In [1]:
```
#IMPORTING ALL NECESSARY LIBRARIES I WILL BE USING.
import pandas as pd
import numpy as np
```

In [4]:
```
#LOADING THE DATASET.
df=pd.read_excel("Flyzy Flight Cancellation.xlsx")
```

In [5]:
```
#ENSURING THERE ARE NO DUPLICATES AND DROPPING THEM.
df.drop_duplicates(inplace=True)
```

In [6]:
```
#ENSURING THERE ARE NO NULL VALUES AND DROPPING THEM.
df.dropna(inplace=True)
```

In [7]:
```
from sklearn.impute import SimpleImputer
```

In [10]:
```
imputer=SimpleImputer(strategy='mean')
```

In [11]:
```
from scipy.stats import zscore
```

In [12]:
```
#IMPORTING AND SAVING MY CLEANED DATASET.
z_scores=np.abs(zscore(df.select_dtypes(include=[np.number])))
outliers=(z_scores>3).any(axis=1)
df=df[~outliers]
```

In [13]:
```
df.to_excel('cleaned_dataset.xlsx',index=False)
print("Cleaning complete. Cleaned dataset saved as 'cleaned_dataset.xlsx'")
```

```
Cleaning complete. Cleaned dataset saved as 'cleaned_dataset.xlsx'
```

In [14]:
```
#IMPORTING ALL THE LIBRARIES I WILL BE USING TO CREATE MY LOGISTIC REGRESSION MODEL
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [15]:
```
#IMPORTING THE CLEANED DATASET.
dataset=pd.read_excel('cleaned_dataset.xlsx')
```

In [16]:
```
#DISPLAYING THE FIRST 5 ROWS OF THE DATASET.
print(dataset.head())
```

```
   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
```

In [17]:
```
#DISPLAYING THE COLUMNS.
print(dataset.columns)
```

```
Index(['Flight ID', 'Airline', 'Flight_Distance', 'Origin_Airport',
       'Destination_Airport', 'Scheduled_Departure_Time', 'Day_of_Week',
       'Month', 'Airplane_Type', 'Weather_Score',
       'Previous_Flight_Delay_Minutes', 'Airline_Rating', 'Passenger_Load',
       'Flight_Cancelled'],
      dtype='object')
```

In [23]:
```
##DISPLAYING THE DATA TYPES OF EACH COLUMN.
print(dataset.dtypes)
```

```
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
```

In [24]:
```
#CHANGING ALL CATEGORICAL VARIABLES TO NUMERIC VARIABLES BECAUSE A LOGISTIC REGRESS
#REQUIRES NUMERICAL INPUT AS IT CANNOT DIRECTLY PROCESS CATEGORICAL DATA.
from sklearn.preprocessing import LabelEncoder
label_encoder=LabelEncoder()
```

In [25]:
```
dataset['Airline']=label_encoder.fit_transform(dataset['Airline'])
dataset['Origin_Airport']=label_encoder.fit_transform(dataset['Origin_Airport'])
```

```
dataset['Destination_Airport']=label_encoder.fit_transform(dataset['Destination_Air
dataset['Airplane_Type']=label_encoder.fit_transform(dataset['Airplane_Type'])
```

In [26]:
```
#DISPLAYING THE DATA TYPES OF ERACH COLUMN TO ENSURE THEY ARE ALL NUMERIC VARIABLES
dataset.dtypes
```

Out[26]:
```
Flight ID                          int64
Airline                            int32
Flight_Distance                    int64
Origin_Airport                     int32
Destination_Airport                int32
Scheduled_Departure_Time           int64
Day_of_Week                        int64
Month                              int64
Airplane_Type                      int32
Weather_Score                    float64
Previous_Flight_Delay_Minutes    float64
Airline_Rating                   float64
Passenger_Load                   float64
Flight_Cancelled                   int64
dtype: object
```

In [30]:
```
#DEFINING THE FEATURES AND THE TARGET VARIABLE
x=dataset.drop(columns=['Flight ID', 'Airline', 'Flight_Distance', 'Origin_Airport'
       'Destination_Airport', 'Scheduled_Departure_Time', 'Day_of_Week',
       'Month', 'Airplane_Type', 'Weather_Score',
       'Previous_Flight_Delay_Minutes', 'Airline_Rating', 'Passenger_Load'])
y=dataset['Flight_Cancelled']
```

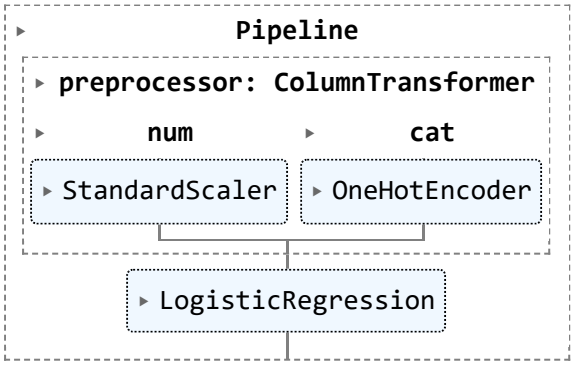
In [37]:
```
#PREPROCESSING STEPS FOR NUMERICAL FEATURES.
preprocessor=ColumnTransformer(transformers=[('num', StandardScaler(), numerical_fe
```

In [38]:
```
#DEFINING THE LOGISTIC REGRESSION MODEL PIPELINE.
model=Pipeline(steps=[('preprocessor', preprocessor), ('classifier', LogisticRegres
```

In [40]:
```
#SPLITTING THE DATA INTO TRAINING AND TESTING SETS.
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.2, random_state
```

In [41]:
```
#TRAINING THE MODEL.
model.fit(x_train, y_train)
```

Out[41]:



In [42]:
```
#MAKING PREDICTIONS ON THE TEST SET.
y_pred=model.predict(x_test)
```

In [44]:
```
#EVALUATING THE MODEL.
accuracy=accuracy_score(y_test, y_pred)
conf_matrix=confusion_matrix(y_test, y_pred)
class_report=classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print('Confusion Matrix')
```

```
print(conf_matrix)
print('Classification Report:')
print(class_report)
```

```
Accuracy: 1.0
Confusion Matrix
[[184   0]
 [  0 404]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       184
           1       1.00      1.00      1.00       404

    accuracy                           1.00       588
   macro avg       1.00      1.00      1.00       588
weighted avg       1.00      1.00      1.00       588
```

In [ ]: