Exploratory data analysis Flyzy Thangavhuelo Rolivhuwa

May 23, 2024

1 Flyzy Flight Cancellation Exploratory Data Analysis

1.1 Problem Statement

Flyzy is a company focused on providing a smooth and hassle-free air travel experience. They offer personalized in-flight and airport recommendations, and they also provide real-time flight tracking, mobile check-in, and more. Flyzy aims to redefine the future of air travel with a more personalized and connected experience from the beginning of the trip to the end.

Flight cancellation is a significant issue in the aviation industry. It not only disrupts the customers' plans but also impacts the airlines' reputation and profitability. Therefore, predicting flight cancellations can help airlines take preventive measures and minimize disruptions.

1.2 Importing Libraries

[1]:
```
#Import Libraries
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
# loading the dataset
data = pd.read_excel("Flyzy Flight Cancellation.xlsx")
```

1.3 Data Cleaning

[3]:
```
#Display top 5 rows
data.head()
```

[3]:

	Flight ID	Airline	Flight_Distance	Origin_Airport	Destination_Airport	\
0	7319483	Airline D	475	Airport 3	Airport 2	
1	4791965	Airline E	538	Airport 5	Airport 4	
2	2991718	Airline C	565	Airport 1	Airport 2	
3	4220106	Airline E	658	Airport 5	Airport 3	
4	2263008	Airline E	566	Airport 2	Airport 2	

	Scheduled_Departure_Time	Day_of_Week	Month	Airplane_Type	Weather_Score	\
0	4	6	1	Type C	0.225122	

```
1                       12            1      6        Type B       0.060346
2                       17            3      9        Type C       0.093920
3                        1            1      8        Type B       0.656750
4                       19            7     12        Type E       0.505211

   Previous_Flight_Delay_Minutes  Airline_Rating  Passenger_Load  \
0                            5.0        2.151974        0.477202
1                           68.0        1.600779        0.159718
2                           18.0        4.406848        0.256803
3                           13.0        0.998757        0.504077
4                            4.0        3.806206        0.019638

   Flight_Cancelled
0                 0
1                 1
2                 0
3                 1
4                 0
```

[4]:
```
#Display bottom 5 rows
data.tail()
```

[4]:
```
      Flight ID    Airline  Flight_Distance Origin_Airport  \
2995    1265781  Airline D              395      Airport 2
2996    5440150  Airline E              547      Airport 1
2997     779080  Airline C              461      Airport 1
2998    4044431  Airline B              464      Airport 3
2999    2806578  Airline A              369      Airport 1

     Destination_Airport  Scheduled_Departure_Time  Day_of_Week  Month  \
2995           Airport 3                        0            6      1
2996           Airport 4                       22            4      7
2997           Airport 3                        8            3      1
2998           Airport 3                        5            5      3
2999           Airport 2                        1            1     10

     Airplane_Type  Weather_Score  Previous_Flight_Delay_Minutes  \
2995        Type B       0.190018                        1.00000
2996        Type E       0.719271                       91.00000
2997        Type B       0.458724                        3.00000
2998        Type E       0.443373                       46.00000
2999        Type A       0.704563                       18.66667

      Airline_Rating  Passenger_Load  Flight_Cancelled
2995        2.451216        0.283440                 1
2996        0.027039        0.665294                 1
2997        1.131315        0.991307                 0
```

```
2998        0.968651        0.254808                 1
2999        1.879411        0.532486                 1
```

[5]:
```
# Display information about the DataFrame
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Flight ID                       3000 non-null   int64
 1   Airline                         3000 non-null   object
 2   Flight_Distance                 3000 non-null   int64
 3   Origin_Airport                  3000 non-null   object
 4   Destination_Airport             3000 non-null   object
 5   Scheduled_Departure_Time        3000 non-null   int64
 6   Day_of_Week                     3000 non-null   int64
 7   Month                           3000 non-null   int64
 8   Airplane_Type                   3000 non-null   object
 9   Weather_Score                   3000 non-null   float64
 10  Previous_Flight_Delay_Minutes   3000 non-null   float64
 11  Airline_Rating                  3000 non-null   float64
 12  Passenger_Load                  3000 non-null   float64
 13  Flight_Cancelled                3000 non-null   int64
dtypes: float64(4), int64(6), object(4)
memory usage: 328.2+ KB
```

1.4 Descriptive Statistics

Obtain summary statistics for categorical columns, such as count, unique values, most frequent value (top), and frequency of the top value. This helps to understand the distribution and frequency of categories.

[6]:
```
# Get descriptive statistics for categorical columns
data.describe()
```

[6]:
```
          Flight ID  Flight_Distance  Scheduled_Departure_Time  Day_of_Week  \
count  3.000000e+03      3000.000000               3000.000000  3000.000000
mean   4.997429e+06       498.909333                 11.435000     3.963000
std    2.868139e+06        98.892266                  6.899298     2.016346
min    3.681000e+03       138.000000                  0.000000     1.000000
25%    2.520313e+06       431.000000                  6.000000     2.000000
50%    5.073096e+06       497.000000                 12.000000     4.000000
75%    7.462026e+06       566.000000                 17.000000     6.000000
max    9.999011e+06       864.000000                 23.000000     7.000000

             Month  Weather_Score  Previous_Flight_Delay_Minutes  \
```

```
       Flight ID      Flight_Distance  Scheduled_Departure_Time
count  3000.000000    3000.000000                   3000.000000
mean      6.381000       0.524023                     26.793383
std       3.473979       0.290694                     27.874733
min       1.000000       0.000965                      0.000000
25%       3.000000       0.278011                      7.000000
50%       6.000000       0.522180                     18.000000
75%       9.000000       0.776323                     38.000000
max      12.000000       1.099246                    259.000000

       Airline_Rating  Passenger_Load  Flight_Cancelled
count     3000.000000     3000.000000       3000.000000
mean         2.317439        0.515885          0.690667
std          1.430386        0.295634          0.462296
min          0.000103        0.001039          0.000000
25%          1.092902        0.265793          0.000000
50%          2.126614        0.517175          1.000000
75%          3.525746        0.770370          1.000000
max          5.189038        1.123559          1.000000
```

[7]:
```
#Type of data
data.dtypes
```

[7]:
```
Flight ID                          int64
Airline                           object
Flight_Distance                    int64
Origin_Airport                    object
Destination_Airport               object
Scheduled_Departure_Time           int64
Day_of_Week                        int64
Month                              int64
Airplane_Type                     object
Weather_Score                    float64
Previous_Flight_Delay_Minutes    float64
Airline_Rating                   float64
Passenger_Load                   float64
Flight_Cancelled                   int64
dtype: object
```

[8]:
```
#Finding duplicates
data.duplicated()
```

[8]:
```
0       False
1       False
2       False
3       False
4       False
        …
```

```
2995    False
2996    False
2997    False
2998    False
2999    False
Length: 3000, dtype: bool
```

[9]:
```
#Droping duplicates
data = data.drop_duplicates()
```

[10]:
```
# Check for missing values
missing_values = data.isnull().sum()
print(missing_values)
```

```
Flight ID                          0
Airline                            0
Flight_Distance                    0
Origin_Airport                     0
Destination_Airport                0
Scheduled_Departure_Time           0
Day_of_Week                        0
Month                              0
Airplane_Type                      0
Weather_Score                      0
Previous_Flight_Delay_Minutes      0
Airline_Rating                     0
Passenger_Load                     0
Flight_Cancelled                   0
dtype: int64
```

[11]:
```
# Handle missing values (if any)
# Fill missing values for numeric columns with their mean
numeric_columns = data.select_dtypes(include=[np.number]).columns
data[numeric_columns] = data[numeric_columns].fillna(data[numeric_columns].
 ↪mean())
```

[12]:
```
#Display top 5 raws
data.head()
```

[12]:
```
   Flight ID    Airline  Flight_Distance Origin_Airport Destination_Airport  \
0    7319483  Airline D              475      Airport 3           Airport 2
1    4791965  Airline E              538      Airport 5           Airport 4
2    2991718  Airline C              565      Airport 1           Airport 2
3    4220106  Airline E              658      Airport 5           Airport 3
4    2263008  Airline E              566      Airport 2           Airport 2

   Scheduled_Departure_Time  Day_of_Week  Month Airplane_Type  Weather_Score  \
```

```
0                          4            6      1        Type C       0.225122
1                         12            1      6        Type B       0.060346
2                         17            3      9        Type C       0.093920
3                          1            1      8        Type B       0.656750
4                         19            7     12        Type E       0.505211

   Previous_Flight_Delay_Minutes  Airline_Rating  Passenger_Load  \
0                            5.0        2.151974        0.477202
1                           68.0        1.600779        0.159718
2                           18.0        4.406848        0.256803
3                           13.0        0.998757        0.504077
4                            4.0        3.806206        0.019638

   Flight_Cancelled
0                 0
1                 1
2                 0
3                 1
4                 0
```

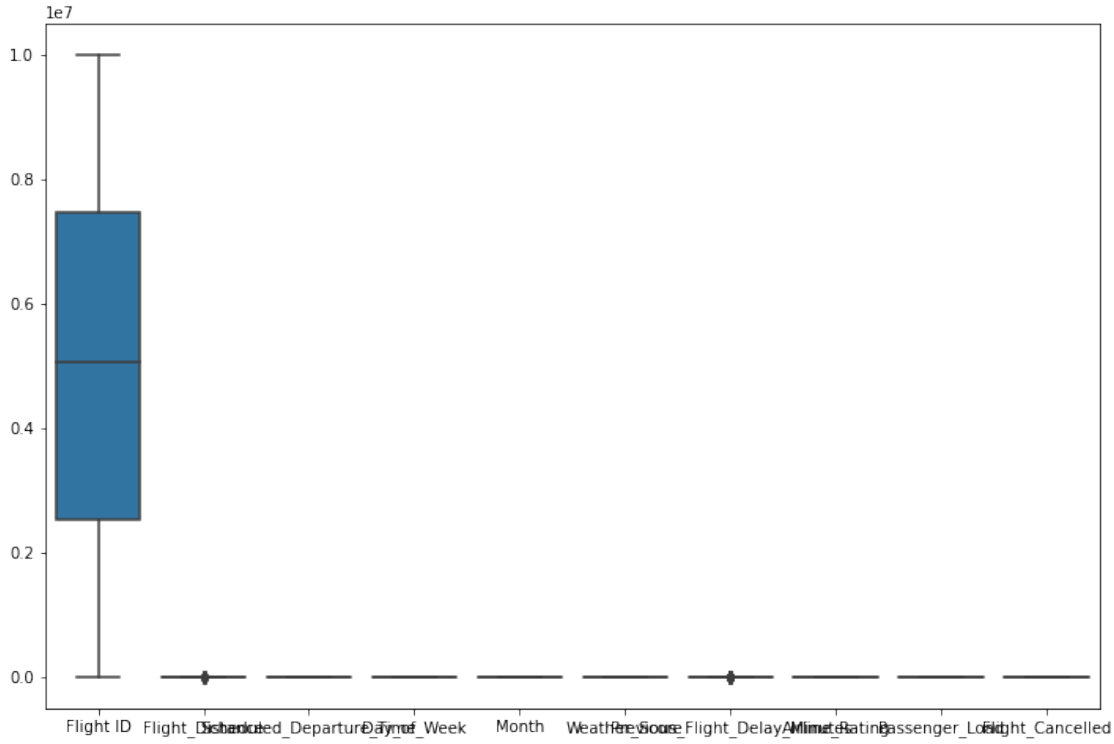
[13]:
```
#Display bottom 5 rows
data.tail()
```

[13]:
```
      Flight ID    Airline  Flight_Distance Origin_Airport  \
2995    1265781  Airline D              395      Airport 2
2996    5440150  Airline E              547      Airport 1
2997     779080  Airline C              461      Airport 1
2998    4044431  Airline B              464      Airport 3
2999    2806578  Airline A              369      Airport 1

     Destination_Airport  Scheduled_Departure_Time  Day_of_Week  Month  \
2995           Airport 3                         0            6      1
2996           Airport 4                        22            4      7
2997           Airport 3                         8            3      1
2998           Airport 3                         5            5      3
2999           Airport 2                         1            1     10

     Airplane_Type  Weather_Score  Previous_Flight_Delay_Minutes  \
2995        Type B       0.190018                        1.00000
2996        Type E       0.719271                       91.00000
2997        Type B       0.458724                        3.00000
2998        Type E       0.443373                       46.00000
2999        Type A       0.704563                       18.66667

      Airline_Rating  Passenger_Load  Flight_Cancelled
2995        2.451216        0.283440                 1
2996        0.027039        0.665294                 1
```

2997	1.131315	0.991307	0
2998	0.968651	0.254808	1
2999	1.879411	0.532486	1

1.5 Handling Outliers

[14]:
```
# Identify outliers using boxplots
plt.figure(figsize=(12, 8))
sns.boxplot(data=data.select_dtypes(include=[np.number]))
plt.show()
```



[15]:
```
# Handling outliers (using Z-score)
from scipy.stats import zscore

numeric_columns = data.select_dtypes(include=[np.number]).columns
data = data[(np.abs(zscore(data[numeric_columns])) < 3).all(axis=1)]
```

1.6 Descriptive Statistics

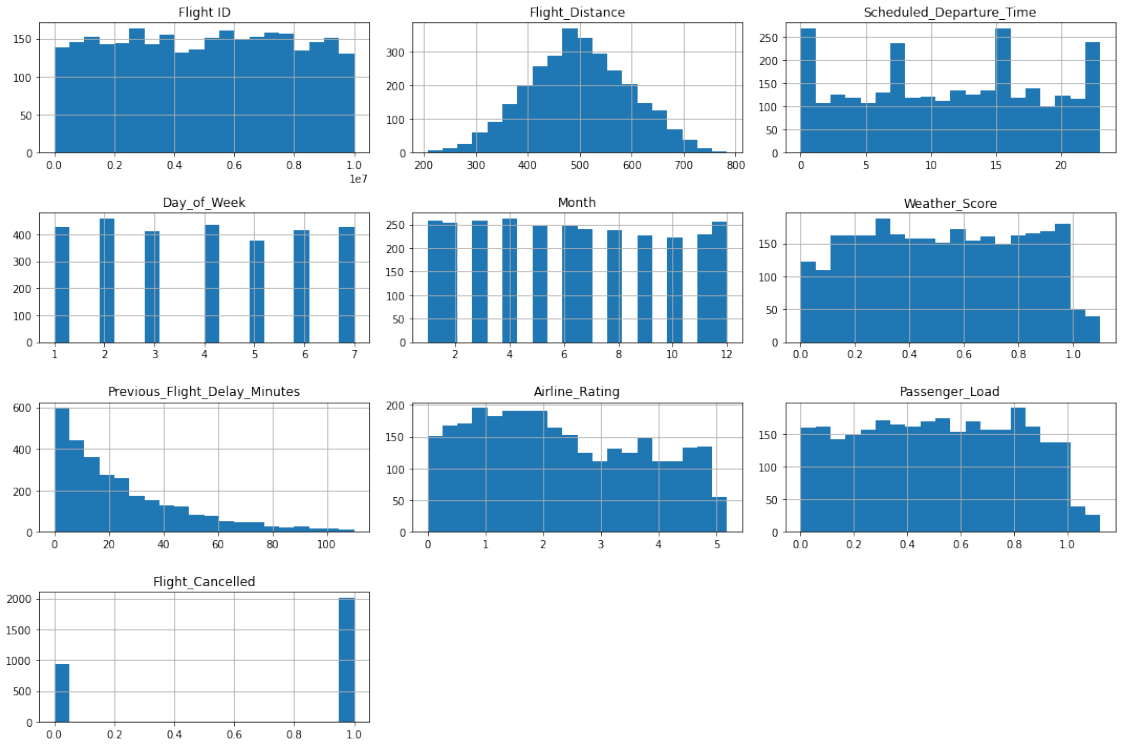
[16]:
```
# Get descriptive statistics for numerical columns
numerical_summary = data.describe()
```

[17]:
```
# Get descriptive statistics for categorical columns
categorical_summary = data.describe(include=['object', 'category'])

numerical_summary, categorical_summary
```

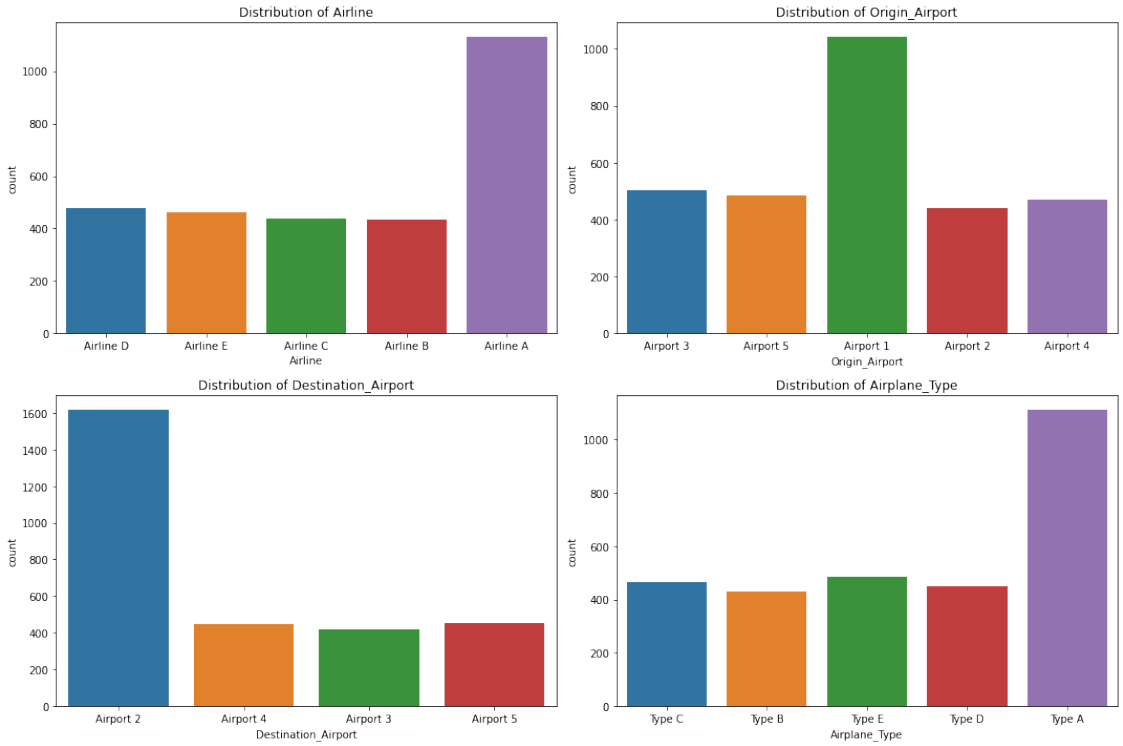
[17]:
```
(          Flight ID  Flight_Distance  Scheduled_Departure_Time  Day_of_Week  \
 count  2.939000e+03      2939.000000               2939.000000  2939.000000
 mean   4.998676e+06       498.381763                 11.473971     3.957469
 std    2.859490e+06        97.289714                  6.899718     2.015063
 min    3.681000e+03       207.000000                  0.000000     1.000000
 25%    2.547248e+06       431.000000                  6.000000     2.000000
 50%    5.081258e+06       497.000000                 12.000000     4.000000
 75%    7.438063e+06       565.000000                 17.000000     6.000000
 max    9.999011e+06       784.000000                 23.000000     7.000000

              Month  Weather_Score  Previous_Flight_Delay_Minutes  \
 count  2939.000000    2939.000000                    2939.000000
 mean      6.381082       0.525968                      24.755296
 std       3.474951       0.291237                      23.017804
 min       1.000000       0.000965                       0.000000
 25%       3.000000       0.279113                       7.000000
 50%       6.000000       0.524675                      17.777780
 75%       9.000000       0.779039                      36.444440
 max      12.000000       1.099246                     110.000000

        Airline_Rating  Passenger_Load  Flight_Cancelled
 count     2939.000000     2939.000000       2939.000000
 mean         2.320053        0.517560          0.684587
 std          1.430754        0.295229          0.464759
 min          0.000103        0.001039          0.000000
 25%          1.095369        0.267479          0.000000
 50%          2.128631        0.518406          1.000000
 75%          3.530915        0.772711          1.000000
 max          5.189038        1.123559          1.000000  ,
           Airline Origin_Airport Destination_Airport Airplane_Type
 count        2939           2939                2939          2939
 unique          5              5                   4             5
 top     Airline A      Airport 1           Airport 2        Type A
 freq         1129           1040                1617          1112)
```

1.7 Data Distribution

[18]:
```
# Plot histograms for numerical columns
data.hist(bins=20, figsize=(15, 10))
plt.tight_layout()
plt.show()
```



[19]:
```
# Plot bar charts for categorical columns
categorical_columns = ['Airline', 'Origin_Airport', 'Destination_Airport',␣
 ↪'Airplane_Type']
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

for i, col in enumerate(categorical_columns):
    sns.countplot(ax=axes[i//2, i%2], x=col, data=data)
    axes[i//2, i%2].set_title(f'Distribution of {col}')

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

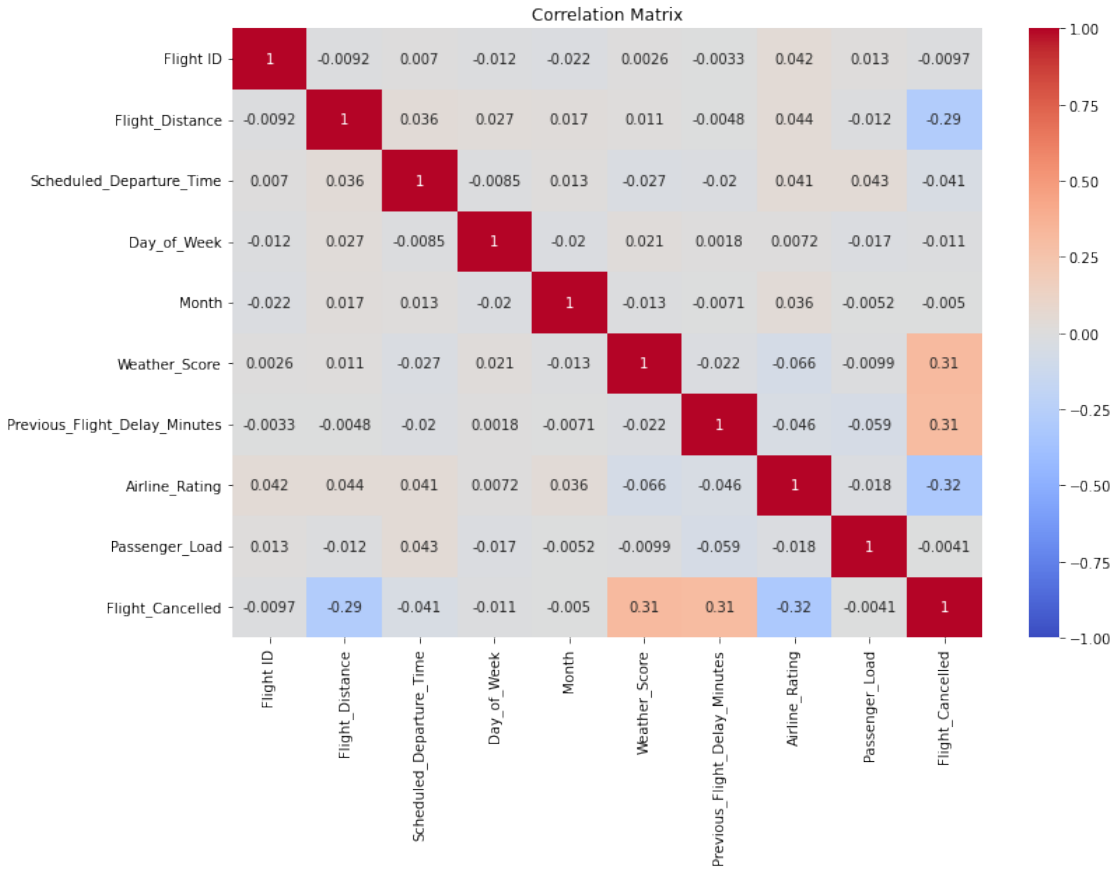


1.8 Relationship Between Features

Calculate and visualize the correlation matrix for numerical columns using a heatmap. This helps to identify linear relationships between numerical features. Create pair plots for numerical columns to visualize the relationships between pairs of features. This helps to see potential patterns, trends, and correlations.

[20]:
```
# Correlation matrix for numerical columns
corr_matrix = data.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix')
plt.show()
```

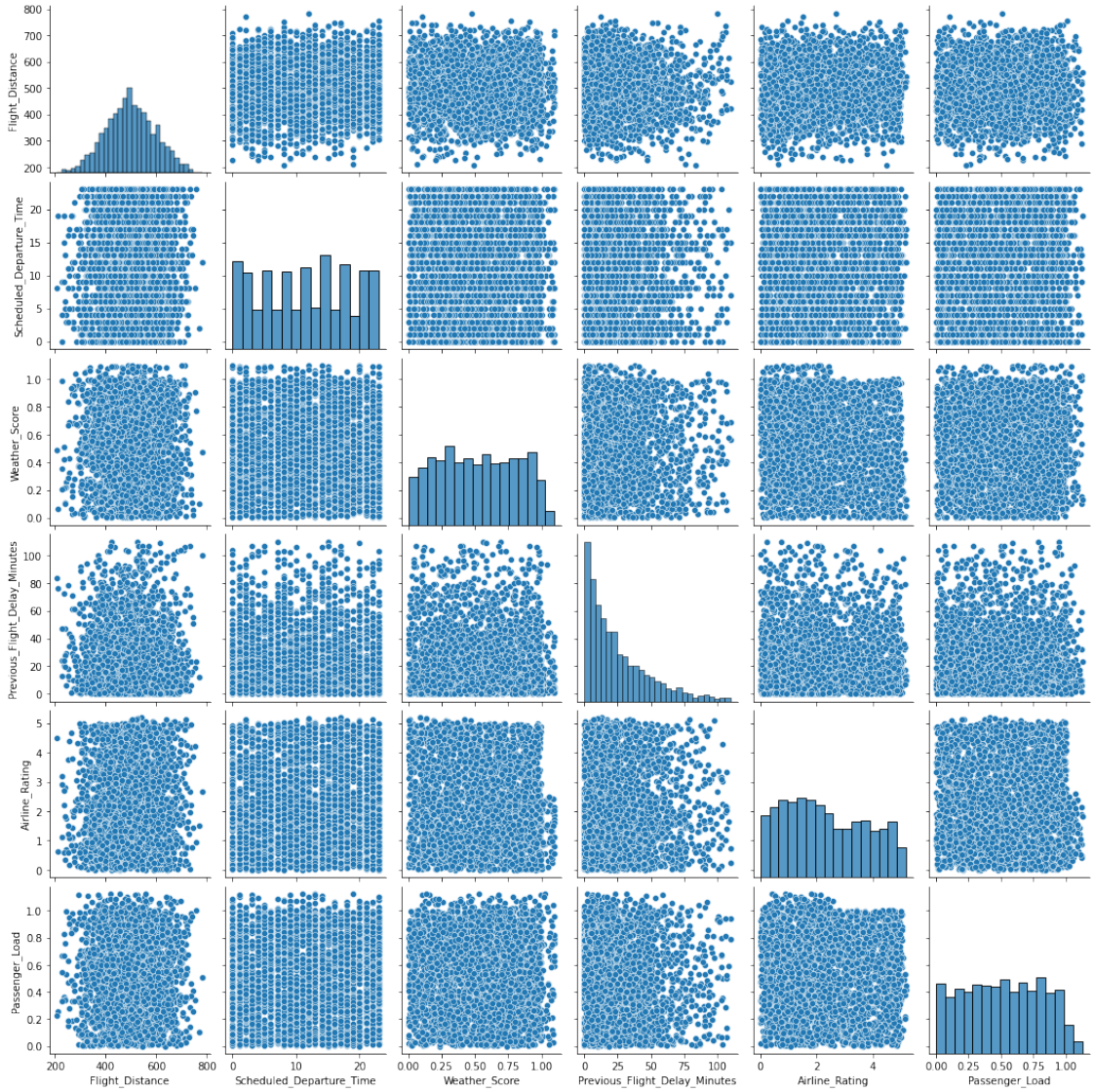
```
/tmp/ipykernel_105/510550350.py:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  corr_matrix = data.corr()
```



1.8.1 Create pair plots for numerical columns to visualize the relationships between pairs of features. This helps to see potential patterns, trends, and correlations.

1.8.2

[21]:
```
# Pair plot for numerical columns to visualize relationships
sns.pairplot(data, vars=['Flight_Distance', 'Scheduled_Departure_Time',␣
 ↪'Weather_Score', 'Previous_Flight_Delay_Minutes', 'Airline_Rating',␣
 ↪'Passenger_Load'])
plt.show()
```

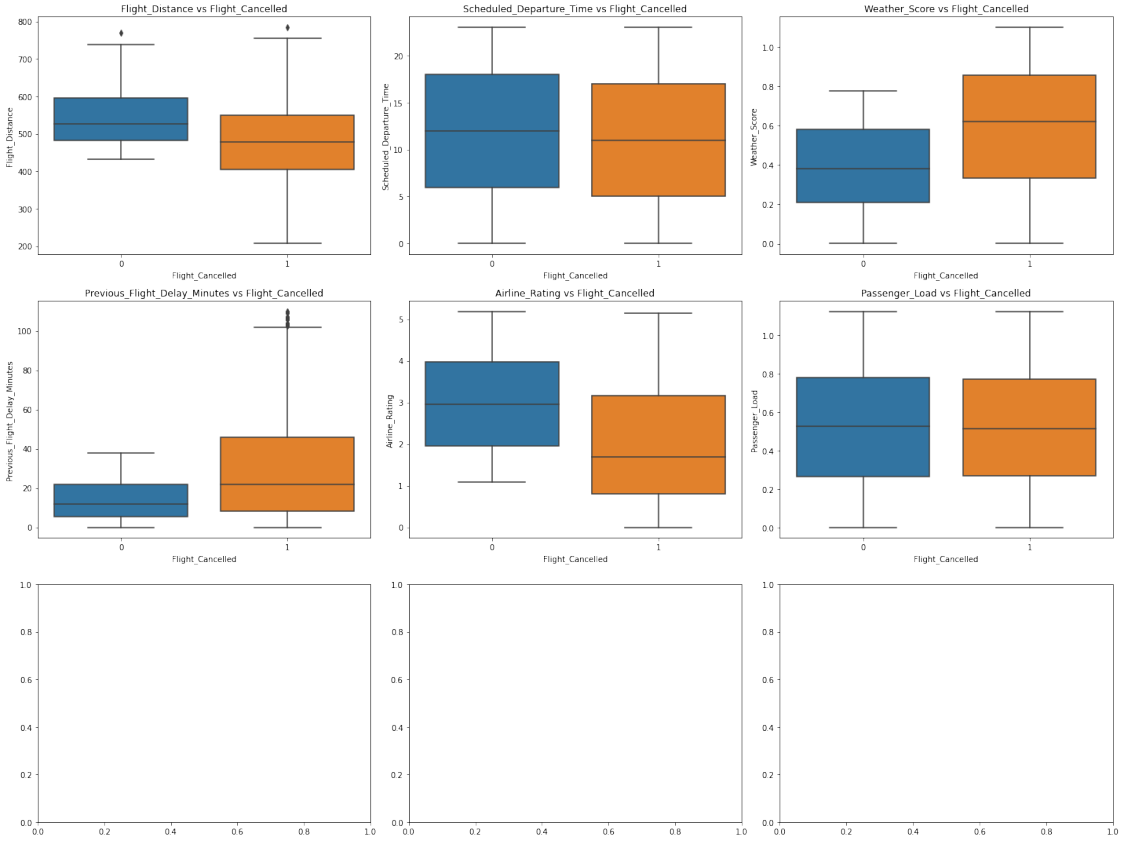


1.9 Relationship Between Features and Target Variable

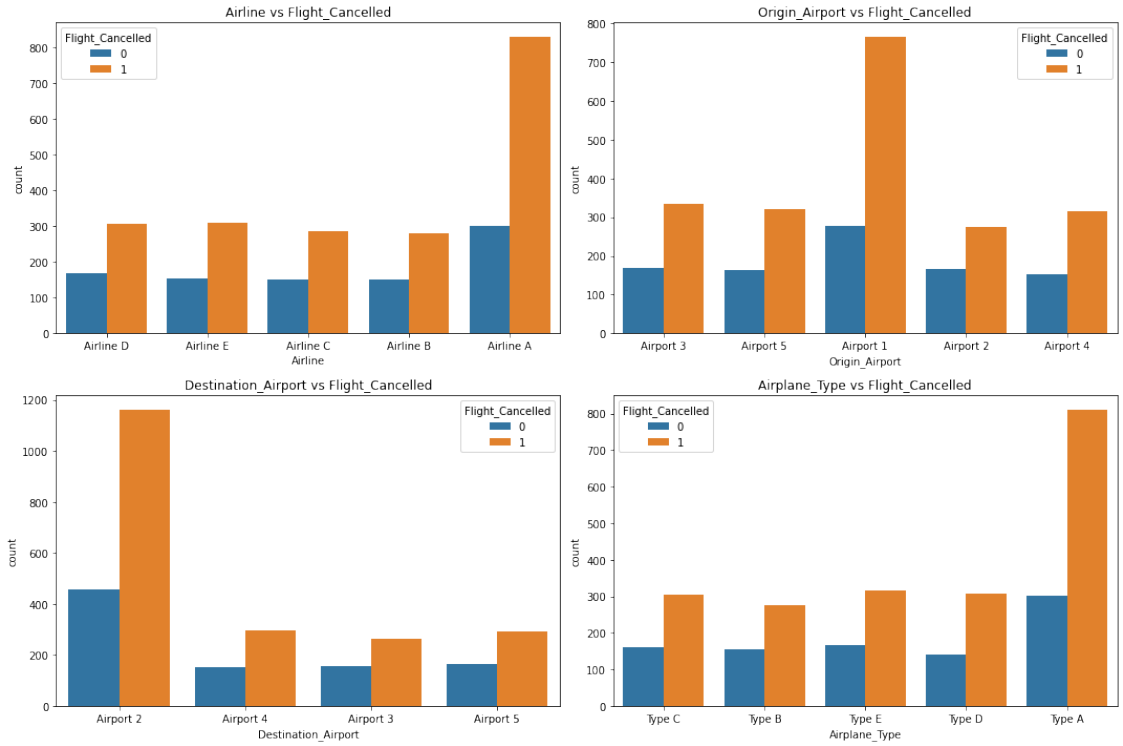
Use box plots to visualize the relationship between numerical features and the target variable (Flight_Cancelled). This helps to see if there are significant differences in the distribution of numerical features between cancelled and non-cancelled flights. Use bar plots to visualize the relationship between categorical features and the target variable (Flight_Cancelled). This helps to see the distribution of cancellations across different categories.

[22]:
```
# Box plots to show the relationship between numerical features and the target␣
↪variable
fig, axes = plt.subplots(3, 3, figsize=(20, 15))
```

```
numerical_columns = ['Flight_Distance', 'Scheduled_Departure_Time',␣
 ↪'Weather_Score', 'Previous_Flight_Delay_Minutes', 'Airline_Rating',␣
 ↪'Passenger_Load']
for i, col in enumerate(numerical_columns):
    sns.boxplot(ax=axes[i//3, i%3], x='Flight_Cancelled', y=col, data=data)
    axes[i//3, i%3].set_title(f'{col} vs Flight_Cancelled')

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



[23]:
```
# Bar plots for categorical columns against the target variable
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

for i, col in enumerate(categorical_columns):
    sns.countplot(ax=axes[i//2, i%2], x=col, hue='Flight_Cancelled', data=data)
    axes[i//2, i%2].set_title(f'{col} vs Flight_Cancelled')

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



1.10 Summary of Insights

After running the above code, you'll be able to:

Get a quick statistical summary of the dataset. Visualize the distribution of each feature. Understand the relationships between different features. See how each feature relates to the target variable, which in this case is whether a flight is cancelled or not ##

[ ]: