In [201…

```
import os
os.getcwd()
```

Out[201…

```
'C:\\Users\\guy74\\Documents\\NU Stuff\\ANA500\\ANA500_MicroProject'
```

In [202…

```
import pandas as pd
import numpy as np

#Load in the data
df = pd.read_csv("airline.csv")

#There is a column which is presumably actually not named in the source but at some
#That variable is presumably not useful here analytically, and same goes for the "i
#The rest could serve useful and so we don't want to drop anything else yet at this
df = df.drop(columns=["Unnamed: 0", "id"])
#print(df)

#Loop through each column and calculate some data integrity metrics for each one to
column_checks = []
total_rows = df.shape[0]
for col in df.columns:
    col_data = df[col]

    populated_count = col_data.notnull().sum()
    missing_count = col_data.isnull().sum()
    distinct_count = col_data.nunique(dropna=True)
    data_type = col_data.dtype

    column_checks.append({
        "Column": col,
        "Populated Values": populated_count,
        "Missing Values": missing_count,
        "Distinct Values": distinct_count,
        "Data Type": str(data_type)
    })

integrity_df = pd.DataFrame(column_checks)

#This can sort it by missing values if we want the highest of those to rise to the
#integrity_df = integrity_df.sort_values(by="Missing Values", ascending=False)


print(integrity_df.to_string(index=False))
```

	Column	Populated Values	Missing Values	Distinct Values	Data Type
	Gender	129880	0	2	object
	Customer Type	129880	0	2	object
	Age	129880	0	75	int64
	Type of Travel	129880	0	2	object
	Class	129880	0	3	object
	Flight Distance	129880	0	3821	int64
	Inflight wifi service	129880	0	6	int64
	Departure/Arrival time convenient	129880	0	6	int64
	Ease of Online booking	129880	0	6	int64
	Gate location	129880	0	6	int64
	Food and drink	129880	0	6	int64
	Online boarding	129880	0	6	int64
	Seat comfort	129880	0	6	int64
	Inflight entertainment	129880	0	6	int64
	On-board service	129880	0	6	int64
	Leg room service	129880	0	6	int64
	Baggage handling	129880	0	5	int64
	Checkin service	129880	0	6	int64
	Inflight service	129880	0	6	int64
	Cleanliness	129880	0	6	int64
	Departure Delay in Minutes	129880	0	466	int64
	Arrival Delay in Minutes	129487	393	472	float64
	satisfaction	129880	0	2	object

In [203…
```
#Let's also check for any duplicate rows where every single variable is exactly the
duplicate_count = df.duplicated().sum()
print(f"Total number of exact duplicate rows (excluding the two dropped columns): {
```

```
Total number of exact duplicate rows (excluding the two dropped columns): 0
```

In [204…
```
#Swap out the column names so if any columns have spaces we change them to undersco
# Replace all spaces in column names with underscores
```

```
df.columns = df.columns.str.replace(" ", "_")
print(df)
```

```
        Gender      Customer_Type  Age   Type_of_Travel     Class  \
0         Male     Loyal Customer   13  Personal Travel  Eco Plus
1         Male  disloyal Customer   25  Business travel  Business
2       Female     Loyal Customer   26  Business travel  Business
3       Female     Loyal Customer   25  Business travel  Business
4         Male     Loyal Customer   61  Business travel  Business
...        ...                ...  ...              ...       ...
129875    Male  disloyal Customer   34  Business travel  Business
129876    Male     Loyal Customer   23  Business travel  Business
129877  Female     Loyal Customer   17  Personal Travel       Eco
129878    Male     Loyal Customer   14  Business travel  Business
129879  Female     Loyal Customer   42  Personal Travel       Eco

        Flight_Distance  Inflight_wifi_service  \
0                   460                      3
1                   235                      3
2                  1142                      2
3                   562                      2
4                   214                      3
...                 ...                    ...
129875              526                      3
129876              646                      4
129877              828                      2
129878             1127                      3
129879              264                      2

        Departure/Arrival_time_convenient  Ease_of_Online_booking  \
0                                       4                       3
1                                       2                       3
2                                       2                       2
3                                       5                       5
4                                       3                       3
...                                   ...                     ...
129875                                  3                       3
129876                                  4                       4
129877                                  5                       1
129878                                  3                       3
129879                                  5                       2

        Gate_location  ...  Inflight_entertainment  On-board_service  \
0                   1  ...                       5                 4
1                   3  ...                       1                 1
2                   2  ...                       5                 4
3                   5  ...                       2                 2
4                   3  ...                       3                 3
...               ...  ...                     ...               ...
129875              1  ...                       4                 3
129876              4  ...                       4                 4
129877              5  ...                       2                 4
129878              3  ...                       4                 3
129879              5  ...                       1                 1

        Leg_room_service  Baggage_handling  Checkin_service  Inflight_service  \
0                      3                 4                4                 5
1                      5                 3                1                 4
2                      3                 4                4                 4
```

```
3                    5                  3                  1                  4
4                    4                  4                  3                  3
...                ...                ...                ...                ...
129875               2                  4                  4                  5
129876               5                  5                  5                  5
129877               3                  4                  5                  4
129878               2                  5                  4                  5
129879               2                  1                  1                  1

        Cleanliness  Departure_Delay_in_Minutes  Arrival_Delay_in_Minutes  \
0                 5                         25                      18.0
1                 1                          1                       6.0
2                 5                          0                       0.0
3                 2                         11                       9.0
4                 3                          0                       0.0
...             ...                        ...                       ...
129875            4                          0                       0.0
129876            4                          0                       0.0
129877            2                          0                       0.0
129878            4                          0                       0.0
129879            1                          0                       0.0

                  satisfaction
0       neutral or dissatisfied
1       neutral or dissatisfied
2                     satisfied
3       neutral or dissatisfied
4                     satisfied
...                         ...
129875  neutral or dissatisfied
129876                satisfied
129877  neutral or dissatisfied
129878                satisfied
129879  neutral or dissatisfied

[129880 rows x 23 columns]
```

In [205…
```
#If/when needed, export the new version of the file to another named .csv for furth
df.to_csv('AirlineSatisfaction_Transformed.csv', index=False)
```

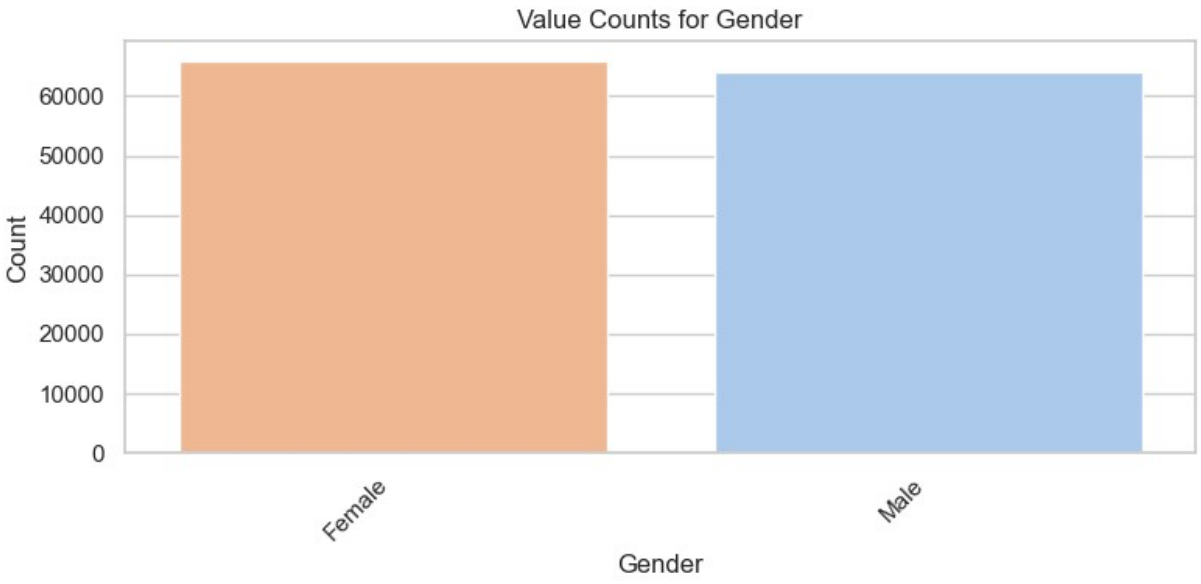
In [206…
```
#This marks the end of week 1 and now continuing on for week 2
```

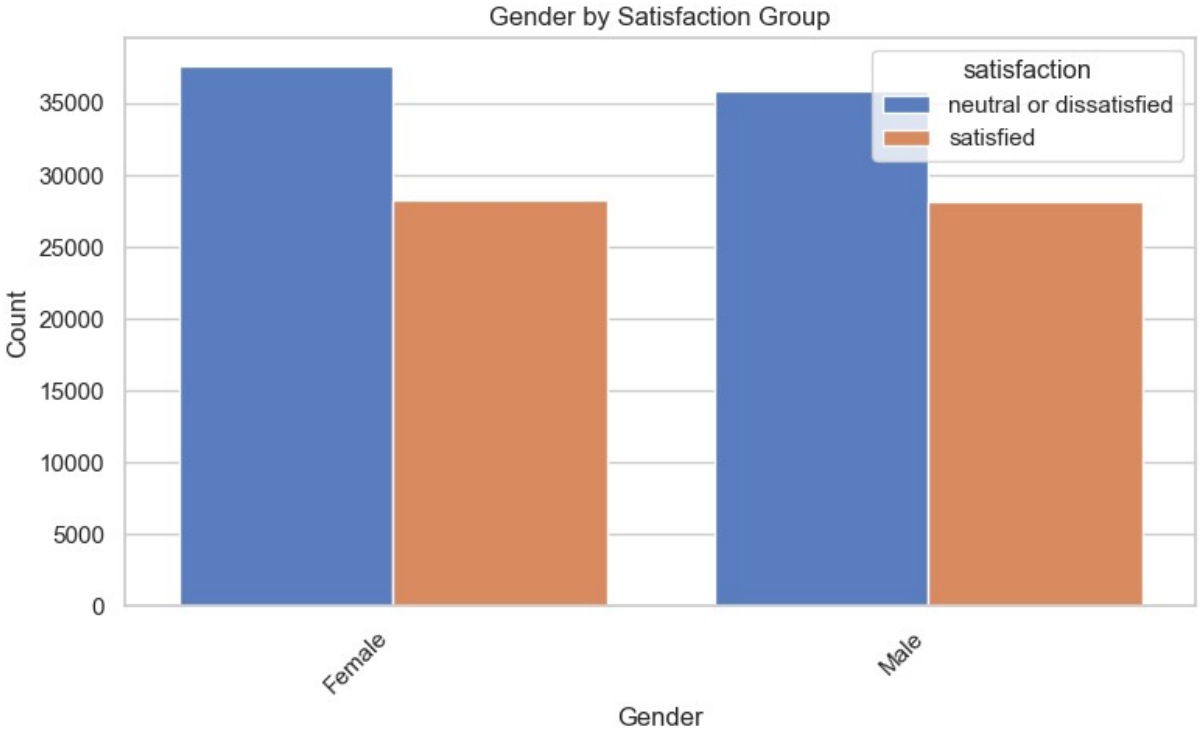
In [207…
```
#Feeback from last week indicated:
#To strengthen your analysis, please dive deeper into your columns.
#For example, explore how the Age variable is distributed overall and within satisf
#How many records are labeled as "satisfied" versus "unsatisfied"? What does the ag
#These insights can help guide your feature engineering and modeling steps.

#Therefore below going to do a bit further exploratory analysis for distribution ov
#(for each column present in the dataset via a loop)
```

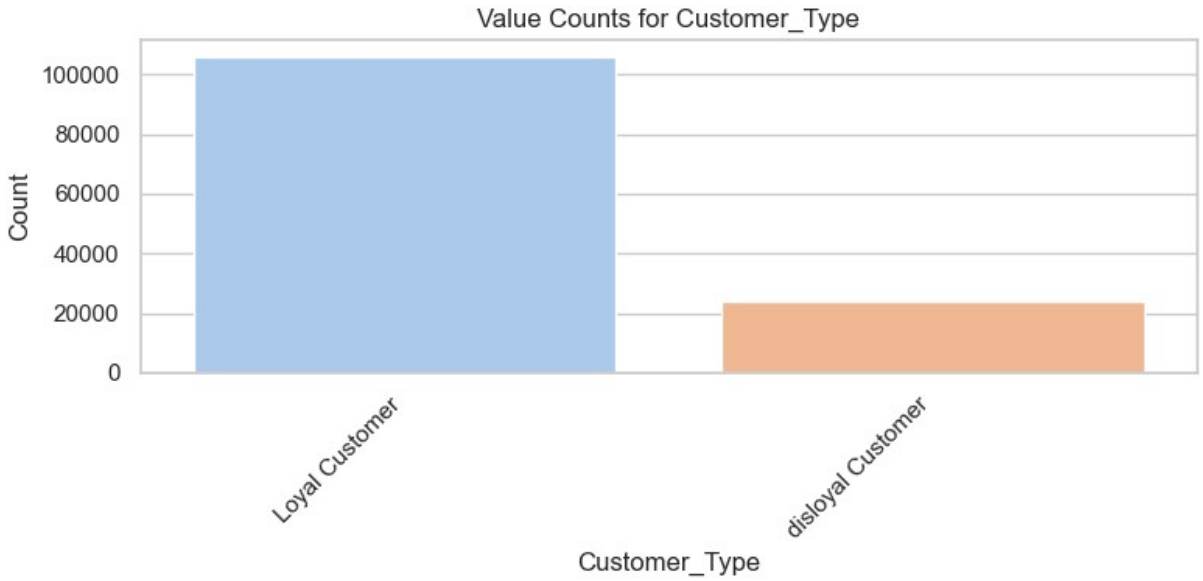
In [208…
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

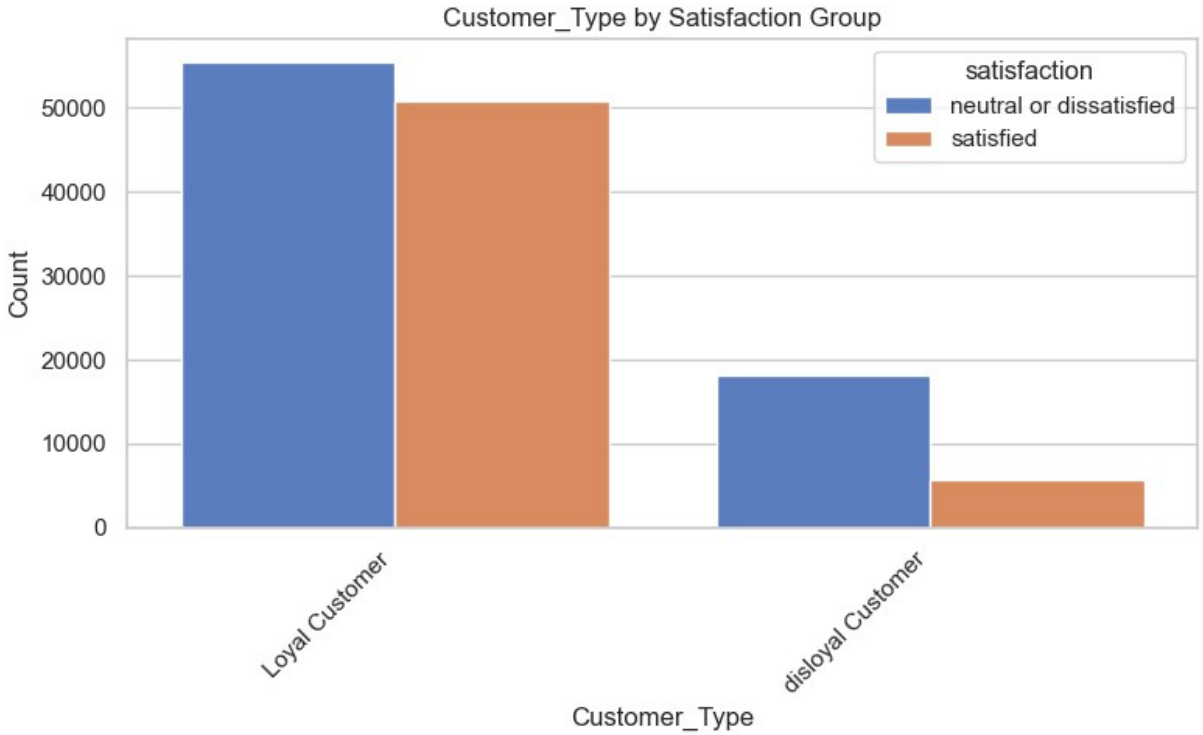
```
df = pd.read_csv("AirlineSatisfaction_Transformed.csv")
sns.set(style="whitegrid")

for col in df.columns:
    dtype = df[col].dtype

    if pd.api.types.is_numeric_dtype(dtype):
    #If it's numeric run distribution histograms
        #Overall distribution
        plt.figure(figsize=(8, 4))
        sns.histplot(df[col], bins=30, color="skyblue")
        plt.title(f"Overall Distribution of {col}")
        plt.xlabel(col)
        plt.ylabel("Count")
        plt.tight_layout()
        plt.show()

        #Grouped by satisfaction
        plt.figure(figsize=(8, 5))
        sns.histplot(
            data=df,
            x=col,
            hue="satisfaction",
            bins=30,
            element="step",
            stat="count",
            common_norm=False,
            palette="muted"
        )
        plt.title(f"{col} Distribution by Satisfaction Group")
        plt.xlabel(col)
        plt.ylabel("Count")
        plt.tight_layout()
        plt.show()

    elif pd.api.types.is_object_dtype(dtype) or pd.api.types.is_categorical_dtype(d
    #If it's categorical instead of numeric then now we'll run the counts

        #Overall
        plt.figure(figsize=(8, 4))
        sns.countplot(
            data=df,
            x=col,
            hue=col,
            order=df[col].value_counts().index,
            palette="pastel",
            legend=False
        )
        plt.title(f"Value Counts for {col}")
        plt.xlabel(col)
        plt.ylabel("Count")
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()

        #by satisfaction
```

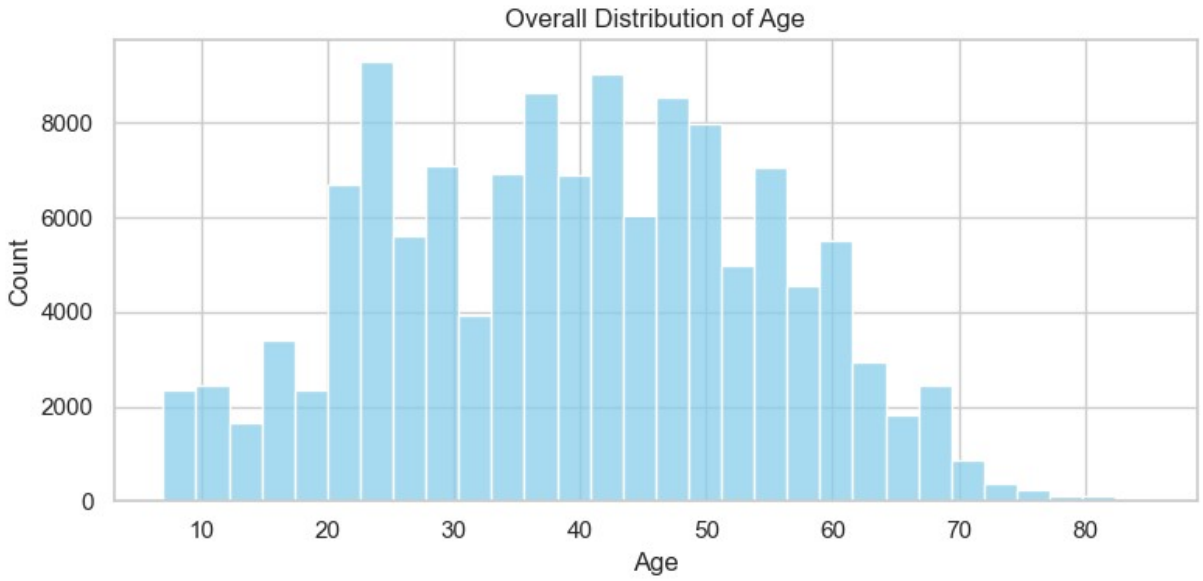
```
        plt.figure(figsize=(8, 5))
        sns.countplot(
            data=df,
            x=col,
            hue="satisfaction",
            order=df[col].value_counts().index,
            palette="muted"
        )
        plt.title(f"{col} by Satisfaction Group")
        plt.xlabel(col)
        plt.ylabel("Count")
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()
```

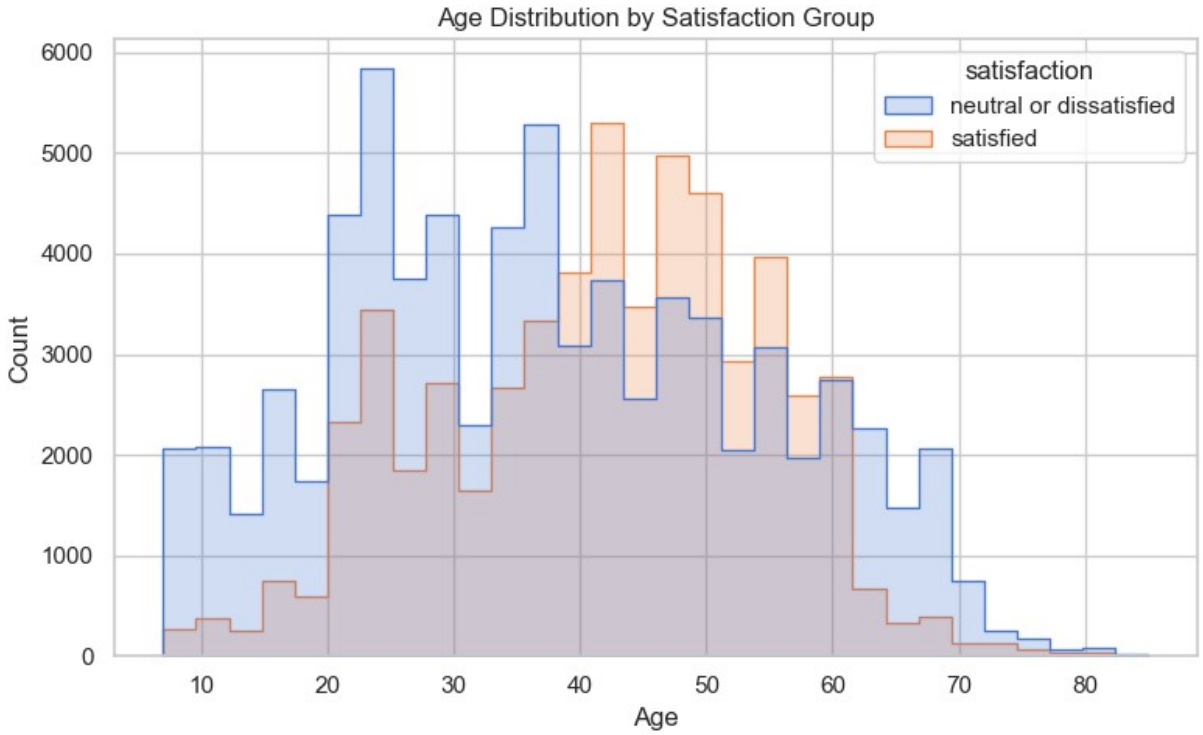


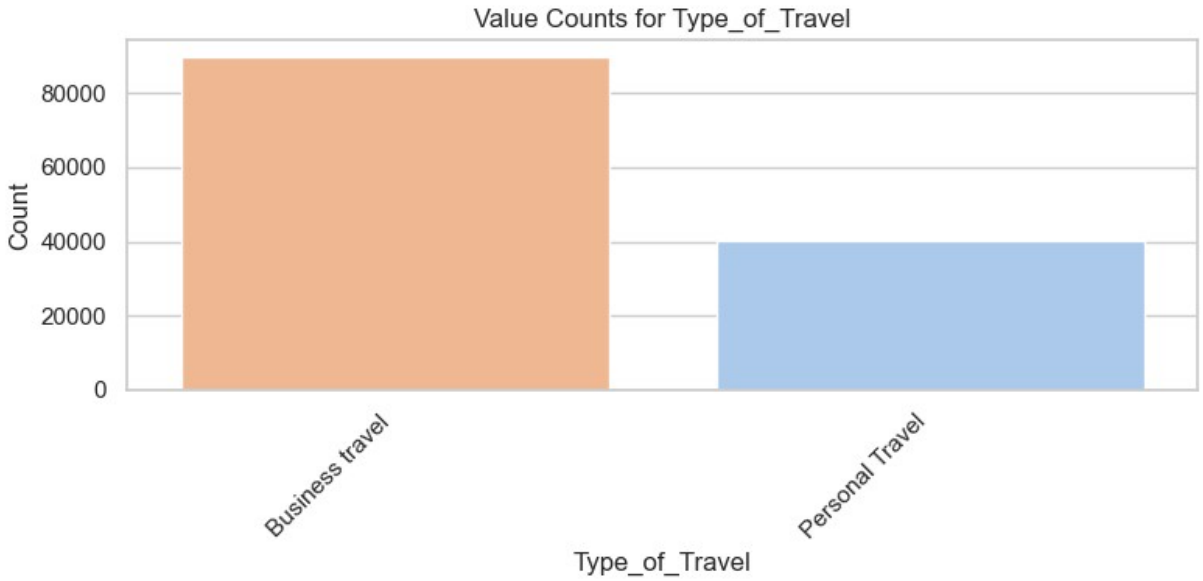


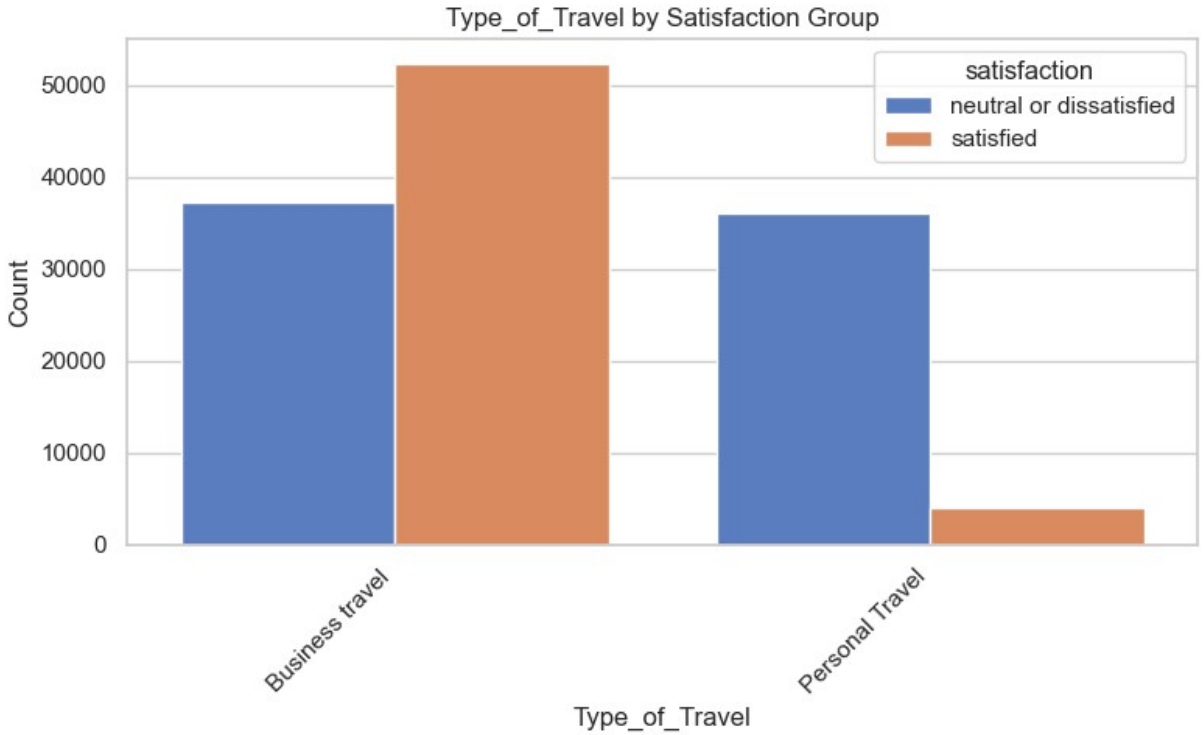

Value Counts for Customer_Type
Count
100000
80000
60000
40000
20000
0
Loyal Customer
disloyal Customer
Customer_Type


Customer_Type by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
50000
40000
30000
20000
10000
0
Loyal Customer
disloyal Customer
Customer_Type

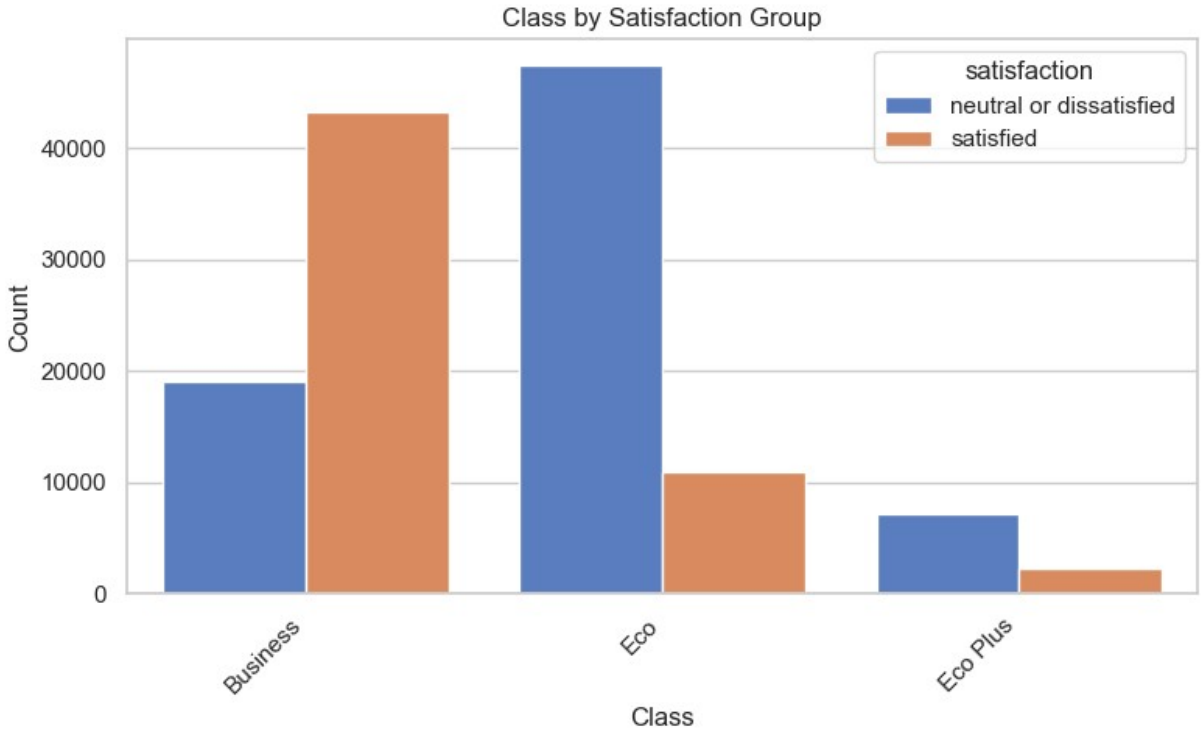

Overall Distribution of Age
Count
Age
0
2000
4000
6000
8000
10
20
30
40
50
60
70
80

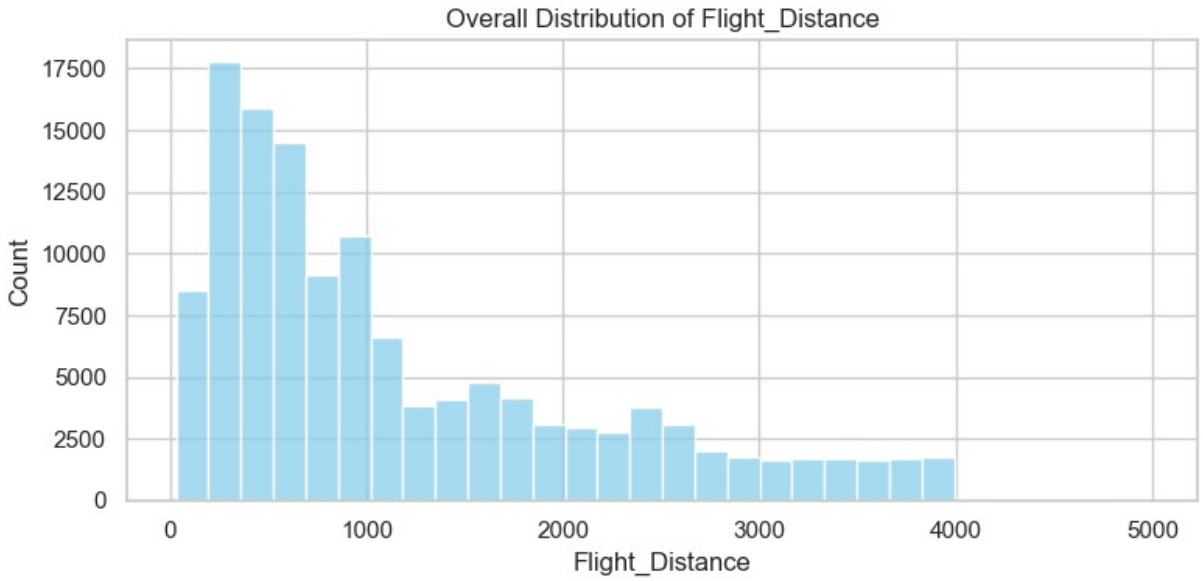

Age Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
Age
0
1000
2000
3000
4000
5000
6000
10
20
30
40
50
60
70
80

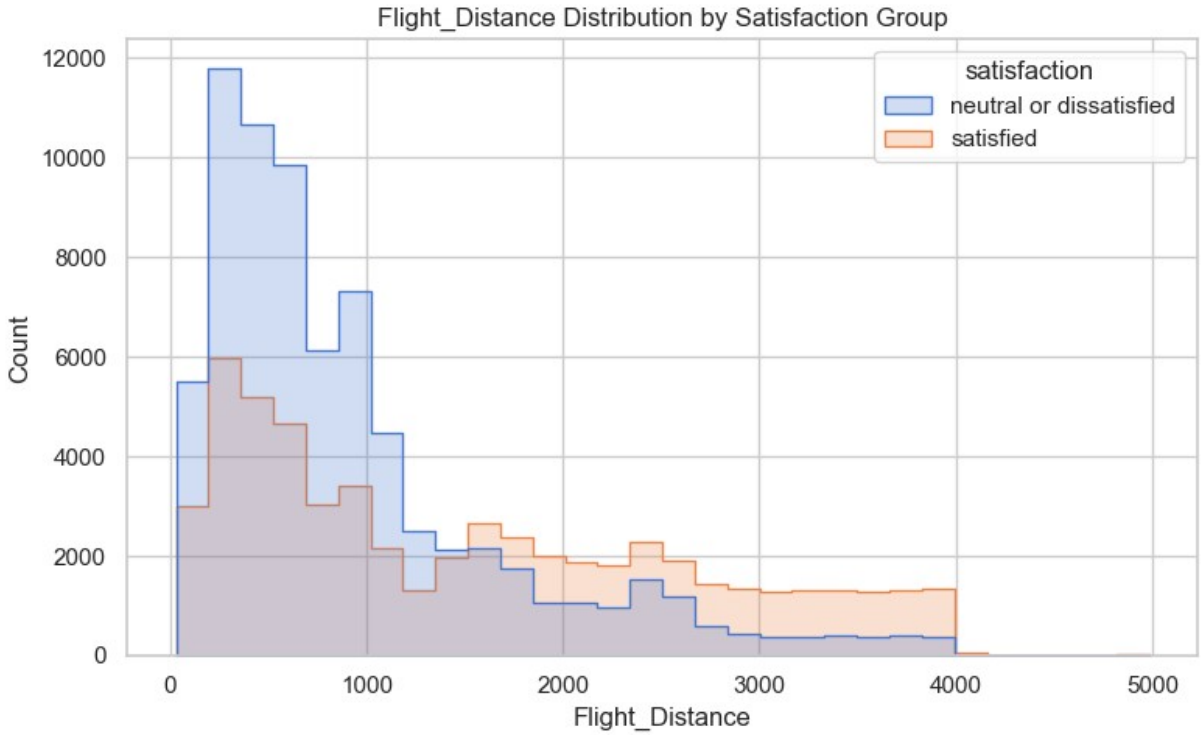

Value Counts for Type_of_Travel
Count
80000
60000
40000
20000
0
Business travel
Personal Travel
Type_of_Travel

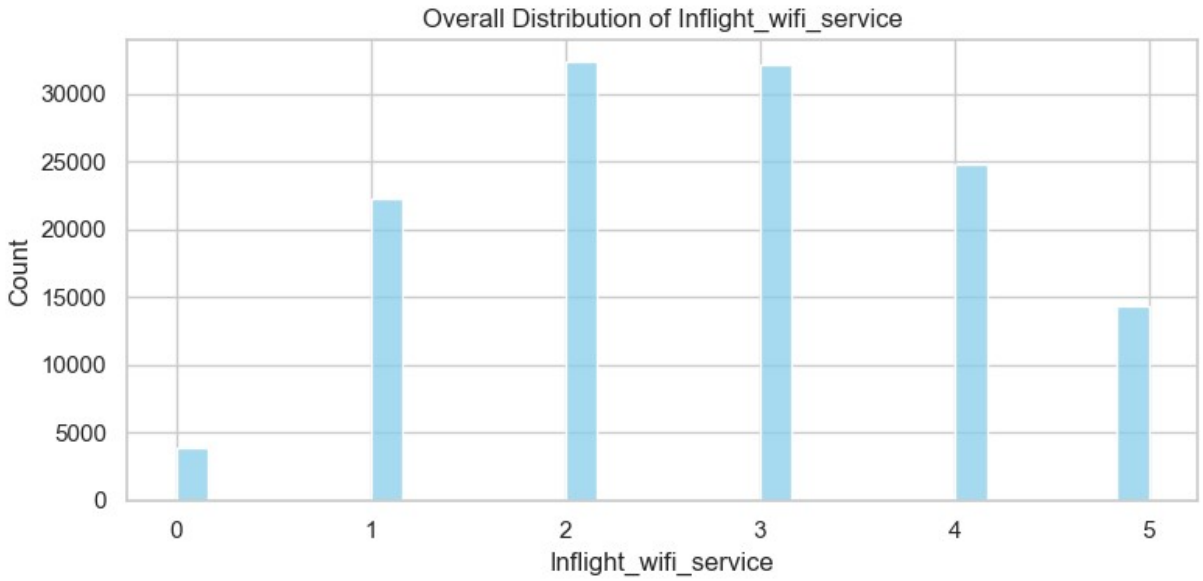

Type_of_Travel by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
50000
40000
30000
20000
10000
0
Business travel
Personal Travel
Type_of_Travel

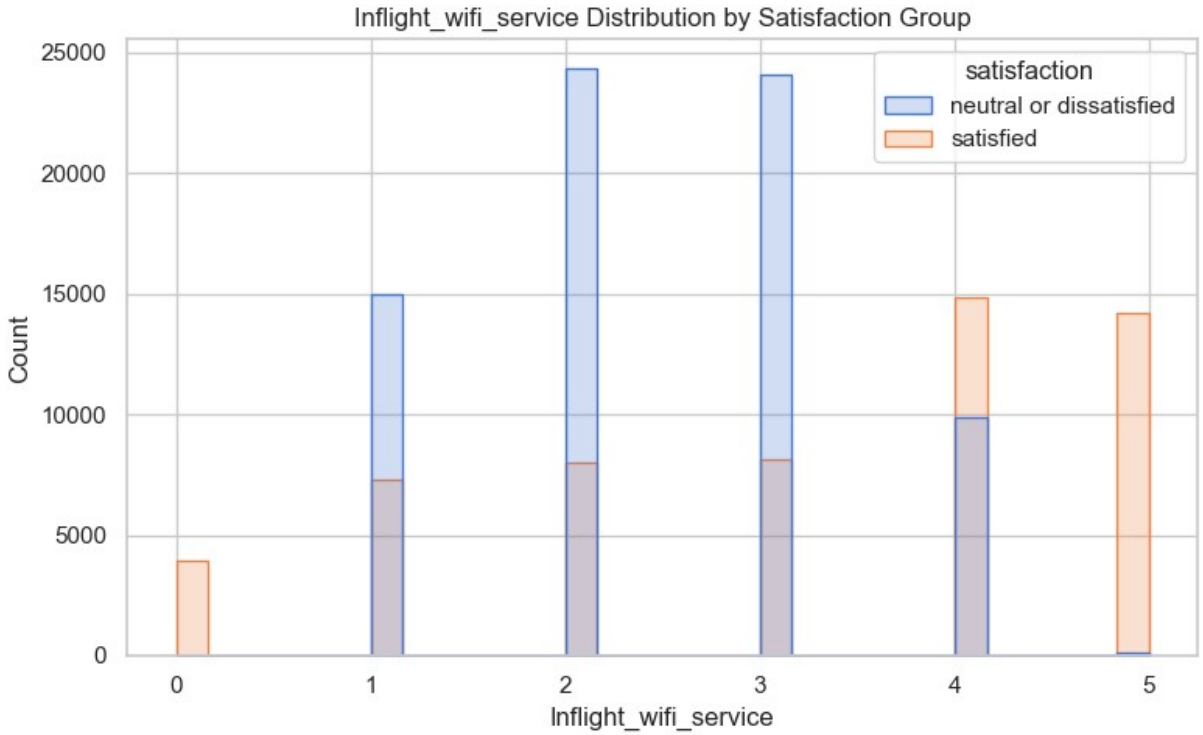

Value Counts for Class
60000
50000
40000
30000
20000
10000
0
Count
Business
Eco
Eco Plus
Class

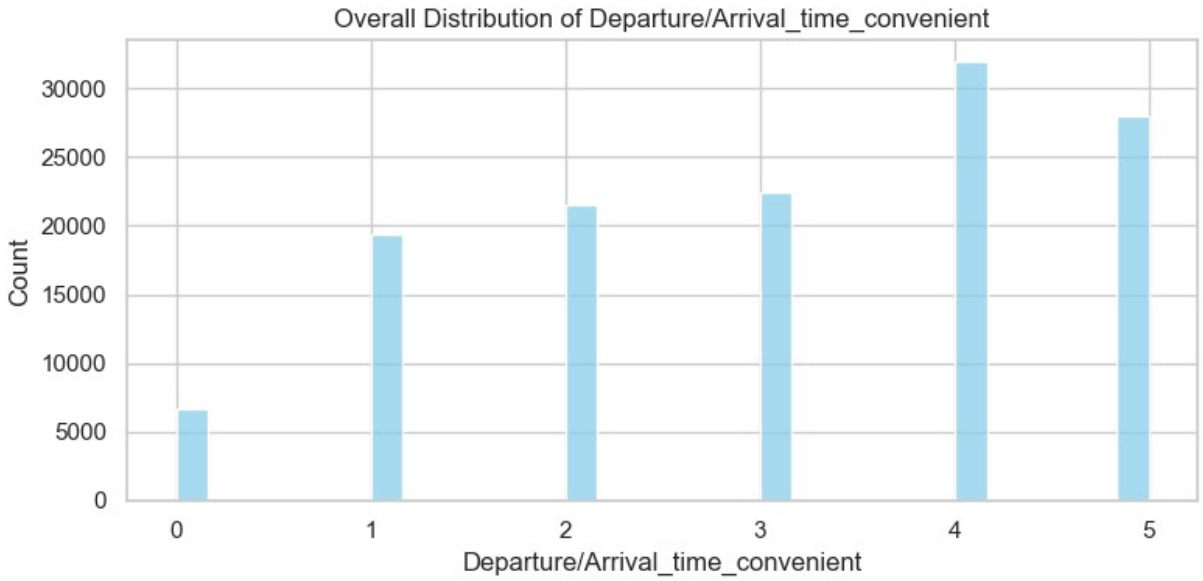

Class by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
40000
30000
20000
10000
0
Count
Business
Eco
Eco Plus
Class

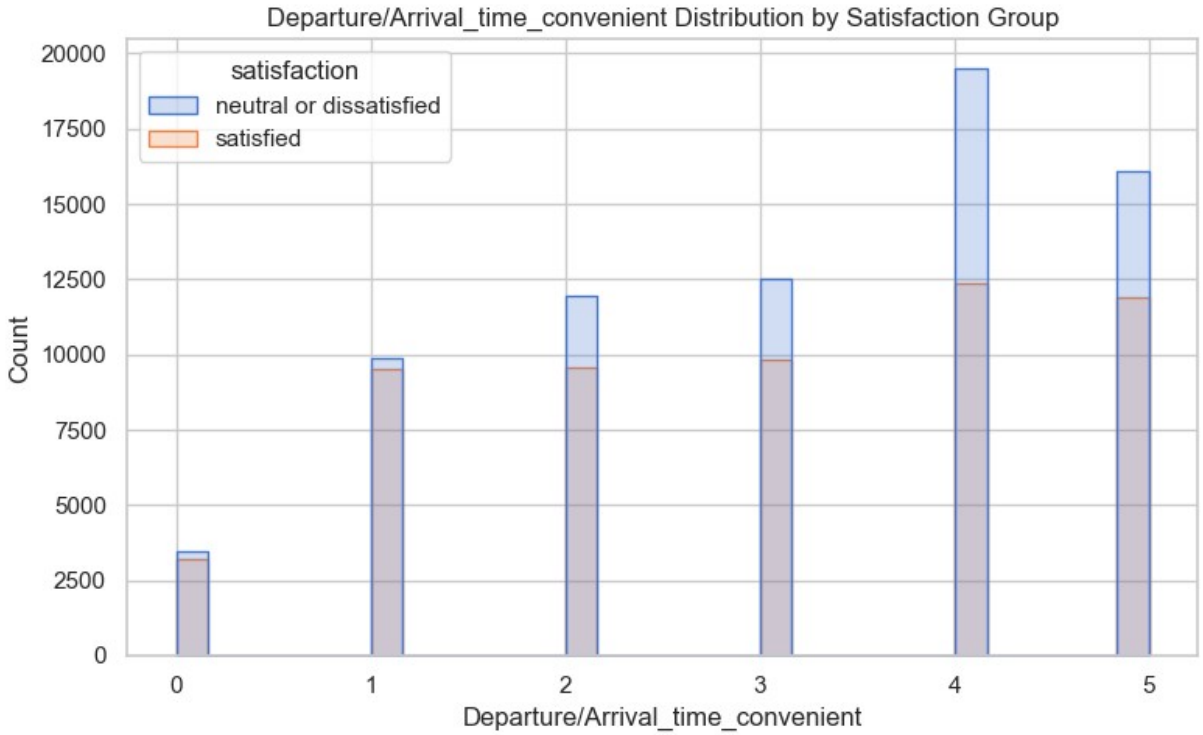

Overall Distribution of Flight_Distance
Count
Flight_Distance
0
2500
5000
7500
10000
12500
15000
17500
0
1000
2000
3000
4000
5000


Flight_Distance Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
Flight_Distance
0
2000
4000
6000
8000
10000
12000
0
1000
2000
3000
4000
5000

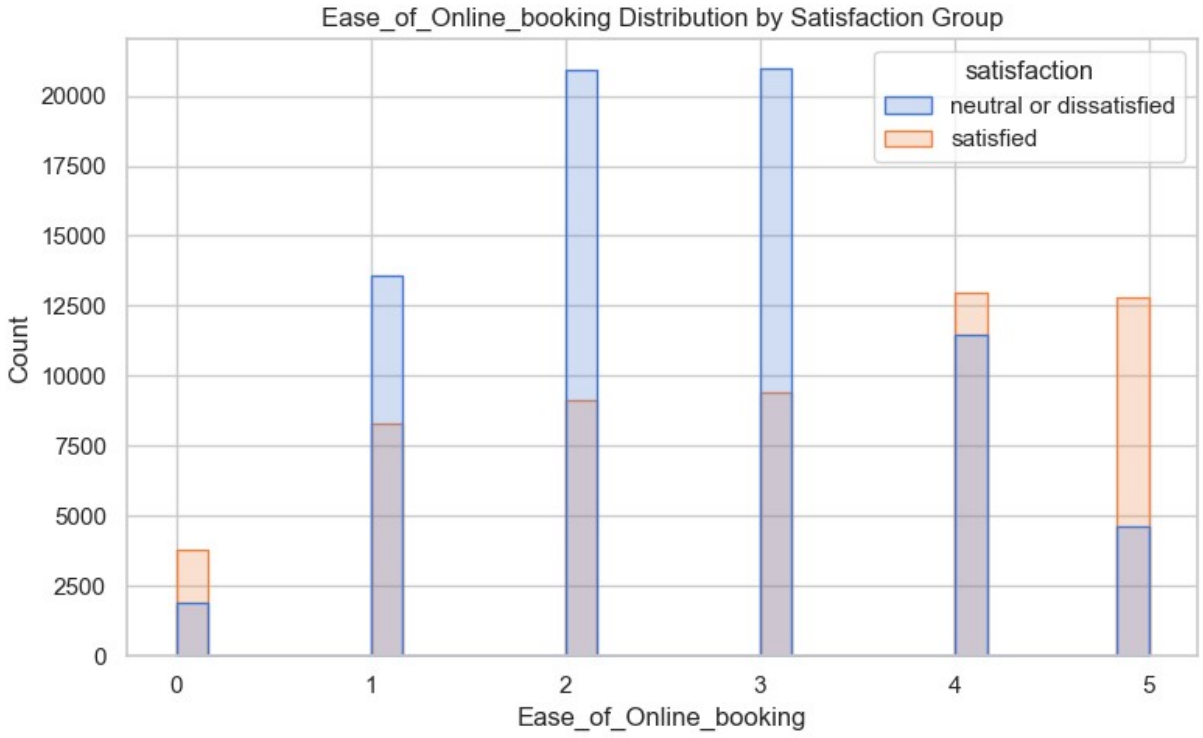

Overall Distribution of Inflight_wifi_service
Count
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Inflight_wifi_service

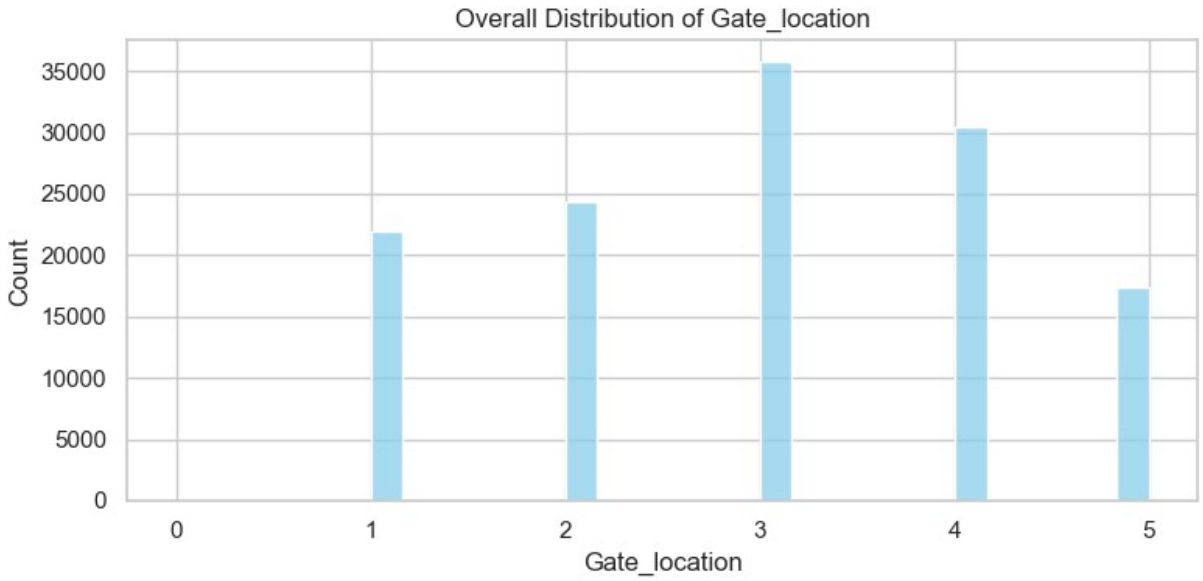

Inflight_wifi_service Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Inflight_wifi_service

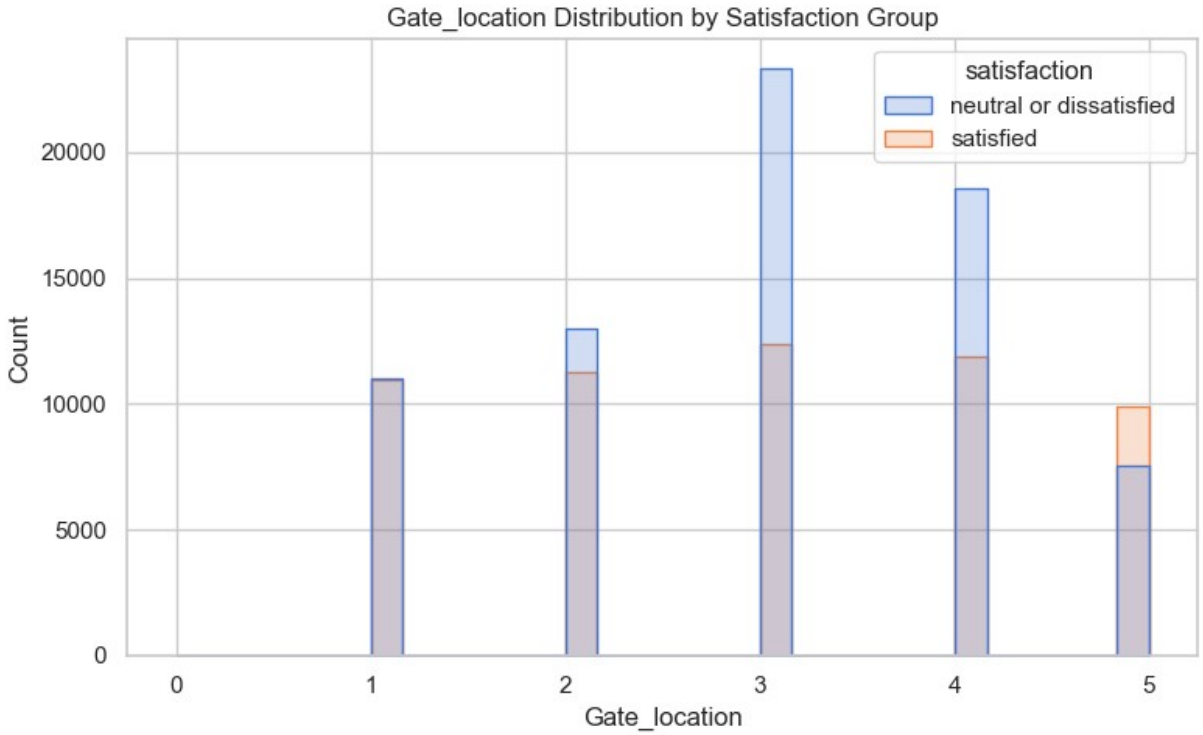

Overall Distribution of Departure/Arrival_time_convenient
Count
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Departure/Arrival_time_convenient

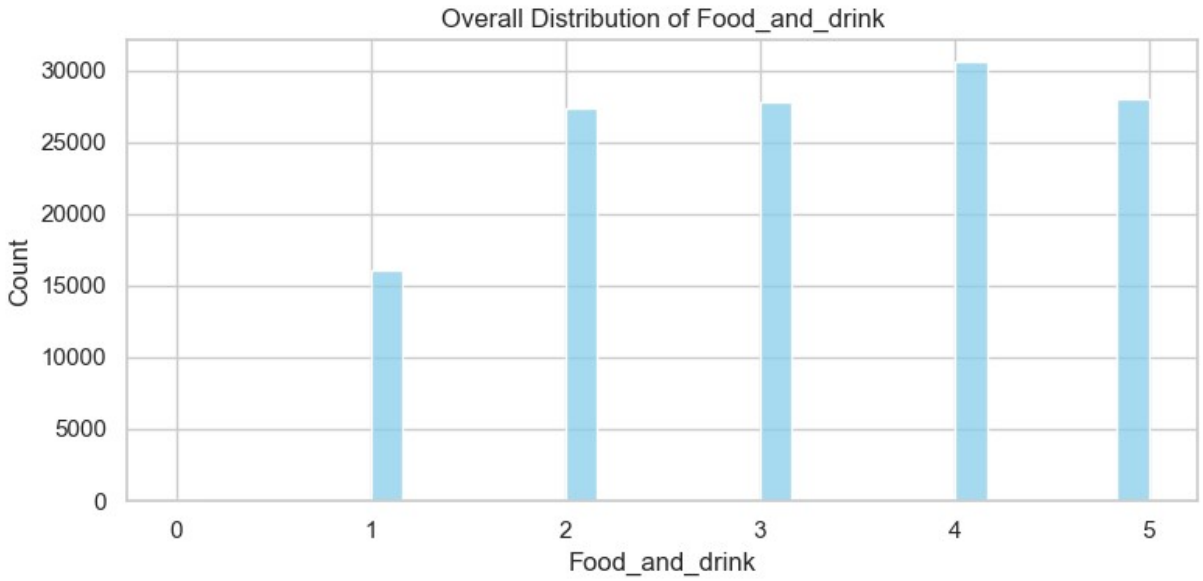

Departure/Arrival_time_convenient Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Departure/Arrival_time_convenient

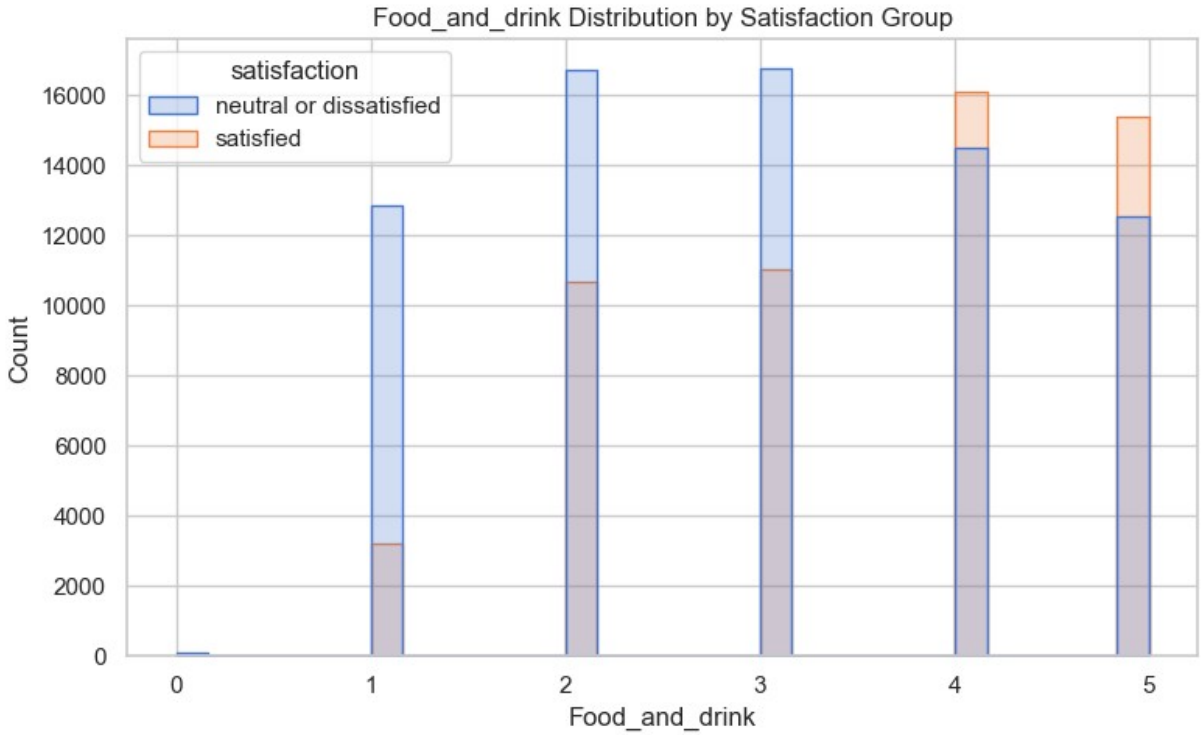

Overall Distribution of Ease_of_Online_booking
Count
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Ease_of_Online_booking

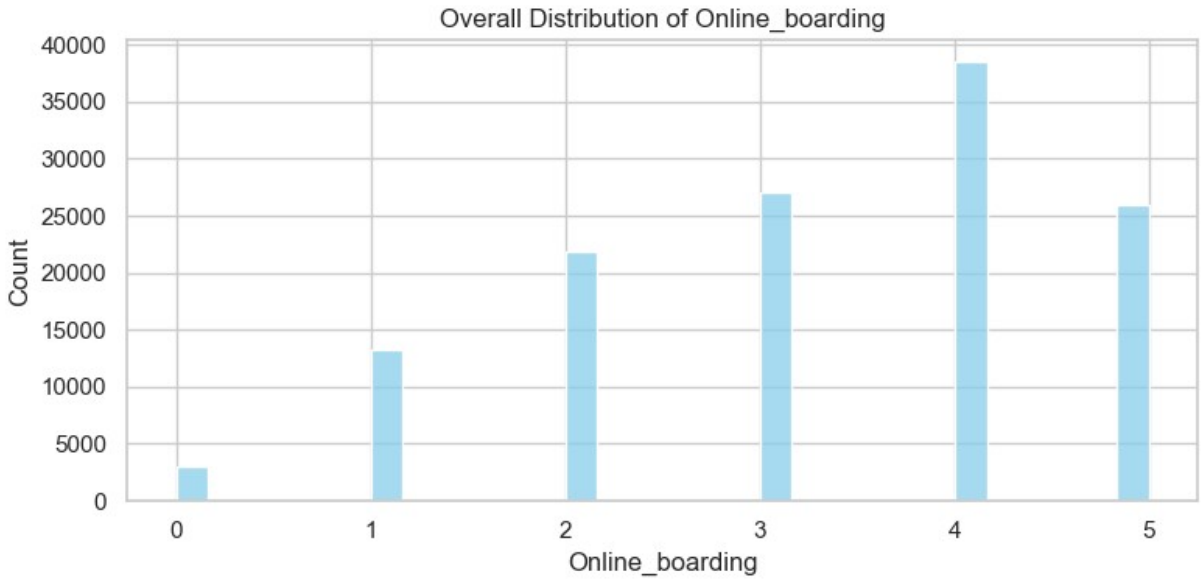

Ease_of_Online_booking Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Ease_of_Online_booking

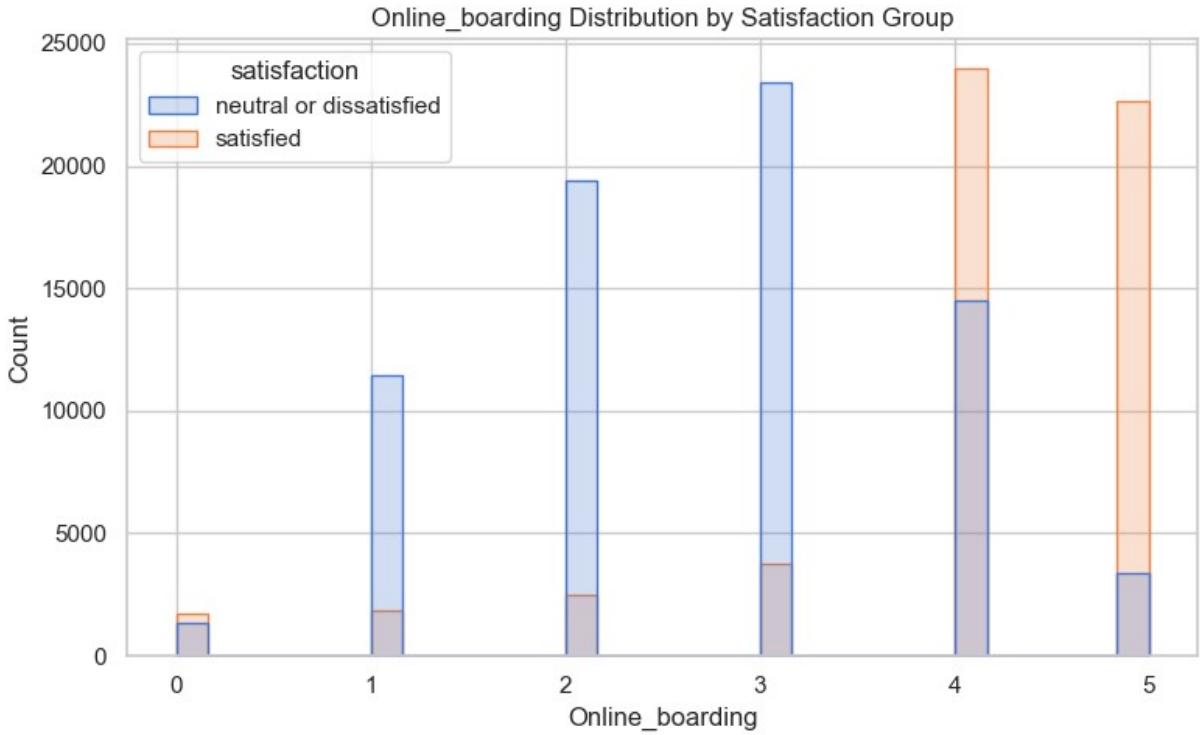

Overall Distribution of Gate_location
Count
35000
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Gate_location

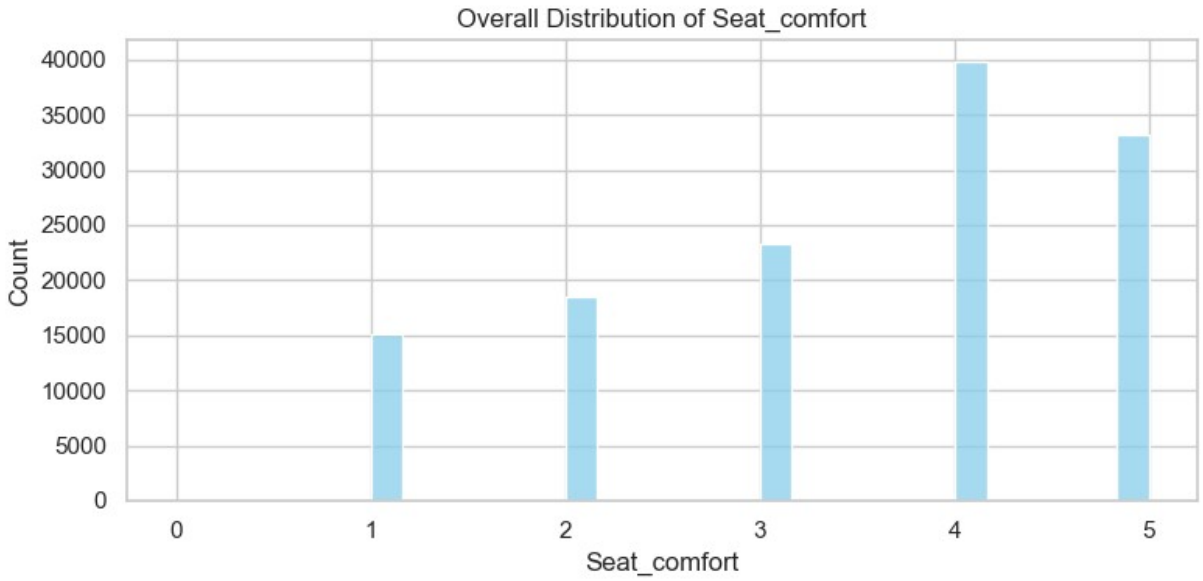

Gate_location Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
20000
15000
10000
5000
0
0
1
2
3
4
5
Gate_location

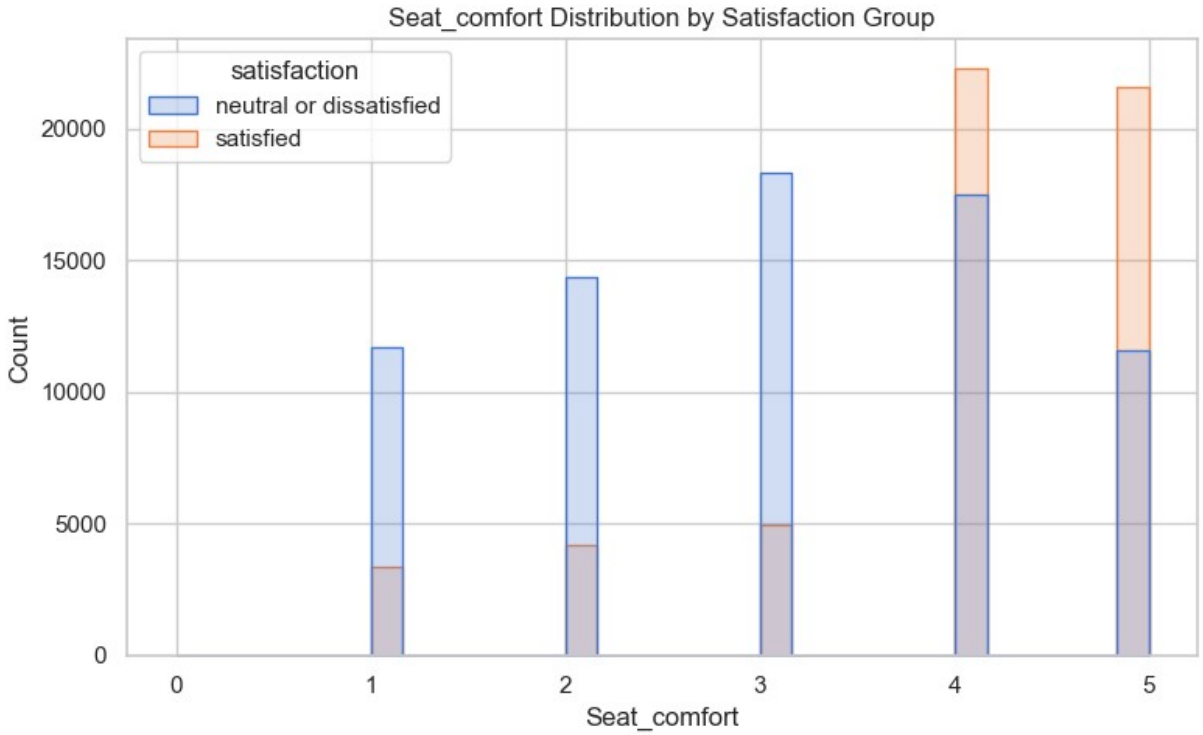

Overall Distribution of Food_and_drink
Count
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Food_and_drink

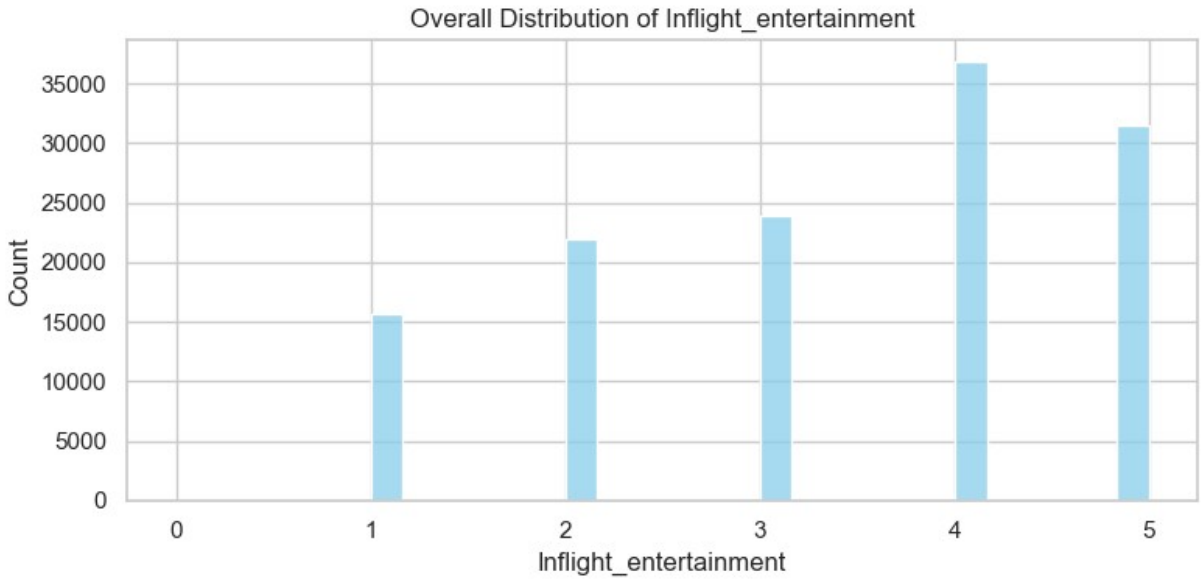

Food_and_drink Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
16000
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Food_and_drink

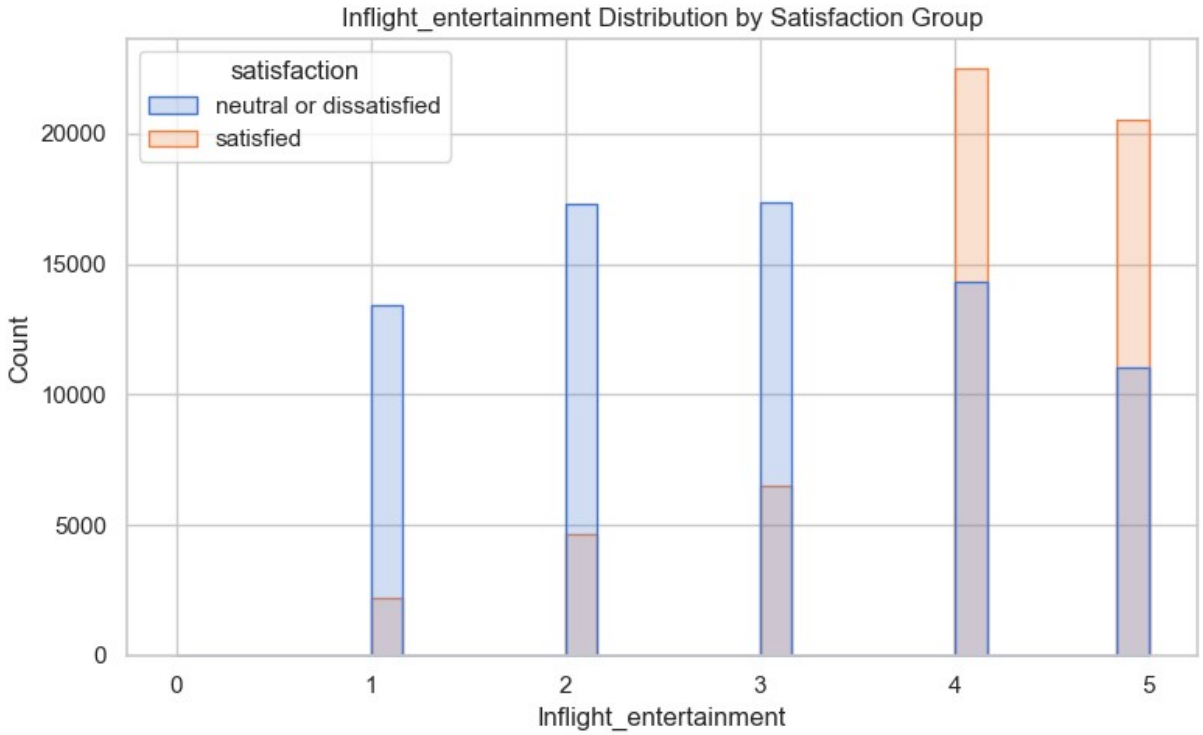

Overall Distribution of Online_boarding
Count
Online_boarding
0
5000
10000
15000
20000
25000
30000
35000
40000
0
1
2
3
4
5

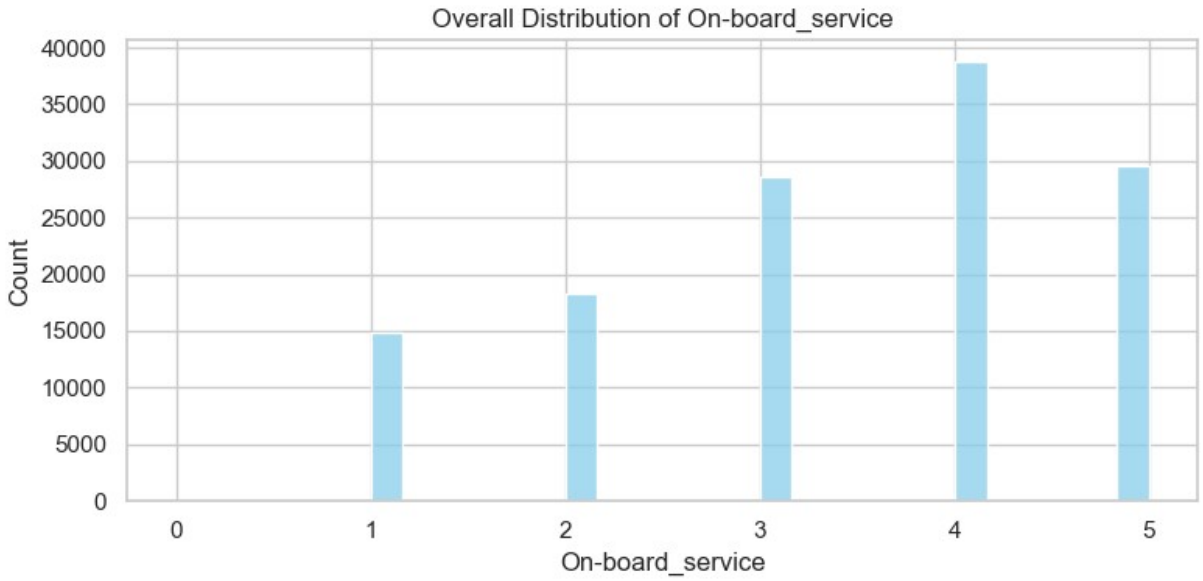

Online_boarding Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
Online_boarding
0
5000
10000
15000
20000
25000
0
1
2
3
4
5

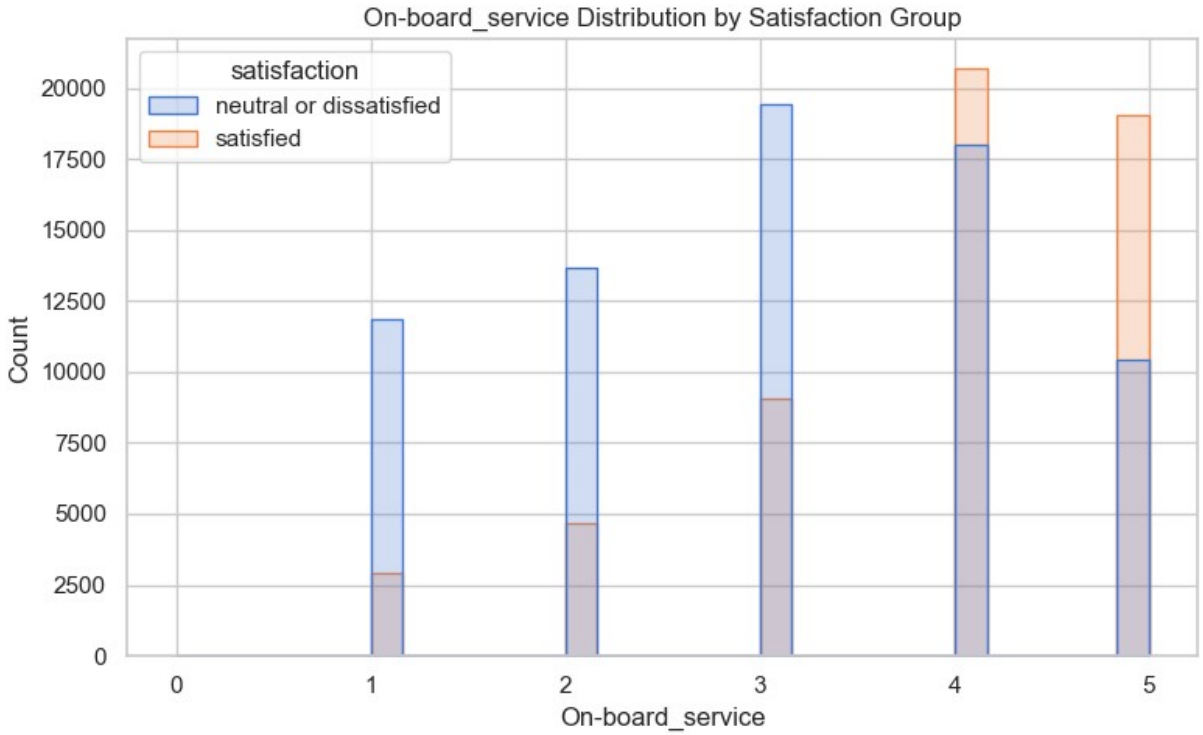

Overall Distribution of Seat_comfort
Count
Seat_comfort
0
5000
10000
15000
20000
25000
30000
35000
40000
0
1
2
3
4
5

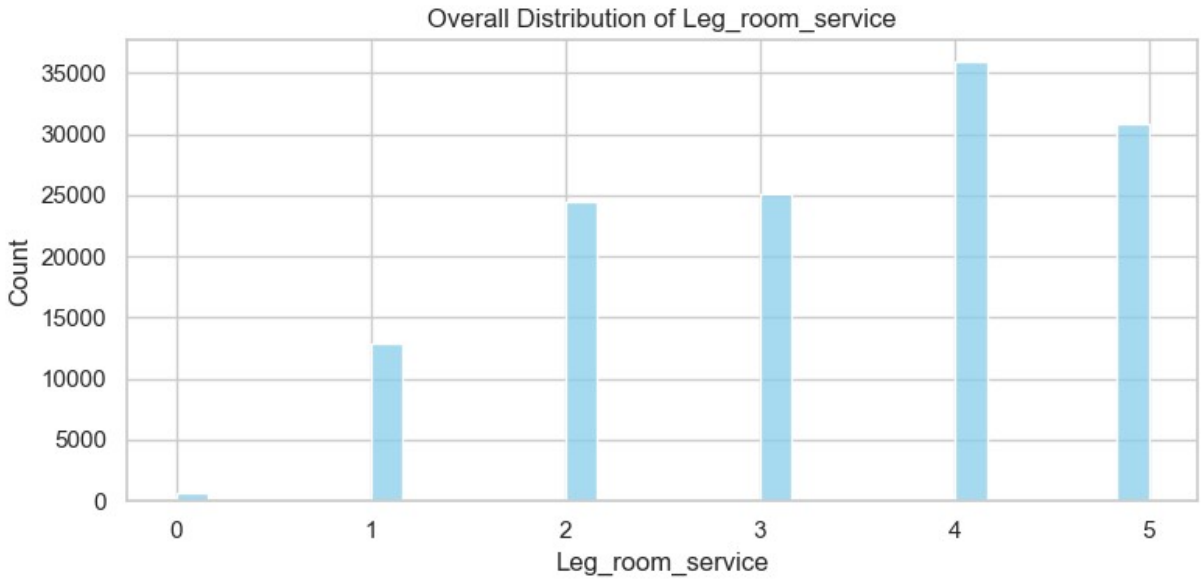

Seat_comfort Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
Seat_comfort
0
5000
10000
15000
20000
0
1
2
3
4
5

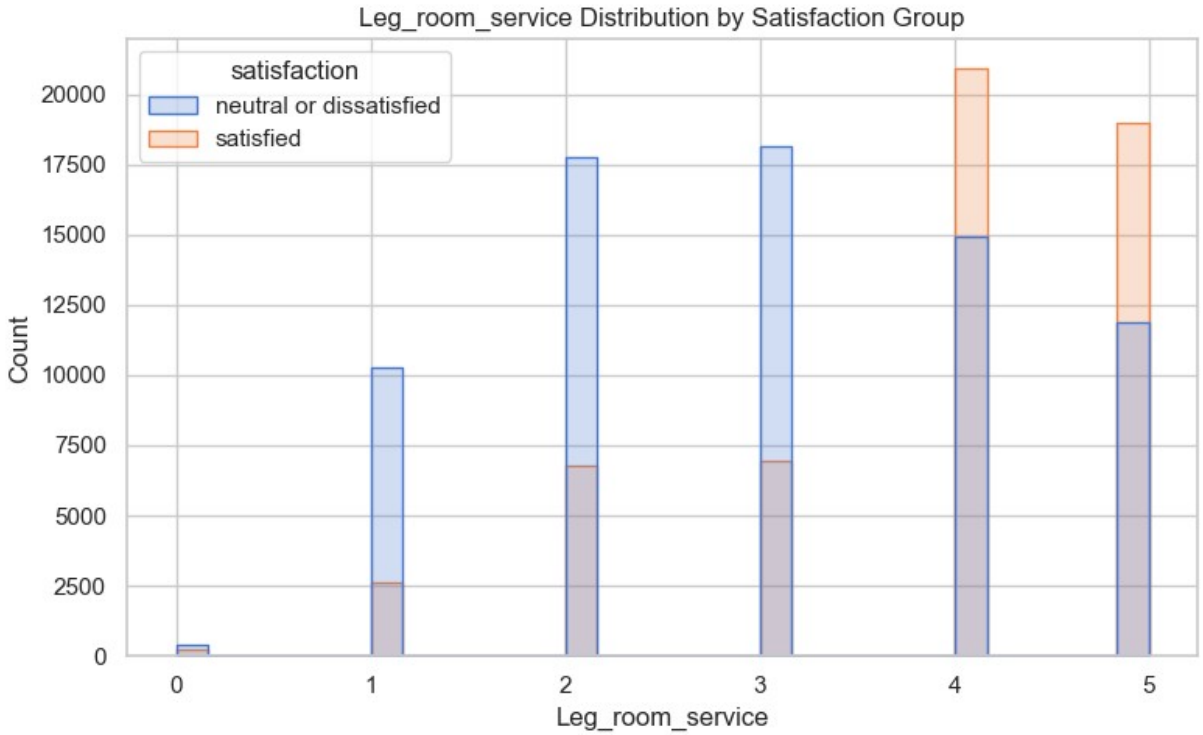

Overall Distribution of Inflight_entertainment
Count
35000
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Inflight_entertainment

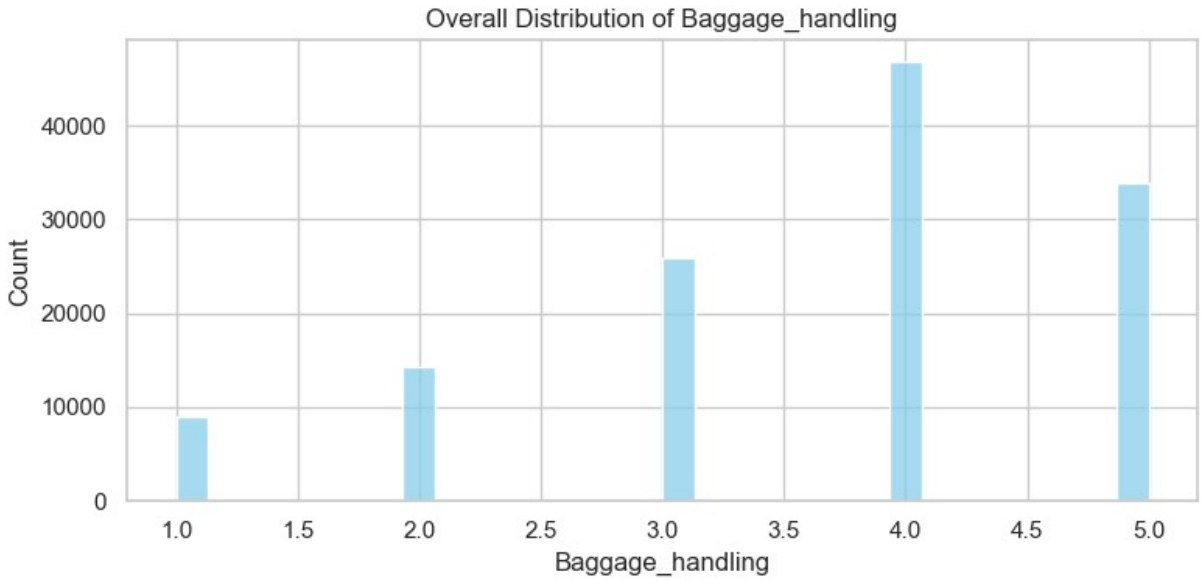

Inflight_entertainment Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
20000
15000
10000
5000
0
0
1
2
3
4
5
Inflight_entertainment

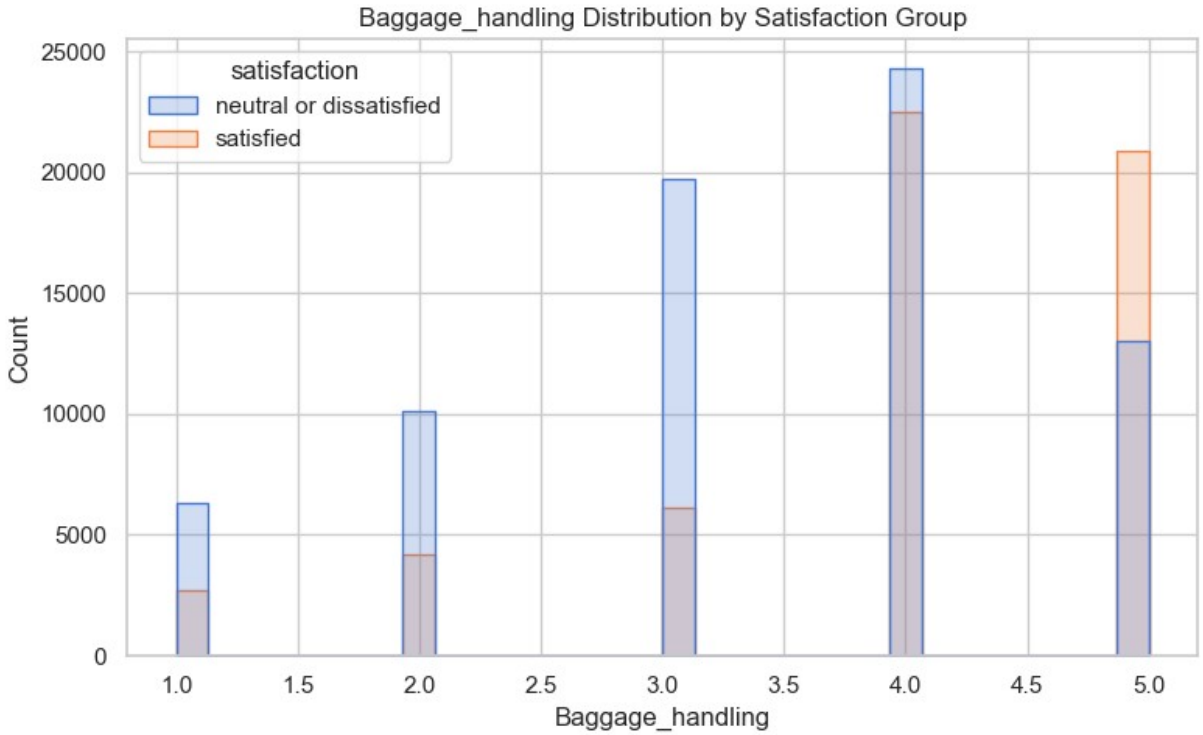

Overall Distribution of On-board_service
Count
On-board_service
0
5000
10000
15000
20000
25000
30000
35000
40000
0
1
2
3
4
5

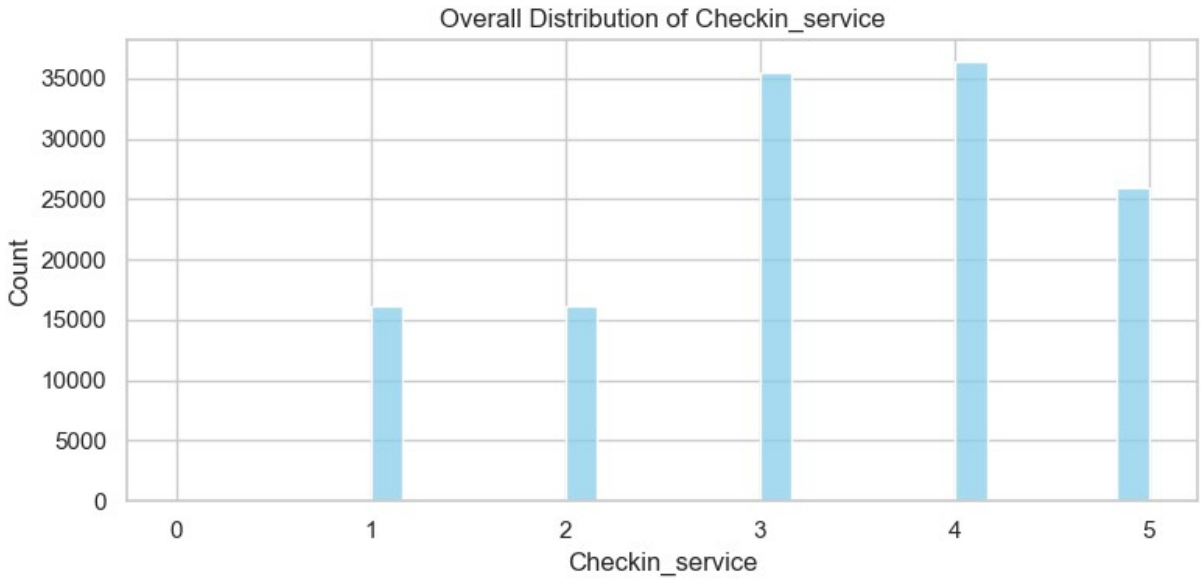

On-board_service Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
On-board_service
0
2500
5000
7500
10000
12500
15000
17500
20000
0
1
2
3
4
5

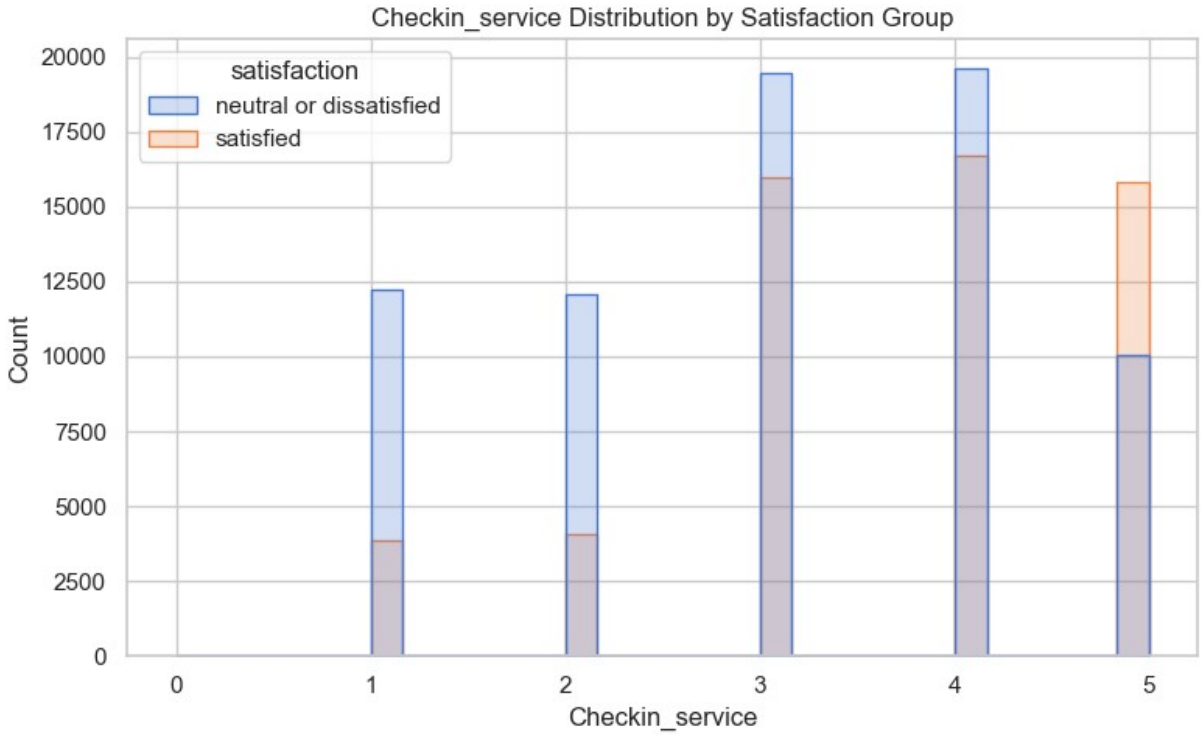

Overall Distribution of Leg_room_service
Count
35000
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Leg_room_service

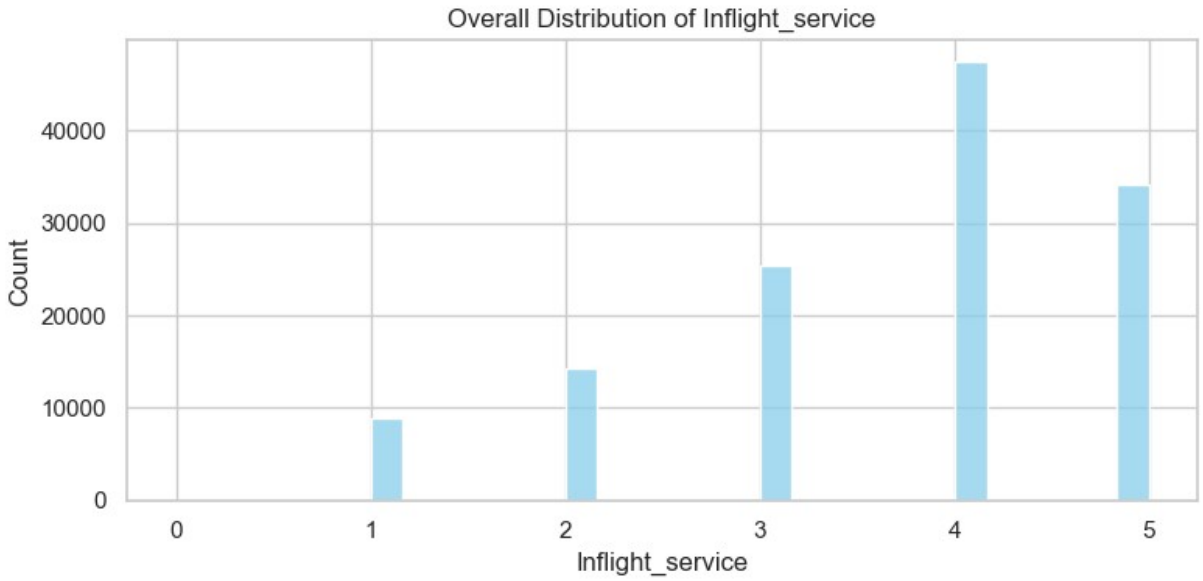

Leg_room_service Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Leg_room_service

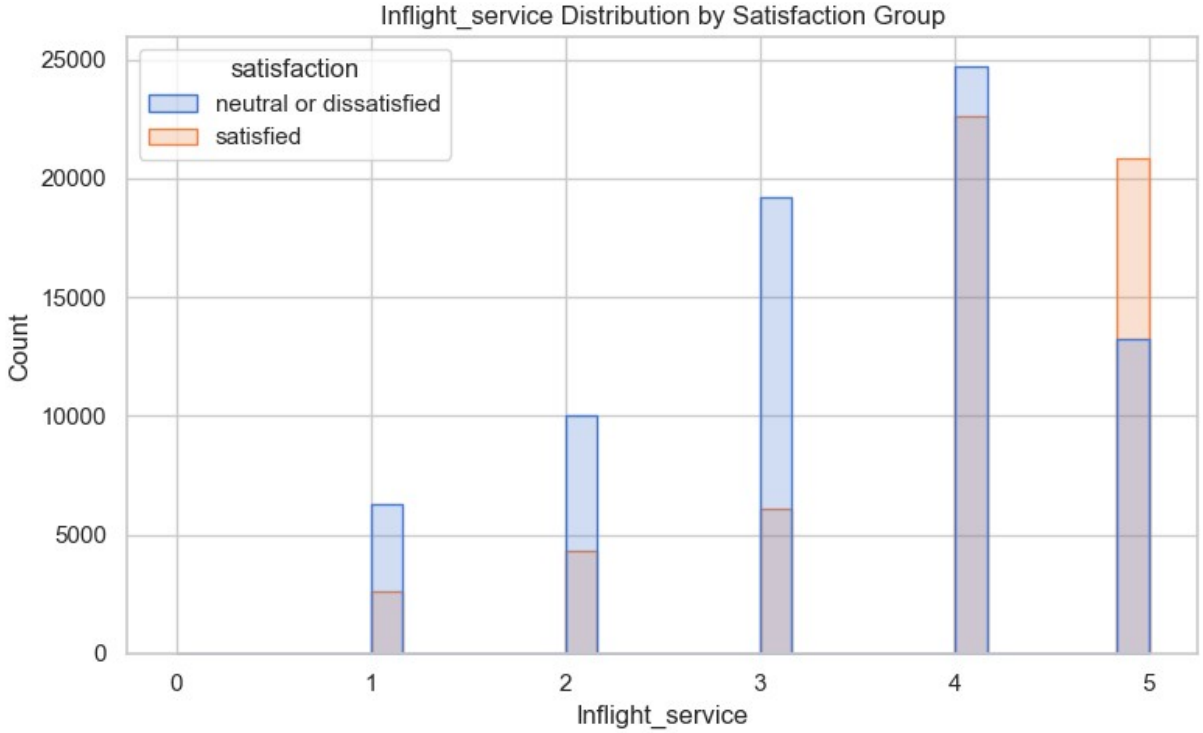

Overall Distribution of Baggage_handling
Count
40000
30000
20000
10000
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Baggage_handling

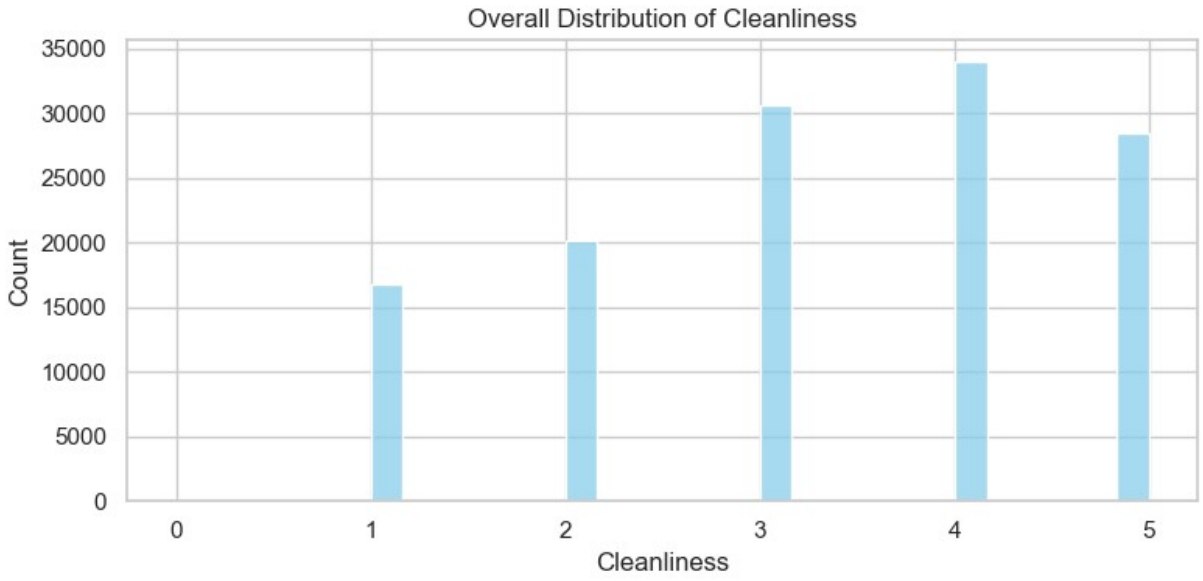

Baggage_handling Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
25000
20000
15000
10000
5000
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Baggage_handling

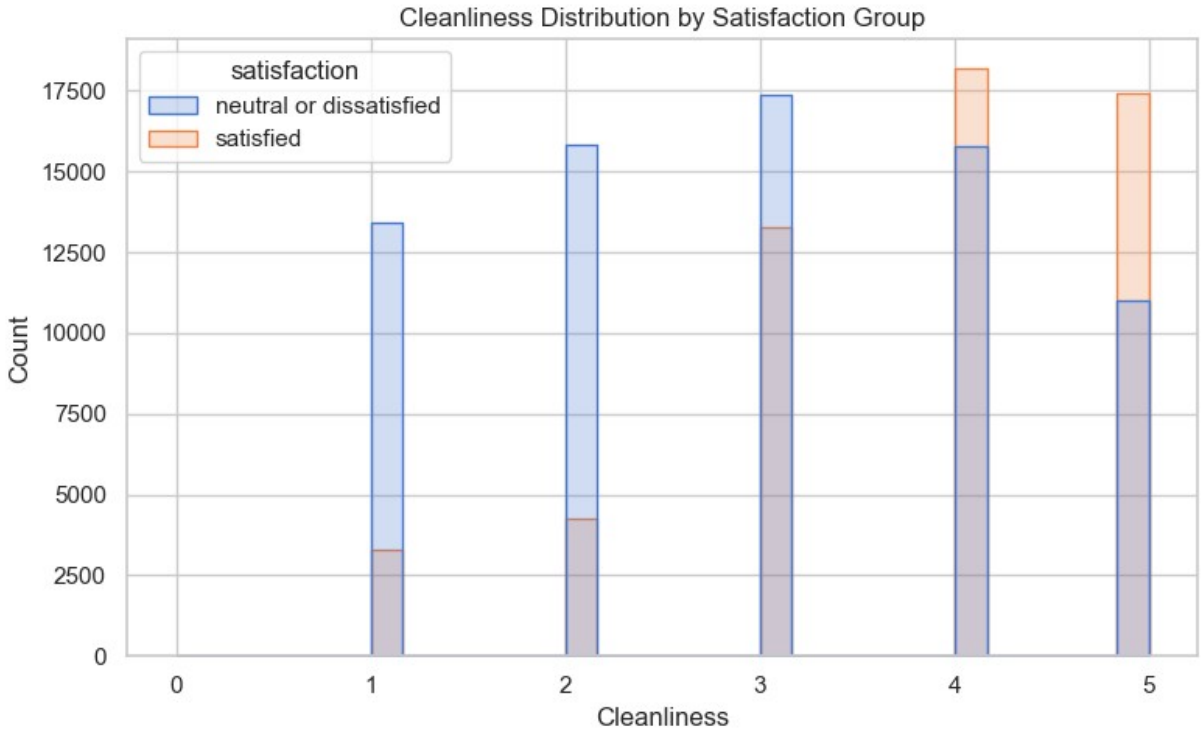

Overall Distribution of Checkin_service
Count
Checkin_service
0
5000
10000
15000
20000
25000
30000
35000
0
1
2
3
4
5

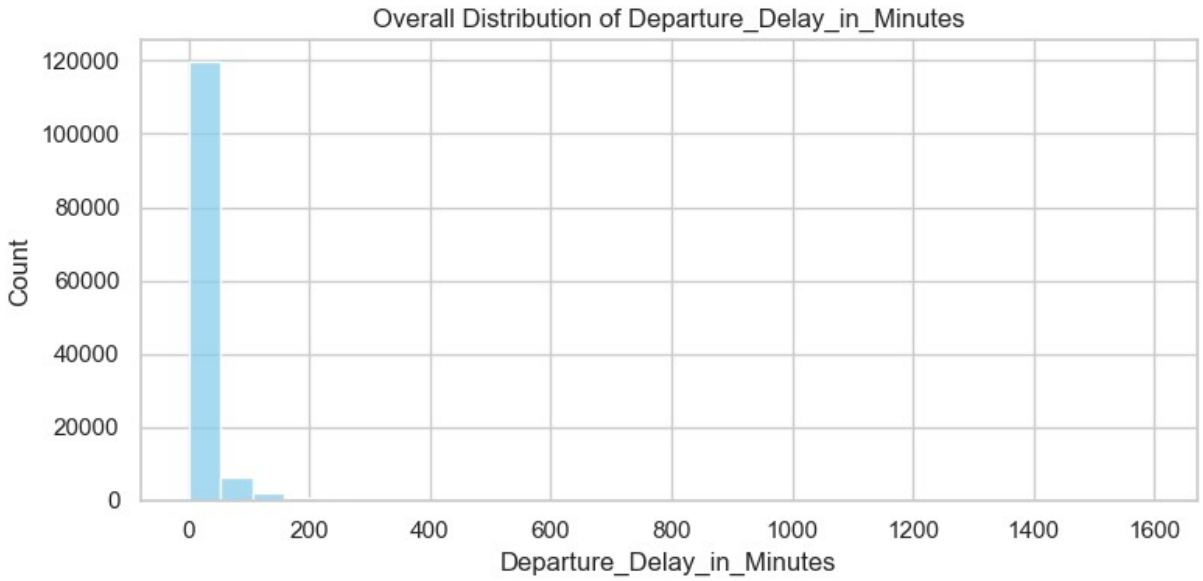

Checkin_service Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
Checkin_service
0
2500
5000
7500
10000
12500
15000
17500
20000
0
1
2
3
4
5

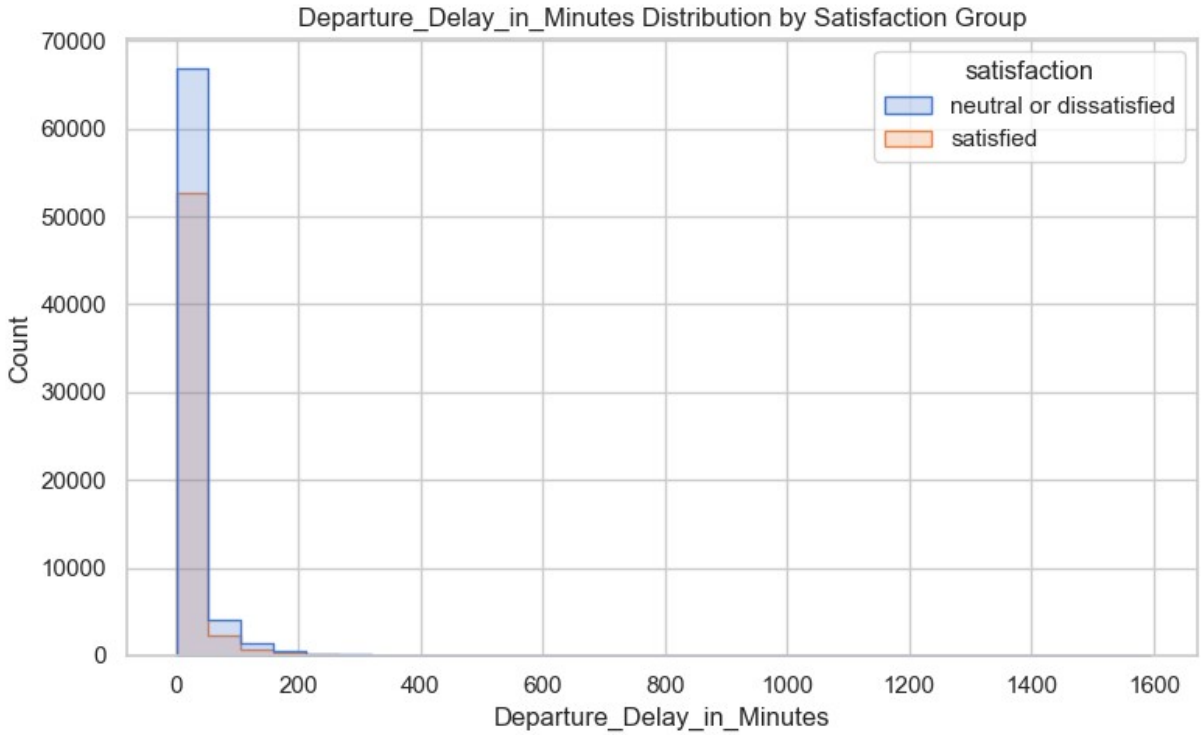

Overall Distribution of Inflight_service
Count
0
10000
20000
30000
40000
0
1
2
3
4
5
Inflight_service

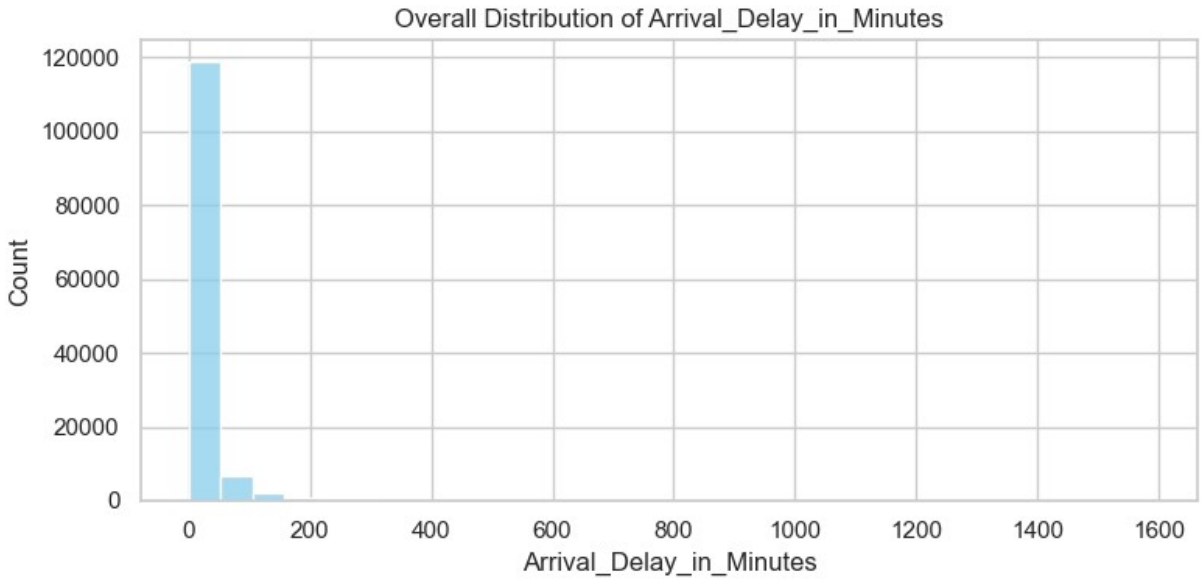

Inflight_service Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
0
5000
10000
15000
20000
25000
0
1
2
3
4
5
Inflight_service

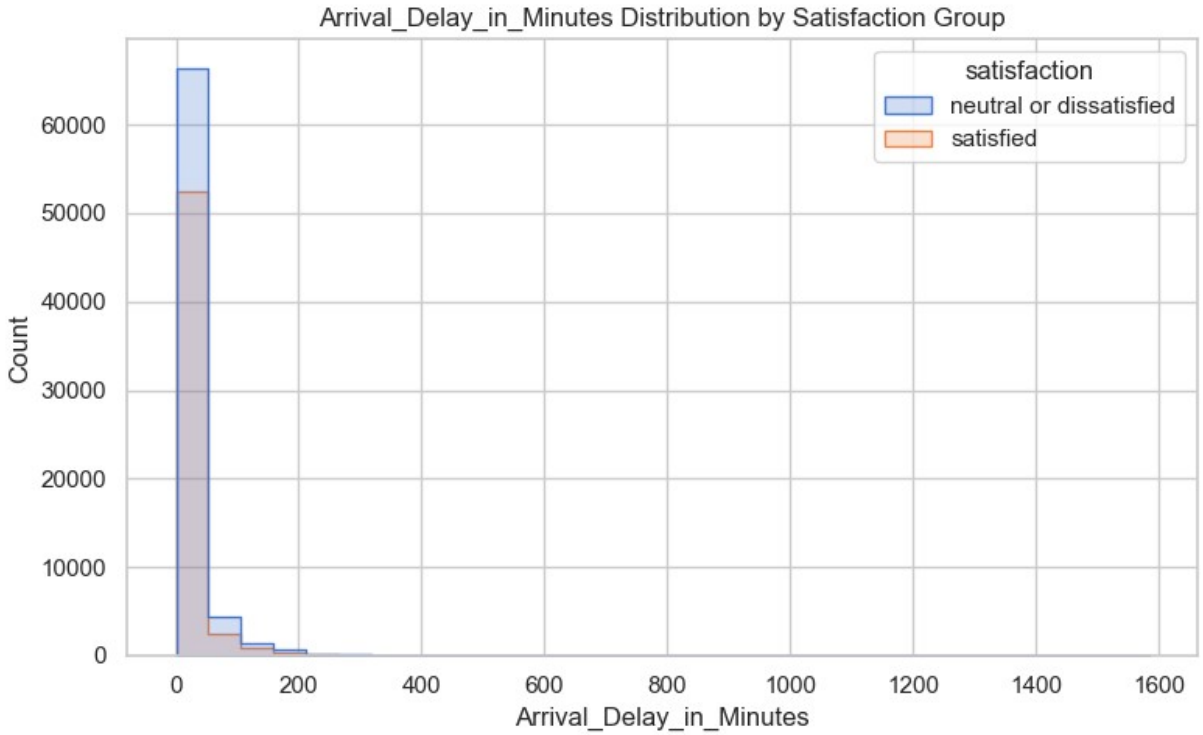

Overall Distribution of Cleanliness
Count
Cleanliness
0
5000
10000
15000
20000
25000
30000
35000
0
1
2
3
4
5

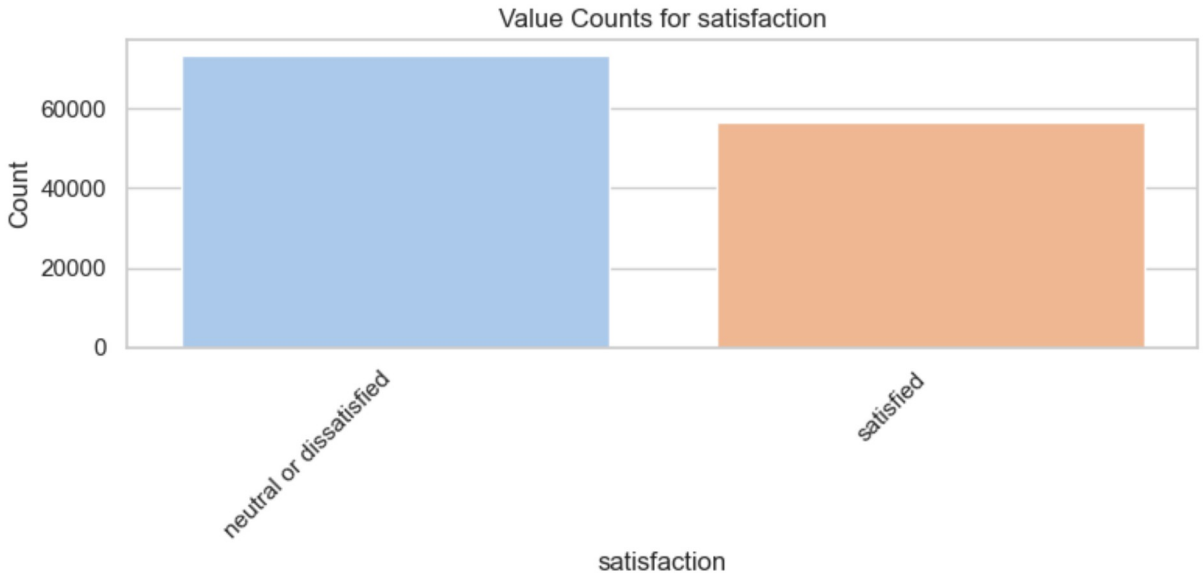

Cleanliness Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
Cleanliness
0
2500
5000
7500
10000
12500
15000
17500
0
1
2
3
4
5

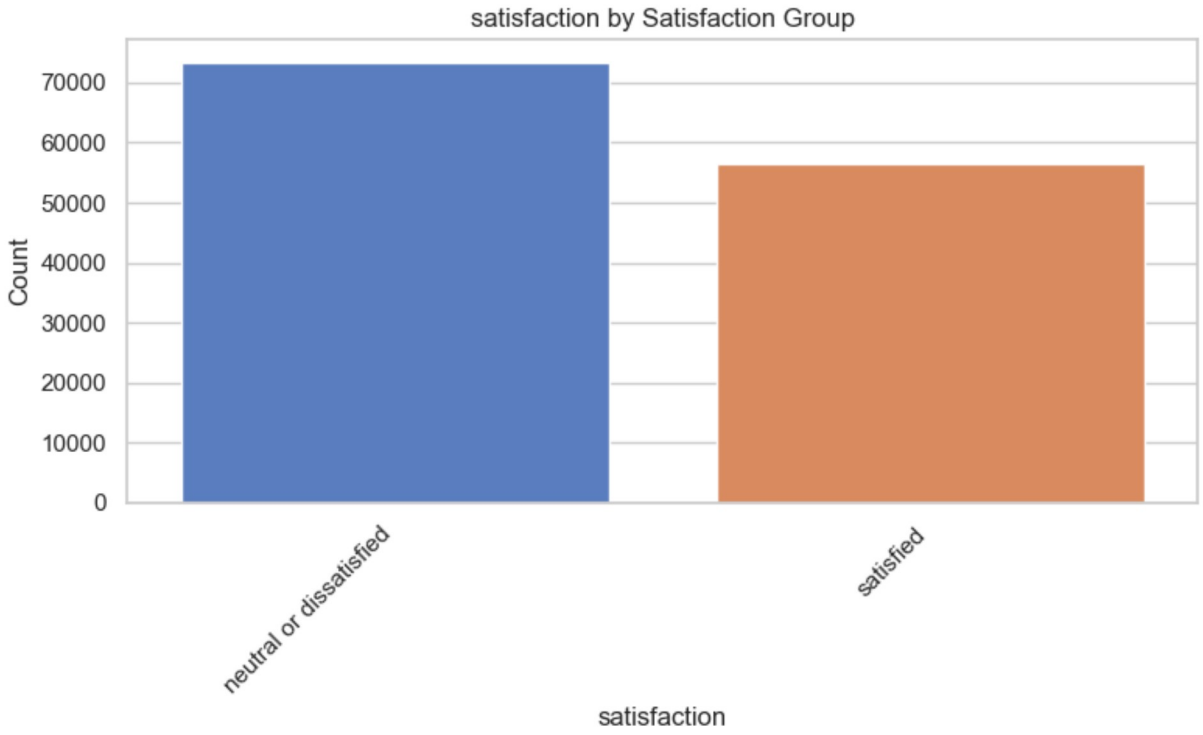

Overall Distribution of Departure_Delay_in_Minutes
Count
Departure_Delay_in_Minutes
0
20000
40000
60000
80000
100000
120000
0
200
400
600
800
1000
1200
1400
1600


Departure_Delay_in_Minutes Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
Departure_Delay_in_Minutes
0
10000
20000
30000
40000
50000
60000
70000
0
200
400
600
800
1000
1200
1400
1600


Overall Distribution of Arrival_Delay_in_Minutes
Count
120000
100000
80000
60000
40000
20000
0
0
200
400
600
800
1000
1200
1400
1600
Arrival_Delay_in_Minutes


Arrival_Delay_in_Minutes Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
60000
50000
40000
30000
20000
10000
0
0
200
400
600
800
1000
1200
1400
1600
Arrival_Delay_in_Minutes





In [209…

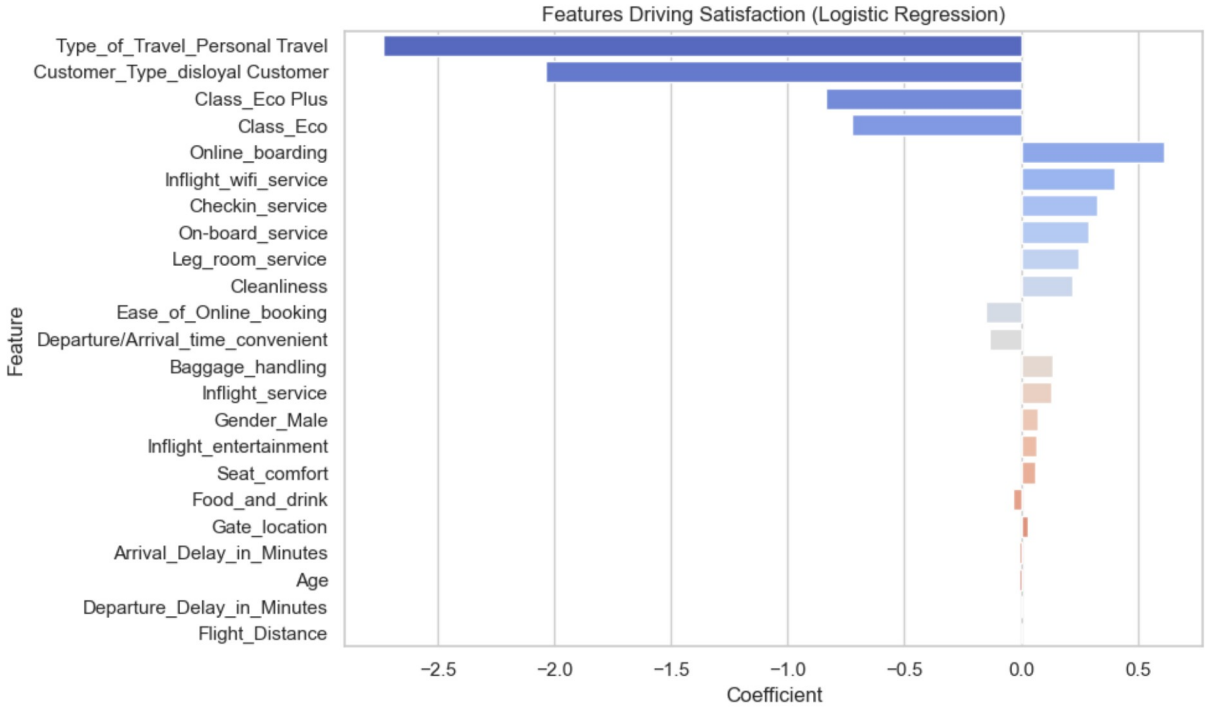
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np


df = pd.read_csv("AirlineSatisfaction_Transformed.csv")

#Drop rows with missing values (will only drop due to Arrival Delay in Minutes, for
```

```
df = df.dropna()

#Encode target variable for satisfaction; 1 for satisfied and 0 if not
df["satisfaction_binary"] = df["satisfaction"].apply(lambda x: 1 if x.strip().lower

#Build logistic regression model
X = df.drop(columns=["satisfaction", "satisfaction_binary"])
y = df["satisfaction_binary"]
categorical_cols = X.select_dtypes(include="object").columns.tolist()
numeric_cols = X.select_dtypes(include=["int64", "float64"]).columns.tolist()
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'), categorical_c
        ('num', 'passthrough', numeric_cols)
    ]
)

logreg_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=10000))
])

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2

logreg_pipeline.fit(X_train, y_train)

#Also adding a section to output the actual statistics about the fitted model so we
#But looks like we can't get it from sklearn so instead have to use statsmodels for
import statsmodels.api as sm
X_train_encoded = preprocessor.fit_transform(X_train)



#Looks like I had an issue with order of operations from the kernal persisting over
#need to establish all_feature_names first here before we can use it to build the d
ohe = logreg_pipeline.named_steps["preprocessor"].named_transformers_["cat"]
ohe_feature_names = ohe.get_feature_names_out(categorical_cols)
all_feature_names = np.concatenate([ohe_feature_names, numeric_cols])


#Build DataFrame for statsmodels with column names
X_train_df = pd.DataFrame(
    X_train_encoded.toarray() if hasattr(X_train_encoded, "toarray") else X_train_e
    columns=all_feature_names
)


#Add intercept manually
X_train_df = sm.add_constant(X_train_df)
#Fit statsmodels logistic regression
sm_model = sm.Logit(y_train.values, X_train_df)
sm_result = sm_model.fit()

#Print detailed model statistical summary (coefficients and p values visible etc...
print(sm_result.summary())
```

```
#Get variable names again post encoding
ohe = logreg_pipeline.named_steps["preprocessor"].named_transformers_["cat"]
ohe_feature_names = ohe.get_feature_names_out(categorical_cols)
all_feature_names = np.concatenate([ohe_feature_names, numeric_cols])

#grab coefficients
coefficients = logreg_pipeline.named_steps["classifier"].coef_[0]
coef_df = pd.DataFrame({
    "Feature": all_feature_names,
    "Coefficient": coefficients
}).sort_values(by="Coefficient", key=np.abs, ascending=False)

#Plot the top 100 variables/features (so should just show all of them unless we exc
plt.figure(figsize=(10, 6))
sns.barplot(data=coef_df.head(100), x="Coefficient", y="Feature", palette="coolwarm
plt.title("Features Driving Satisfaction (Logistic Regression)")
plt.tight_layout()
plt.show()


#Now the only message left is a future warning so nbd
```

```
Optimization terminated successfully.
         Current function value: 0.334868
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:               103589
Model:                          Logit   Df Residuals:                   103565
Method:                           MLE   Df Model:                           23
Date:                Sun, 01 Jun 2025   Pseudo R-squ.:                  0.5108
Time:                        00:30:48   Log-Likelihood:                -34689.
converged:                       True   LL-Null:                       -70911.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
=================
                                       coef    std err          z      P>|z|
[0.025      0.975]
------------------------------------------------------------------------------
-----------------
const                                -5.7439      0.075    -76.552      0.000
-5.891      -5.597
Gender_Male                           0.0712      0.019      3.660      0.000
0.033       0.109
Customer_Type_disloyal Customer      -2.0405      0.030    -68.549      0.000
-2.099      -1.982
Type_of_Travel_Personal Travel       -2.7382      0.031    -87.034      0.000
-2.800      -2.676
Class_Eco                            -0.7223      0.026    -28.180      0.000
-0.773      -0.672
Class_Eco Plus                       -0.8364      0.041    -20.174      0.000
-0.918      -0.755
Age                                  -0.0087      0.001    -12.277      0.000
-0.010      -0.007
Flight_Distance                   -2.652e-05   1.13e-05     -2.343      0.019   -4.8
7e-05   -4.34e-06
Inflight_wifi_service                 0.4000      0.011     34.907      0.000
0.378       0.422
Departure/Arrival_time_convenient    -0.1349      0.008    -16.546      0.000
-0.151      -0.119
Ease_of_Online_booking               -0.1503      0.011    -13.299      0.000
-0.172      -0.128
Gate_location                         0.0277      0.009      3.030      0.002
0.010       0.046
Food_and_drink                       -0.0317      0.011     -2.954      0.003
-0.053      -0.011
Online_boarding                       0.6126      0.010     59.816      0.000
0.593       0.633
Seat_comfort                          0.0623      0.011      5.573      0.000
0.040       0.084
Inflight_entertainment                0.0680      0.014      4.758      0.000
0.040       0.096
On-board_service                      0.2916      0.010     28.665      0.000
0.272       0.312
Leg_room_service                      0.2473      0.009     29.079      0.000
0.231       0.264
Baggage_handling                      0.1354      0.011     11.840      0.000
0.113       0.158
```

```
Checkin_service                      0.3304      0.009     38.645      0.000
0.314       0.347
Inflight_service                     0.1290      0.012     10.728      0.000
0.105       0.153
Cleanliness                          0.2191      0.012     18.040      0.000
0.195       0.243
Departure_Delay_in_Minutes           0.0044      0.001      4.422      0.000
0.002       0.006
Arrival_Delay_in_Minutes            -0.0095      0.001     -9.641      0.000
-0.011      -0.008
==============================================================================
=================
C:\Users\guy74\AppData\Local\Temp\ipykernel_16992\4104577527.py:87: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14
.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(data=coef_df.head(100), x="Coefficient", y="Feature", palette="coolwar
m")
```

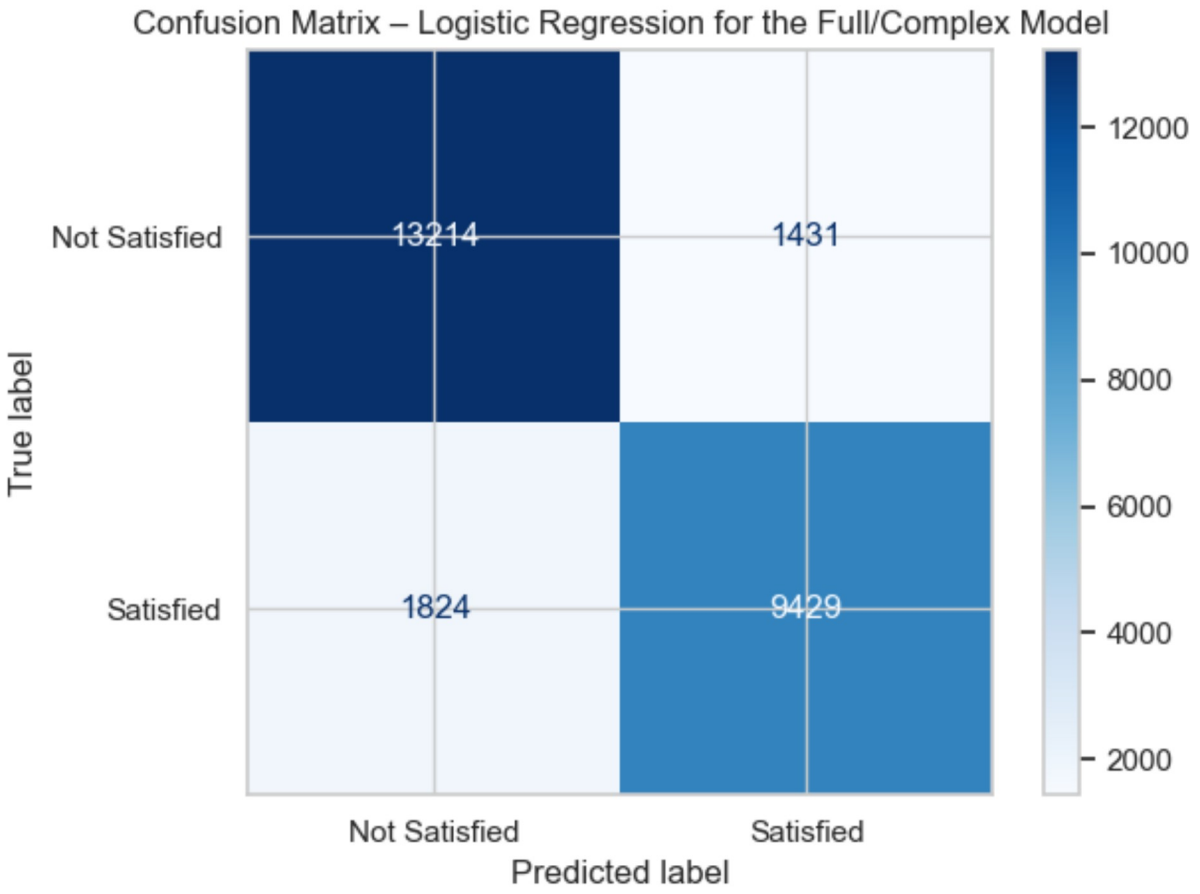


In [210…

```python
#Finally test out how the "big model" is doing but we will likely want to trim this
#(to avoid overfitting but still try to retain as much explanatory power as possibl
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_scor

#Predict on the test set
y_pred = logreg_pipeline.predict(X_test)

#Build out a confusion matrix and print the accuracy, precision, and recall
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Not Satisfied",

plt.figure(figsize=(6, 6))
disp.plot(cmap="Blues", values_format='d')
plt.title("Confusion Matrix – Logistic Regression for the Full/Complex Model")
```

```
plt.tight_layout()
plt.show()

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred))  #Adding F1
```

```
<Figure size 600x600 with 0 Axes>
```




```
Accuracy: 0.8743146188894896
Precision: 0.868232044198895
Recall: 0.8379098906958145
F1 Score: 0.8528015194681862
```

In [211…

```
#As seen above, good performance on the unseen testing data still, but likely more
#More to come in future iterations
```

In [212…

```
#Next I want to use some code to find the ACTUAL best subset of variables to use fo
#Brute forcing all models to analyze the AIC but only for combinations from 1 throu
#NOTE first approach was flawed but this take is matching AIC's now
import pandas as pd
import numpy as np
import statsmodels.api as sm
from itertools import combinations
from tqdm import tqdm #Lets us make progress bars during the run

df = pd.read_csv("AirlineSatisfaction_Transformed.csv").dropna()
df["satisfaction_binary"] = df["satisfaction"].apply(lambda x: 1 if x.strip().lower
exclude = ["Unnamed: 0", "id", "satisfaction", "satisfaction_binary"]
```

```
candidate_vars = [col for col in df.columns if col not in exclude]
from sklearn.model_selection import train_test_split
X = df[candidate_vars]
y = df["satisfaction_binary"]
X_train, _, y_train, _ = train_test_split(X, y, test_size=0.2, stratify=y, random_s


results = []

#Loop over all combinations of 1 to 4 variables, fit them, get AIC
for k in range(1, 5):
    for combo in tqdm(list(combinations(candidate_vars, k)), desc=f"Evaluating comb
        try:
            X_combo = X_train[list(combo)].copy()
            X_encoded = pd.get_dummies(X_combo, drop_first=True)
            X_encoded = sm.add_constant(X_encoded)
            model = sm.Logit(y_train, X_encoded).fit(disp=False)
            results.append({
                "Num_Vars": k,
                "Variables": combo,
                "AIC": model.aic
            })

        except Exception as e:
            continue

#Sort by the AIC and convert to a dataframe
aic_df = pd.DataFrame(results).sort_values(by="AIC").reset_index(drop=True)


print("\nTop 10 models by AIC:")
print(aic_df.head(10))
```

```
Evaluating combinations of 1 variables: 100%|████████| 22/22 [00:01<00:00, 16.37i
t/s]
Evaluating combinations of 2 variables: 100%|████████| 231/231 [00:15<00:00, 14.62
it/s]
Evaluating combinations of 3 variables: 100%|████████| 1540/1540 [01:41<00:00, 1
5.20it/s]
Evaluating combinations of 4 variables: 100%|████████| 7315/7315 [08:24<00:00, 1
4.50it/s]
```

```
Top 10 models by AIC:
   Num_Vars  \
0         4
1         4
2         4
3         4
4         4
5         4
6         4
7         4
8         4
9         4


Variables  \
0                   (Flight_Distance, Online_boarding, Inflight_entertainment, Leg_
room_service)
1                   (Flight_Distance, Online_boarding, Inflight_entertainment, On-b
oard_service)
2                         (Flight_Distance, Online_boarding, On-board_service, Leg_
room_service)
3                  (Online_boarding, Inflight_entertainment, On-board_service, Leg_
room_service)
4                   (Online_boarding, Inflight_entertainment, Leg_room_service, Che
ckin_service)
5                    (Flight_Distance, Online_boarding, Inflight_entertainment, Che
ckin_service)
6  (Departure/Arrival_time_convenient, Online_boarding, Inflight_entertainment, Leg_
room_service)
7                   (Flight_Distance, Online_boarding, Inflight_entertainment, Bagg
age_handling)
8                   (Flight_Distance, Online_boarding, Inflight_entertainment, Infl
ight_service)
9                  (Online_boarding, Inflight_entertainment, Leg_room_service, Bagg
age_handling)

            AIC
0  93996.830276
1  95309.465752
2  95654.683162
3  96046.680058
4  96287.059640
5  96299.235448
6  96505.311690
7  96506.095550
8  96647.565633
9  97140.152008
```

In [213…

```python
#Using that full list of all the AICs by each model to examine some of the other on
pd.options.display.max_colwidth = None
top_models = aic_df.sort_values(by='AIC').head(30).copy()
#top_models = aic_df.sort_values(by='AIC').tail(30).copy()
top_models['Variables'] = top_models['Variables'].apply(lambda x: ", ".join(x))
print(top_models)
```

```
    Num_Vars  \
0          4
1          4
2          4
3          4
4          4
5          4
6          4
7          4
8          4
9          4
10         4
11         4
12         4
13         4
14         4
15         4
16         4
17         4
18         4
19         4
20         4
21         4
22         4
23         4
24         4
25         4
26         4
27         4
28         3
29         4


Variables  \
0                    Flight_Distance, Online_boarding, Inflight_entertainment, Leg_
room_service
1                    Flight_Distance, Online_boarding, Inflight_entertainment, On-b
oard_service
2                          Flight_Distance, Online_boarding, On-board_service, Leg_
room_service
3                   Online_boarding, Inflight_entertainment, On-board_service, Leg_
room_service
4                    Online_boarding, Inflight_entertainment, Leg_room_service, Che
ckin_service
5                     Flight_Distance, Online_boarding, Inflight_entertainment, Che
ckin_service
6   Departure/Arrival_time_convenient, Online_boarding, Inflight_entertainment, Leg_
room_service
7                    Flight_Distance, Online_boarding, Inflight_entertainment, Bagg
age_handling
8                    Flight_Distance, Online_boarding, Inflight_entertainment, Infl
ight_service
9                   Online_boarding, Inflight_entertainment, Leg_room_service, Bagg
age_handling
10   Flight_Distance, Departure/Arrival_time_convenient, Online_boarding, Inflight_e
ntertainment
```

```
11                              Flight_Distance, Online_boarding, Leg_room_service,
Cleanliness
12                  Online_boarding, Inflight_entertainment, Leg_room_service, Infl
ight_service
13                             Online_boarding, Seat_comfort, On-board_service, Leg_
room_service
14                          Flight_Distance, Online_boarding, Leg_room_service, Infl
ight_service
15                              Online_boarding, On-board_service, Leg_room_service,
Cleanliness
16           Online_boarding, Inflight_entertainment, Leg_room_service, Arrival_Dela
y_in_Minutes
17                          Flight_Distance, Online_boarding, Leg_room_service, Bagg
age_handling
18         Online_boarding, Inflight_entertainment, Leg_room_service, Departure_Dela
y_in_Minutes
19                              Flight_Distance, Online_boarding, On-board_service,
Cleanliness
20               Flight_Distance, Inflight_wifi_service, Online_boarding, Inflight_e
ntertainment
21                              Flight_Distance, Online_boarding, Seat_comfort, Leg_
room_service
22                     Food_and_drink, Online_boarding, Inflight_entertainment, Leg_
room_service
23              Inflight_wifi_service, Online_boarding, Inflight_entertainment, Leg_
room_service
24             Ease_of_Online_booking, Online_boarding, Inflight_entertainment, Leg_
room_service
25                               Age, Online_boarding, Inflight_entertainment, Leg_
room_service
26                       Online_boarding, Seat_comfort, Inflight_entertainment, Leg_
room_service
27                      Flight_Distance, Food_and_drink, Online_boarding, Inflight_e
ntertainment
28                                    Online_boarding, Inflight_entertainment, Leg_
room_service
29                      Gate_location, Online_boarding, Inflight_entertainment, Leg_
room_service

             AIC
0   93996.830276
1   95309.465752
2   95654.683162
3   96046.680058
4   96287.059640
5   96299.235448
6   96505.311690
7   96506.095550
8   96647.565633
9   97140.152008
10  97176.158465
11  97202.746850
12  97215.892542
13  97218.067398
14  97265.094931
15  97295.391806
```

```
16  97300.774501
17  97366.555093
18  97367.251016
19  97383.532500
20  97392.849196
21  97458.311705
22  97469.166477
23  97494.507828
24  97528.841033
25  97536.796757
26  97539.577456
27  97588.176789
28  97620.637472
29  97621.187861
```

In [214…

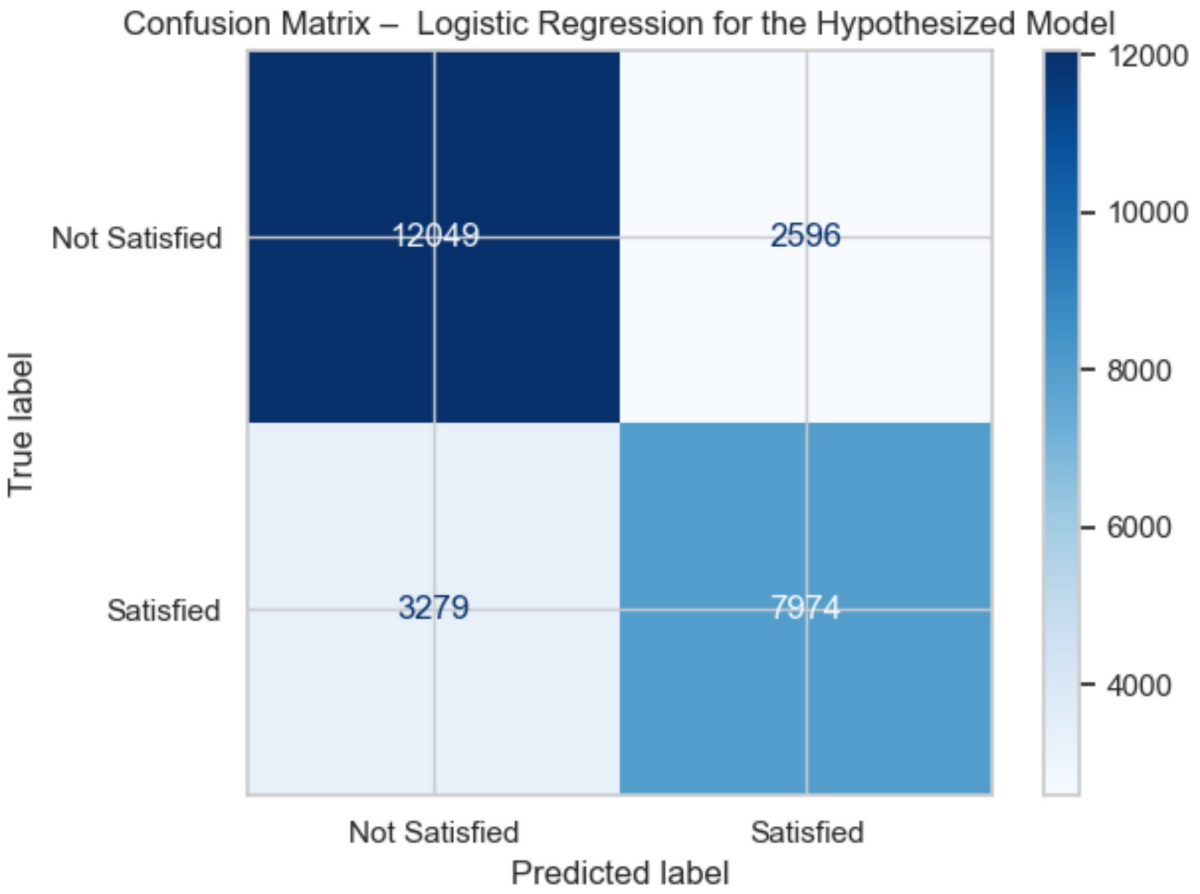
```
#Next I want to compare the three models using AIC and the confusion matrix from ea
#he full/complex one, the hypothesized one from the beginning of the project, and t
```

In [215…

```
#This version for the hypothesized model from the beginning

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import statsmodels.api as sm

df = pd.read_csv("AirlineSatisfaction_Transformed.csv")
df = df.dropna()
df["satisfaction_binary"] = df["satisfaction"].apply(lambda x: 1 if x.strip().lower

#Selecting only specific variables now. Going to first run this for my hypothesized
selected_vars = ["Class", "On-board_service", "Inflight_service", "Arrival_Delay_in
X = df[selected_vars].copy()
y = df["satisfaction_binary"]


#Build hypothesized model
categorical_cols = X.select_dtypes(include="object").columns.tolist()
numeric_cols = X.select_dtypes(include=["int64", "float64"]).columns.tolist()

preprocessor = ColumnTransformer([
    ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'), categorical_cols)
    ('num', 'passthrough', numeric_cols)
])
logreg_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=10000))
])

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2
logreg_pipeline.fit(X_train, y_train)
```

```
X_train_encoded = preprocessor.fit_transform(X_train)

#Get column names after encoding
ohe = preprocessor.transformers_[0][1]
ohe_feature_names = ohe.get_feature_names_out(categorical_cols)
all_feature_names = np.concatenate([ohe_feature_names, numeric_cols])

#Fit model
X_train_df = pd.DataFrame(
    X_train_encoded.toarray() if hasattr(X_train_encoded, "toarray") else X_train_e
    columns=all_feature_names
)
X_train_df = sm.add_constant(X_train_df)
sm_model = sm.Logit(y_train.values, X_train_df)
sm_result = sm_model.fit()

#Show the stats
print(sm_result.summary())

print(f"\nAIC for Hypothesized Model: {sm_result.aic:.2f}")



#And also the confusion matrix
#Next output the confusion matrix for the hypothesized model
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_scor
import matplotlib.pyplot as plt

#Predict probabilities on test set
X_test_hypo_encoded = preprocessor.transform(X_test[selected_vars])
X_test_hypo_df = pd.DataFrame(
    X_test_hypo_encoded.toarray() if hasattr(X_test_hypo_encoded, 'toarray') else X
    columns=all_feature_names
)
X_test_hypo_df = sm.add_constant(X_test_hypo_df)

#Predict class labels using threshold 0.5
y_pred_hypo = sm_result.predict(X_test_hypo_df) > 0.5

#Build confusion matrix and metrics
cm = confusion_matrix(y_test, y_pred_hypo)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Not Satisfied",

plt.figure(figsize=(6, 6))
disp.plot(cmap="Blues", values_format='d')
plt.title("Confusion Matrix -  Logistic Regression for the Hypothesized Model")
plt.tight_layout()
plt.show()

print("Accuracy:", accuracy_score(y_test, y_pred_hypo))
print("Precision:", precision_score(y_test, y_pred_hypo))
print("Recall:", recall_score(y_test, y_pred_hypo))
```

```
Optimization terminated successfully.
         Current function value: 0.516501
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:               103589
Model:                          Logit   Df Residuals:                   103583
Method:                           MLE   Df Model:                            5
Date:                Sun, 01 Jun 2025   Pseudo R-squ.:                  0.2455
Time:                        00:41:14   Log-Likelihood:                -53504.
converged:                       True   LL-Null:                       -70911.
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
========
                               coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
const                       -1.1447      0.029    -39.935      0.000      -1.201
-1.089
Class_Eco                   -2.1835      0.016   -137.015      0.000      -2.215
-2.152
Class_Eco Plus              -1.7723      0.030    -59.907      0.000      -1.830
-1.714
On-board_service             0.4097      0.007     54.801      0.000       0.395
0.424
Inflight_service             0.1496      0.008     18.942      0.000       0.134
0.165
Arrival_Delay_in_Minutes    -0.0034      0.000    -15.989      0.000      -0.004
-0.003
================================================================================
========

AIC for Hypothesized Model: 107019.54
<Figure size 600x600 with 0 Axes>
```




```
Accuracy: 0.773148505676114
Precision: 0.754399243140965
Recall: 0.7086110370567849
```

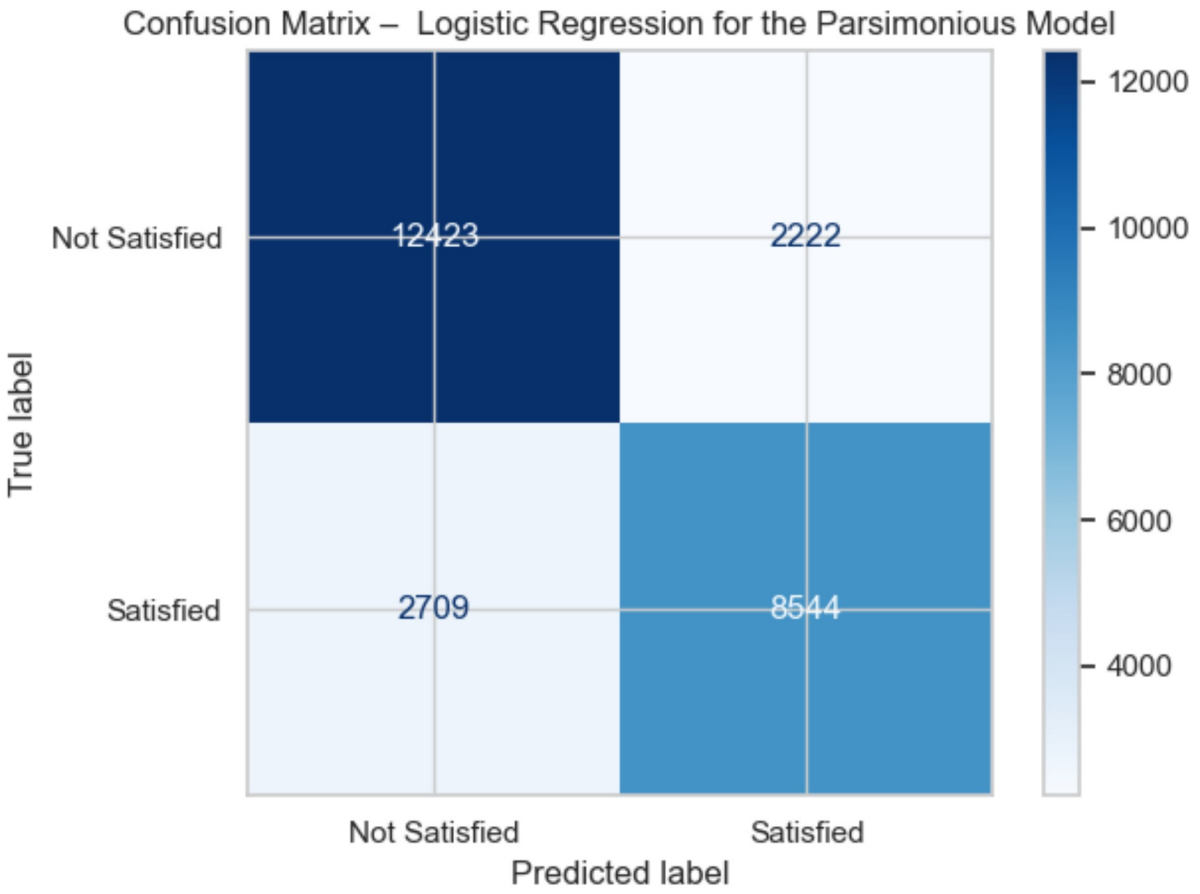
In [216…
```
#Finally below now we do the same for the most parsimonious model we found via the
```

In [217…
```
#This version for the most parsimonious model discovered by the loop of all models
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import statsmodels.api as sm

df = pd.read_csv("AirlineSatisfaction_Transformed.csv")
df = df.dropna()
df["satisfaction_binary"] = df["satisfaction"].apply(lambda x: 1 if x.strip().lower

#Selecting only specific variables now. Now these are changed to the Parsimonious m
selected_vars = ["Flight_Distance", "Online_boarding", "Inflight_entertainment", "L
X = df[selected_vars].copy()
y = df["satisfaction_binary"]


#Build parsimonious model
```

```
categorical_cols = X.select_dtypes(include="object").columns.tolist()
numeric_cols = X.select_dtypes(include=["int64", "float64"]).columns.tolist()

preprocessor = ColumnTransformer([
    ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'), categorical_cols)
    ('num', 'passthrough', numeric_cols)
])
logreg_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=10000))
])

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2
logreg_pipeline.fit(X_train, y_train)


preprocessor.fit(X_train)  # Fit the preprocessor explicitly
X_train_encoded = preprocessor.transform(X_train)


# Get column names after encoding
if categorical_cols:
    ohe = fitted_preprocessor.named_transformers_['cat']
    ohe_feature_names = ohe.get_feature_names_out(categorical_cols)
    all_feature_names = np.concatenate([ohe_feature_names, numeric_cols])
else:
    all_feature_names = numeric_cols


#Fit model
X_train_df = pd.DataFrame(
    X_train_encoded.toarray() if hasattr(X_train_encoded, "toarray") else X_train_e
    columns=all_feature_names
)
X_train_df = sm.add_constant(X_train_df)
sm_model = sm.Logit(y_train.values, X_train_df)
sm_result = sm_model.fit()

#Show the stats
print(sm_result.summary())

print(f"\nAIC for Parsimonious Model: {sm_result.aic:.2f}")



#And also the confusion matrix
#Next output the confusion matrix for the Parsimonious model
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_scor
import matplotlib.pyplot as plt

#Predict probabilities on test set
X_test_pars_encoded = preprocessor.transform(X_test[selected_vars])
X_test_pars_df = pd.DataFrame(
    X_test_pars_encoded.toarray() if hasattr(X_test_pars_encoded, 'toarray') else X
    columns=all_feature_names
)
```

```python
X_test_pars_df = sm.add_constant(X_test_pars_df)

#Predict class labels using threshold 0.5
y_pred_pars = sm_result.predict(X_test_pars_df) > 0.5

#Build confusion matrix and metrics
cm = confusion_matrix(y_test, y_pred_pars)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Not Satisfied",

plt.figure(figsize=(6, 6))
disp.plot(cmap="Blues", values_format='d')
plt.title("Confusion Matrix -  Logistic Regression for the Parsimonious Model")
plt.tight_layout()
plt.show()

print("Accuracy:", accuracy_score(y_test, y_pred_pars))
print("Precision:", precision_score(y_test, y_pred_pars))
print("Recall:", recall_score(y_test, y_pred_pars))
```

```
Optimization terminated successfully.
         Current function value: 0.453653
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:               103589
Model:                          Logit   Df Residuals:                   103584
Method:                           MLE   Df Model:                            4
Date:                Sun, 01 Jun 2025   Pseudo R-squ.:                  0.3373
Time:                        00:41:16   Log-Likelihood:                -46993.
converged:                       True   LL-Null:                       -70911.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
======
                             coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
------
const                     -6.9007      0.043   -161.362      0.000      -6.985
-6.817
Flight_Distance            0.0005   8.61e-06     58.432      0.000       0.000
0.001
Online_boarding            0.8811      0.007    119.690      0.000       0.867
0.896
Inflight_entertainment     0.4802      0.007     71.068      0.000       0.467
0.493
Leg_room_service           0.4143      0.007     62.621      0.000       0.401
0.427
==============================================================================
======

AIC for Parsimonious Model: 93996.83
<Figure size 600x600 with 0 Axes>
```




```
Accuracy: 0.8095991968491776
Precision: 0.793609511424856
Recall: 0.7592641962143428
```

In [218…

```
#Another attempt at automatic feature selection to see about an effective way to te
#i.e. we can try >4 variable combinations without taking way too long to run
#So this time instead of a brute force loop to get AIC using forward stepwise AIC s
```

In [219…

```
#Run the full model and then also run the automatic selection using stepwise
import pandas as pd
import numpy as np
import statsmodels.api as sm
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

#Prep
df = pd.read_csv("AirlineSatisfaction_Transformed.csv").dropna()
df["satisfaction_binary"] = df["satisfaction"].apply(lambda x: 1 if x.strip().lower
X_raw = df.drop(columns=["satisfaction", "satisfaction_binary"])
y = df["satisfaction_binary"]
categorical_cols = X_raw.select_dtypes(include="object").columns.tolist()
numeric_cols = X_raw.select_dtypes(include=["int64", "float64"]).columns.tolist()

preprocessor = ColumnTransformer([
    ("cat", OneHotEncoder(drop="first", handle_unknown="ignore"), categorical_cols)
    ("num", "passthrough", numeric_cols)
])

X_encoded = preprocessor.fit_transform(X_raw)
```

```
ohe = preprocessor.named_transformers_["cat"]
cat_feature_names = ohe.get_feature_names_out(categorical_cols)
all_feature_names = np.concatenate([cat_feature_names, numeric_cols])

X_df = pd.DataFrame(
    X_encoded.toarray() if hasattr(X_encoded, "toarray") else X_encoded,
    columns=all_feature_names,
    index=X_raw.index
)

#Full/complex model
X_full_sm = sm.add_constant(X_df)
full_model = sm.Logit(y, X_full_sm).fit()
print("=== Full/Complex Model ===")
print(f"AIC: {full_model.aic:.2f}")

#Forward stepwise AIC selection model
remaining = list(all_feature_names)
selected = []
current_aic = np.inf

print("\n=== Forward Stepwise AIC Selection ===")
while True:
    improved = False
    best_candidate = None
    best_aic = current_aic

    for candidate in remaining:
        try_vars = selected + [candidate]
        X_try = sm.add_constant(X_df[try_vars])
        model_try = sm.Logit(y, X_try).fit(disp=False)
        if model_try.aic < best_aic:
            improved = True
            best_candidate = candidate
            best_aic = model_try.aic
    if improved:
        selected.append(best_candidate)
        remaining.remove(best_candidate)
        current_aic = best_aic
        print(f"Added: {best_candidate}, AIC: {current_aic:.2f}")
    else:
        break

#Final stepwise model
X_stepwise = sm.add_constant(X_df[selected])
stepwise_model = sm.Logit(y, X_stepwise).fit()
print("\n=== Final Stepwise Model ===")
print(f"AIC: {stepwise_model.aic:.2f}")
print("Selected Variables:", selected)

# === Final Comparison Summary ===
print("\n=== Model AIC Comparison ===")
print(f"Full Model AIC: {full_model.aic:.2f}")
print(f"Stepwise Model AIC: {stepwise_model.aic:.2f}")

if stepwise_model.aic < full_model.aic:
```

```
    print("Stepwise model has lower AIC (more parsimonious and better fit).")
elif stepwise_model.aic == full_model.aic:
    print("Stepwise model matches full model AIC.")
else:
    print("Full model has lower AIC, problematic result? Stepwise selection did not

#Interesting, appears our best bet is still technically the full model even in term
#i.e. the predictive power by adding all of these is still stronger than the penalt
```

```
Optimization terminated successfully.
         Current function value: 0.334658
         Iterations 7
=== Full/Complex Model ===
AIC: 86715.67

=== Forward Stepwise AIC Selection ===
Added: Online_boarding, AIC: 139840.85
Added: Type_of_Travel_Personal Travel, AIC: 117244.50
Added: On-board_service, AIC: 106824.45
Added: Customer_Type_disloyal Customer, AIC: 99070.72
Added: Checkin_service, AIC: 95976.48
Added: Inflight_entertainment, AIC: 93174.51
Added: Leg_room_service, AIC: 91330.94
Added: Inflight_wifi_service, AIC: 90678.33
Added: Class_Eco, AIC: 89863.01
Added: Departure/Arrival_time_convenient, AIC: 89200.82
Added: Class_Eco Plus, AIC: 88630.40
Added: Arrival_Delay_in_Minutes, AIC: 88192.40
Added: Cleanliness, AIC: 87716.46
Added: Baggage_handling, AIC: 87313.04
Added: Ease_of_Online_booking, AIC: 87100.88
Added: Age, AIC: 86923.87
Added: Inflight_service, AIC: 86798.60
Added: Seat_comfort, AIC: 86762.66
Added: Departure_Delay_in_Minutes, AIC: 86741.88
Added: Gender_Male, AIC: 86731.29
Added: Gate_location, AIC: 86723.15
Added: Food_and_drink, AIC: 86716.58
Added: Flight_Distance, AIC: 86715.67
Optimization terminated successfully.
         Current function value: 0.334658
         Iterations 7

=== Final Stepwise Model ===
AIC: 86715.67
Selected Variables: ['Online_boarding', 'Type_of_Travel_Personal Travel', 'On-board_
service', 'Customer_Type_disloyal Customer', 'Checkin_service', 'Inflight_entertainm
ent', 'Leg_room_service', 'Inflight_wifi_service', 'Class_Eco', 'Departure/Arrival_t
ime_convenient', 'Class_Eco Plus', 'Arrival_Delay_in_Minutes', 'Cleanliness', 'Bagga
ge_handling', 'Ease_of_Online_booking', 'Age', 'Inflight_service', 'Seat_comfort', '
Departure_Delay_in_Minutes', 'Gender_Male', 'Gate_location', 'Food_and_drink', 'Flig
ht_Distance']

=== Model AIC Comparison ===
Full Model AIC: 86715.67
Stepwise Model AIC: 86715.67
Stepwise model matches full model AIC.
```

In [220…

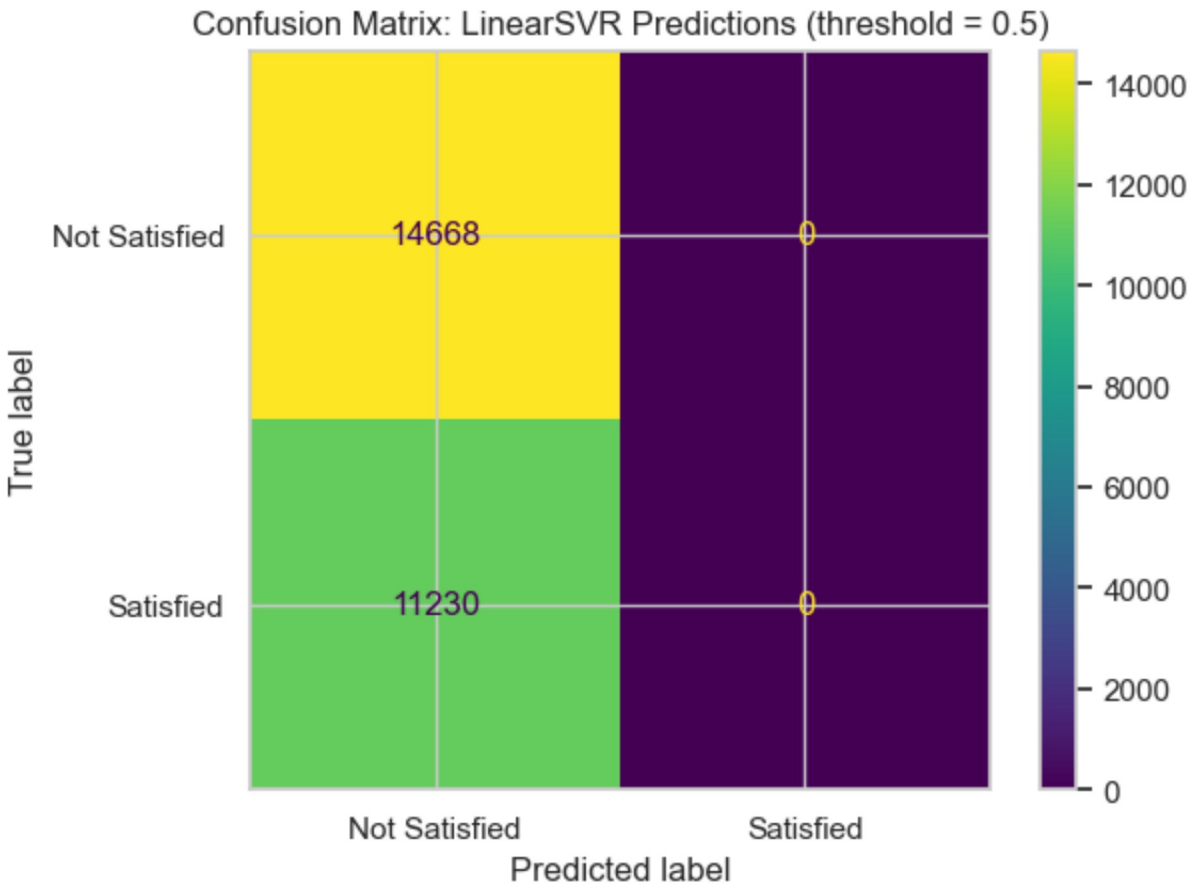
```
#Next during week 4 going to work on the SVM portion now first and then get into th
#(Feedback from last submission indciated we should have SVM included)
```

In [221…

```
#NOTE RETAINING THIS CELL FOR FUTURE POSTERITY JUST IN CASE BUT THIS APPROACH TO SV


#In support of the SVM model/code below now setting things up here again using the
```

```python
from sklearn.svm import LinearSVR
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np


#Use the  existing preprocessed features and binary satisfaction target
X = X_df
y = df["satisfaction_binary"].astype(float)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

#Fit SVM model
svm_reg = LinearSVR(epsilon=1.5, random_state=42)
svm_reg.fit(X_train, y_train)

#Predict using the model on the test set
y_pred = svm_reg.predict(X_test)

#Get and print the metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("LinearSVR Results:")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"Mean Absolute Error (MAE): {mae:.4f}")
print(f"R² Score: {r2:.4f}")


#Also adding a confusion matrix so we can compare them more apples to apples/simila
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
y_pred_binary = (y_pred >= 0.5).astype(int)
cm = confusion_matrix(y_test, y_pred_binary)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Not Satisfied",
disp.plot()
plt.title("Confusion Matrix: LinearSVR Predictions (threshold = 0.5)")
plt.show()


#Print out the accuracy, precision, and recall metrics just like before with the LR
from sklearn.metrics import accuracy_score, precision_score, recall_score

print(f"Accuracy: {accuracy_score(y_test, y_pred_binary):.2%}")
print(f"Precision: {precision_score(y_test, y_pred_binary, zero_division=0):.2%}")
print(f"Recall: {recall_score(y_test, y_pred_binary, zero_division=0):.2%}")
```

```
LinearSVR Results:
Mean Squared Error (MSE): 0.4336
Mean Absolute Error (MAE): 0.4336
R² Score: -0.7656
```

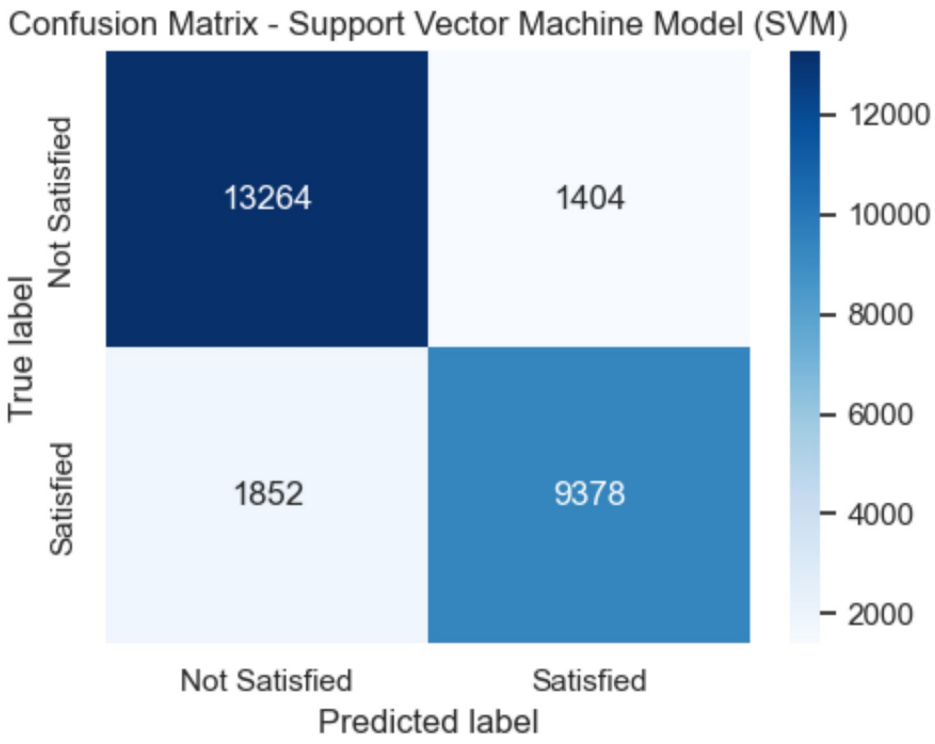



```
Accuracy: 56.64%
Precision: 0.00%
Recall: 0.00%
```

In [249…

```
#In support of the SVM model/code below now setting things up here again using the
#Ahhh upon further investigation it appears we should not be using the LinearSVR th
#The learning activity sort of tripped me up at first because it appears to be tell
#But I think the true intent is for us to use linearSVC instead(???)
#Yep! This is looking much better, clearing out the failed attempt cells above and


from sklearn.svm import LinearSVC
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, reca
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from sklearn.metrics import f1_score


from sklearn.exceptions import ConvergenceWarning
warnings.filterwarnings("ignore", category=ConvergenceWarning)

#Train SVM classifier
svm_clf = LinearSVC(max_iter=10000, random_state=42)

#Confusion matrix is like an order of magnitude larger???
#Oh I see the problem now, we need to fit it on the training set and only run the p
#svm_clf.fit(X_df, y)
```

```
#Train on training set
svm_clf.fit(X_train, y_train)

#Predict on training set
#y_pred = svm_clf.predict(X_df)
#Predict on test set
y_pred = svm_clf.predict(X_test)



#Output the metrics as well as a similar confusion matrix so we can compare them mo
#(similar methodology of comparison for the various models as we used above for LR)
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred, zero_division=0)
rec = recall_score(y_test, y_pred, zero_division=0)
f1 = f1_score(y_test, y_pred, zero_division=0)  #Adding F1 as this may be considere

#Not really as applicable for non regression but leave in for now at least consiste
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)


#Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Not Satisfied", "S
plt.title("Confusion Matrix - Support Vector Machine Model (SVM)")
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.tight_layout()
plt.show()



#print("LinearSVC Results:")
print(f"Accuracy: {acc:.4f}")
print(f"Precision: {prec:.4f}")
print(f"Recall: {rec:.4f}")
print(f"F1 Score: {f1:.4f}")  #Add F1

#print(f"Mean Squared Error (MSE): {mse:.4f}")
#print(f"Mean Absolute Error (MAE): {mae:.4f}")
#print(f"R² Score: {r2:.4f}")
```




```
Accuracy: 0.8743
Precision: 0.8698
Recall: 0.8351
F1 Score: 0.8521
```

In [ ]:
```
#Now finally hitting the week 4 intended stuff for the Deep Learning Recurrent Neur
#But I guess I have to figure out how to run this in collab instead of locally from
#Set up and played with colab but decided I am first going to attempt running this
#even if the runs take a bit longer, as long as it's a matter of minutes and not ho
```

In [251…
```
pip install tensorflow
```

```
Collecting tensorflowNote: you may need to restart the kernel to use updated package
s.

  Downloading tensorflow-2.19.0-cp312-cp312-win_amd64.whl.metadata (4.1 kB)
Collecting absl-py>=1.0.0 (from tensorflow)
  Downloading absl_py-2.3.0-py3-none-any.whl.metadata (2.4 kB)
Collecting astunparse>=1.6.0 (from tensorflow)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flatbuffers>=24.3.25 (from tensorflow)
  Downloading flatbuffers-25.2.10-py2.py3-none-any.whl.metadata (875 bytes)
Collecting gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 (from tensorflow)
  Downloading gast-0.6.0-py3-none-any.whl.metadata (1.3 kB)
Collecting google-pasta>=0.1.1 (from tensorflow)
  Downloading google_pasta-0.2.0-py3-none-any.whl.metadata (814 bytes)
Collecting libclang>=13.0.0 (from tensorflow)
  Downloading libclang-18.1.1-py2.py3-none-win_amd64.whl.metadata (5.3 kB)
Collecting opt-einsum>=2.3.2 (from tensorflow)
  Downloading opt_einsum-3.4.0-py3-none-any.whl.metadata (6.3 kB)
Requirement already satisfied: packaging in c:\users\guy74\anaconda3\lib\site-packag
es (from tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.2
1.4,!=4.21.5,<6.0.0dev,>=3.20.3 in c:\users\guy74\anaconda3\lib\site-packages (from
tensorflow) (4.25.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\guy74\anaconda3\lib\s
ite-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in c:\users\guy74\anaconda3\lib\site-packa
ges (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in c:\users\guy74\anaconda3\lib\site-pack
ages (from tensorflow) (1.16.0)
Collecting termcolor>=1.1.0 (from tensorflow)
  Downloading termcolor-3.1.0-py3-none-any.whl.metadata (6.4 kB)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\guy74\anaconda3
\lib\site-packages (from tensorflow) (4.11.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\guy74\anaconda3\lib\site-pa
ckages (from tensorflow) (1.14.1)
Collecting grpcio<2.0,>=1.24.3 (from tensorflow)
  Downloading grpcio-1.71.0-cp312-cp312-win_amd64.whl.metadata (4.0 kB)
Collecting tensorboard~=2.19.0 (from tensorflow)
  Downloading tensorboard-2.19.0-py3-none-any.whl.metadata (1.8 kB)
Collecting keras>=3.5.0 (from tensorflow)
  Downloading keras-3.10.0-py3-none-any.whl.metadata (6.0 kB)
Requirement already satisfied: numpy<2.2.0,>=1.26.0 in c:\users\guy74\anaconda3\lib
\site-packages (from tensorflow) (1.26.4)
Requirement already satisfied: h5py>=3.11.0 in c:\users\guy74\anaconda3\lib\site-pac
kages (from tensorflow) (3.11.0)
Collecting ml-dtypes<1.0.0,>=0.5.1 (from tensorflow)
  Downloading ml_dtypes-0.5.1-cp312-cp312-win_amd64.whl.metadata (22 kB)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\guy74\anaconda3\lib\si
te-packages (from astunparse>=1.6.0->tensorflow) (0.44.0)
Requirement already satisfied: rich in c:\users\guy74\anaconda3\lib\site-packages (f
rom keras>=3.5.0->tensorflow) (13.7.1)
Collecting namex (from keras>=3.5.0->tensorflow)
  Downloading namex-0.1.0-py3-none-any.whl.metadata (322 bytes)
Collecting optree (from keras>=3.5.0->tensorflow)
  Downloading optree-0.16.0-cp312-cp312-win_amd64.whl.metadata (31 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\guy74\anaconda3
```

```
\lib\site-packages (from requests<3,>=2.21.0->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\guy74\anaconda3\lib\site-pac
kages (from requests<3,>=2.21.0->tensorflow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\guy74\anaconda3\lib\si
te-packages (from requests<3,>=2.21.0->tensorflow) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\guy74\anaconda3\lib\si
te-packages (from requests<3,>=2.21.0->tensorflow) (2025.1.31)
Requirement already satisfied: markdown>=2.6.8 in c:\users\guy74\anaconda3\lib\site-
packages (from tensorboard~=2.19.0->tensorflow) (3.4.1)
Collecting tensorboard-data-server<0.8.0,>=0.7.0 (from tensorboard~=2.19.0->tensorfl
ow)
  Downloading tensorboard_data_server-0.7.2-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\guy74\anaconda3\lib\site-
packages (from tensorboard~=2.19.0->tensorflow) (3.0.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\guy74\anaconda3\lib\sit
e-packages (from werkzeug>=1.0.1->tensorboard~=2.19.0->tensorflow) (2.1.3)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\guy74\anaconda3\lib
\site-packages (from rich->keras>=3.5.0->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\guy74\anaconda3\l
ib\site-packages (from rich->keras>=3.5.0->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\guy74\anaconda3\lib\site-packa
ges (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.0)
Downloading tensorflow-2.19.0-cp312-cp312-win_amd64.whl (376.0 MB)
   ---------------------------------------- 0.0/376.0 MB ? eta -:--:--
   -- ------------------------------------- 21.8/376.0 MB 105.9 MB/s eta 0:00:04
   ---- ----------------------------------- 45.4/376.0 MB 111.1 MB/s eta 0:00:03
   ------- -------------------------------- 69.5/376.0 MB 113.6 MB/s eta 0:00:03
   --------- ------------------------------ 95.4/376.0 MB 110.8 MB/s eta 0:00:03
   ------------ --------------------------- 119.5/376.0 MB 112.3 MB/s eta 0:00:03
   -------------- ------------------------- 143.7/376.0 MB 113.4 MB/s eta 0:00:03
   ---------------- ----------------------- 167.5/376.0 MB 112.7 MB/s eta 0:00:02
   ------------------- -------------------- 191.1/376.0 MB 113.1 MB/s eta 0:00:02
   ---------------------- ----------------- 216.0/376.0 MB 114.2 MB/s eta 0:00:02
   ------------------------ --------------- 239.9/376.0 MB 114.4 MB/s eta 0:00:02
   --------------------------- ------------ 263.7/376.0 MB 114.8 MB/s eta 0:00:01
   ----------------------------- ---------- 288.4/376.0 MB 115.6 MB/s eta 0:00:01
   ------------------------------- -------- 311.7/376.0 MB 115.6 MB/s eta 0:00:01
   --------------------------------- ------ 334.0/376.0 MB 116.4 MB/s eta 0:00:01
   ----------------------------------- ---- 351.3/376.0 MB 112.5 MB/s eta 0:00:01
   ------------------------------------  375.9/376.0 MB 112.4 MB/s eta 0:00:01
   ------------------------------------  375.9/376.0 MB 112.4 MB/s eta 0:00:01
   ------------------------------------  375.9/376.0 MB 112.4 MB/s eta 0:00:01
   ------------------------------------  375.9/376.0 MB 112.4 MB/s eta 0:00:01
   ---------------------------------------- 376.0/376.0 MB 85.0 MB/s eta 0:00:00
Downloading absl_py-2.3.0-py3-none-any.whl (135 kB)
Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Downloading flatbuffers-25.2.10-py2.py3-none-any.whl (30 kB)
Downloading gast-0.6.0-py3-none-any.whl (21 kB)
Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
Downloading grpcio-1.71.0-cp312-cp312-win_amd64.whl (4.3 MB)
   ---------------------------------------- 0.0/4.3 MB ? eta -:--:--
   ---------------------------------------- 4.3/4.3 MB 85.3 MB/s eta 0:00:00
Downloading keras-3.10.0-py3-none-any.whl (1.4 MB)
   ---------------------------------------- 0.0/1.4 MB ? eta -:--:--
   ---------------------------------------- 1.4/1.4 MB 69.9 MB/s eta 0:00:00
Downloading libclang-18.1.1-py2.py3-none-win_amd64.whl (26.4 MB)
```

```
   ---------------------------------------- 0.0/26.4 MB ? eta -:--:--
   ---------------------------------- --- 24.4/26.4 MB 118.2 MB/s eta 0:00:01
   ---------------------------------------- 26.4/26.4 MB 92.7 MB/s eta 0:00:00
Downloading ml_dtypes-0.5.1-cp312-cp312-win_amd64.whl (210 kB)
Downloading opt_einsum-3.4.0-py3-none-any.whl (71 kB)
Downloading tensorboard-2.19.0-py3-none-any.whl (5.5 MB)
   ---------------------------------------- 0.0/5.5 MB ? eta -:--:--
   ---------------------------------------- 5.5/5.5 MB 84.5 MB/s eta 0:00:00
Downloading termcolor-3.1.0-py3-none-any.whl (7.7 kB)
Downloading tensorboard_data_server-0.7.2-py3-none-any.whl (2.4 kB)
Downloading namex-0.1.0-py3-none-any.whl (5.9 kB)
Downloading optree-0.16.0-cp312-cp312-win_amd64.whl (315 kB)
Installing collected packages: namex, libclang, flatbuffers, termcolor, tensorboard-
data-server, optree, opt-einsum, ml-dtypes, grpcio, google-pasta, gast, astunparse,
absl-py, tensorboard, keras, tensorflow
Successfully installed absl-py-2.3.0 astunparse-1.6.3 flatbuffers-25.2.10 gast-0.6.0
google-pasta-0.2.0 grpcio-1.71.0 keras-3.10.0 libclang-18.1.1 ml-dtypes-0.5.1 name
x-0.1.0 opt-einsum-3.4.0 optree-0.16.0 tensorboard-2.19.0 tensorboard-data-serve
r-0.7.2 tensorflow-2.19.0 termcolor-3.1.0
```

In [253…

```python
import tensorflow as tf
print(tf.__version__)
```

```
2.19.0
```

In [257…

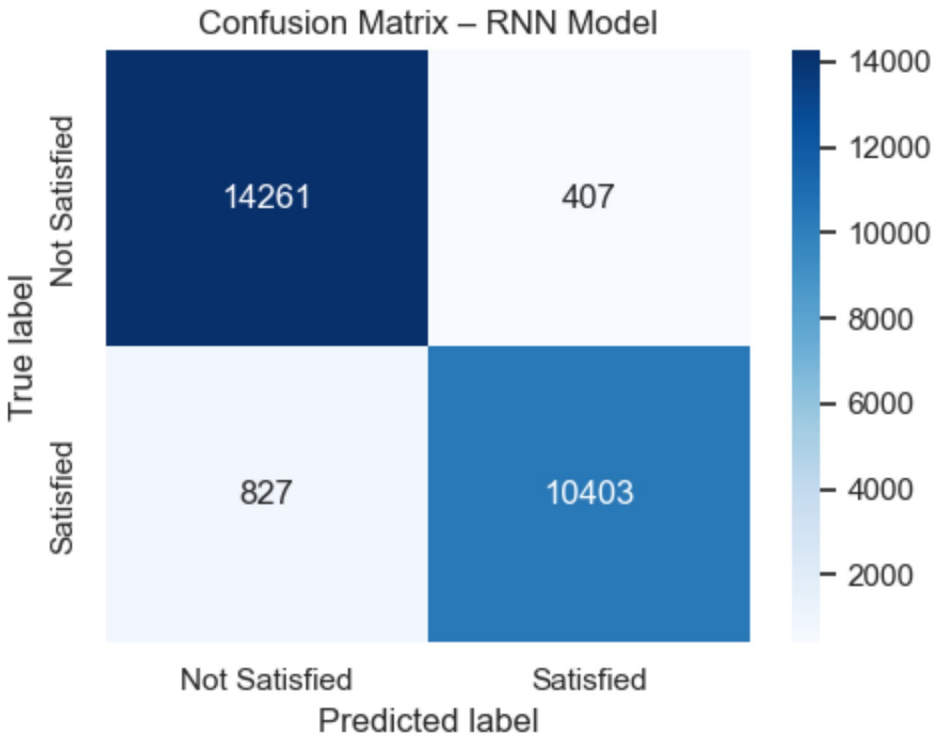
```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense, InputLayer

#Using the same dataframes we already created for the training/testing split from t
X_train, X_test, y_train, y_test = train_test_split(X_df, y, test_size=0.2, random_

#Scale the input features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#Reshape data for RNN input: (samples, timesteps, features)
#We'll treat each row as a single timestep sequence to simulate a recurrent setup
X_train_rnn = X_train_scaled.reshape((X_train_scaled.shape[0], 1, X_train_scaled.sh
X_test_rnn = X_test_scaled.reshape((X_test_scaled.shape[0], 1, X_test_scaled.shape[

#Build the RNN model
model = Sequential()
model.add(InputLayer(input_shape=(1, X_train_scaled.shape[1])))
model.add(SimpleRNN(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))  # binary classification output

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
#Train the model
history = model.fit(X_train_rnn, y_train, epochs=20, batch_size=32, validation_spli

#Predict and evaluate
y_pred_prob = model.predict(X_test_rnn)
y_pred = (y_pred_prob > 0.5).astype("int32").flatten()



#Confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Not Satisfied", "S
plt.title("Confusion Matrix - RNN Model")
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.tight_layout()
plt.show()


#Print evaluation metrics
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred, zero_division=0)
rec = recall_score(y_test, y_pred, zero_division=0)
f1 = f1_score(y_test, y_pred, zero_division=0)

print(f"Accuracy: {acc:.4f}")
print(f"Precision: {prec:.4f}")
print(f"Recall: {rec:.4f}")
print(f"F1 Score: {f1:.4f}")
```

```
Epoch 1/20
C:\Users\guy74\anaconda3\Lib\site-packages\keras\src\layers\core\input_layer.py:27:
UserWarning: Argument `input_shape` is deprecated. Use `shape` instead.
  warnings.warn(
```

```
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step - accuracy: 0.8706 - loss: 0.3180 - val_a
ccuracy: 0.9289 - val_loss: 0.1782
Epoch 2/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9315 - loss: 0.1713 - val_a
ccuracy: 0.9347 - val_loss: 0.1617
Epoch 3/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9361 - loss: 0.1593 - val_a
ccuracy: 0.9393 - val_loss: 0.1517
Epoch 4/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9397 - loss: 0.1485 - val_a
ccuracy: 0.9415 - val_loss: 0.1447
Epoch 5/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9423 - loss: 0.1413 - val_a
ccuracy: 0.9451 - val_loss: 0.1403
Epoch 6/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9459 - loss: 0.1372 - val_a
ccuracy: 0.9452 - val_loss: 0.1355
Epoch 7/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9463 - loss: 0.1348 - val_a
ccuracy: 0.9475 - val_loss: 0.1321
Epoch 8/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9489 - loss: 0.1287 - val_a
ccuracy: 0.9490 - val_loss: 0.1251
Epoch 9/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9516 - loss: 0.1201 - val_a
ccuracy: 0.9506 - val_loss: 0.1211
Epoch 10/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9517 - loss: 0.1202 - val_a
ccuracy: 0.9521 - val_loss: 0.1195
Epoch 11/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9522 - loss: 0.1163 - val_a
ccuracy: 0.9527 - val_loss: 0.1156
Epoch 12/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9521 - loss: 0.1140 - val_a
ccuracy: 0.9526 - val_loss: 0.1168
Epoch 13/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9532 - loss: 0.1118 - val_a
ccuracy: 0.9532 - val_loss: 0.1133
Epoch 14/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9552 - loss: 0.1092 - val_a
ccuracy: 0.9554 - val_loss: 0.1137
Epoch 15/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9554 - loss: 0.1083 - val_a
ccuracy: 0.9537 - val_loss: 0.1140
Epoch 16/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9552 - loss: 0.1090 - val_a
ccuracy: 0.9550 - val_loss: 0.1097
Epoch 17/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9545 - loss: 0.1066 - val_a
ccuracy: 0.9552 - val_loss: 0.1088
Epoch 18/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9560 - loss: 0.1054 - val_a
ccuracy: 0.9552 - val_loss: 0.1077
Epoch 19/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9563 - loss: 0.1038 - val_a
ccuracy: 0.9548 - val_loss: 0.1077
```

```
Epoch 20/20
2590/2590 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9564 - loss: 0.1042 - val_a
ccuracy: 0.9537 - val_loss: 0.1080
810/810 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
```




```
Accuracy: 0.9524
Precision: 0.9623
Recall: 0.9264
F1 Score: 0.9440
```

In [ ]: