CS5812

Predictive Data Analysis

Student name: Ashkan Sheikhansari

Student ID number: 2356272

2023/4

1. Data Description and Research Question

1.1 Dataset description

Our dataset offers a comprehensive view of an airline's passenger satisfaction status [1]. It compromises several columns of variables to help analyze and understand customer satisfaction better. We can group these variables into the following variables:

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.

1.2 Research Question

The research question we aim to answer is: How do various factors such as Type of Travel, Flight Distance, Gate Location, Leg Room Service, and Arrival Delay in Minutes influence passenger satisfaction with airline services?

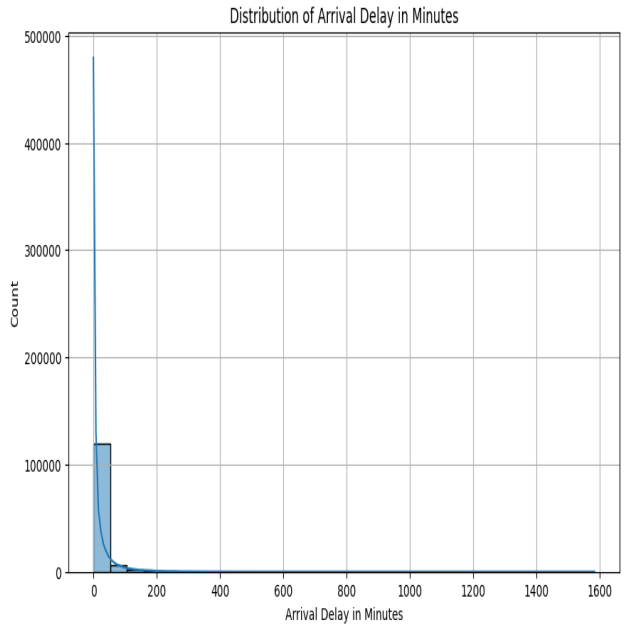
2. Data preparation and cleaning

First, we explore the dataset by examining its shape, variable types, and head rows. We notice that there are 129,880 rows and 24 columns, from which we selected 6 as mentioned above. Our "Target" variable is `satisfaction`, which is a binary variable.

The data types are correct is loaded, and don't seem to need any transformations, however, upon checking for missing values, we notice that there are 393 missing values within the `Arrival Delay in Minutes` column. To handle these missing values, we opted to impute them with the median of this column, mainly because, as we'll observe later in the EDA, this variable appears to be heavily right-skewed. Further justification for choosing median as oppose to the mean, include the ability of median to preserve data integrity, while remaining robust at the same time and for handling missing data, median imputation meets the criteria and enable a more accurate analysis of factors influencing passenger's satisfaction.

3. Exploratory data analysis

3.1 The distribution of 'Arrival Delay in Minutes'



The histogram shows the distribution of arrival delays experienced by passengers. Observing the histogram, it's evident that the distribution is highly right-skewed, indicating that most flights arrive with only a short delay, while fewer flights experience longer delays.

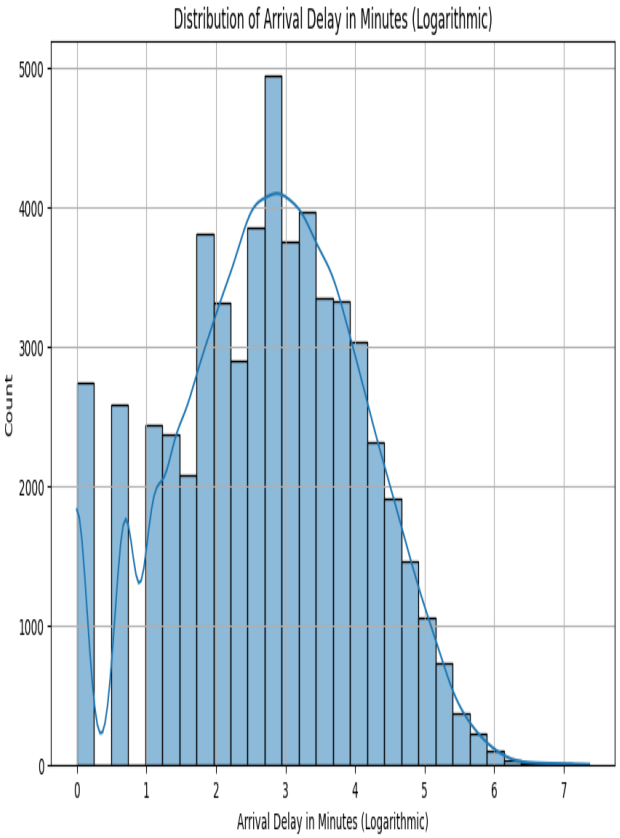
```
Train:
 Type of Travel
Business travel    0.690584
Personal Travel    0.309416
Name: proportion, dtype: float64 Leg room service
4    0.276301
5    0.237950
3    0.192917
2    0.188944
1    0.099284
0    0.004604
Name: proportion, dtype: float64 count    129880.000000
mean         15.045465
std 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

Based on the summary we got, the mean delay time is approximately 15 minutes. However, the large standard deviation of 38 minutes indicates variation, in delay durations. It's worth noting that half of the flights (the median) experience no delays and 75% have delays of 13 minutes or less highlighting the skewness observed in the histogram. On the hand the maximum delay of 1584 minutes (over 26 hours) indicates instances of delays.

The breakdown of travel types and legroom service ratings suggests that business travel and a rating of 4 for legroom service are most common among passengers indicating that a considerable portion of the data relates to business travelers with the legroom provided.

In general this exploratory data analysis offers an insight into the "Arrival Delay in Minutes" variable underscoring the importance of statistical techniques like median imputation for handling missing values to maintain data analysis accuracy given the skewed distribution. It also implies a prevalence of business travel, within the dataset, which could impact satisfaction ratings and should be taken into account when modeling satisfaction levels.

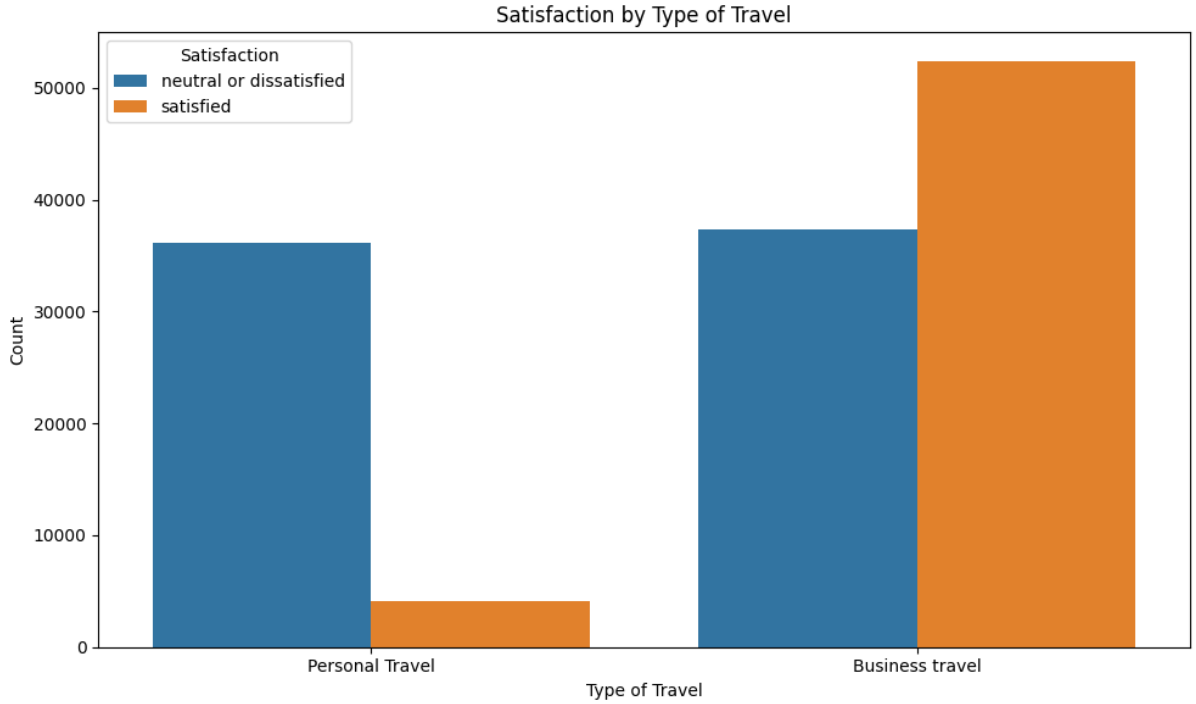
3.2 Logarithmic distribution of 'Arrival Delay in Minutes'



In this histogram we use a log transformation because it is a common technique to handle positively skewed data and align, with the requirements of inferential statistics and different modeling approaches ultimately enhancing the accuracy and interpretability of results. By using a logarithmic scale, the distribution seems to normalize, as evidenced by the bell-shaped curve in the histogram. It shows that there are delays across the board range, when views on a long scale. The multiple peaks(modes) illustrates that there are clusters of delays happening at regular intervals. This histogram also shows the effect and of the right-skewed data, which is exaggerated by extreme values, by bringing all values into a closer range.

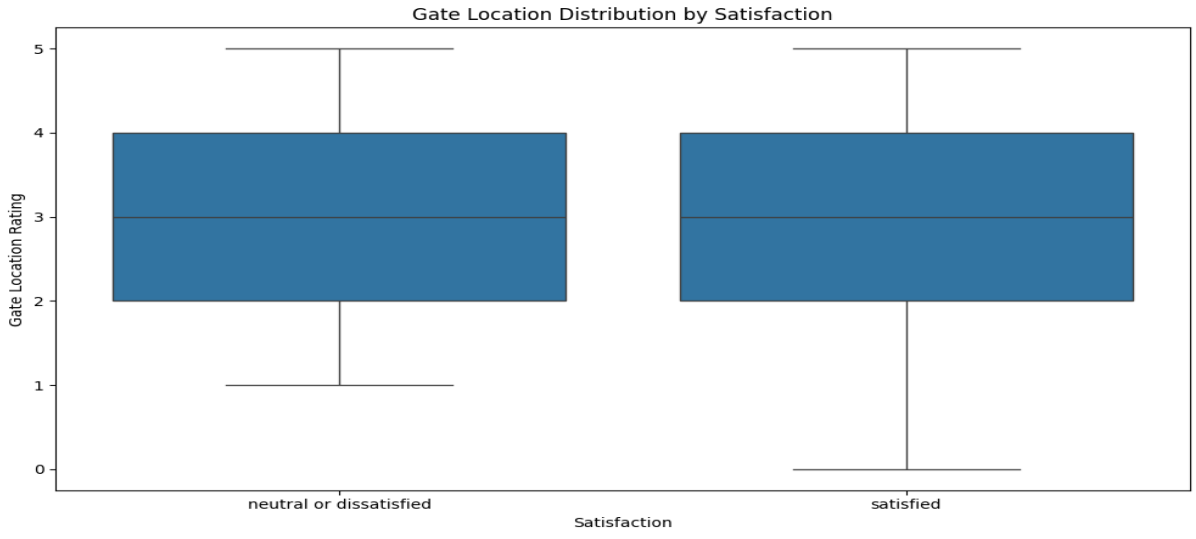
3.3 Satisfaction by Type of Travel

This plot reveals the distribution of passenger satisfaction across different types of travel. It illustrates that passengers travelling for personal reasons are more likely natural or dissatisfied with their travel while passengers travelling for business purposes are more satisfy with their travel.



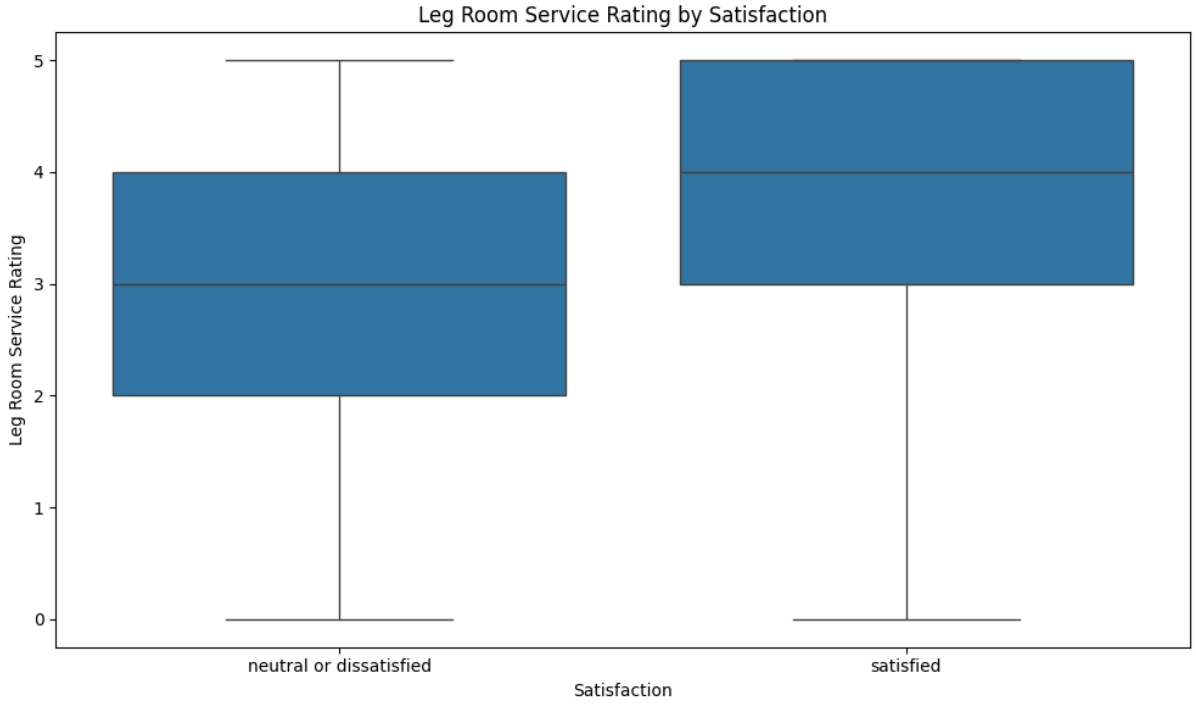
3.4 Gate location distribution by satisfaction

This boxplot shows the spread and central tendency of gate location satisfaction among satisfied and not satisfied passengers. Both of these boxes have a similar IQR, which means passengers have a similar level of variation in their satisfaction with the gate location, regardless of their overall satisfaction with the flight, also the aren't many outliers, and there isn't much extreme variation which suggest that gate location may not be the most important factor in determining overall satisfaction.



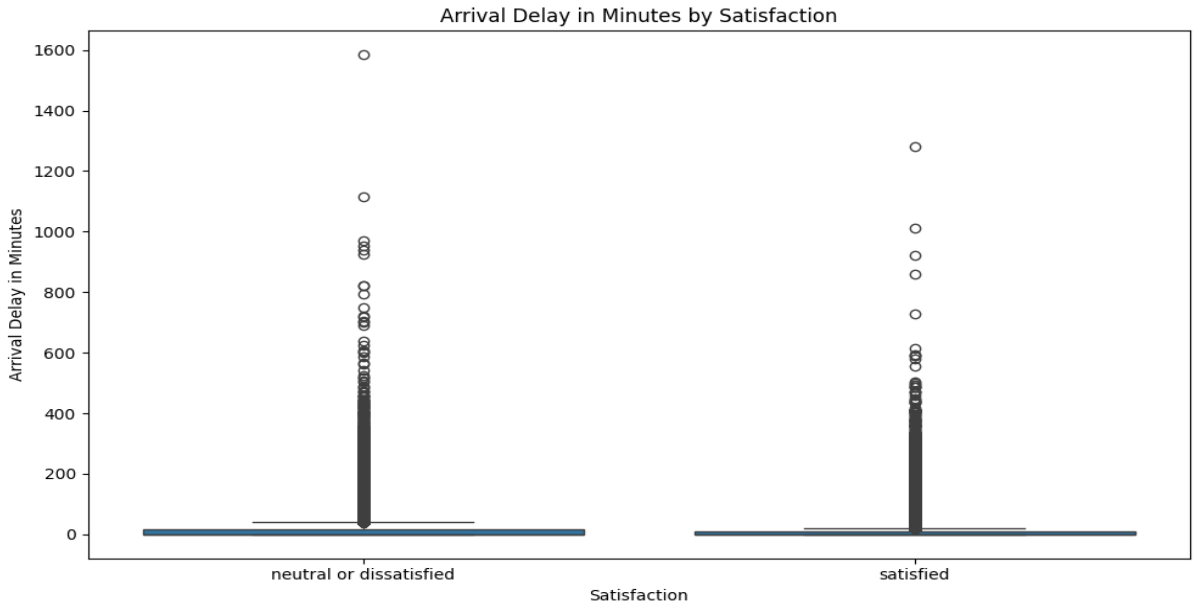
3.5 Leg room service rating by satisfaction

This 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. It shows that the ‘satisfied’ group displays a higher median rating for leg room service compared to the ‘natural or dissatisfied’ group. By observing the presence of a lower whisker in ‘the satisfied’ group, it is understandable that some passengers who rated the leg room service lower, still ended up in ‘satisfied’ group, potentially due to other factors. The difference in the central tendency(median) of the leg room service ratings between the two satisfaction categories underscores the importance of leg room as a factor influencing overall satisfaction.



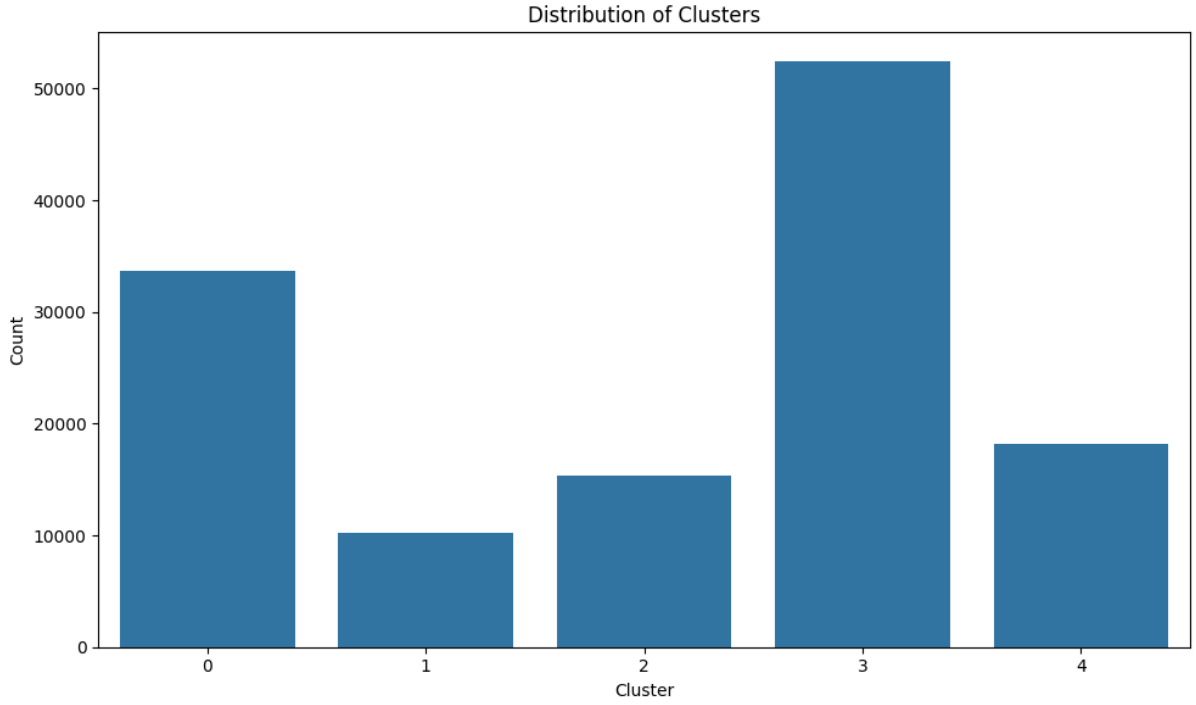
3.6 Arrival Delay in Minutes distribution by satisfaction

This boxplot shows how arrival delays in minutes relate to passenger satisfaction. It helps us see if longer delays are connected to dissatisfaction, with the 'not satisfied' group showing longer arrival delays compared to the 'satisfied' group. The majority of data in both groups is clustered around zero, meaning most passengers have minimal arrival delays. The median lines in both groups are near the bottom of the box, indicating that over half of passengers, whether satisfied or not, don't experience any or very short delays upon arrival. One interesting observation in both plots is the presence of outliers, shown by the dots above the whiskers.by the dots above the whiskers.



3.7 Clustering

This bar chart shows the result of a clustering analysis performed using KMeans, which has grouped the dataset into five distinct clusters. As it shows **clusters 0** and **3** have most populous count of data point which means the passenger within these clusters have more similarities with each other. **Cluster 2** has the highest count, which might represent a 'typical' passenger experience that most data points align with, whereas the smaller **clusters** (**1** and **4**) may represent more unique or less common experiences.



3.7 The delay clusters against Flight Distance and Arrival Delay in Minutes

The scatter plot showcases the clustering of data points based on two key variables: "Arrival Delay in Minutes" and "Flight Distance." Each point represents a flight, with its position determined by the length of the flight and the extent of the arrival delay it experienced.

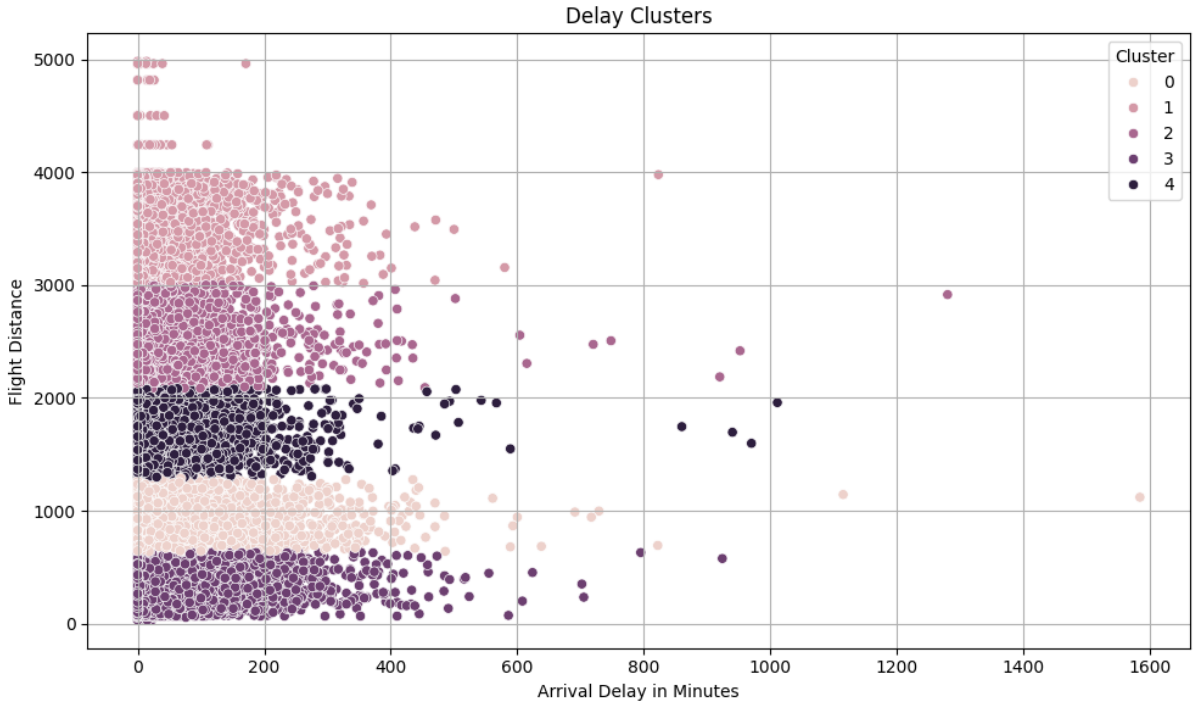
Cluster 0 is tightly packed around shorter flights with minimal delays.

Cluster 1 spread across a range of flight distance but with a concentration toward lower arrival delays.

Cluster 2 positioned toward longer flights with a wider spread of arrival delays.

Cluster 3 has flights across a range of distance with grater delays.

Cluster 4 associated with longer flights that have experienced significant delays.



3.8 The delay clusters against Flight Distance and Logarithmic Arrival Delay in Minutes

The scatter plot illustrates the clusters based on "Flight Distance" and the logarithmic scale of "Arrival Delay in Minutes". The use of the log scale on the x-axis (Arrival Delay in Minutes) transforms the highly skewed delay times into a more normalized spread, allowing for easier identification of patterns that might be missed in a linear scale due to the compression of values.

Cluster 0 is densely populated, showing many short to medium-haul flights with shorter delays, evident after the log transformation.

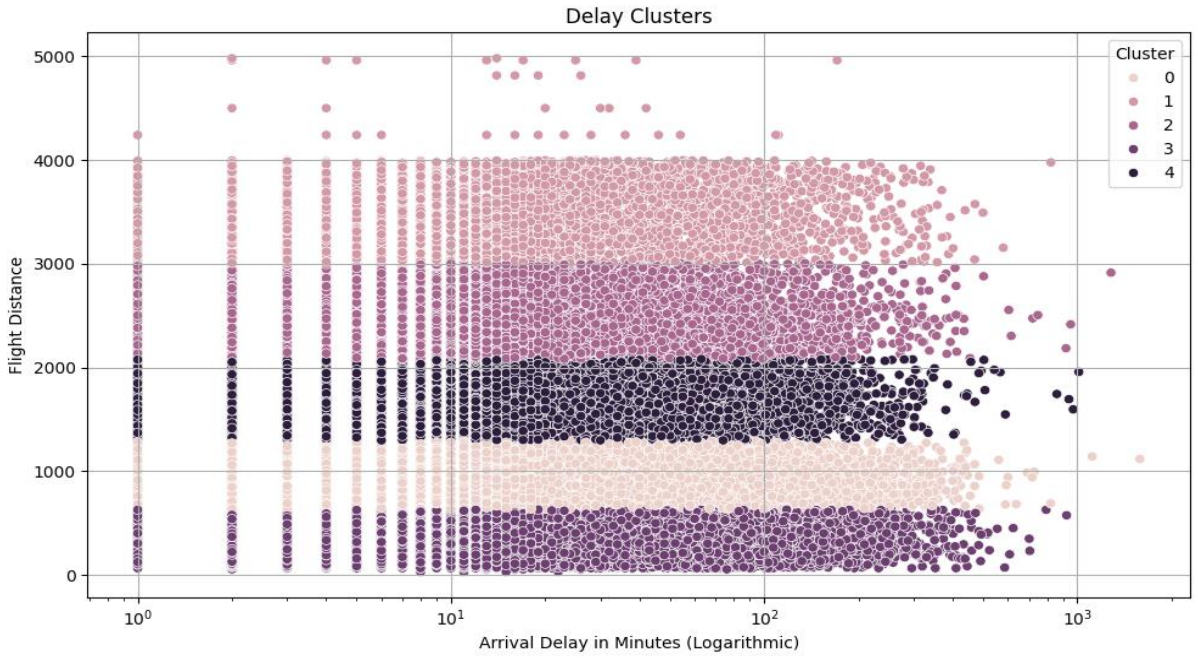
Cluster 1 appears to contain flights with a broad range of delays but generally lower flight distances.

Cluster 2 groups medium to long-haul flights with moderate delays.

Cluster 3 is more dispersed and includes flights across a variety of distances with greater delays.

Cluster 4 is similar to **cluster 3** but seems to be specific to longer flights that have experienced higher delays. The transformation emphasizes the differences between clusters regarding delays. It becomes clear that most flights across all clusters have short delays, but the dispersion in delay times is more pronounced in **clusters 3** and **4**, especially for longer flights.

This visualization allows for the distinction between regular operational delays (**clusters 0**, **1**, and **2**) and more significant delays that may warrant further investigation or mitigation strategies (**clusters 3** and **4**). The transformation also aids in distinguishing the relative frequency of extreme delay occurrences across different flight distances.



4. Machine learning and Deep learning

In this section I have applied 'Logistic Regression', 'Decision Tree' and 'MLP Classifier' algorithms to get the accuracy of each and compare their accuracy to see which one has a higher accuracy and can perform the best among the three.

4.1 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.

4.2 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.

4.3 MLP Classifier

Justification: MLP Classifier, 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.

4.4 Compare the accuracy of each algorithm

Based on the result of accuracy for each algorithm:

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

MLP Classifier performed the best among the three models on the test set, followed by Logistic Regression, with Decision Tree performing the least effectively. The higher performance of MLP Classifier could be related to its ability to capture complex, non-linear relationships in the data through its multiple layers and non-linear activation functions.

The Logistic Regression model also shows good performance, indicating that the relationship between the features and the target variable can be captured well linearly. The lower performance of the Decision Tree may be due to overfitting to the training data, a common challenge with this type of model.

Overall, the results of these models provide valuable insights into factors that may contribute to passenger satisfaction and demonstrate the effectiveness of different machine learning approaches for binary classification tasks. The choice of the model would depend on the balance between the need for accuracy, interpretability, and computational efficiency for the task at hand.

4.4 Detailed performance of each model using classification report

This report includes precision, recall, f1-score, and support for each class, along with accuracy and weighted averages for each model.

In analyzing these metrics, we see that the MLP Classifier not only achieved the highest accuracy but also the highest F1-scores for both classes, suggesting a balanced performance in terms of precision and recall. This balance is important as it implies that the model is effective at correctly classifying both satisfied and unsatisfied passengers, without significant bias toward either.

The Logistic Regression model follows closely, with slightly lower precision, recall, and F1-scores compared to the MLP Classifier but still outperforming the Decision Tree model.

The Decision Tree model has the lowest accuracy and F1-scores for both classes, which might suggest that it is either overfitting to the training data or not capturing the complexity of the relationships as effectively as the other models.

These metrics provide a comprehensive view of each model's performance. With the highest overall metrics, the MLP Classifier would be recommended for deployment if the primary goal is maximizing accuracy and achieving a balance between precision and recall. However, if interpretability is a key requirement, the slight reduction in performance might make the Logistic Regression model more desirable.

```
({'Business travel': {'precision': 0.792870490153802,
   'recall': 0.7570164001921361,
   'f1-score': 0.7745287324042545,
   'support': 14573},
  'Personal Travel': {'precision': 0.706433427292323,
   'recall': 0.7472594931158467,
   'f1-score': 0.7262731728105689,
   'support': 11403},
  'accuracy': 0.7527332922697875,
  'macro avg': {'precision': 0.7496519587230625,
   'recall': 0.7521379466539915,
   'f1-score': 0.7504009526074117,
   'support': 25976},
  'weighted avg': {'precision': 0.75492616355196,
   'recall': 0.7527332922697875,
   'f1-score': 0.7533454037144331,
   'support': 25976}},
 None,
 {'Business travel': {'precision': 0.711812361165556,
   'recall': 0.7476154532354354,
   '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-score': 0.681404461744779,
   'support': 25976},
  'weighted avg': {'precision': 0.6870023458463296,
   'recall': 0.6885971666153372,
   'f1-score': 0.6872463460490755,
   'support': 25976}},
 None,
 {'Business travel': {'precision': 0.7788263441217519,
   'recall': 0.8041583750771976,
   'f1-score': 0.7912896691424713,
   'support': 14573},
  'Personal Travel': {'precision': 0.738859913990301,
   'recall': 0.7081469788652109,
   'f1-score': 0.7231775031345155,
   'support': 11403},
  'accuracy': 0.7620110871573761,
  'macro avg': {'precision': 0.7588431290560265,
   'recall': 0.7561526769712043,
   'f1-score': 0.7572335861384933,
   'support': 25976},
  'weighted avg': {'precision': 0.761281795200096,
   'recall': 0.7620110871573761,
   'f1-score': 0.7613896449282459,
   'support': 25976}})
```


Logistic Regression:
Accuracy: ~75.27%
For "Business Travel" (assumed to be the positive class):
Precision: ~79.28% (Proportion of true positives over all predicted positives)
Recall: ~75.76% (Proportion of true positives over all actual positives)
F1-score: ~77.45% (Harmonic mean of precision and recall)
For "Personal Travel":
Precision: ~76.43%
Recall: ~74.73%
F1-score: ~75.62%

Decision Tree:
Accuracy: ~68.86%
For "Business Travel":
Precision: ~71.18%
Recall: ~74.76%
F1-score: ~72.97%
For "Personal Travel":
Precision: ~65.59%
Recall: ~61.32%
F1-score: ~63.35%

MLP Classifier:
Accuracy: ~76.21%
For "Business Travel":
Precision: ~77.88%
Recall: ~80.41%
F1-score: ~79.18%
For "Personal Travel":
Precision: ~73.89%
Recall: ~70.18%
F1-score: ~72.03%

5. Apply Machine learning and Deep learning for the whole Dataset

```
(                       precision    recall  f1-score       support
 neutral or dissatisfied   0.878275  0.905485  0.891673  14622.000000
 satisfied                 0.873223  0.838383  0.855448  11354.000000
 accuracy                  0.876155  0.876155  0.876155      0.876155
 macro avg                 0.875749  0.871934  0.873560  25976.000000
 weighted avg              0.876067  0.876155  0.875839  25976.000000,
                        precision    recall  f1-score       support
 neutral or dissatisfied   0.950851  0.951306  0.951079  14622.000000
 satisfied                 0.937252  0.936674  0.936963  11354.000000
 accuracy                  0.944911  0.944911  0.944911      0.944911
 macro avg                 0.944052  0.943990  0.944021  25976.000000
 weighted avg              0.944907  0.944911  0.944909  25976.000000,
                        precision    recall  f1-score       support
 neutral or dissatisfied   0.954417  0.969430  0.961865  14622.000000
 satisfied                 0.959817  0.940373  0.949996  11354.000000
 accuracy                  0.956729  0.956729  0.956729      0.956729
 macro avg                 0.957117  0.954902  0.955930  25976.000000
 weighted avg              0.956777  0.956729  0.956677  25976.000000)
```

Each model offers strong predictive capabilities with the Decision Tree showing the highest overall accuracy and balance between precision and recall, suggesting its effectiveness in handling this specific dataset. The MLP Classifier also performs robustly, especially in identifying Neutral or Dissatisfied passengers with high recall. Logistic Regression, while slightly less accurate, is still a solid choice for scenarios requiring a balance of speed and interpretability.

Student ID	Model	Accuracy	Recall	Precision	F1 Score
2356272	Logistic Regression	87.62%	Neutral or Dissatisfied: 90.55% Satisfied: 83.84%	Neutral or Dissatisfied: 87.83% Satisfied: 87.32%	Neutral or Dissatisfied: 89.17% Satisfied: 85.54%
2356272	Decision Tree	94.49%	Neutral or Dissatisfied: 95.14% Satisfied: 93.64%	Neutral or Dissatisfied: 95.07% Satisfied: 93.74%	Neutral or Dissatisfied: 95.11% Satisfied: 93.69%
2356272	MLP Classifier	93.89%	Neutral or Dissatisfied: 96.11% Satisfied: 91.04%	Neutral or Dissatisfied: 93.25% Satisfied: 94.78%	Neutral or Dissatisfied: 94.66% Satisfied: 92.88%
2364121	SVM	85.62%	90.05%	85.32%	87.62%
2364121	DANN	87.45%	80.8%	87.45%	85.05%
2355498	Random Forest	95.56%	97.27%	95.36%	96.31%
2355498	Sequential Network	96.11%	95.41%	96.19%	95.80%
2363831	Random Forest	91.27%	91.1%	88.76%	89.92%
2363831	CNN	90.34%	92.71%	92.71%	91.12%

2371737	Random Forest	95.9%	93.86%	96.61%	95.21%
2371737	DNN	93.91%	89.10%	96.65%	92.72%

References

1. https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction

Appendix:

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

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

In [2]:

```
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 [3]:

```
# 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 [4]:

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

data_info
```

Out[4]:

```
{'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 [9]:

```
# 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[9]:

```
{'Type of Travel': {'data_type': dtype('O'),
  'description': "Indicates the purpose of the flight from the passenger's
perspective, 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 satisfact
ion with the gate's location."},
 'Leg room service': {'data_type': dtype('int64'),
  'description': 'Numerical rating for the satisfaction with the leg room s
ervice offered during the flight.'},
 'Arrival Delay in Minutes': {'data_type': dtype('float64'),
  'description': 'Delay in minutes from the scheduled arrival time of the f
light, 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 [6]:

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

Out[6]:

5 rows × 24 columns

Data Preparation and Cleaning

In [7]:

```
# 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 [10]:

```
# 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:

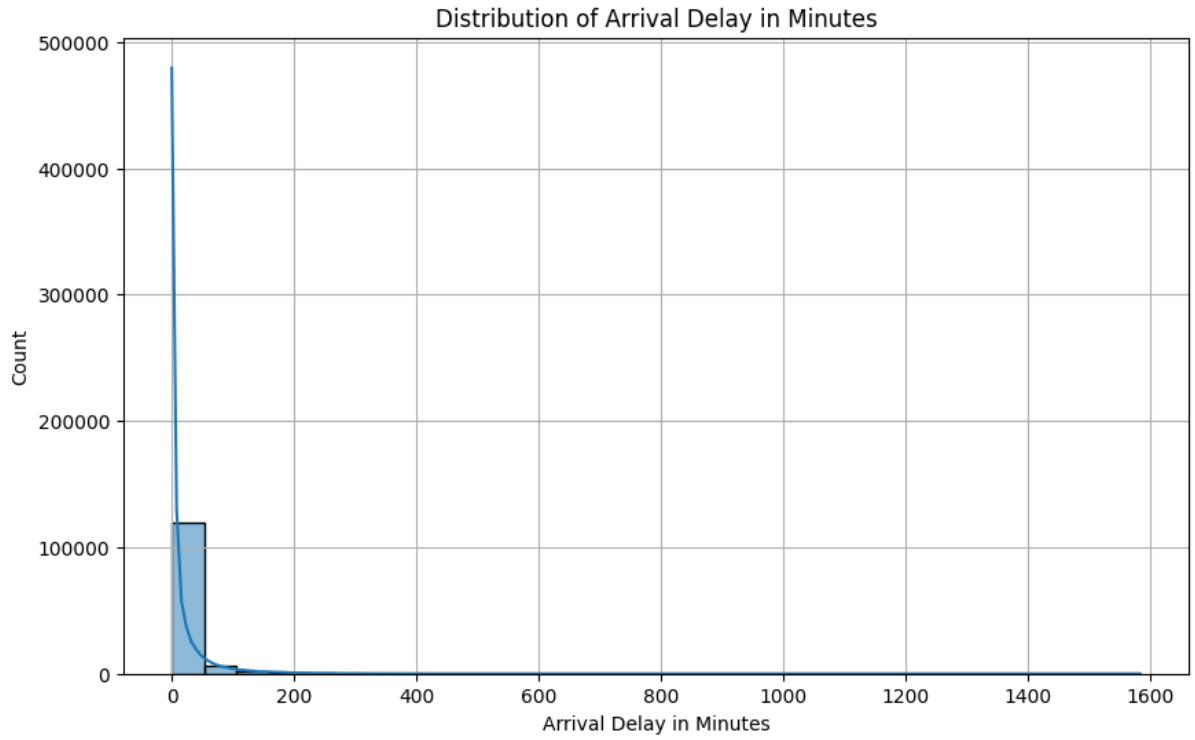
For the numerical column Arrival Delay in Minutes, we imputed NaN values with the median of the column for both training and test datasets.

Exploratory Data Analysis (EDA)

In [11]:

```
# 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)
```



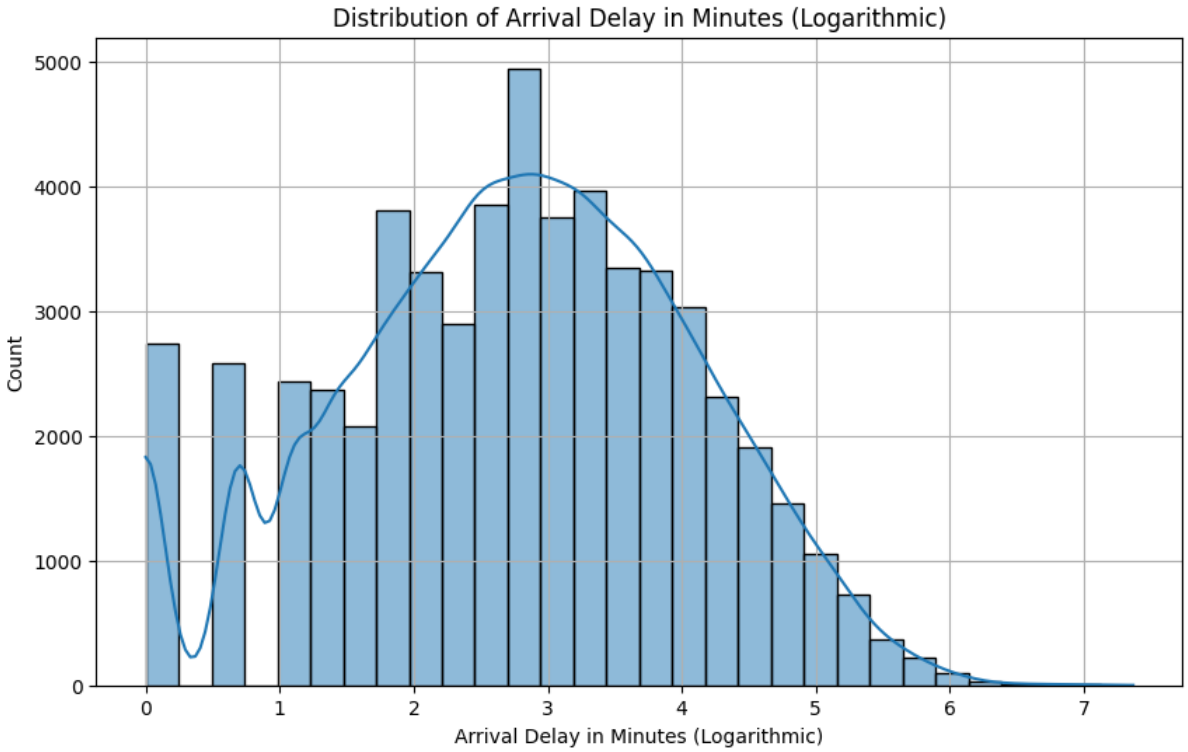

```
Train:
 Type of Travel
Business travel    0.690584
Personal Travel    0.309416
Name: proportion, dtype: float64 Leg room service
4    0.276301
5    0.237950
3    0.192917
2    0.188944
1    0.099284
0    0.004604
Name: proportion, dtype: float64 count    129880.000000
mean         15.045465
std          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 [12]:

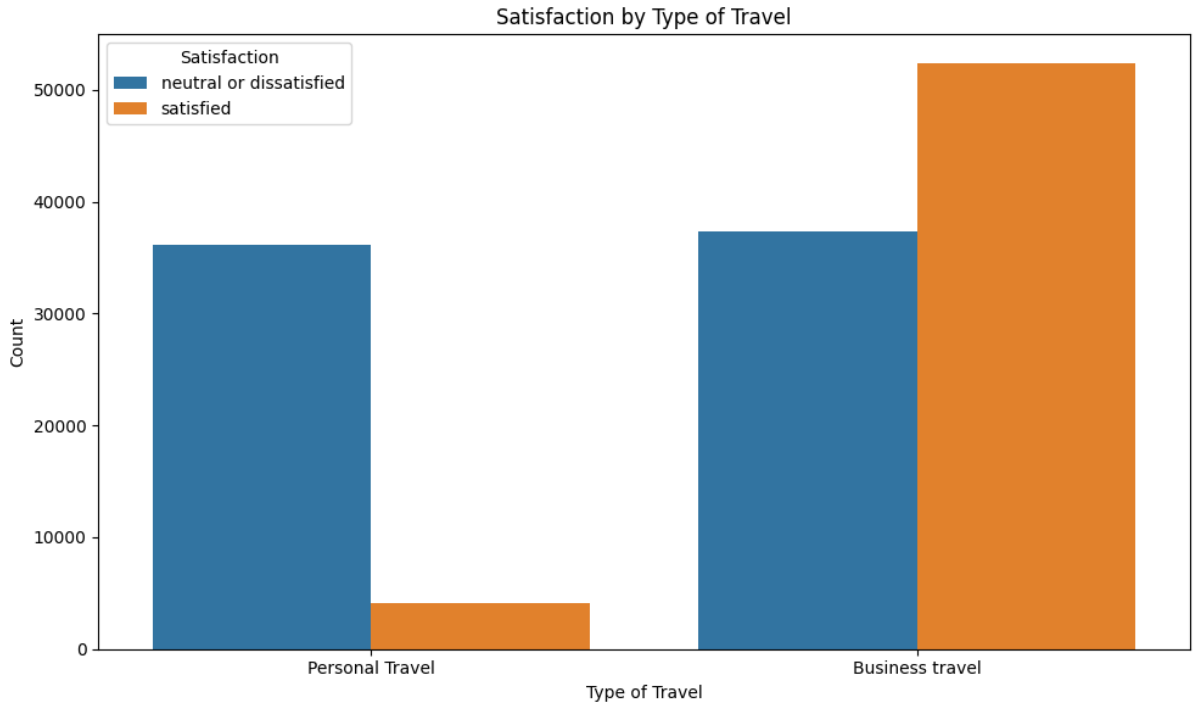
```
# Plotting the logarithmic distribution of 'Arrival Delay in Minutes'
plt.figure(figsize=(10, 6))
sns.histplot(np.log(data["Arrival Delay in Minutes"]), bins=30, kde=True)
plt.title('Distribution of Arrival Delay in Minutes (Logarithmic)')
plt.xlabel('Arrival Delay in Minutes (Logarithmic)')
plt.grid()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

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



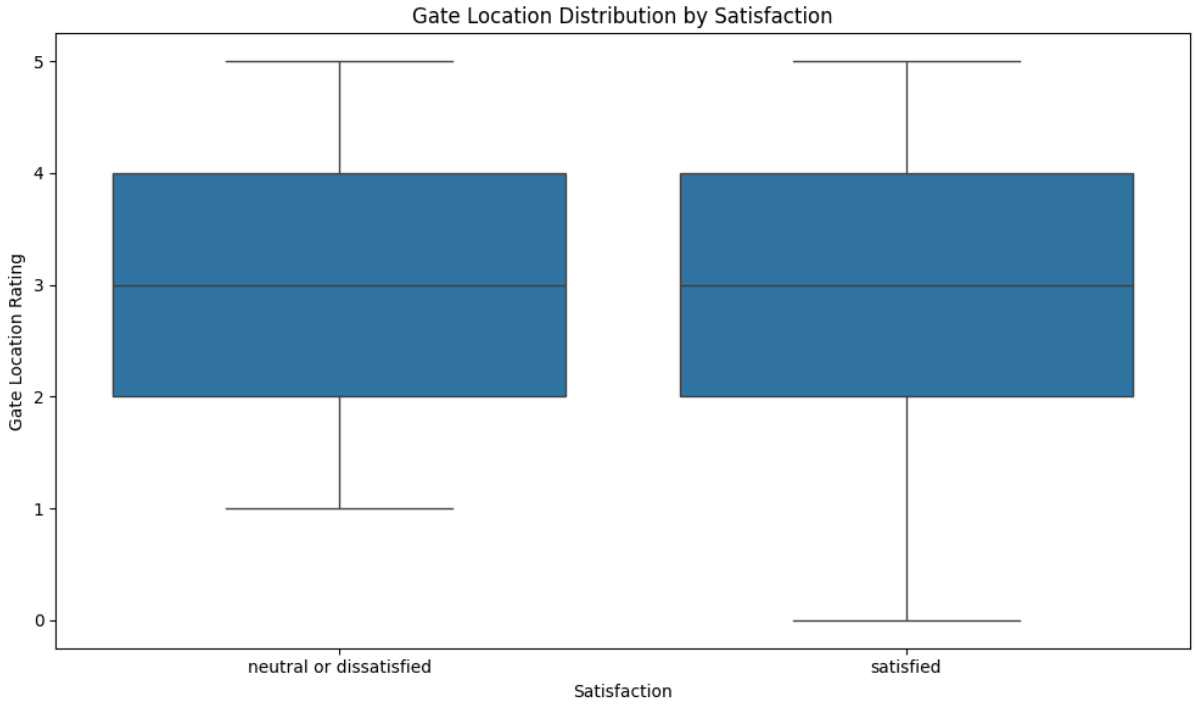
In [13]:

```
# 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()
```



In [14]:

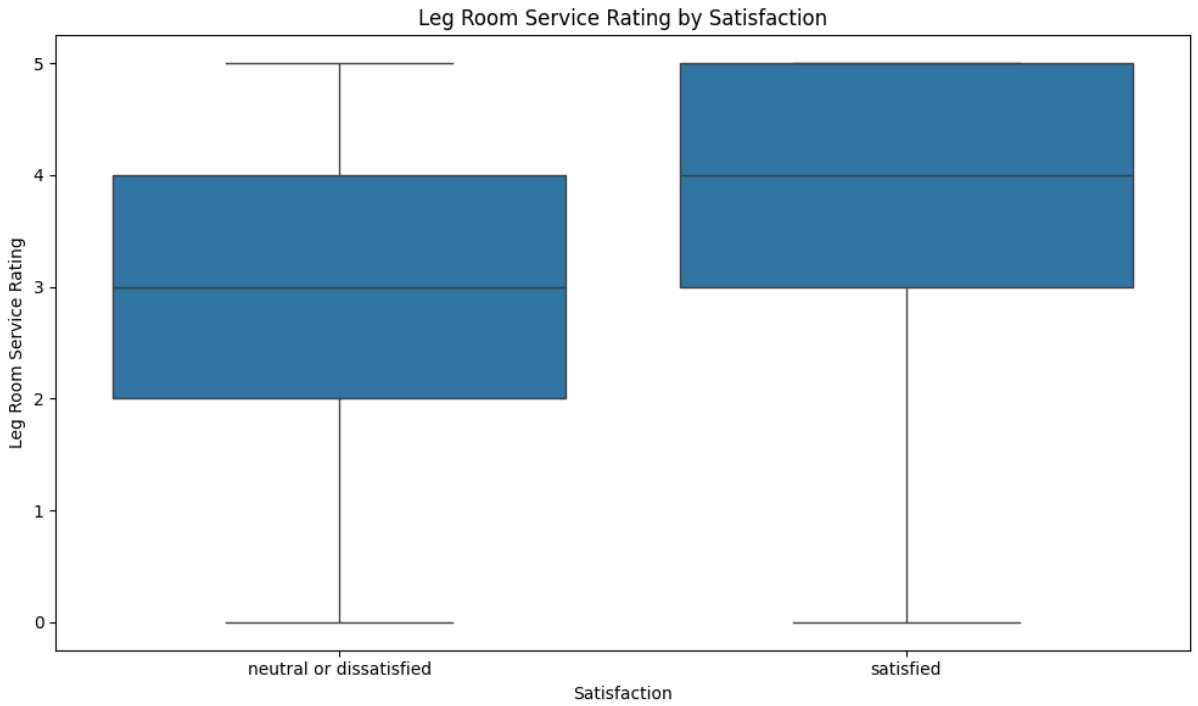
```
# Gate location distribution by satisfaction
plt.figure(figsize=(10, 6))
sns.boxplot(x='satisfaction', y='Gate location', data=data)
plt.title('Gate Location Distribution by Satisfaction')
plt.xlabel('Satisfaction')
plt.ylabel('Gate Location Rating')
plt.tight_layout()
plt.show()
```



In [15]:

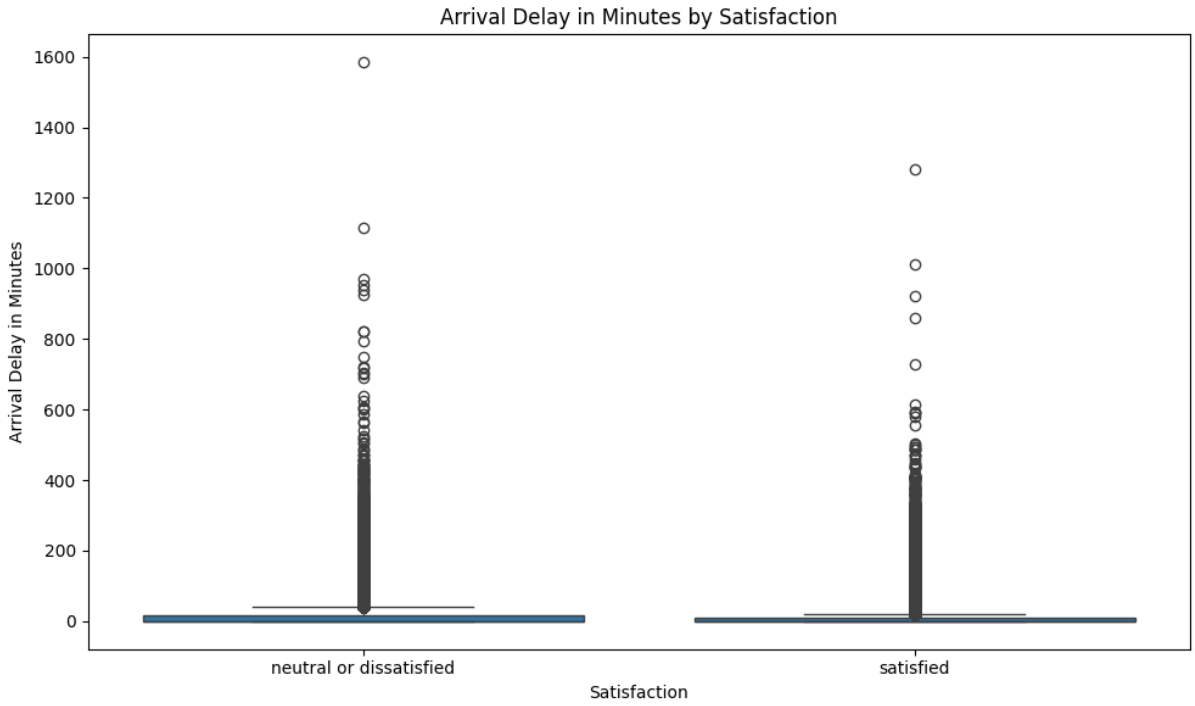
```
# Leg room service rating by satisfaction
```

```
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()
```



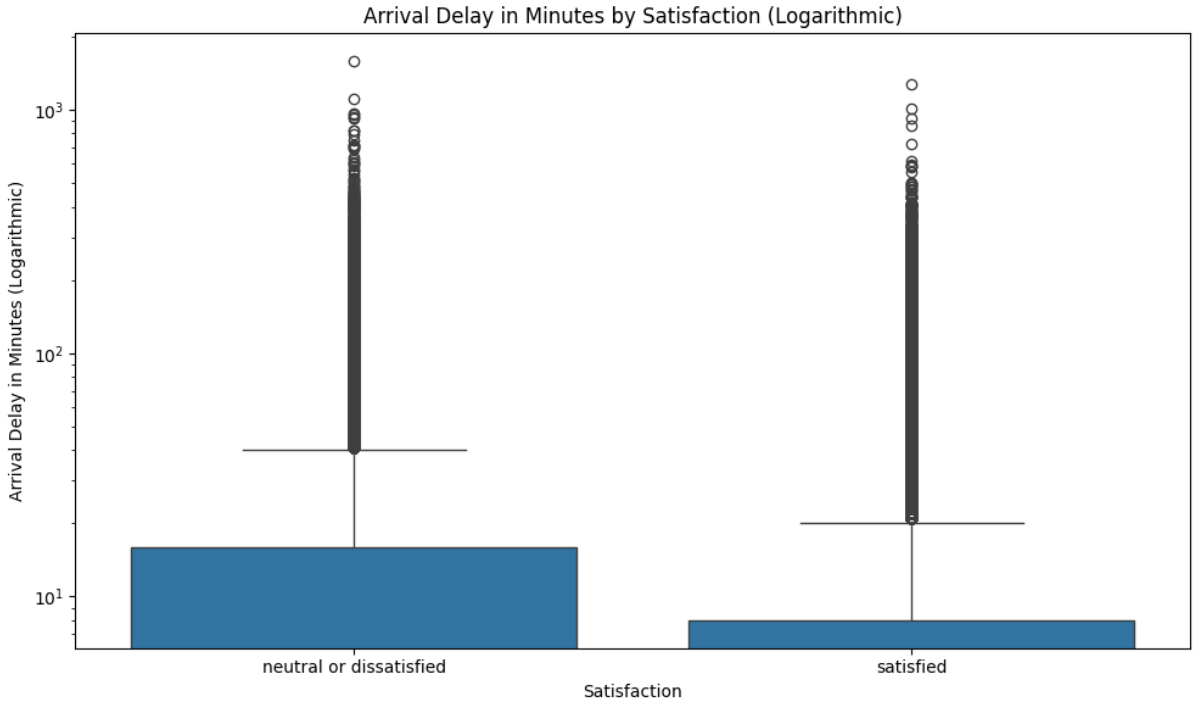
In [16]:

```
# 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 [17]:

```
# 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()
```

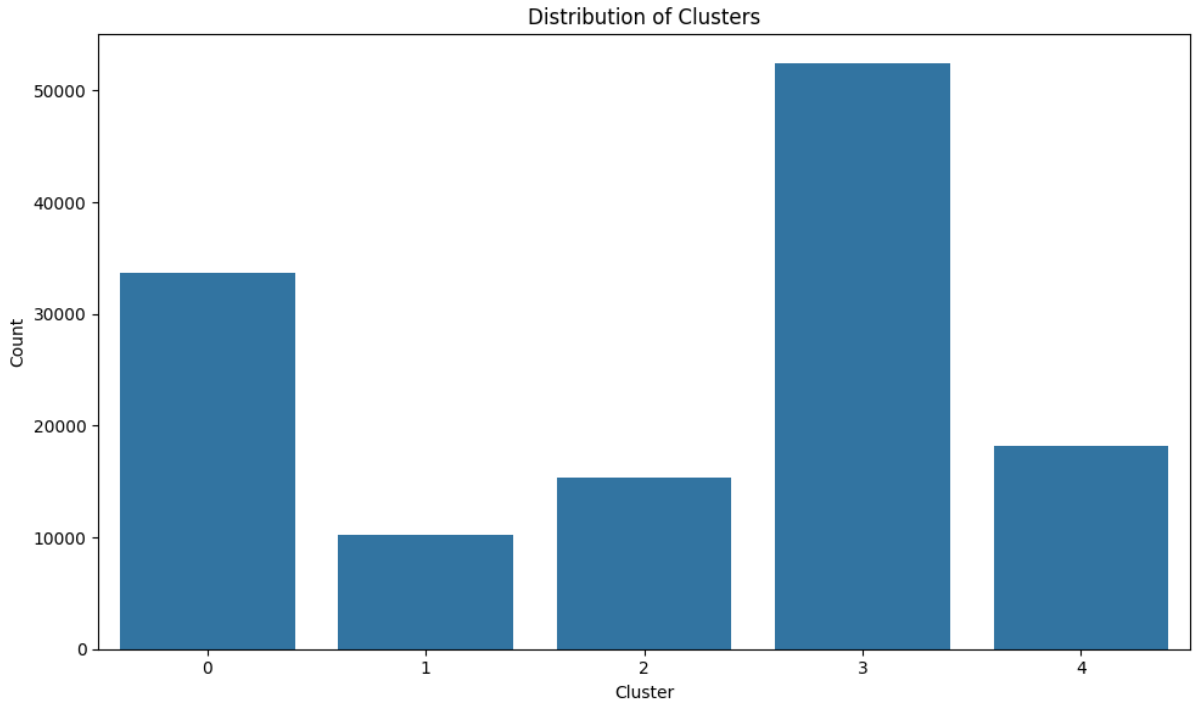


Clustering:

In [18]:

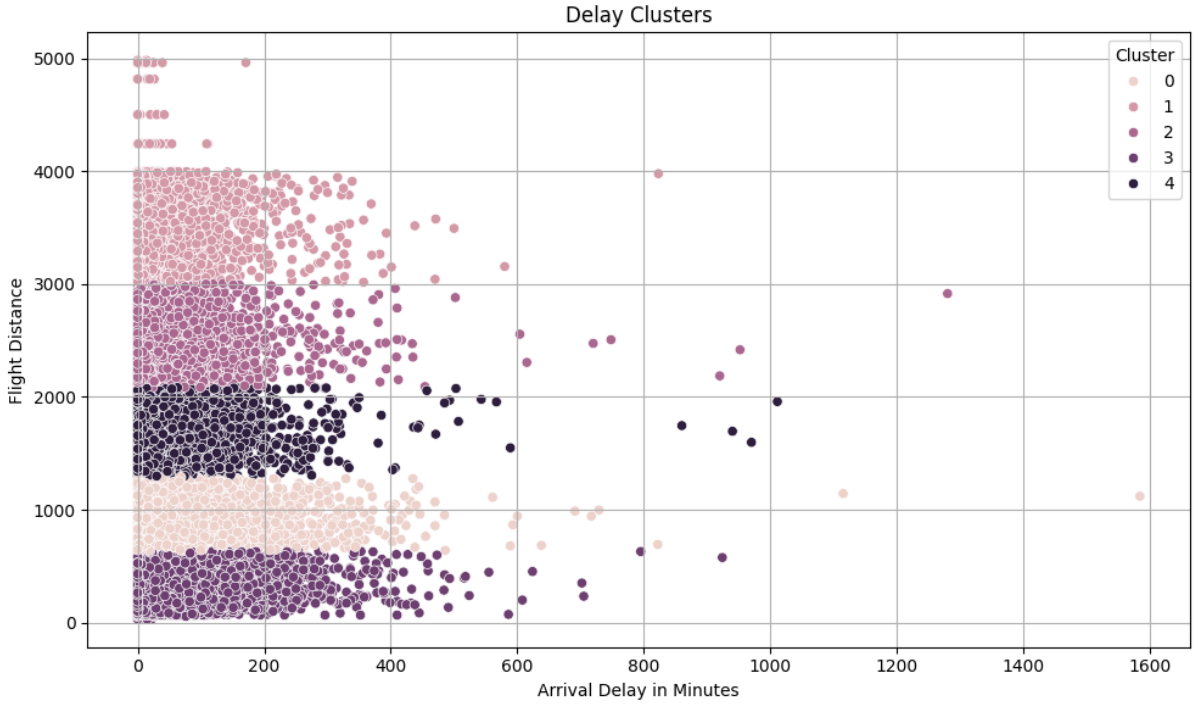
```
# Clustering: Using KMeans for clustering based on the specified features
features_for_clustering = data[['Flight Distance', 'Arrival Delay in
Minutes']]
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(features_for_clustering)

# 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: Fut
ureWarning: 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(
```



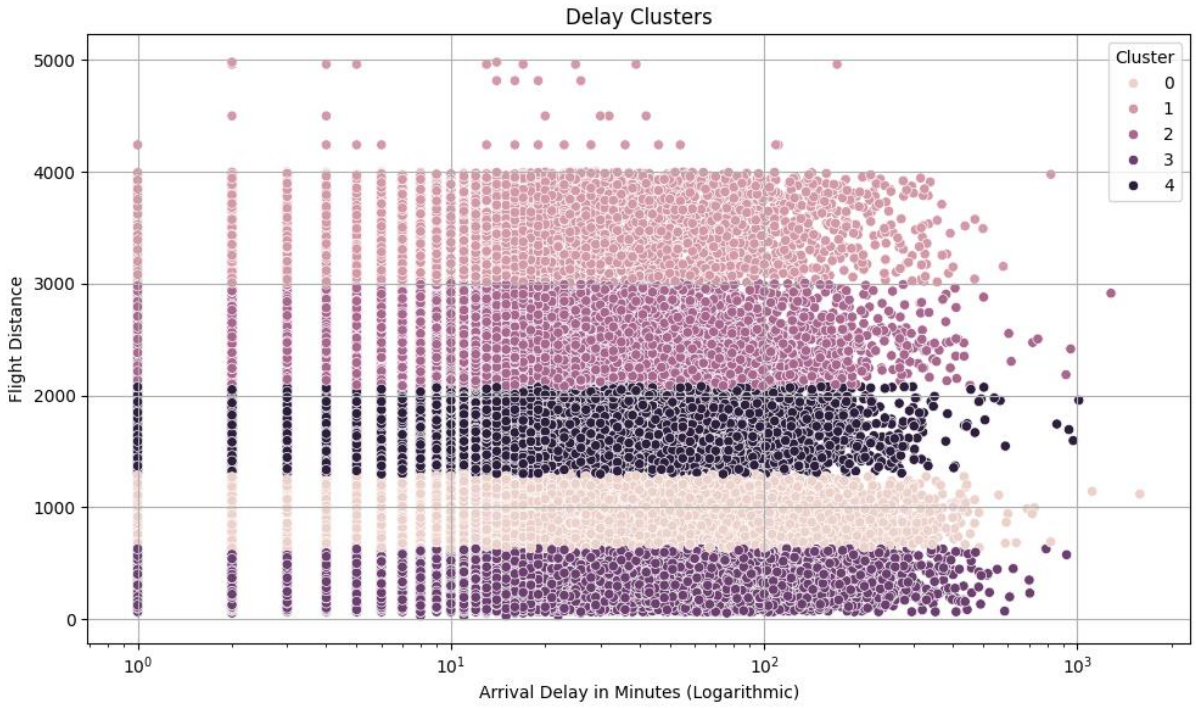
In [19]:

```
# 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 [20]:

```
# 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()
```



Apply the ML and DL

Preprocessing:

In [21]:

```
# 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 [22]:

```
# 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
```

In [23]:

```
# 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': 0.757016400
1921361, 'f1-score': 0.7745287324042545, 'support': 14573}, 'Personal Trave
l': {'precision': 0.706433427292323, 'recall': 0.7472594931158467, 'f1-scor
e': 0.7262731728105689, 'support': 11403}, 'accuracy': 0.7527332922697875,
'macro avg': {'precision': 0.7496519587230625, 'recall': 0.7521379466539915
, 'f1-score': 0.7504009526074117, 'support': 25976}, 'weighted avg': {'prec
ision': 0.75492616355196, 'recall': 0.7527332922697875, 'f1-score': 0.75334
54037144331, 'support': 25976}}
 Decision Tree
 {'Business travel': {'precision': 0.711812361165556, 'recall': 0.747615453
2354354, 'f1-score': 0.7292747414572108, 'support': 14573}, 'Personal Trave
l': {'precision': 0.6552952202436738, 'recall': 0.6131719722879944, 'f1-sco
re': 0.6335341820323472, 'support': 11403}, 'accuracy': 0.6885971666153372,
'macro avg': {'precision': 0.683553790704615, 'recall': 0.6803937127617149,
'f1-score': 0.681404461744779, 'support': 25976}, 'weighted avg': {'precisi
```

```
on': 0.6870023458463296, 'recall': 0.6885971666153372, 'f1-score': 0.687246
3460490755, 'support': 25976}}
 MLPClassifier:
 {'Business travel': {'precision': 0.7788263441217519, 'recall': 0.80415837
50771976, 'f1-score': 0.7912896691424713, 'support': 14573}, 'Personal Trav
el': {'precision': 0.738859913990301, 'recall': 0.7081469788652109, 'f1-sco
re': 0.7231775031345155, 'support': 11403}, 'accuracy': 0.7620110871573761,
'macro avg': {'precision': 0.7588431290560265, 'recall': 0.7561526769712043
, 'f1-score': 0.7572335861384933, 'support': 25976}, 'weighted avg': {'prec
ision': 0.761281795200096, 'recall': 0.7620110871573761, 'f1-score': 0.7613
896449282459, 'support': 25976}}
```

In [25]:

```
print("Train : \n", data.isna().sum())
```

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

Apply DL and ML for the whole Dataset

In [49]:

```
# Label encoding for categorical variables
label_encoder = LabelEncoder()
categorical_cols = ['Gender', 'Customer Type', 'Type of Travel', 'Class',
'satisfaction']

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

# Standardizing the data
```

```
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)
accuracy_lr = accuracy_score(y_test, y_pred_lr)

# Decision Tree Classifier
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)
accuracy_dt = accuracy_score(y_test, y_pred_dt)

# MLP Classifier
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)
accuracy_mlp = accuracy_score(y_test, y_pred_mlp)

accuracy_lr, accuracy_dt, accuracy_mlp
```

Out[49]:

```
(0.8761549122266707, 0.9449106867878041, 0.9567292885740684)
```

In [48]:

```
# Calculating the detailed performance metrics for each model
report_lr = classification_report(y_test, y_pred_lr,
target_names=label_encoder.classes_, output_dict=True)
report_dt = classification_report(y_test, y_pred_dt,
target_names=label_encoder.classes_, output_dict=True)
report_mlp = classification_report(y_test, y_pred_mlp,
target_names=label_encoder.classes_, output_dict=True)

df_report_lr = pd.DataFrame(report_lr).transpose()
df_report_dt = pd.DataFrame(report_dt).transpose()
df_report_mlp = pd.DataFrame(report_mlp).transpose()

df_report_lr, df_report_dt, df_report_mlp
```

Out[48]:

```
(              precision    recall  f1-score       support
 0              0.878275  0.905485  0.891673  14622.000000
 1              0.873223  0.838383  0.855448  11354.000000
 accuracy       0.876155  0.876155  0.876155      0.876155
 macro avg      0.875749  0.871934  0.873560  25976.000000
 weighted avg   0.876067  0.876155  0.875839  25976.000000,
               precision    recall  f1-score       support
 0              0.950851  0.951306  0.951079  14622.000000
 1              0.937252  0.936674  0.936963  11354.000000
 accuracy       0.944911  0.944911  0.944911      0.944911
 macro avg      0.944052  0.943990  0.944021  25976.000000
 weighted avg   0.944907  0.944911  0.944909  25976.000000,
               precision    recall  f1-score       support
 0              0.954417  0.969430  0.961865  14622.000000
 1              0.959817  0.940373  0.949996  11354.000000
 accuracy       0.956729  0.956729  0.956729      0.956729
 macro avg      0.957117  0.954902  0.955930  25976.000000
```

```
weighted avg   0.956777  0.956729  0.956677  25976.000000)
```