CS5812

Predictive Data Analysis

Student name: Ashkan Sheikhansari

Student ID:2356272

2023/24

Appendix:

```
# Connect to Google drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [3]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

Data Collection

In [4]:

```
# Load the data
train_df = pd.read_csv('/content/drive/My Drive/Airline/train.csv')
test_df = pd.read_csv('/content/drive/My Drive/Airline/test.csv')
data = pd.concat([train_df, test_df])
```

In [5]:

```
# Display basic information for both datasets
data_info = {
    'shape': data.shape,
    'columns': data.dtypes.to_dict()
}

data_info
```

Out[5]:

```
{'shape': (129880, 24),
 'columns': {'id': dtype('int64'),
  'Gender': dtype('O'),
  'Customer Type': dtype('O'),
  'Age': dtype('int64'),
  'Type of Travel': dtype('O'),
  'Class': dtype('O'),
  'Flight Distance': dtype('int64'),
  'Inflight wifi service': dtype('int64'),
  'Departure/Arrival time convenient': dtype('int64'),
  'Ease of Online booking': dtype('int64'),
  'Gate location': dtype('int64'),
  'Food and drink': dtype('int64'),
  'Online boarding': dtype('int64'),
  'Seat comfort': dtype('int64'),
  'Inflight entertainment': dtype('int64'),
  'On-board service': dtype('int64'),
  'Leg room service': dtype('int64'),
  'Baggage handling': dtype('int64'),
  'Checkin service': dtype('int64'),
  'Inflight service': dtype('int64'),
  'Cleanliness': dtype('int64'),
  'Departure Delay in Minutes': dtype('int64'),
  'Arrival Delay in Minutes': dtype('float64'),
  'satisfaction': dtype('O')}}
```

Shape: 129,880 rows and 24 columns

Columns and Data Types: The columns and their data types in the test set are identical to those in the training set.

In [6]:

```
# Extracting specific column information for both datasets
specific_columns_info = {
    'Type of Travel': {
        'data_type': data['Type of Travel'].dtype,
        'description': "Indicates the purpose of the flight from the 
passenger's perspective, e.g., Personal or Business."
    },
    'Flight Distance': {
        'data_type': data['Flight Distance'].dtype,
        'description': "The total distance traversed by the aircraft in Km."
    },
    'Gate location': {
        'data_type': data['Gate location'].dtype,
        'description': "A numerical rating representing the passenger's 
satisfaction with the gate's location."
    },
    'Leg room service': {
        'data_type': data['Leg room service'].dtype,
        'description': "Numerical rating for the satisfaction with the leg 
room service offered during the flight."
    },
    'Arrival Delay in Minutes': {
        'data_type': data['Arrival Delay in Minutes'].dtype,
        'description': "Delay in minutes from the scheduled arrival time of 
the flight, as a continuous numerical value."
    },
    'Satisfaction': {
        'data_type': data['satisfaction'].dtype,
        'description': "Final customer satisfaction status. (Target 
variable)"
```

```
    }
}

specific_columns_info
```

Out[6]:

```
{'Type of Travel': {'data_type': dtype('O'),
  'description': "Indicates the purpose of the flight from the passenger's pe
rspective, e.g., Personal or Business."},
 'Flight Distance': {'data_type': dtype('int64'),
  'description': 'The total distance traversed by the aircraft in Km.'},
 'Gate location': {'data_type': dtype('int64'),
  'description': "A numerical rating representing the passenger's satisfactio
n with the gate's location."},
 'Leg room service': {'data_type': dtype('int64'),
  'description': 'Numerical rating for the satisfaction with the leg room ser
vice offered during the flight.'},
 'Arrival Delay in Minutes': {'data_type': dtype('float64'),
  'description': 'Delay in minutes from the scheduled arrival time of the fli
ght, as a continuous numerical value.'},
 'Satisfaction': {'data_type': dtype('O'),
  'description': 'Final customer satisfaction status. (Target variable)'}}
```

Type of Travel

Data Type: Categorical

Description: Indicates the purpose of the flight from the passenger's perspective. This could include categories such as personal, business, or other types of travel. It reflects whether the journey is primarily for business purposes or personal reasons.

Flight Distance

Data Type: Integer

Description: This feature quantifies the length of each journey taken by passengers. Analyzing this column in conjunction with passenger satisfaction ratings could reveal insights into how distance impacts passenger perceptions and expectations, potentially identifying specific needs or preferences associated with short-haul versus long-haul flights.

Gate Location

Data Type: Integer

Description: A numerical rating that likely represents the passenger's satisfaction with the location of the gate. Although the exact scale is not described, it's reasonable to infer that higher values might indicate more favorable perceptions of the gate's convenience, accessibility, or proximity to related airport services.

Leg Room Service

Data Type: Integer

Description: This is a numerical rating given by passengers regarding the leg room service offered during the flight. Similar to "Gate location," the specifics of the scale are not provided, but higher values presumably denote higher satisfaction with the amount of leg room or the quality of accommodations related to leg space.

Arrival Delay in Minutes

Data Type: Float

Description: Represents the delay in minutes from the scheduled arrival time of the flight. This is a continuous numerical value that quantifies the length of delay experienced upon arrival. The precision of this measurement being a float suggests it can account for partial minutes, offering a detailed view of delays.

Satisfaction

Data Type: Categorical (Binary)

Description: The target variable, reflecting the passengers' overall assessment of their flight experience. It categorizes responses into two principal classes: 'satisfied' and 'not satisfied', encapsulating the comprehensive evaluation of various flight service dimensions. This binary distinction enables us to investigate the multifaceted factors contributing to passenger satisfaction, thus serving as a critical indicator for assessing and enhancing airline service quality.

In [7]:

```
# Preview the first few rows of each dataset to understand their structure
data_head = data.head()
data_head
```

Out[7]:

5 rows × 24 columns

Data Preparation and Cleaning

In [8]:

```
# Checking for NaN values in specified columns
print("Train : \n", data[['Type of Travel', 'Flight Distance', 'Gate
location', 'Leg room service', 'Arrival Delay in Minutes',
'satisfaction']].isna().sum())
```

```
Train :
 Type of Travel              0
Flight Distance             0
Gate location               0
Leg room service            0
Arrival Delay in Minutes    393
satisfaction                0
dtype: int64
```

In [9]:

```
# Filling the missing values
data['Arrival Delay in Minutes'].fillna(data['Arrival Delay in
Minutes'].median(), inplace=True)

# Train data
train_df['Arrival Delay in Minutes'].fillna(train_df['Arrival Delay in
Minutes'].median(), inplace=True)
```

```
# Test data
test_df['Arrival Delay in Minutes'].fillna(test_df['Arrival Delay in
Minutes'].median(), inplace=True)
```

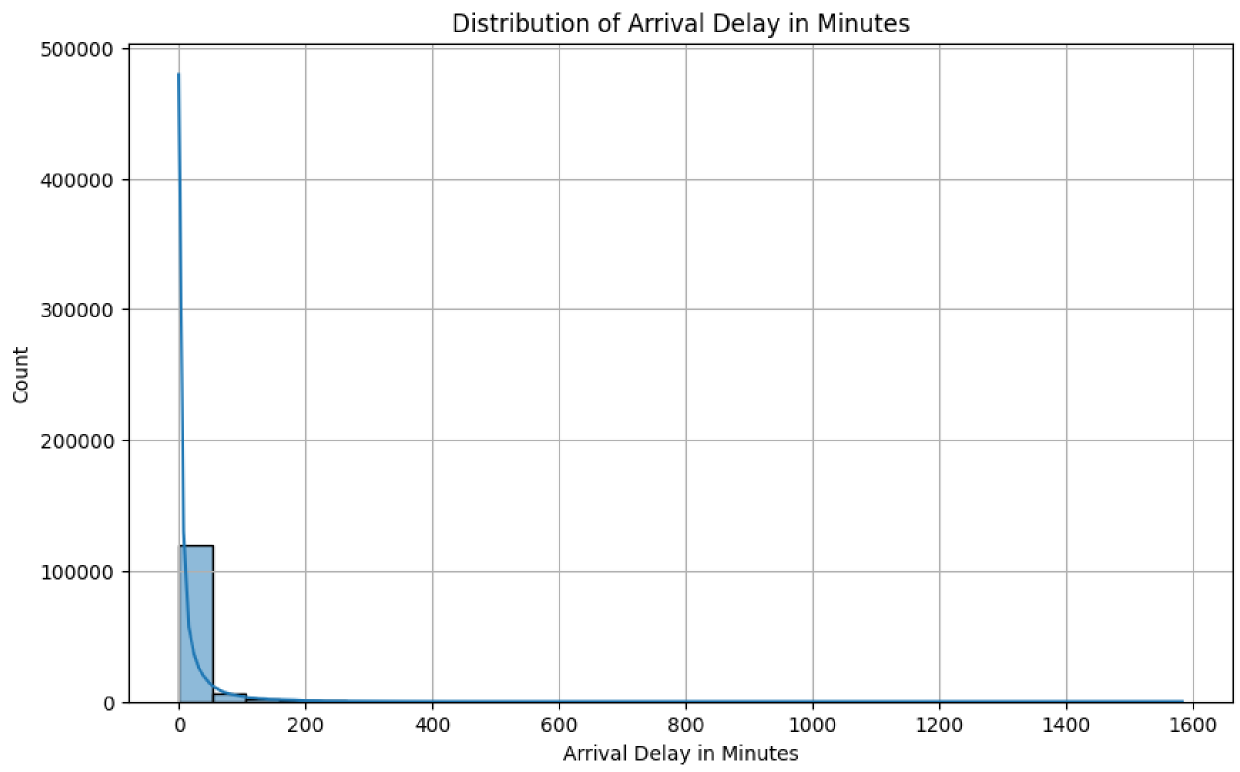
Since we have missing values in Arrival Delay in minutes:

For the numerical column Arrival Delay in Minutes, we imputed NaN values with the median of the column for both training and test datasets.

Exploratory Data Analysis (EDA)

In [10]:

```
# Summary for 'Type of Travel' and 'Leg room service'
type_of_travel_distribution = data["Type of
Travel"].value_counts(normalize=True)
leg_room_service_distribution = data["Leg room
service"].value_counts(normalize=True)

# Summary statistics for 'Arrival Delay in Minutes'
arrival_delay_summary = data["Arrival Delay in Minutes"].describe()

# Plotting the distribution of 'Arrival Delay in Minutes'
plt.figure(figsize=(10, 6))
sns.histplot(data["Arrival Delay in Minutes"], bins=30, kde=True)
plt.title('Distribution of Arrival Delay in Minutes')
plt.grid()
plt.show()
print("Train:", "\n", type_of_travel_distribution,
leg_room_service_distribution, arrival_delay_summary)
```

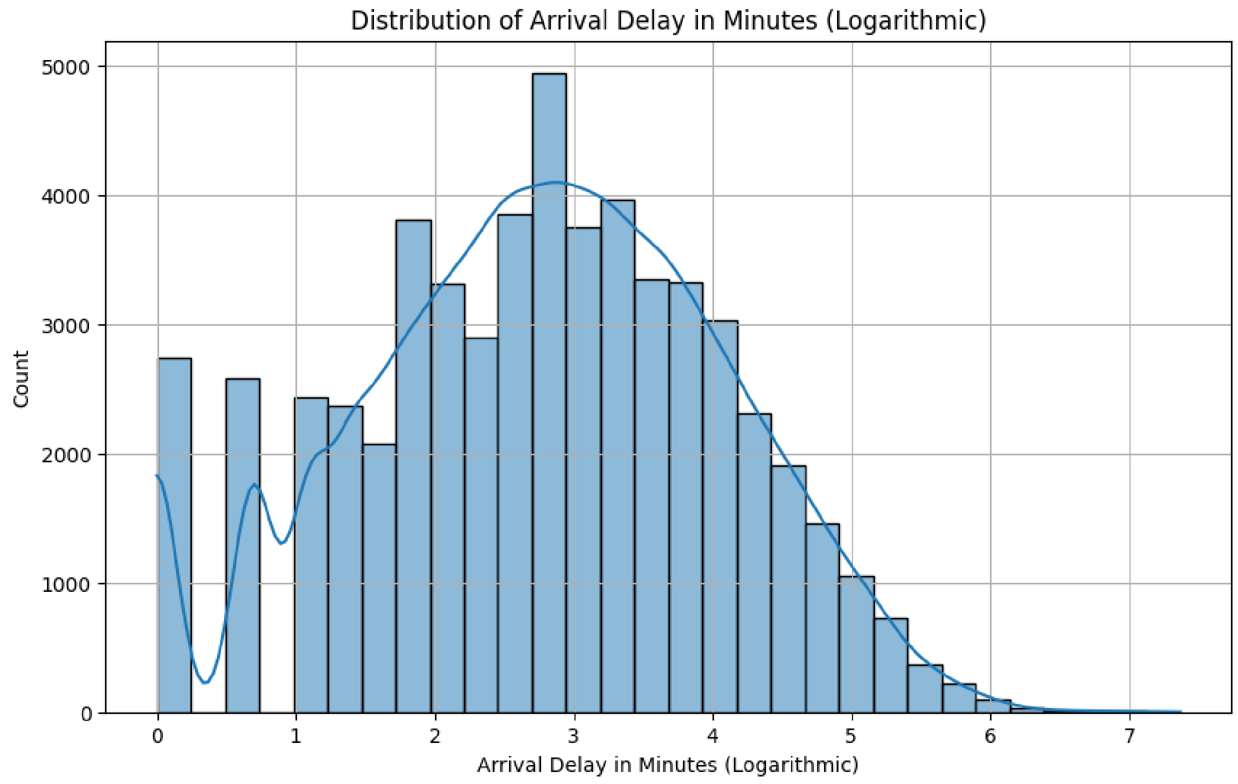


```
Train:
 Type of Travel
Business travel    0.690584
Personal Travel    0.309416
Name: proportion, dtype: float64 Leg room service
4    0.276301
5    0.237950
3    0.192917
2    0.188944
1    0.099284
0    0.004604
Name: proportion, dtype: float64 count    129880.000000
mean         15.045465
std          38.416353
min           0.000000
25%           0.000000
50%           0.000000
75%          13.000000
max        1584.000000
Name: Arrival Delay in Minutes, dtype: float64
```

In [11]:

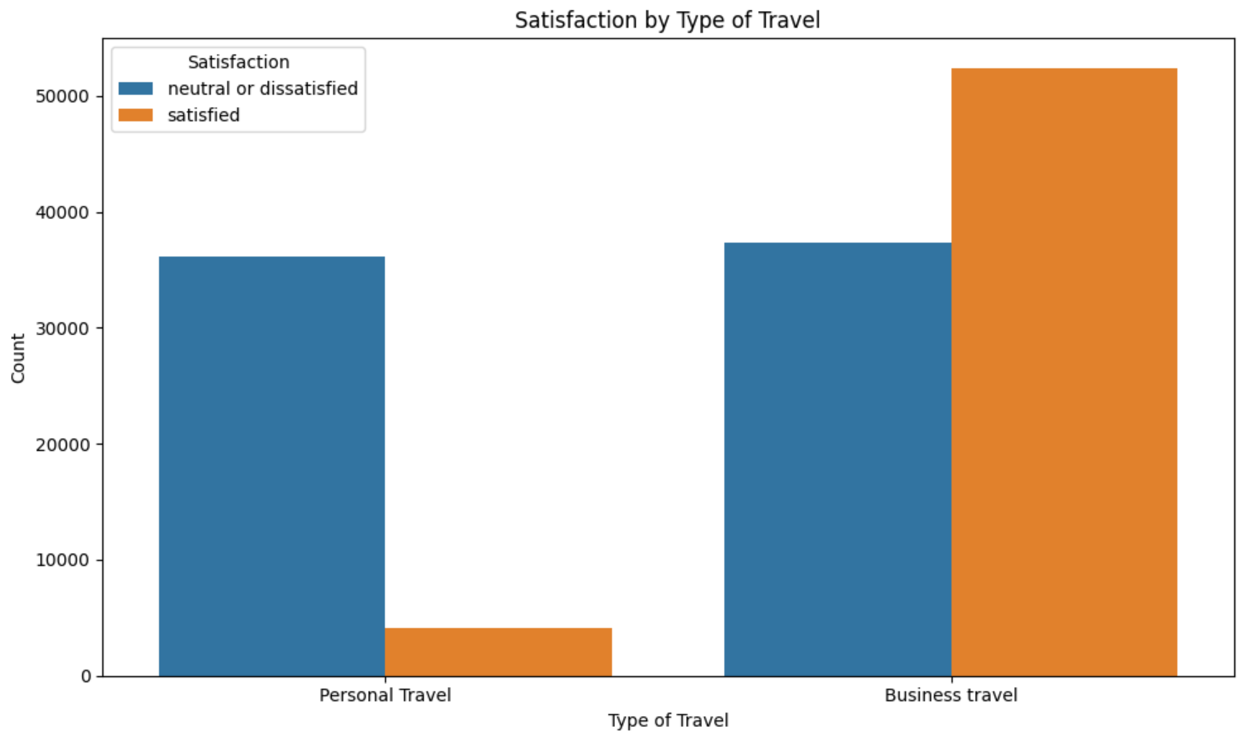
```
# Plotting the logarithmic distribution of 'Arrival Delay in Minutes'
plt.figure(figsize=(10, 6))
sns.histplot(np.log(data["Arrival Delay in Minutes"]), bins=30, kde=True)
plt.title('Distribution of Arrival Delay in Minutes (Logarithmic)')
plt.xlabel('Arrival Delay in Minutes (Logarithmic)')
plt.grid()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:396: Runtime
Warning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



In [12]:

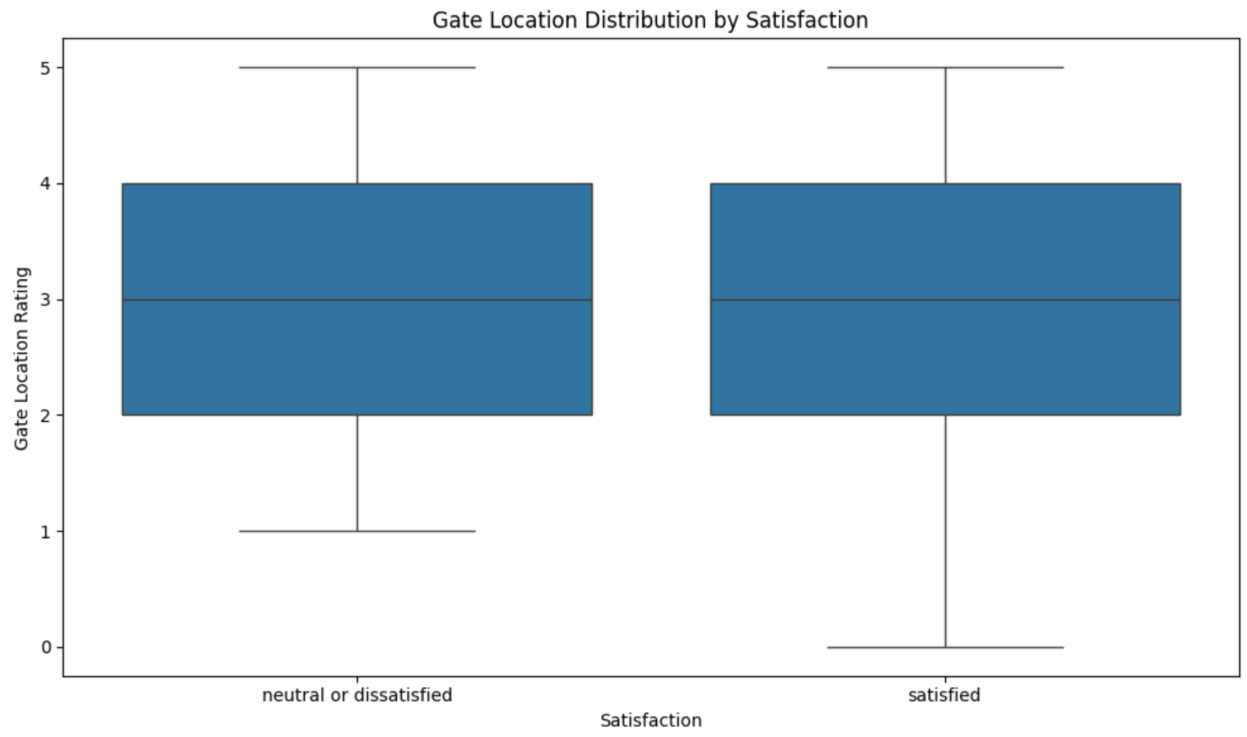
```
# Satisfaction by Type of Travel
plt.figure(figsize=(10, 6))
sns.countplot(x='Type of Travel', hue='satisfaction', data=data)
plt.title('Satisfaction by Type of Travel')
plt.xlabel('Type of Travel')
plt.ylabel('Count')
plt.legend(title='Satisfaction')
plt.tight_layout()
plt.show()
```



This plot reveals the distribution of passenger satisfaction across different types of travel. It indicates how the purpose of travel might influence passengers' overall satisfaction, showing a clear distinction in satisfaction levels between different travel types.

In [13]:

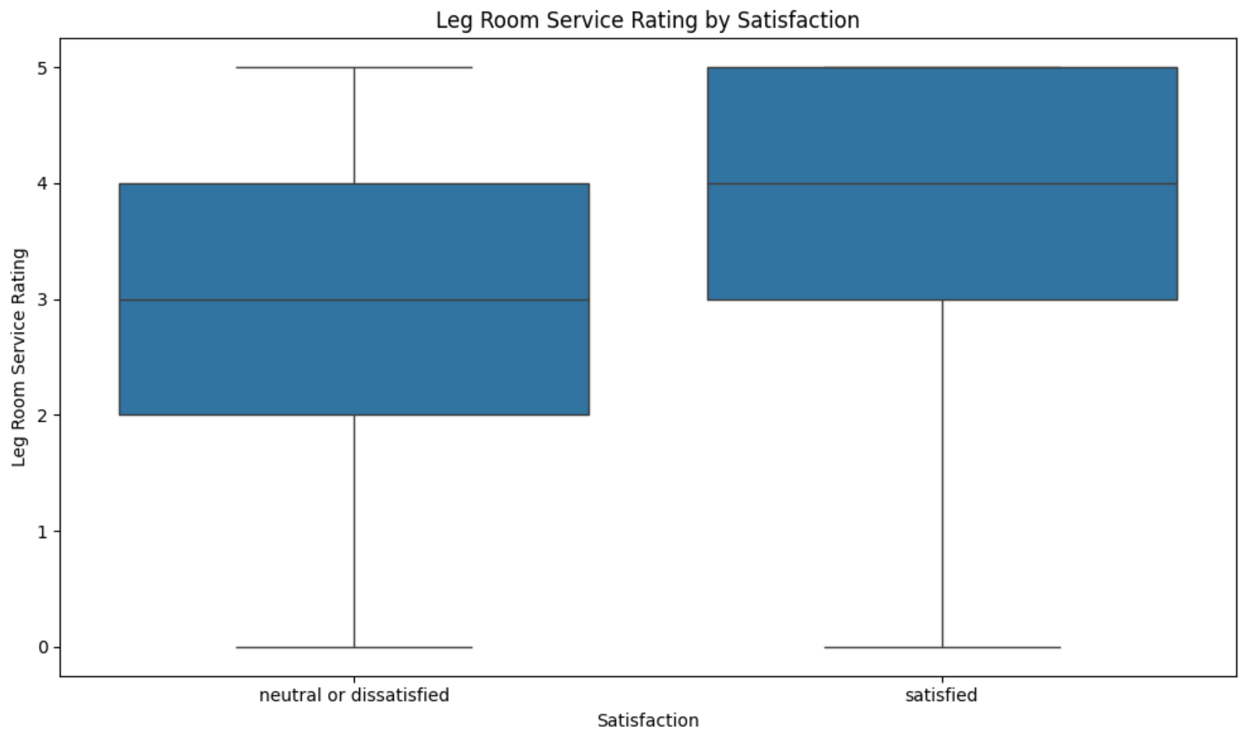
```
# Gate location distribution by satisfaction
plt.figure(figsize=(10, 6))
sns.boxplot(x='satisfaction', y='Gate location', data=data)
plt.title('Gate Location Distribution by Satisfaction')
plt.xlabel('Satisfaction')
plt.ylabel('Gate Location Rating')
plt.tight_layout()
plt.show()
```



The boxplot for gate location ratings by satisfaction status shows the spread and central tendency of gate location satisfaction among satisfied and not satisfied passengers. This could imply how significant the gate's location is to overall satisfaction, with potential insights into whether less satisfied passengers commonly report lower gate location ratings.

In [14]:

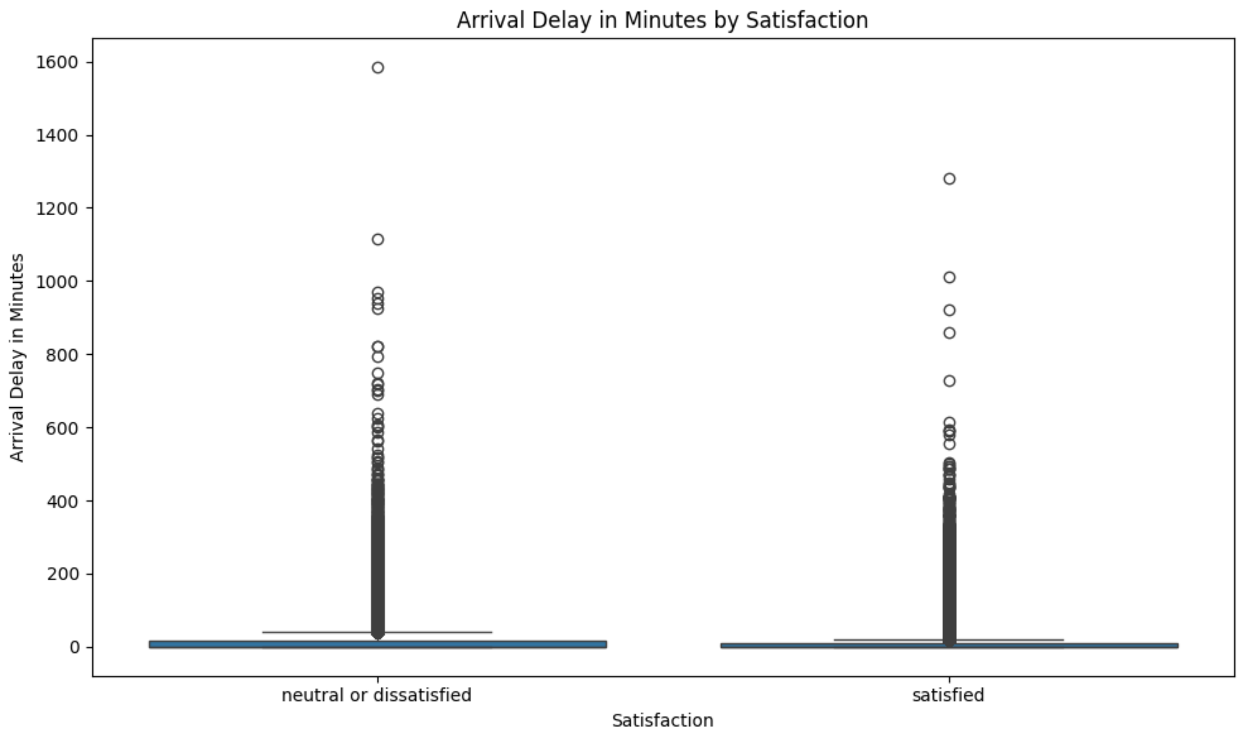
```
# Leg room service rating by satisfaction
plt.figure(figsize=(10, 6))
sns.boxplot(x='satisfaction', y='Leg room service', data=data)
plt.title('Leg Room Service Rating by Satisfaction')
plt.xlabel('Satisfaction')
plt.ylabel('Leg Room Service Rating')
plt.tight_layout()
plt.show()
```



Similar to gate location, the leg room service rating's boxplot categorizes passengers into satisfied and not satisfied groups. It highlights the importance of leg room service in determining overall passenger satisfaction, showing variations in median ratings and spread between the two groups.

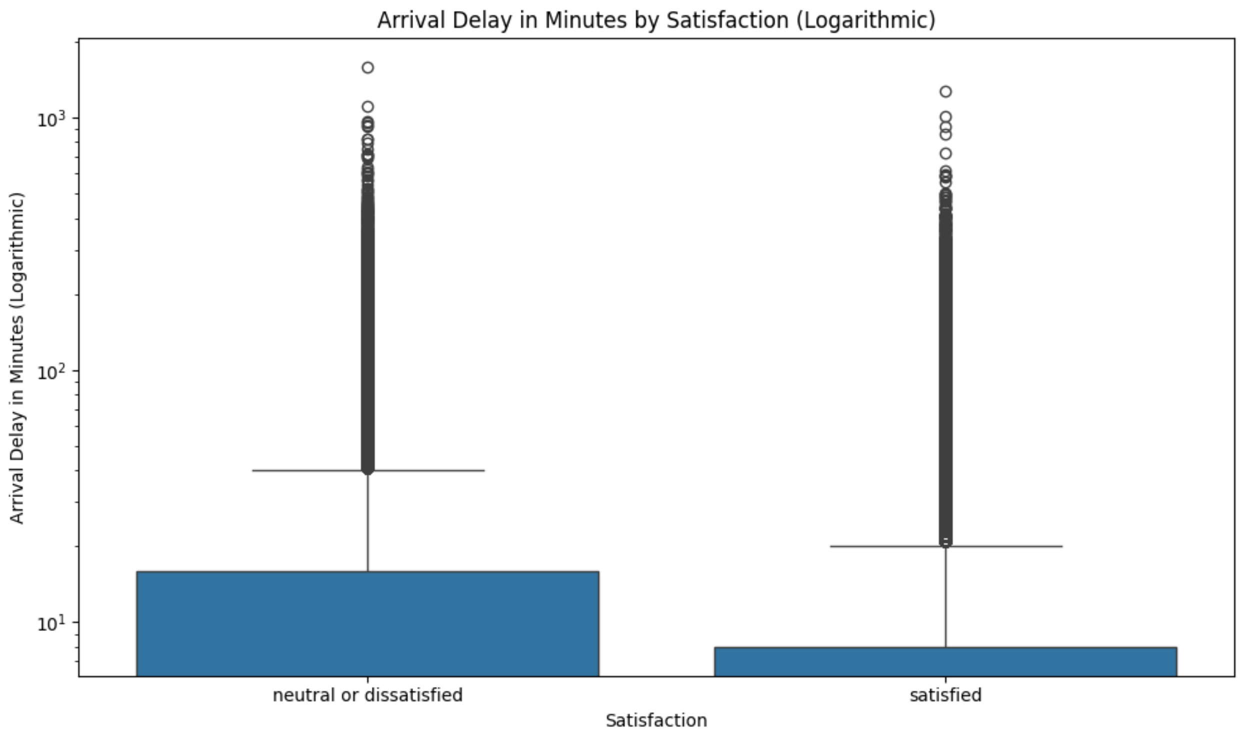
In [15]:

```
# Arrival Delay in Minutes distribution by satisfaction
plt.figure(figsize=(10, 6))
sns.boxplot(x='satisfaction', y='Arrival Delay in Minutes', data=data)
plt.title('Arrival Delay in Minutes by Satisfaction')
plt.xlabel('Satisfaction')
plt.ylabel('Arrival Delay in Minutes')
plt.tight_layout()
plt.show()
```



In [16]:

```
# Logarithmic Arrival Delay in Minutes distribution by satisfaction
plt.figure(figsize=(10, 6))
sns.boxplot(x='satisfaction', y='Arrival Delay in Minutes', data=data)
plt.title('Arrival Delay in Minutes by Satisfaction (Logarithmic)')
plt.xlabel('Satisfaction')
plt.ylabel('Arrival Delay in Minutes (Logarithmic)')
plt.yscale('log')
plt.tight_layout()

plt.show()
```



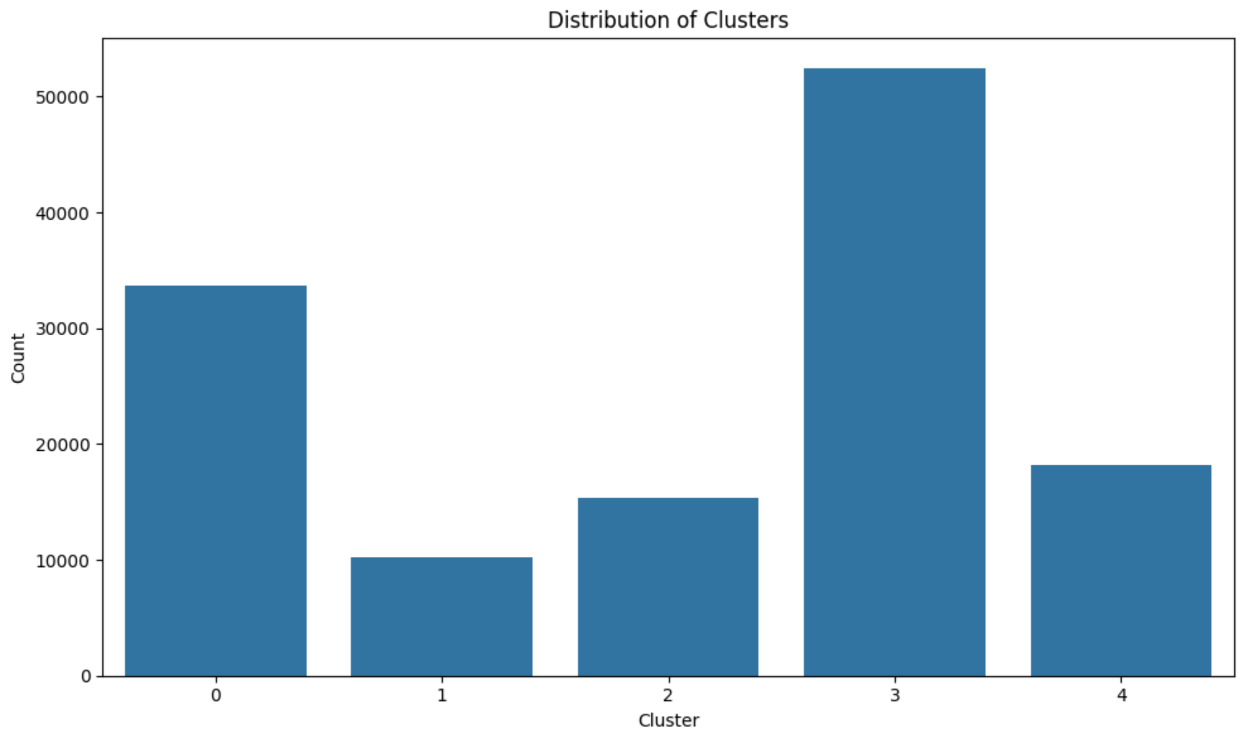
This boxplot compares the arrival delay in minutes against passenger satisfaction. It's crucial for understanding if longer delays correlate strongly with dissatisfaction, indicated by longer arrival delays in the 'not satisfied' group versus the 'satisfied' group. The plot suggests a relationship between delays and satisfaction but also underscores the presence of outliers, indicating that factors other than delay might influence satisfaction.

Clustering:

In [17]:

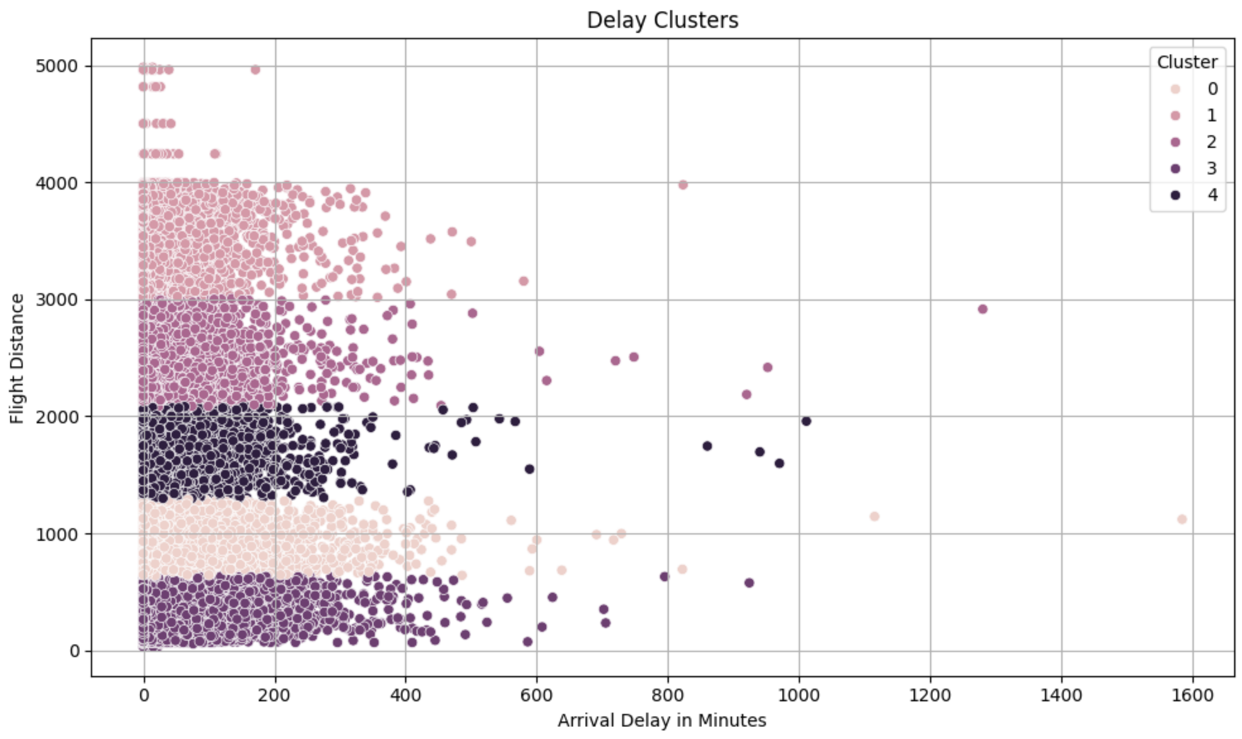
```
# Clustering: Using KMeans for clustering based on the specified features
features_for_clustering = data[['Flight Distance', 'Arrival Delay in
Minutes']]
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(features_for_clustering)

# Adding cluster information to the dataframe
data['Cluster'] = clusters

# Plotting the distribution of clusters
plt.figure(figsize=(10, 6))
sns.countplot(x='Cluster', data=data)
plt.title('Distribution of Clusters')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Futur
eWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```



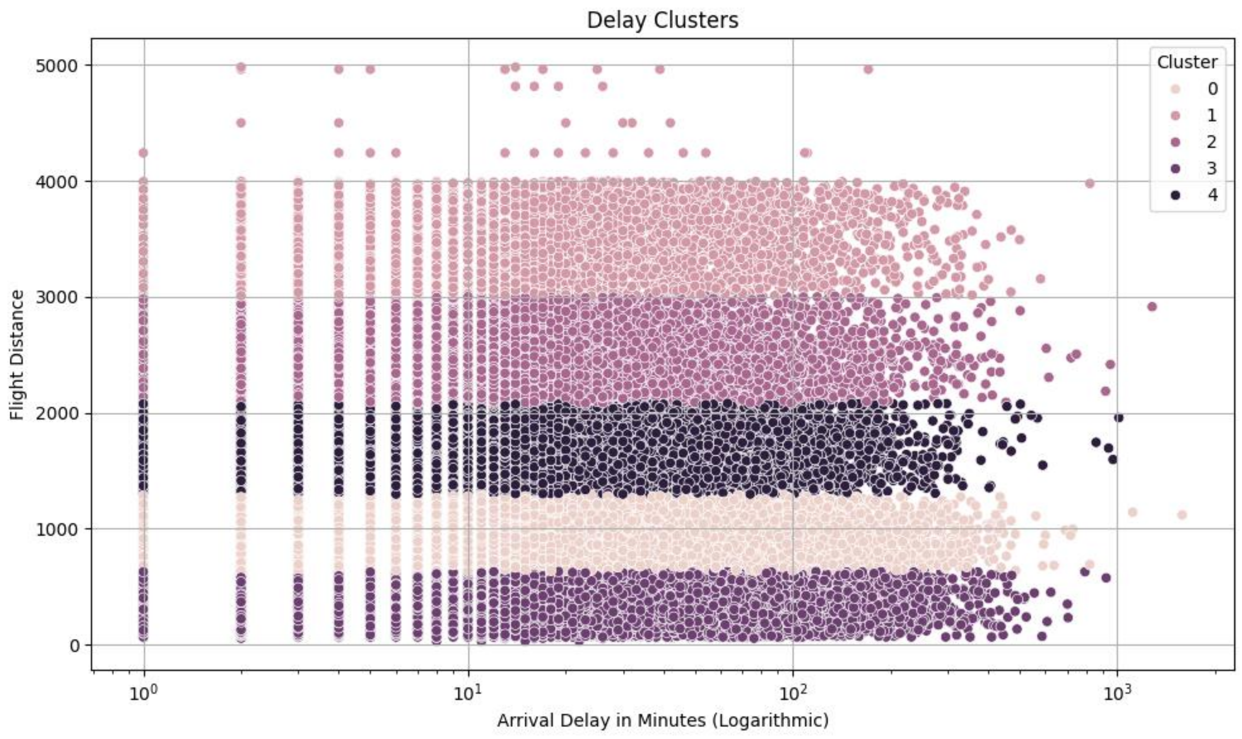
In [18]:

```
# Plotting the delay clusters againts Flight Distance and Arrival Delay in
Minutes
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Arrival Delay in Minutes', y='Flight Distance',
hue='Cluster', data=data)
plt.title('Delay Clusters')
plt.xlabel('Arrival Delay in Minutes')
plt.ylabel('Flight Distance')
plt.grid()
plt.tight_layout()
plt.show()
```



In [19]:

```
# Plotting the delay clusters againts Flight Distance and Logarithmic Arrival
Delay in Minutes
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Arrival Delay in Minutes', y='Flight Distance',
hue='Cluster', data=data)
plt.title('Delay Clusters')
plt.xlabel('Arrival Delay in Minutes (Logarithmic)')
plt.ylabel('Flight Distance')
plt.xscale('log')
plt.grid()
plt.tight_layout()
plt.show()
```



Clustering Distribution By applying KMeans clustering on the specified features, we segmented the data into two clusters. This clustering could reflect underlying patterns or groupings in the data based on travel type, gate location, leg room service, and arrival delay, which may not be explicitly related to satisfaction but show natural groupings of passenger experiences or preferences.

Through these plots and clustering, we've gained valuable insights into factors influencing passenger satisfaction. The visualizations underscore the importance of understanding passenger preferences and experiences across different service aspects, providing a foundation for more detailed analysis or predictive modeling.

Apply the ML and DL

Preprocessing:

In [20]:

```
# Encoding 'satisfaction' to numeric
le = LabelEncoder()
train_df['satisfaction_encoded'] = le.fit_transform(train_df['satisfaction'])

# Encoding 'Type of Travel' to numeric for clustering
train_df['Type of Travel_encoded'] = le.fit_transform(train_df['Type of
Travel'])

le = LabelEncoder()
test_df['satisfaction_encoded'] = le.fit_transform(test_df['satisfaction'])
test_df['Type of Travel_encoded'] = le.fit_transform(test_df['Type of
Travel'])
```

Train:

In [21]:

```
# Preparing the dataset
X_train = train_df[['Type of Travel_encoded', 'Flight Distance', 'Gate
location', 'Leg room service', 'Arrival Delay in Minutes']]
y_train = train_df['satisfaction_encoded']

X_test = test_df[['Type of Travel_encoded', 'Flight Distance', 'Gate
location', 'Leg room service', 'Arrival Delay in Minutes']]
y_test = test_df['satisfaction_encoded']

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

# Logistic Regression
lr_model = LogisticRegression(random_state=42)
lr_model.fit(X_train_scaled, y_train)
y_pred_lr = lr_model.predict(X_test_scaled)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))

# Decision Tree
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)  # No need to scale for Decision Trees
y_pred_dt = dt_model.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))

# MLPClassifier
mlp_model = MLPClassifier(random_state=42, max_iter=300)
mlp_model.fit(X_train_scaled, y_train)
y_pred_mlp = mlp_model.predict(X_test_scaled)
print("MLPClassifier Accuracy:", accuracy_score(y_test, y_pred_mlp))
```

```
Logistic Regression Accuracy: 0.7527332922697875
Decision Tree Accuracy: 0.6885971666153372
MLPClassifier Accuracy: 0.7620110871573761
```

1. Logistic Regression

Justification: Logistic Regression is a straightforward and efficient model for binary classification tasks. It works well with linearly separable data and is highly interpretable, making it a good baseline model. It can serve as a benchmark to evaluate the complexity and performance of more sophisticated models.

2. Decision Tree

Justification: Decision Trees are versatile models that can handle both numerical and categorical data. They are useful for their interpretability, as they provide clear insight into how decisions are made, making it easier to understand the importance and impact of different features on the prediction. Decision Trees can capture non-linear relationships between features and the target variable.

3. MLPClassifier (Multi-layer Perceptron Classifier)

Justification: MLPClassifier, a type of neural network, is suitable for complex datasets with non-linear relationships. It can model intricate patterns through its hidden layers, making it powerful for tasks where Logistic Regression and Decision Trees might fall short. However, it requires careful tuning of parameters and can be less interpretable.

In [25]:

```
# Performance Evaluation
performance = {}

# Logistic Regression Performance
performance['Logistic Regression'] = classification_report(y_test, y_pred_lr,
target_names=le.classes_, output_dict=True)

# Decision Tree Performance
performance['Decision Tree'] = classification_report(y_test, y_pred_dt,
target_names=le.classes_, output_dict=True)

# MLPClassifier Performance
performance['MLPClassifier'] = classification_report(y_test, y_pred_mlp,
target_names=le.classes_, output_dict=True)

print('Logistic Regression: ', '\n', performance['Logistic Regression'],
'\n', 'Decision Tree', '\n' ,performance['Decision Tree'], '\n',
'MLPClassifier:', '\n',performance['MLPClassifier'])
```

```
Logistic Regression: 
 {'Business travel': {'precision': 0.792870490153802, 'recall': 0.75701640019
21361, 'f1-score': 0.7745287324042545, 'support': 14573}, 'Personal Travel':
{'precision': 0.706433427292323, 'recall': 0.7472594931158467, 'f1-score': 0.
7262731728105689, 'support': 11403}, 'accuracy': 0.7527332922697875, 'macro a
vg': {'precision': 0.7496519587230625, 'recall': 0.7521379466539915, 'f1-scor
e': 0.7504009526074117, 'support': 25976}, 'weighted avg': {'precision': 0.75
492616355196, 'recall': 0.7527332922697875, 'f1-score': 0.7533454037144331, '
support': 25976}}
 Decision Tree
 {'Business travel': {'precision': 0.711812361165556, 'recall': 0.74761545323
54354, 'f1-score': 0.7292747414572108, 'support': 14573}, 'Personal Travel':
{'precision': 0.6552952202436738, 'recall': 0.6131719722879944, 'f1-score': 0
.6335341820323472, 'support': 11403}, 'accuracy': 0.6885971666153372, 'macro
avg': {'precision': 0.683553790704615, 'recall': 0.6803937127617149, 'f1-scor
e': 0.681404461744779, 'support': 25976}, 'weighted avg': {'precision': 0.687
0023458463296, 'recall': 0.6885971666153372, 'f1-score': 0.6872463460490755,
'support': 25976}}
 MLPClassifier:
 {'Business travel': {'precision': 0.7788263441217519, 'recall': 0.8041583750
771976, 'f1-score': 0.7912896691424713, 'support': 14573}, 'Personal Travel':
{'precision': 0.738859913990301, 'recall': 0.7081469788652109, 'f1-score': 0.
7231775031345155, 'support': 11403}, 'accuracy': 0.7620110871573761, 'macro a
vg': {'precision': 0.7588431290560265, 'recall': 0.7561526769712043, 'f1-scor
e': 0.7572335861384933, 'support': 25976}, 'weighted avg': {'precision': 0.76
1281795200096, 'recall': 0.7620110871573761, 'f1-score': 0.7613896449282459,
'support': 25976}}
```