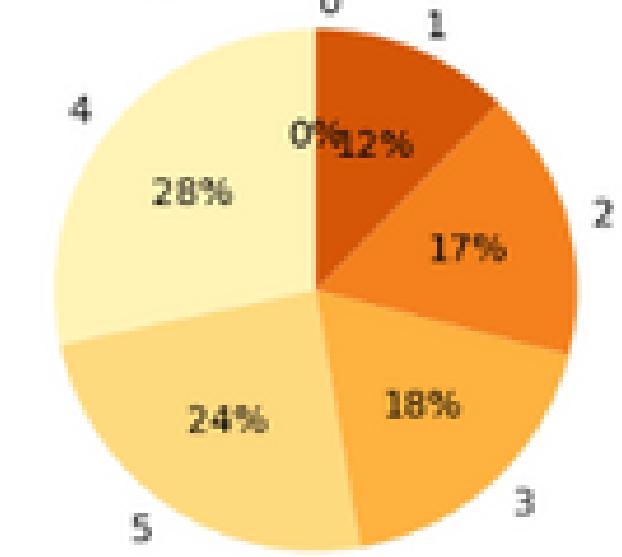
Online boarding

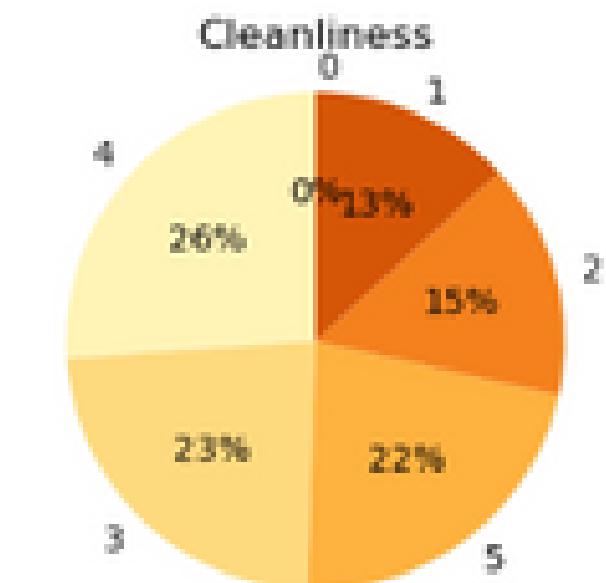
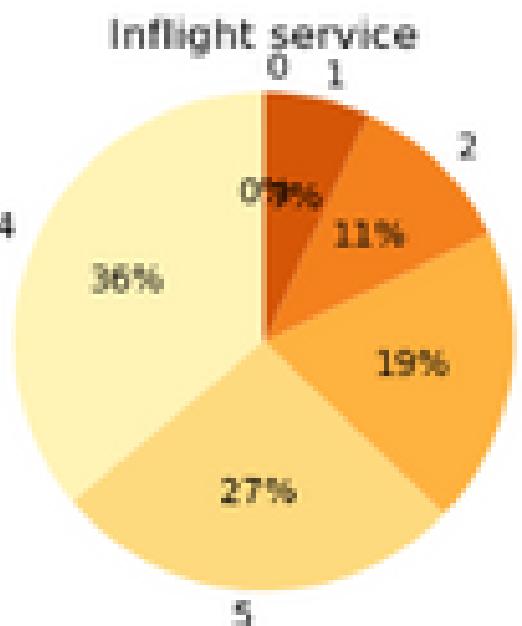
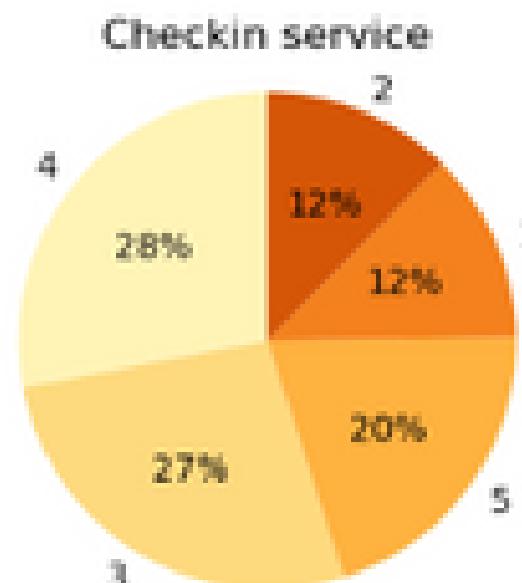
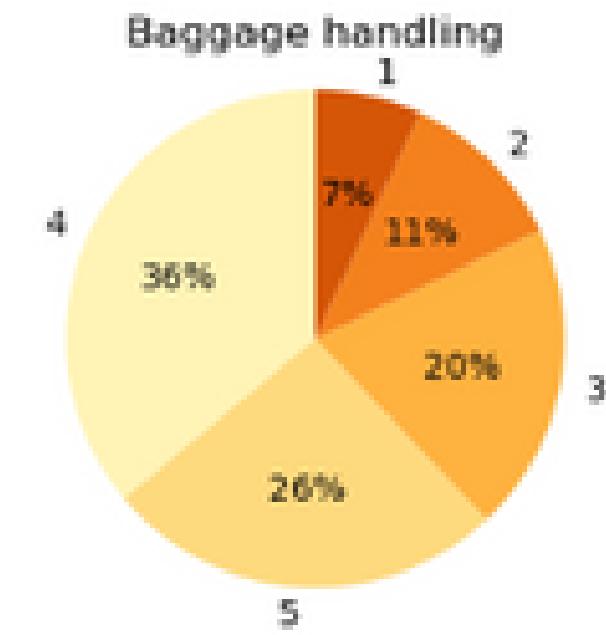
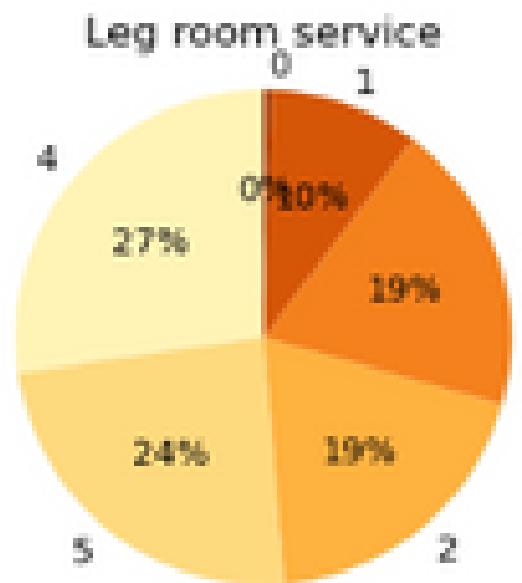
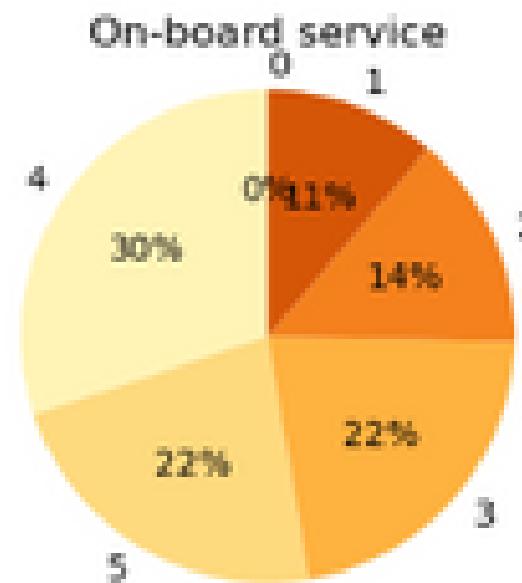


Seat comfort



Inflight entertainment





5). Feature Engineering

5.1). Encoding Categorical Variables

In classification problems, the goal is to predict the category of a new observation based on a set of training data. However, machine learning algorithms cannot directly work with categorical variables. They can only work with numerical data. Therefore, it is necessary to encode categorical variables into numerical values before they can be used in a classification model.

```
from sklearn.preprocessing import LabelEncoder

def encode_dataframe(df):
    encoding_mapping = {}

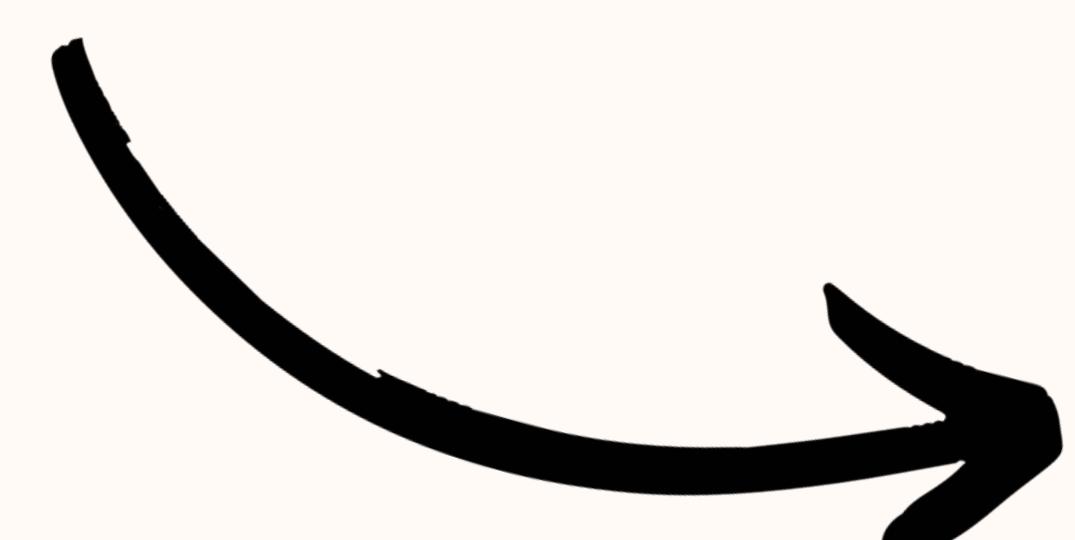
    # Iterate over each column
    for column in df.columns:
        # Check if the column dtype is object (i.e., categorical)
        if df[column].dtype == 'object':
            # Create a LabelEncoder object
            label_encoder = LabelEncoder()
            # Fit and transform the column to obtain encoded values
            encoded_values = label_encoder.fit_transform(df[column])
            # Create a dictionary mapping original values to encoded values
            encoding_mapping[column] = dict(zip(df[column], encoded_values))
            # Replace the column values with encoded values
            df[column] = encoded_values

    return df, encoding_mapping
```

```
encoded_df, encoding_mapping = encode_dataframe(df)

encoded_df
```

Gender	Customer_Type	Age	Type_of_Travel	Class	satisfaction
Male	Loyal Customer	13	Personal Travel	Eco Plus	neutral or dissatisfied
Male	disloyal Customer	25	Business travel	Business	neutral or dissatisfied
Female	Loyal Customer	26	Business travel	Business	satisfied
Female	Loyal Customer	25	Business travel	Business	neutral or dissatisfied
Male	Loyal Customer	61	Business travel	Business	satisfied



Gender	Customer_Type	Age	Type_of_Travel	Class	satisfaction
1	0	13	1	2	0
1	1	25	0	0	0
0	0	26	0	0	1
0	0	25	0	0	0
1	0	61	0	0	1
...
1	1	34	0	0	0
1	0	23	0	0	1
0	0	17	1	1	0
1	0	14	0	0	1
0	0	42	1	1	0

5.2). Feature Selection and Feature Importance

Setting our categorical
& target variable

```
categorical_features = X[[  
    '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', 'Checkin_service',  
    'Inflight_service', 'Cleanliness']].copy()  
y_target_cat = y
```

Feature Selection by Chi-Sqaure
Test

```
selector = SelectKBest(chi2, k=10)  
X_10_best = selector.fit_transform(categorical_features, y)  
  
# Get the mask of selected features  
selected_features_mask = selector.get_support()  
  
# Get the names of selected features  
chi_10_selected_features = categorical_features.columns[selected_features_mask]  
  
print("The 10 most important features selected by chi-square test:")  
print(chi_10_selected_features)
```

```
The 10 most important features selected by chi-square test:  
Index(['Customer_Type', 'Type_of_Travel', 'Class', 'Inflight_wifi_service',  
       'Online_boarding', 'Seat_comfort', 'Inflight_entertainment',  
       'On-board_service', 'Leg_room_service', 'Cleanliness'],  
      dtype='object')
```

```
Online_boarding          0.163768  
Inflight_wifi_service   0.145169  
Type_of_Travel          0.099629  
Class                   0.097405  
Inflight_entertainment  0.053983  
Seat_comfort            0.047860  
Flight_Distance         0.035487  
Customer_Type           0.035108  
Ease_of_Online_booking   0.034178  
On-board_service        0.033440  
Leg_room_service        0.033235  
Age                     0.030020  
id                      0.029826  
Cleanliness             0.026593  
Checkin_service         0.023300  
Inflight_service        0.022528  
Baggage_handling        0.021653  
Departure/Arrival_time_convenient 0.016849  
Gate_location           0.015630  
Arrival_Delay_in_Minutes 0.010873  
Food_and_drink          0.010354  
Departure_Delay_in_Minutes 0.009383  
Gender                  0.003728  
dtype: float64
```

```
from sklearn.feature_selection import SelectFromModel  
  
selector = SelectFromModel(rf(n_estimators=100, random_state=0))  
selected_features = selector.fit_transform(X, y)  
support = selector.get_support()  
wrapper_features = X.loc[:, support].columns.tolist()  
print(wrapper_features)  
  
model = rf(n_estimators=100, random_state=0)  
model.fit(X, y)  
feature_importances = pd.Series(model.feature_importances_, index=X.columns)  
print(feature_importances)
```



Feature Importance by Wrapper Method (random forest)

Feature Importance by Permutation Method

```
import warnings
warnings.filterwarnings("ignore")
import eli5
from eli5.sklearn import PermutationImportance

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())
```

Weight	Feature
0.1466 ± 0.0008	Inflight_wifi_service
0.1350 ± 0.0016	Type_of_Travel
0.0532 ± 0.0006	Customer_Type
0.0411 ± 0.0011	Online_boarding
0.0339 ± 0.0005	Class
0.0259 ± 0.0004	Checkin_service
0.0186 ± 0.0005	Seat_comfort
0.0179 ± 0.0003	Baggage_handling
0.0158 ± 0.0002	Inflight_service
0.0151 ± 0.0006	Cleanliness
0.0146 ± 0.0007	id
0.0108 ± 0.0004	Age
0.0101 ± 0.0005	On-board_service
0.0084 ± 0.0003	Leg_room_service
0.0082 ± 0.0003	Flight_Distance
0.0074 ± 0.0003	Inflight_entertainment
0.0073 ± 0.0001	Arrival_Delay_in_Minutes
0.0060 ± 0.0002	Ease_of_Online_booking
0.0043 ± 0.0003	Gate_location
0.0040 ± 0.0002	Departure_Delay_in_Minutes
... 3 more ...	

Now, let's analyze the result:

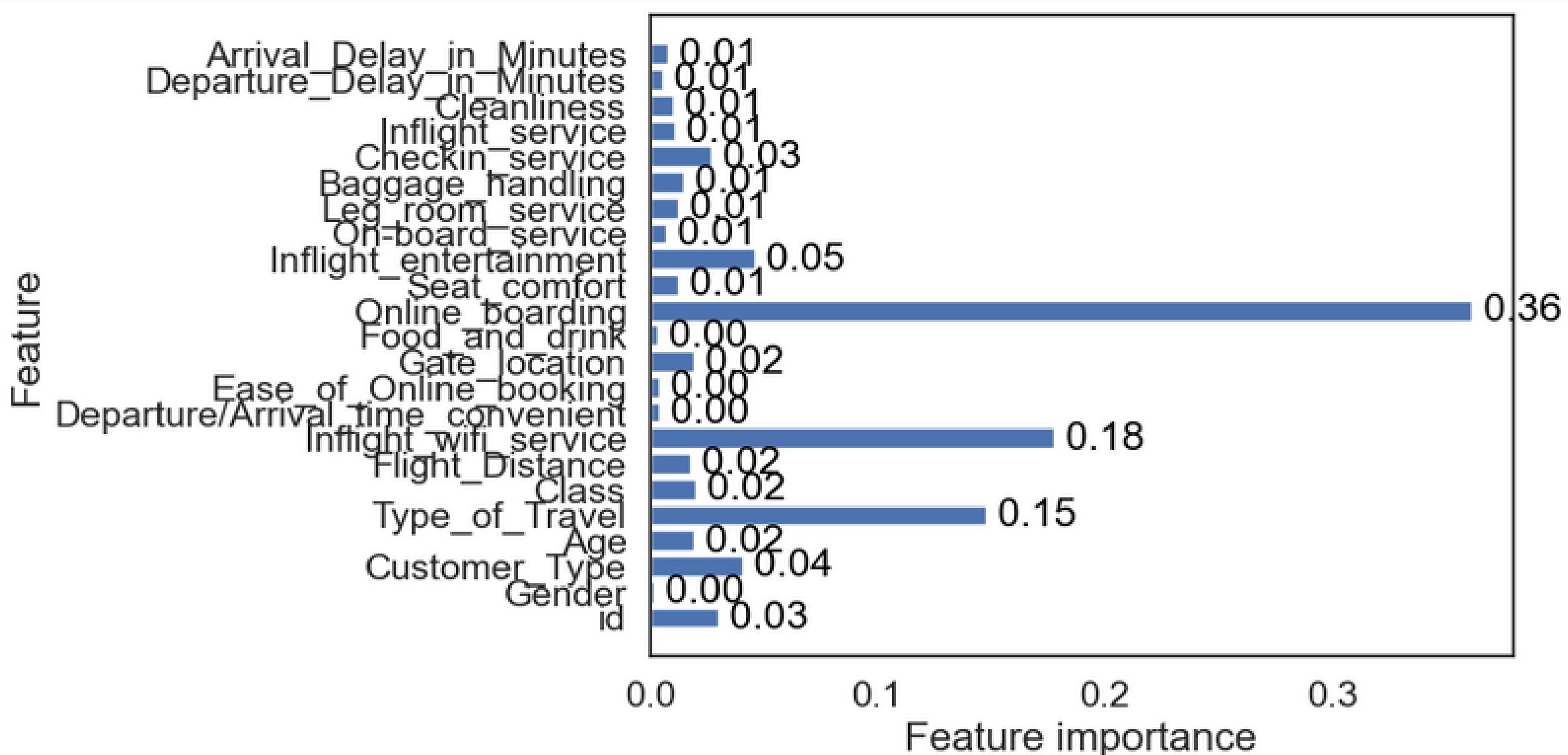
1. Inflight_wifi_service: This feature has the highest weight (0.1466) and is considered the most important feature according to Permutation Importance. It means that if you randomly permute the values of this feature, the model's performance would decrease significantly.
2. Type_of_Travel: This feature has the second-highest weight (0.1350). It is another important feature for the model's predictions.
3. Customer_Type: This feature has a weight of 0.0532, indicating its importance but to a lesser extent than the previous two features.
4. Online_boarding: This feature has a weight of 0.0411, suggesting it has some impact on the model's predictions.
5. Class: This feature has a weight of 0.0339, indicating its importance but to a lesser extent than the previous features.

```

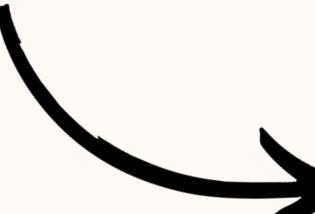
def plot_feature_importances_cancer(model, feature_names):
    n_features = len(feature_names)
    feature_importances = model.feature_importances_
    plt.barh(range(n_features), feature_importances, align='center')
    plt.yticks(np.arange(n_features), feature_names)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature")
    for i, v in enumerate(feature_importances):
        plt.text(v, i, f" {v:.2f}", color='black', va='center')
# Assuming you have the following variables: tree, X_train_tree, y_train_tree, and df
tree = DecisionTreeClassifier(random_state=1)
tree.fit(X_train_tree, y_train_tree)
# Call the plot_feature_importances_cancer function
plot_feature_importances_cancer(tree, X.columns)
plt.show()

```

Feature Importance by Decision Tree



```
from sklearn.preprocessing import StandardScaler  
from sklearn.linear_model import LogisticRegression  
from sklearn.feature_selection import RFE  
  
# Scale the data  
scaler = StandardScaler()  
X_scaled = scaler.fit_transform(X)  
  
# Create a logistic regression model  
logreg = LogisticRegression(max_iter=1000)  
  
# Apply Recursive Feature Elimination (RFE)  
rfe = RFE(logreg, n_features_to_select=10)  
rfe.fit(X_scaled, y)  
  
selected_features_rfe = X.columns[rfe.support_].tolist()  
selected_features_rfe
```



Feature Importance by Recursive Feature Elimination



```
['Customer_Type',  
 'Type_of_Travel',  
 'Class',  
 'Inflight_wifi_service',  
 'Ease_of_Online_booking',  
 'Online_boarding',  
 'On-board_service',  
 'Leg_room_service',  
 'Checkin_service',  
 'Cleanliness']
```

forward_features

```
[ 'Customer_Type',  
  'Type_of_Travel',  
  'Inflight_wifi_service',  
  'Gate_location',  
  'Online_boarding',  
  'Inflight_entertainment',  
  'On-board_service',  
  'Leg_room_service',  
  'Checkin_service',  
  'Cleanliness' ]
```

backward_features

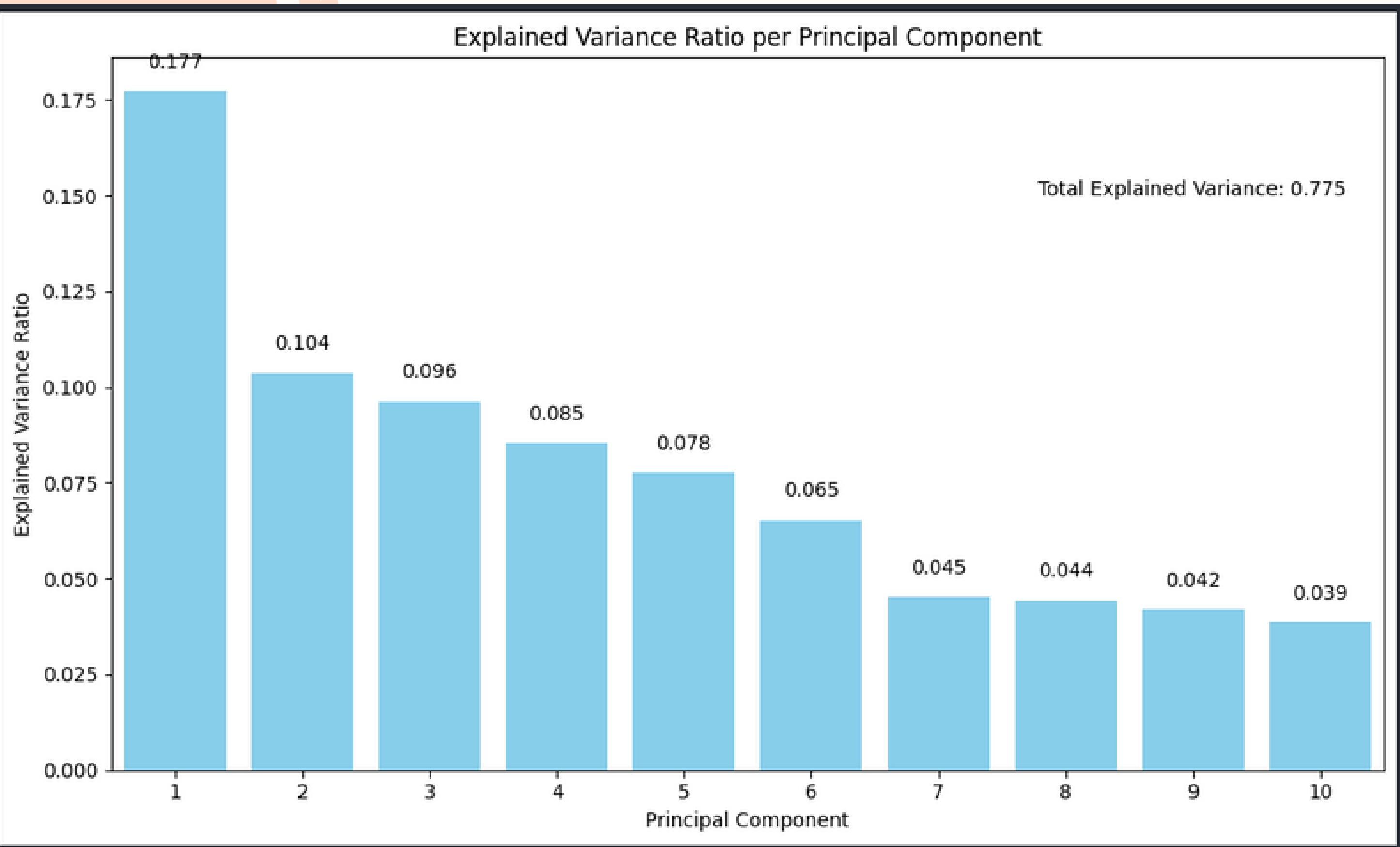
✓ 0.0s

```
[ 'Customer_Type',  
  'Type_of_Travel',  
  'Class',  
  'Inflight_wifi_service',  
  'Ease_of_Online_booking',  
  'Online_boarding',  
  'On-board_service',  
  'Leg_room_service',  
  'Checkin_service',  
  'Cleanliness' ]
```

```
def backward_selection(X, y, k_features=10):  
    scaler = StandardScaler()  
    X_scaled = scaler.fit_transform(X)  
  
    estimator = LogisticRegression()  
    selector = RFE(estimator, n_features_to_select=k_features, step=1)  
    selector = selector.fit(X_scaled, y)  
    selected_features = X.columns[selector.support_].tolist()  
    return selected_features  
  
def forward_selection(X, y, k_features=10):  
    scaler = StandardScaler()  
    X_scaled = scaler.fit_transform(X)  
  
    estimator = LogisticRegression()  
    selector = SequentialFeatureSelector(estimator, n_features_to_select=k_features, direction='forward')  
    selector = selector.fit(X_scaled, y)  
    selected_features = X.columns[selector.support_].tolist()  
    return selected_features  
  
def stepwise_selection(X, y, k_features=10):  
    scaler = StandardScaler()  
    X_scaled = scaler.fit_transform(X)  
  
    estimator = LogisticRegression()  
    selector = SequentialFeatureSelector(estimator, k_features=k_features, forward=True, floating=True, scoring='accuracy', cv=5)  
    selector = selector.fit(X_scaled, y)  
    selected_features = X.columns[list(selector.k_feature_idx_)].tolist()  
    return selected_features
```

✓ 0.0s

5.3). Dimentionality Reduction



Full_Set.shape

(129880, 25)

Projected_data.shape

(129880, 10)

PERFORM MODEL ALGORITHM



Linear regression analysis

satisfaction ~ Departure_Delay_in_Minutes + Arrival_Delay_in_Minutes

OLS Regression Results						
Dep. Variable:	satisfaction	R-squared:	0.004			
Model:	OLS	Adj. R-squared:	0.004			
Method:	Least Squares	F-statistic:	241.2			
Date:	Tue, 23 May 2023	Prob (F-statistic):	2.79e-105			
Time:	23:16:39	Log-Likelihood:	-92930.			
No. Observations:	129888	AIC:	1.858e+05			
Df Residuals:	129877	BIC:	1.858e+05			
Df Model:	2					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.4454	0.001	301.903	0.000	0.443	0.448
Departure_Delay_in_Minutes	0.0008	0.000	6.455	0.000	0.001	0.001
Arrival_Delay_in_Minutes	-0.0015	0.000	-12.115	0.000	-0.002	-0.001
Omnibus:	466876.768	Durbin-Watson:	2.006			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	21368.532			
Skew:	0.263	Prob(JB):	0.00			
Kurtosis:	1.084	Cond. No.	61.8			

satisfaction ~ Arrival_Delay_in_Minutes

OLS Regression Results						
Dep. Variable:	satisfaction	R-squared:	0.003			
Model:	OLS	Adj. R-squared:	0.003			
Method:	Least Squares	F-statistic:	449.6			
Date:	Sat, 20 May 2023	Prob (F-statistic):	1.17e-97			
Time:	17:20:38	Log-Likelihood:	-92920.			
No. Observations:	129880	AIC:	1.858e+05			
Df Residuals:	129878	BIC:	1.859e+05			
Df Model:	1					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.4458	0.001	302.274	0.000	0.443	0.449
Arrival_Delay_in_Minutes	-0.0008	3.57e-05	-20.990	0.000	-0.001	-0.001
Omnibus:	466538.303	Durbin-Watson:		2.006		
Prob(Omnibus):	0.000	Jarque-Bera (JB):		21395.028		
Skew:	0.263	Prob(JB):		0.00		
Kurtosis:	1.083	Cond. No.		44.3		
Notes:						
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.						

satisfaction ~ Departure_Delay_in_Minutes

OLS Regression Results

Dep. Variable:	satisfaction	R-squared:	0.003			
Model:	OLS	Adj. R-squared:	0.003			
Method:	Least Squares	F-statistic:	335.2			
Date:	Sat, 20 May 2023	Prob (F-statistic):	8.63e-75			
Time:	17:20:40	Log-Likelihood:	-92973.			
No. Observations:	129880	AIC:	1.859e+05			
Df Residuals:	129878	BIC:	1.860e+05			
Df Model:	1					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.4442	0.001	301.615	0.000	0.441	0.447
Departure_Delay_in_Minutes	-0.0007	3.61e-05	-18.310	0.000	-0.001	-0.001
Omnibus:	465598.855	Durbin-Watson:	2.086			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	21462.111			
Skew:	0.263	Prob(JB):	0.00			
Kurtosis:	1.079	Cond. No.	43.8			

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Multicollinearity

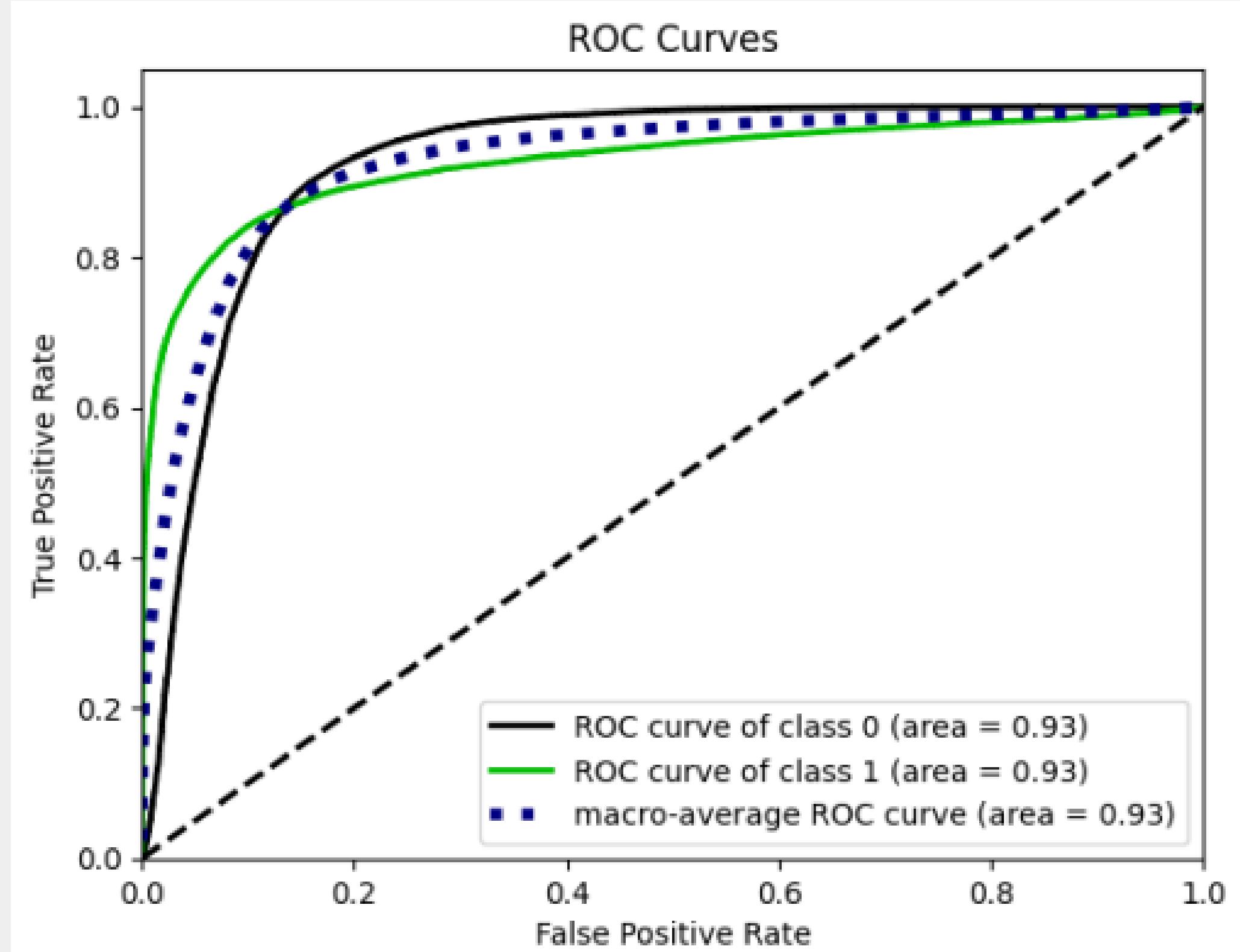
VIF Factor	features
0 12.565103	Departure_Delay_in_Minutes
1 12.565103	Arrival_Delay_in_Minutes
2 1.154836	Intercept

	Departure_Delay_in_Minutes	Arrival_Delay_in_Minutes	Intercept
0	25	18.0	1
1	1	6.0	1
2	0	0.0	1
3	11	9.0	1
4	0	0.0	1
...
129875	0	0.0	1
129876	0	0.0	1
129877	0	0.0	1
129878	0	0.0	1
129879	0	0.0	1

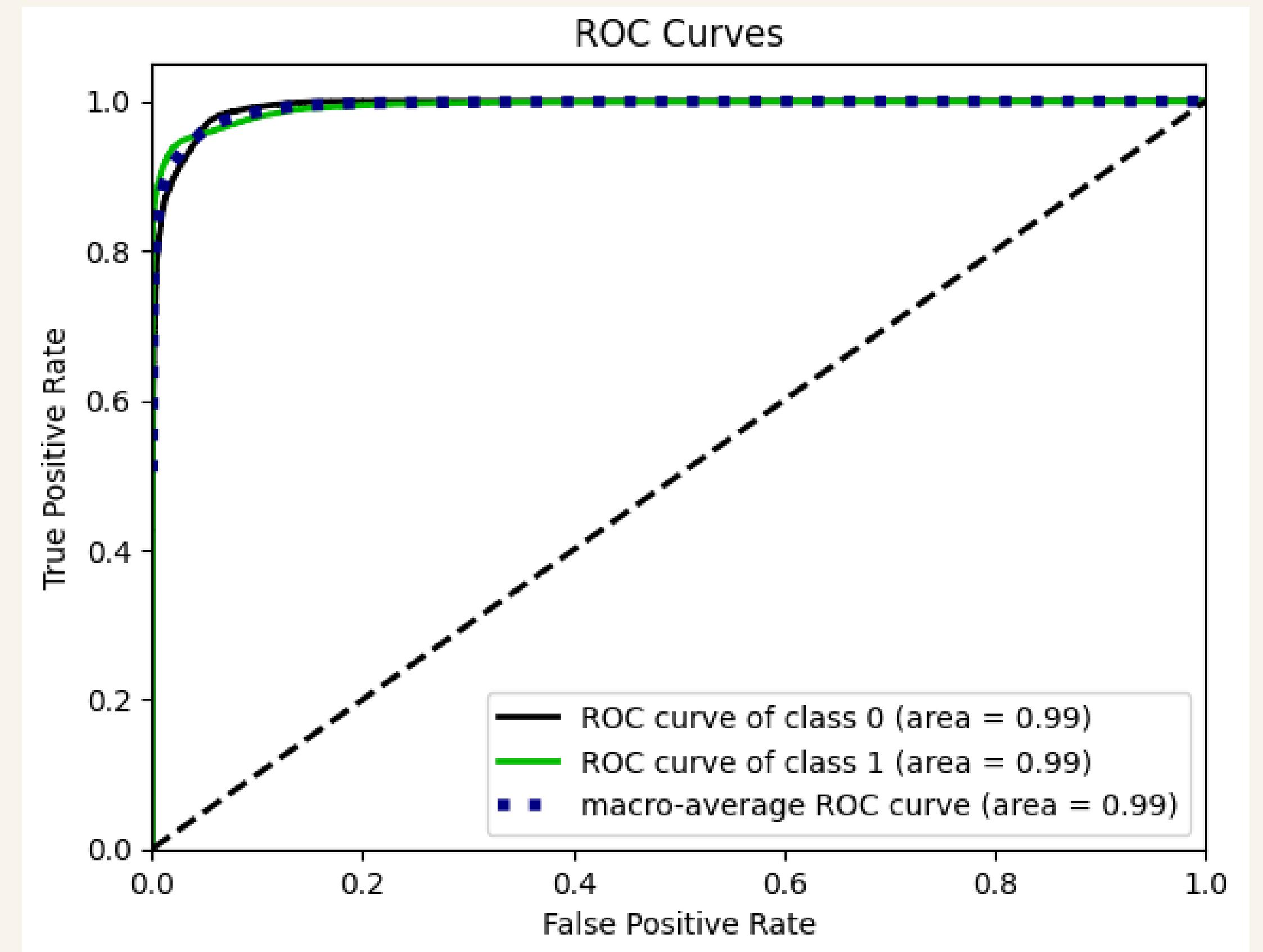
129880 rows × 3 columns

Both 'Departure_Delay_in_Minutes' and 'Arrival_Delay_in_Minutes' have relatively high VIF factors of approximately 12.565103. VIF values above 5 or 10 are often considered indicative of significant multicollinearity

ROC Curve of Simple Logistic regression



ROC Curve of RFC



6). Model Selection

Model1: Normal Logistic Regression
(all features)

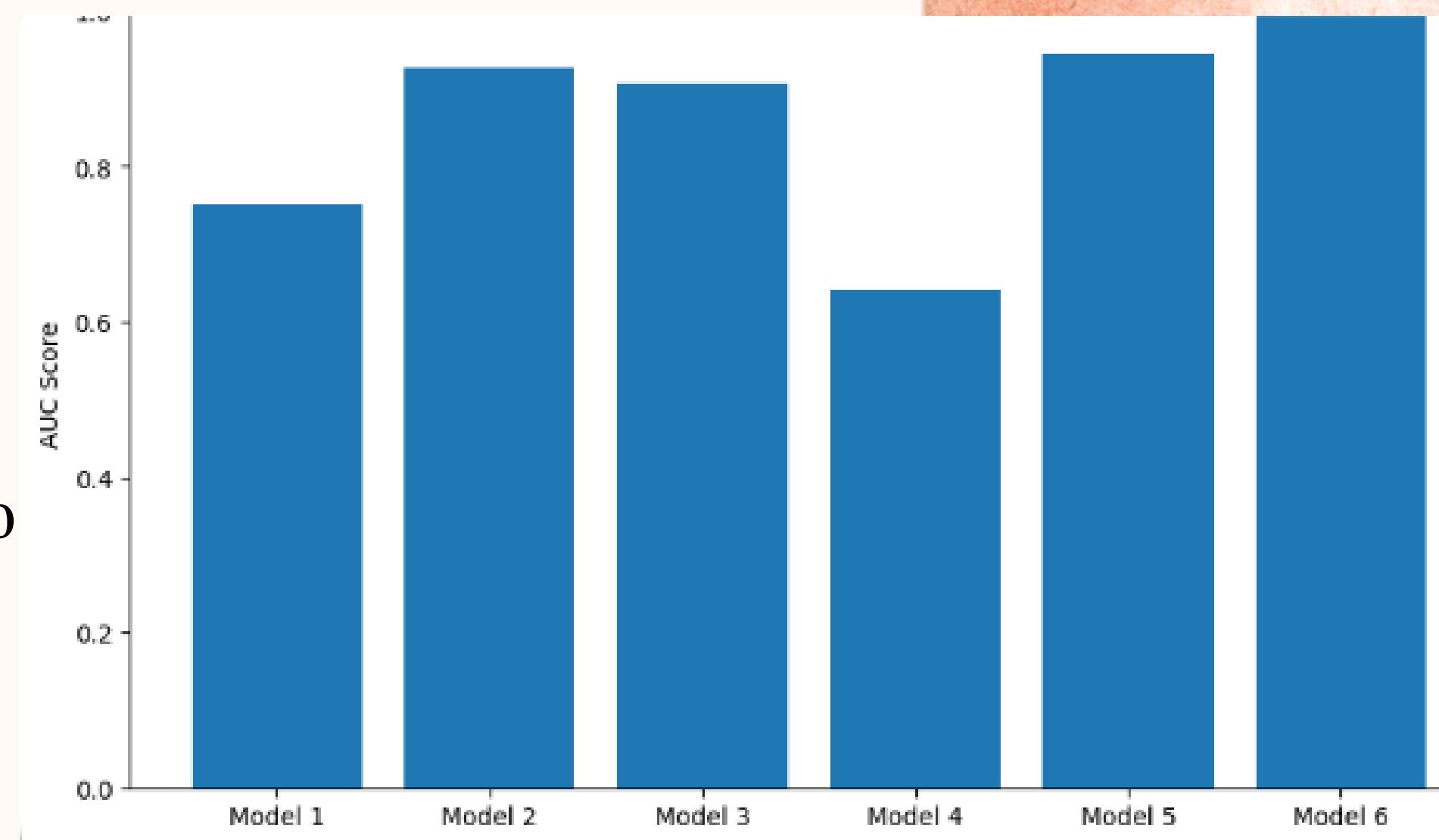
Model2: Logistic Regression Penalized with Lasso
(regularization-strength = 0.1)

Model3: Logistic Regression Penalized with Ridge
(L2 penalty = 50%)

Model4: Logistic Regression Penalized with Elastic Net
(L1 penalty = 50%,
L2 penalty = 50%)

Model5: Decision Tree

Model6: Random Forest



Model 1: AUC = 0.7506861718066697
Model 2: AUC = 0.9267654577702052
Model 3: AUC = 0.9079386048955072
Model 4: AUC = 0.6412692385246368
Model 5: AUC = 0.9459782563100856
Model 6: AUC = 0.9935930212658386

7). Decision Making

In terms of algorithmic complexity, it is best that we go with random forest classifier. However, in terms of logistic regression model, we select the mentioned features.

The important factors that make effect to customer satisfaction are :

- **Seat_comfort**
- **Inflight_entertainment**
- **class**
- **Type_of_Travel**
- **Inflight_wifi_service**
- **Online_boarding**



Thank
you