```
#installing & importing modules and packages
%pip install --upgrade pip
%pip install pandas numpy openpyxl seaborn matplotlib

import pandas as pd
import numpy as np
import openpyxl
import matplotlib.pyplot as plt
import seaborn as sns

#loading training and testing dataset
train_data = pd.read_excel("TEAM15_DATASET1(TRAIN).xlsx")
test_data = pd.read_excel("TEAM15_DATASET2(TEST).xlsx")

#saving 'price' column in a variable
y = train_data['Price']

# Create price categories
# You can adjust these thresholds based on the distribution of your
prices
price_bins = [0, 6000, 12000, float('inf')]
price_labels = ['Low', 'Medium', 'High']
y_class = pd.cut(y, bins=price_bins, labels=price_labels, right=False)

# Display the new categorical target
print(y_class.value_counts())

#removing price column
train_data = train_data.drop('Price', axis=1)

#flagging to differentiate tran and test dataset
train_data['is_train'] = 1
test_data['is_train'] = 0

#merging both training and testing dataset
df = pd.concat([train_data, test_data], ignore_index=True)

print(f"Combined dataset shape: {df.shape}")
print(f"Training samples: {df['is_train'].sum()}")
print(f"Test samples: {(df['is_train'] == 0).sum()}")
print("\nFirst few rows of combined dataset:")
df.head()
```

```
Requirement already satisfied: pip in e:\python313\lib\site-packages
(25.2)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: pandas in e:\python313\lib\site-
packages (2.3.2)
Requirement already satisfied: numpy in e:\python313\lib\site-packages
(2.3.2)
```

```
Requirement already satisfied: openpyxl in e:\python313\lib\site-
packages (3.1.5)
Requirement already satisfied: seaborn in e:\python313\lib\site-
packages (0.13.2)
Requirement already satisfied: matplotlib in e:\python313\lib\site-
packages (3.10.6)
Requirement already satisfied: python-dateutil>=2.8.2 in e:\python313\
lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in e:\python313\lib\site-
packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in e:\python313\lib\
site-packages (from pandas) (2025.2)
Requirement already satisfied: et-xmlfile in e:\python313\lib\site-
packages (from openpyxl) (2.0.0)
Requirement already satisfied: contourpy>=1.0.1 in e:\python313\lib\
site-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in e:\python313\lib\site-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in e:\python313\lib\
site-packages (from matplotlib) (4.60.0)
Requirement already satisfied: kiwisolver>=1.3.1 in e:\python313\lib\
site-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in c:\users\jishn\
appdata\roaming\python\python313\site-packages (from matplotlib)
(25.0)
Requirement already satisfied: pillow>=8 in e:\python313\lib\site-
packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in e:\python313\lib\
site-packages (from matplotlib) (3.2.5)
Requirement already satisfied: six>=1.5 in e:\python313\lib\site-
packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
Price
Medium    4620
Low       3167
High      2896
Name: count, dtype: int64
Combined dataset shape: (13354, 11)
Training samples: 10683
Test samples: 2671

First few rows of combined dataset:

       Airline Date_of_Journey    Source Destination
Route  \
0       IndiGo      24/03/2019  Banglore   New Delhi              BLR
→ DEL
1    Air India       1/05/2019   Kolkata    Banglore  CCU → IXR → BBI
→ BLR
2  Jet Airways       9/06/2019     Delhi      Cochin  DEL → LKO → BOM
```

```
→ COK
3       IndiGo      12/05/2019   Kolkata    Banglore        CCU → NAG
→ BLR
4       IndiGo      01/03/2019  Banglore   New Delhi        BLR → NAG
→ DEL

  Dep_Time  Arrival_Time Duration Total_Stops Additional_Info
is_train
0    22:20  01:10 22 Mar   2h 50m    non-stop         No info
1
1    05:50         13:15   7h 25m     2 stops         No info
1
2    09:25  04:25 10 Jun      19h     2 stops         No info
1
3    18:05         23:30   5h 25m      1 stop         No info
1
4    16:50         21:35   4h 45m      1 stop         No info
1
```

```
#before preprocessing
df.shape
```

```
(13354, 11)
```

```
#removing 'Route' column
df.drop('Route', axis=1, inplace=True)
```

```
##checking for null values
df.isnull().sum()
```

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        1
Additional_Info    0
is_train           0
dtype: int64
```

```
#accessing row with null value
rows_with_nulls = df[df.isnull().any(axis=1)]
rows_with_nulls
```

```
        Airline Date_of_Journey Source Destination Dep_Time
Arrival_Time  \
9039  Air India       6/05/2019  Delhi      Cochin    09:45  09:25 07
May
```

```
     Duration Total_Stops Additional_Info  is_train
9039  23h 40m         NaN         No info         1
```

```
df.shape
```

```
(13354, 10)
```

```
#getting count of variuos data in 'additional info'
df['Additional_Info'].value_counts()
```

```
Additional_Info
No info                         10493
In-flight meal not included      2426
No check-in baggage included      396
1 Long layover                     20
Change airports                     8
Business class                      5
No Info                             3
1 Short layover                     1
Red-eye flight                      1
2 Long layover                      1
Name: count, dtype: int64
```

```
#dividing it to two categories
df['Additional_Info'] = df['Additional_Info'].apply(lambda x: 'No
info' if x == 'No info' else 'Others')
df['Additional_Info'].value_counts()
```

```
Additional_Info
No info    10493
Others      2861
Name: count, dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13354 entries, 0 to 13353
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          13354 non-null  object
 1   Date_of_Journey  13354 non-null  object
 2   Source           13354 non-null  object
 3   Destination      13354 non-null  object
 4   Dep_Time         13354 non-null  object
 5   Arrival_Time     13354 non-null  object
 6   Duration         13354 non-null  object
 7   Total_Stops      13353 non-null  object
 8   Additional_Info  13354 non-null  object
 9   is_train         13354 non-null  int64
```

```
dtypes: int64(1), object(9)
memory usage: 1.0+ MB
```

```
df.head()
```

```
       Airline Date_of_Journey    Source Destination Dep_Time
Arrival_Time  \
0       IndiGo      24/03/2019  Banglore   New Delhi    22:20  01:10
22 Mar
1    Air India       1/05/2019   Kolkata    Banglore    05:50
13:15
2  Jet Airways       9/06/2019     Delhi      Cochin    09:25  04:25
10 Jun
3       IndiGo      12/05/2019   Kolkata    Banglore    18:05
23:30
4       IndiGo      01/03/2019  Banglore   New Delhi    16:50
21:35

  Duration Total_Stops Additional_Info  is_train
0   2h 50m    non-stop         No info         1
1   7h 25m     2 stops         No info         1
2      19h     2 stops         No info         1
3   5h 25m      1 stop         No info         1
4   4h 45m      1 stop         No info         1
```

```
#feature engineering
df['Date'] = df['Date_of_Journey'].str.split('/').str[0]
df['Month'] = df['Date_of_Journey'].str.split('/').str[1]
df['Year'] = df['Date_of_Journey'].str.split('/').str[2]
df.drop('Date_of_Journey',axis=1,inplace=True)
df.head()
```

```
       Airline    Source Destination Dep_Time  Arrival_Time
Duration  \
0       IndiGo  Banglore   New Delhi    22:20  01:10 22 Mar   2h 50m

1    Air India   Kolkata    Banglore    05:50         13:15   7h 25m

2  Jet Airways     Delhi      Cochin    09:25  04:25 10 Jun      19h

3       IndiGo   Kolkata    Banglore    18:05         23:30   5h 25m

4       IndiGo  Banglore   New Delhi    16:50         21:35   4h 45m


  Total_Stops Additional_Info  is_train Date Month  Year
0    non-stop         No info         1   24    03  2019
1     2 stops         No info         1    1    05  2019
2     2 stops         No info         1    9    06  2019
3      1 stop         No info         1   12    05  2019
4      1 stop         No info         1   01    03  2019
```

```
df['Dep_hour'] = df['Dep_Time'].str.split(':').str[0]
df['Dep_minute'] = df['Dep_Time'].str.split(':').str[1]
df.drop('Dep_Time', axis=1, inplace=True)

df['Arrival_hour'] = df['Arrival_Time'].str.split('
').str[0].str.split(':').str[0]
df['Arrival_minute'] = df['Arrival_Time'].str.split('
').str[0].str.split(':').str[1]
df.drop('Arrival_Time', axis=1, inplace=True)

df.head()
```

```
       Airline    Source Destination Duration Total_Stops
Additional_Info  \
0       IndiGo  Banglore   New Delhi   2h 50m    non-stop         No
info
1    Air India   Kolkata    Banglore   7h 25m     2 stops         No
info
2  Jet Airways     Delhi      Cochin      19h     2 stops         No
info
3       IndiGo   Kolkata    Banglore   5h 25m      1 stop         No
info
4       IndiGo  Banglore   New Delhi   4h 45m      1 stop         No
info

   is_train Date Month  Year Dep_hour Dep_minute Arrival_hour
Arrival_minute
0         1   24    03  2019       22         20           01
10
1         1    1    05  2019       05         50           13
15
2         1    9    06  2019       09         25           04
25
3         1   12    05  2019       18         05           23
30
4         1   01    03  2019       16         50           21
35
```

```
#splitting duration to hours and minutes
def extract_duration_hours(duration_str):
    try:
        if 'h' in duration_str:
            return int(duration_str.split('h')[0])
        else:
            return 0
    except:
        return 0

def extract_duration_minutes(duration_str):
    try:
```

```
        if 'm' in duration_str:
            parts = duration_str.split(' ')
            for part in parts:
                if 'm' in part:
                    return int(part.replace('m', ''))
            return 0
        else:
            return 0
    except:
        return 0

# Apply the functions
df['Duration_hour'] = df['Duration'].apply(extract_duration_hours)
df['Duration_minute'] = df['Duration'].apply(extract_duration_minutes)
df.drop('Duration', axis=1, inplace=True)

#converting hours to minutes to redcue the features
df['Duration_minutes'] = df['Duration_hour']*60 +
df['Duration_minute']
df.drop(['Duration_hour','Duration_minute'], axis=1, inplace=True)

df.head()
```

```
       Airline    Source Destination Total_Stops Additional_Info
is_train  \
0       IndiGo  Banglore   New Delhi    non-stop         No info
1
1    Air India   Kolkata    Banglore     2 stops         No info
1
2  Jet Airways     Delhi      Cochin     2 stops         No info
1
3       IndiGo   Kolkata    Banglore      1 stop         No info
1
4       IndiGo  Banglore   New Delhi      1 stop         No info
1

  Date Month  Year Dep_hour Dep_minute Arrival_hour Arrival_minute  \
0   24    03  2019       22         20           01             10
1    1    05  2019       05         50           13             15
2    9    06  2019       09         25           04             25
3   12    05  2019       18         05           23             30
4   01    03  2019       16         50           21             35

   Duration_minutes
0               170
1               445
2              1140
3               325
4               285
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13354 entries, 0 to 13353
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Airline           13354 non-null  object
 1   Source            13354 non-null  object
 2   Destination       13354 non-null  object
 3   Total_Stops       13353 non-null  object
 4   Additional_Info   13354 non-null  object
 5   is_train          13354 non-null  int64 
 6   Date              13354 non-null  object
 7   Month             13354 non-null  object
 8   Year              13354 non-null  object
 9   Dep_hour          13354 non-null  object
 10  Dep_minute        13354 non-null  object
 11  Arrival_hour      13354 non-null  object
 12  Arrival_minute    13354 non-null  object
 13  Duration_minutes  13354 non-null  int64 
dtypes: int64(2), object(12)
memory usage: 1.4+ MB
```

```
df[['Date','Month','Year','Dep_hour','Dep_minute','Arrival_hour','Arrival_minute','Duration_minutes']].isnull().sum()
```

```
Date              0
Month             0
Year              0
Dep_hour          0
Dep_minute        0
Arrival_hour      0
Arrival_minute    0
Duration_minutes  0
dtype: int64
```

```
#converting to int datatype
col_to_convert =
['Date','Month','Year','Dep_hour','Dep_minute','Arrival_hour','Arrival_minute']
df[col_to_convert] = df[col_to_convert].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13354 entries, 0 to 13353
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Airline           13354 non-null  object
```

```
 1   Source            13354 non-null  object
 2   Destination       13354 non-null  object
 3   Total_Stops       13353 non-null  object
 4   Additional_Info   13354 non-null  object
 5   is_train          13354 non-null  int64
 6   Date              13354 non-null  int64
 7   Month             13354 non-null  int64
 8   Year              13354 non-null  int64
 9   Dep_hour          13354 non-null  int64
 10  Dep_minute        13354 non-null  int64
 11  Arrival_hour      13354 non-null  int64
 12  Arrival_minute    13354 non-null  int64
 13  Duration_minutes  13354 non-null  int64
dtypes: int64(9), object(5)
memory usage: 1.4+ MB
```

```
df.head()
```

```
       Airline    Source Destination Total_Stops Additional_Info
is_train  \
0       IndiGo  Banglore   New Delhi    non-stop         No info
1
1    Air India   Kolkata    Banglore     2 stops         No info
1
2  Jet Airways     Delhi      Cochin     2 stops         No info
1
3       IndiGo   Kolkata    Banglore      1 stop         No info
1
4       IndiGo  Banglore   New Delhi      1 stop         No info
1

   Date  Month  Year  Dep_hour  Dep_minute  Arrival_hour
Arrival_minute  \
0    24      3  2019        22          20             1
10
1     1      5  2019         5          50            13
15
2     9      6  2019         9          25             4
25
3    12      5  2019        18           5            23
30
4     1      3  2019        16          50            21
35

   Duration_minutes
0               170
1               445
2              1140
3               325
4               285
```

```
#accessing unique values in total_stops column
df['Total_Stops'].unique()
```

```
array(['non-stop', '2 stops', '1 stop', '3 stops', nan, '4 stops'],
      dtype=object)
```

```
#converting text data to numerical values
df['Total_Stops'] = df['Total_Stops'].map({'non-stop':0, '2 stops':2,
'1 stop':1, '3 stops':3, '4 stops':4})
df.head()
```

```
       Airline    Source Destination  Total_Stops Additional_Info
is_train  \
0       IndiGo  Banglore   New Delhi          0.0         No info
1
1    Air India   Kolkata    Banglore          2.0         No info
1
2  Jet Airways     Delhi      Cochin          2.0         No info
1
3       IndiGo   Kolkata    Banglore          1.0         No info
1
4       IndiGo  Banglore   New Delhi          1.0         No info
1

   Date  Month  Year  Dep_hour  Dep_minute  Arrival_hour
Arrival_minute  \
0    24      3  2019        22          20             1
10
1     1      5  2019         5          50            13
15
2     9      6  2019         9          25             4
25
3    12      5  2019        18           5            23
30
4     1      3  2019        16          50            21
35

   Duration_minutes
0               170
1               445
2              1140
3               325
4               285
```

```
#filling null values with mode
df['Total_Stops'].fillna(df['Total_Stops'].mode()[0],inplace=True)
```

```
C:\Users\jishn\AppData\Local\Temp\ipykernel_22444\3100553634.py:2:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
```

```
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df['Total_Stops'].fillna(df['Total_Stops'].mode()[0],inplace=True)
```

```
df.isnull().sum()
```

```
Airline             0
Source              0
Destination         0
Total_Stops         0
Additional_Info     0
is_train            0
Date                0
Month               0
Year                0
Dep_hour            0
Dep_minute          0
Arrival_hour        0
Arrival_minute      0
Duration_minutes    0
dtype: int64
```

```
#printing unique values for the columns
print(df['Airline'].unique(),df['Source'].unique(),df['Destination'].u
nique())
```

```
['IndiGo' 'Air India' 'Jet Airways' 'SpiceJet' 'Multiple carriers'
'GoAir'
 'Vistara' 'Air Asia' 'Vistara Premium economy' 'Jet Airways Business'
 'Multiple carriers Premium economy' 'Trujet'] ['Banglore' 'Kolkata'
'Delhi' 'Chennai' 'Mumbai'] ['New Delhi' 'Banglore' 'Cochin' 'Kolkata'
'Delhi' 'Hyderabad']
```

```
#encoding categorical columns
%pip install scikit-learn
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
import pandas as pd

le = LabelEncoder()
df['Additional_Info'] = le.fit_transform(df['Additional_Info'])

cols_to_encode = ['Airline', 'Source', 'Destination']
```

```
ohe = OneHotEncoder(sparse_output=False, drop=None)
encoded = ohe.fit_transform(df[cols_to_encode])
encoded_cols = ohe.get_feature_names_out(cols_to_encode)
encoded_df = pd.DataFrame(encoded, columns=encoded_cols,
index=df.index)
df = pd.concat([df.drop(cols_to_encode, axis=1), encoded_df], axis=1)

df.head()
```

```
Requirement already satisfied: scikit-learn in e:\python313\lib\site-
packages (1.7.1)
Requirement already satisfied: numpy>=1.22.0 in e:\python313\lib\site-
packages (from scikit-learn) (2.3.2)
Requirement already satisfied: scipy>=1.8.0 in e:\python313\lib\site-
packages (from scikit-learn) (1.16.1)
Requirement already satisfied: joblib>=1.2.0 in e:\python313\lib\site-
packages (from scikit-learn) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in e:\python313\
lib\site-packages (from scikit-learn) (3.6.0)
Note: you may need to restart the kernel to use updated packages.
```

```
   Total_Stops  Additional_Info  is_train  Date  Month  Year  Dep_hour
\
0          0.0                0         1    24      3  2019        22

1          2.0                0         1     1      5  2019         5

2          2.0                0         1     9      6  2019         9

3          1.0                0         1    12      5  2019        18

4          1.0                0         1     1      3  2019        16


   Dep_minute  Arrival_hour  Arrival_minute  ...  Source_Chennai  \
0          20             1              10  ...             0.0
1          50            13              15  ...             0.0
2          25             4              25  ...             0.0
3           5            23              30  ...             0.0
4          50            21              35  ...             0.0

   Source_Delhi  Source_Kolkata  Source_Mumbai
Destination_Banglore  \
0           0.0             0.0            0.0                    0.0

1           0.0             1.0            0.0                    1.0

2           1.0             0.0            0.0                    0.0

3           0.0             1.0            0.0                    1.0
```

```
4           0.0              0.0             0.0                      0.0


   Destination_Cochin  Destination_Delhi  Destination_Hyderabad  \
0                 0.0                0.0                    0.0
1                 0.0                0.0                    0.0
2                 1.0                0.0                    0.0
3                 0.0                0.0                    0.0
4                 0.0                0.0                    0.0

   Destination_Kolkata  Destination_New Delhi
0                  0.0                    1.0
1                  0.0                    0.0
2                  0.0                    0.0
3                  0.0                    0.0
4                  0.0                    1.0

[5 rows x 34 columns]
```

```
#scale numeric features to mean 0 and std dev 1
from sklearn.preprocessing import StandardScaler
import pandas as pd

#splitting dataset back to training and testing using flag
train_clean = df[df['is_train'] == 1].drop('is_train', axis=1)
test_clean = df[df['is_train'] == 0].drop('is_train', axis=1)

scaler = StandardScaler()
X_train = scaler.fit_transform(train_clean)
X_test = scaler.transform(test_clean)

X_train_df = pd.DataFrame(X_train, columns=train_clean.columns)
X_test_df = pd.DataFrame(X_test, columns=test_clean.columns)

X_train_df.head()
```

```
   Total_Stops  Additional_Info      Date     Month  Year  Dep_hour  \
0    -1.220744        -0.529309  1.237383 -1.467490   0.0  1.654259
1     1.741483        -0.529309 -1.475239  0.250276   0.0 -1.303095
2     1.741483        -0.529309 -0.531719  1.109160   0.0 -0.607247
3     0.260370        -0.529309 -0.177898  0.250276   0.0  0.958411
4     0.260370        -0.529309 -1.475239 -1.467490   0.0  0.610487

   Dep_minute  Arrival_hour  Arrival_minute  Duration_minutes  ...  \
0   -0.235050     -1.800427       -0.890057         -0.931583  ...
1    1.363492     -0.050851       -0.587124         -0.390072  ...
2    0.031373     -1.363033        0.018744          0.978475  ...
3   -1.034321      1.407129        0.321677         -0.626367  ...
4    1.363492      1.115533        0.624611         -0.705132  ...

   Source_Chennai  Source_Delhi  Source_Kolkata  Source_Mumbai  \
```

```
0        -0.19231     -0.859188       -0.606227      -0.264193
1        -0.19231     -0.859188        1.649546      -0.264193
2        -0.19231      1.163890       -0.606227      -0.264193
3        -0.19231     -0.859188        1.649546      -0.264193
4        -0.19231     -0.859188       -0.606227      -0.264193

   Destination_Banglore  Destination_Cochin  Destination_Delhi  \
0             -0.606227           -0.859188          -0.366493
1              1.649546           -0.859188          -0.366493
2             -0.606227            1.163890          -0.366493
3              1.649546           -0.859188          -0.366493
4             -0.606227           -0.859188          -0.366493

   Destination_Hyderabad  Destination_Kolkata  Destination_New Delhi
0              -0.264193             -0.19231               3.234571
1              -0.264193             -0.19231              -0.309160
2              -0.264193             -0.19231              -0.309160
3              -0.264193             -0.19231              -0.309160
4              -0.264193             -0.19231               3.234571

[5 rows x 33 columns]
```

```
X_test_df.head()
```

```
   Total_Stops  Additional_Info      Date     Month  Year  Dep_hour  \
0     0.260370        -0.529309 -0.885539  1.109160   0.0  0.784449
1     0.260370        -0.529309 -0.177898  0.250276   0.0 -1.129133
2     0.260370         1.889256  0.883563  0.250276   0.0  1.132373
3     0.260370        -0.529309  0.883563  0.250276   0.0 -0.781209
4    -1.220744        -0.529309  1.237383  1.109160   0.0  1.828221

   Dep_minute  Arrival_hour  Arrival_minute  Duration_minutes  ...  \
0    0.297797     -1.363033        0.018744          0.023446  ...
1   -0.235050     -0.488245       -0.284190         -0.793743  ...
2   -0.501474      0.823937       -1.495925          1.539677  ...
3   -1.300745      1.115533       -1.495925          0.269587  ...
4    1.629915     -1.654629        1.230478         -0.931583  ...

   Source_Chennai  Source_Delhi  Source_Kolkata  Source_Mumbai  \
0        -0.19231      1.163890       -0.606227      -0.264193
1        -0.19231     -0.859188        1.649546      -0.264193
2        -0.19231      1.163890       -0.606227      -0.264193
3        -0.19231      1.163890       -0.606227      -0.264193
4        -0.19231     -0.859188       -0.606227      -0.264193

   Destination_Banglore  Destination_Cochin  Destination_Delhi  \
0             -0.606227            1.163890          -0.366493
1              1.649546           -0.859188          -0.366493
2             -0.606227            1.163890          -0.366493
3             -0.606227            1.163890          -0.366493
```

```
4              -0.606227           -0.859188           2.728564

   Destination_Hyderabad  Destination_Kolkata  Destination_New Delhi
0              -0.264193             -0.19231               -0.30916
1              -0.264193             -0.19231               -0.30916
2              -0.264193             -0.19231               -0.30916
3              -0.264193             -0.19231               -0.30916
4              -0.264193             -0.19231               -0.30916

[5 rows x 33 columns]
```

```
X_train_df.info()
```

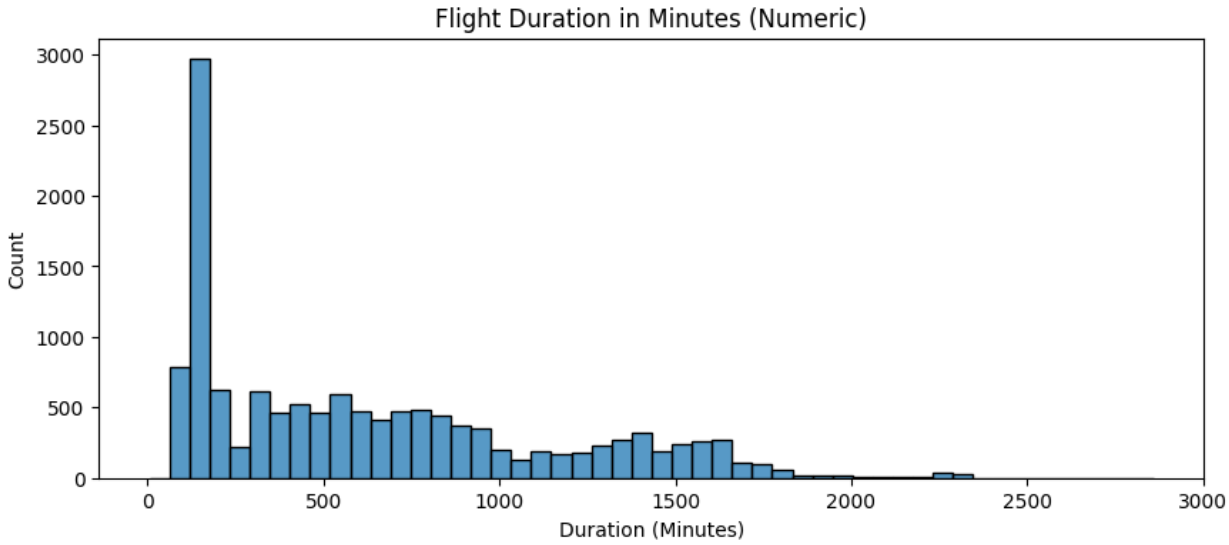
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 33 columns):
 #   Column                                          Non-Null Count  Dtype

---  ------                                          --------------  -----

 0   Total_Stops                                     10683 non-null
float64
 1   Additional_Info                                 10683 non-null
float64
 2   Date                                            10683 non-null
float64
 3   Month                                           10683 non-null
float64
 4   Year                                            10683 non-null
float64
 5   Dep_hour                                        10683 non-null
float64
 6   Dep_minute                                      10683 non-null
float64
 7   Arrival_hour                                    10683 non-null
float64
 8   Arrival_minute                                  10683 non-null
float64
 9   Duration_minutes                                10683 non-null
float64
 10  Airline_Air Asia                                10683 non-null
float64
 11  Airline_Air India                               10683 non-null
float64
 12  Airline_GoAir                                   10683 non-null
float64
 13  Airline_IndiGo                                  10683 non-null
float64
 14  Airline_Jet Airways                             10683 non-null
float64
```

```
 15  Airline_Jet Airways Business                  10683 non-null
float64
 16  Airline_Multiple carriers                     10683 non-null
float64
 17  Airline_Multiple carriers Premium economy     10683 non-null
float64
 18  Airline_SpiceJet                              10683 non-null
float64
 19  Airline_Trujet                                10683 non-null
float64
 20  Airline_Vistara                               10683 non-null
float64
 21  Airline_Vistara Premium economy               10683 non-null
float64
 22  Source_Banglore                               10683 non-null
float64
 23  Source_Chennai                                10683 non-null
float64
 24  Source_Delhi                                  10683 non-null
float64
 25  Source_Kolkata                                10683 non-null
float64
 26  Source_Mumbai                                 10683 non-null
float64
 27  Destination_Banglore                          10683 non-null
float64
 28  Destination_Cochin                            10683 non-null
float64
 29  Destination_Delhi                             10683 non-null
float64
 30  Destination_Hyderabad                         10683 non-null
float64
 31  Destination_Kolkata                           10683 non-null
float64
 32  Destination_New Delhi                         10683 non-null
float64
dtypes: float64(33)
memory usage: 2.7 MB

train_clean_scaled = pd.DataFrame(X_train,
columns=train_clean.columns)
train_clean_scaled['Price'] = y.values

test_clean_scaled = pd.DataFrame(X_test, columns=test_clean.columns)

train_clean_scaled.to_excel("Cleaned_Train.xlsx", index=False)
test_clean_scaled.to_excel("Cleaned_Test.xlsx", index=False)

print("Cleaned and scaled train/test datasets saved successfully!!")
```

```
Cleaned and scaled train/test datasets saved successfully!!
```

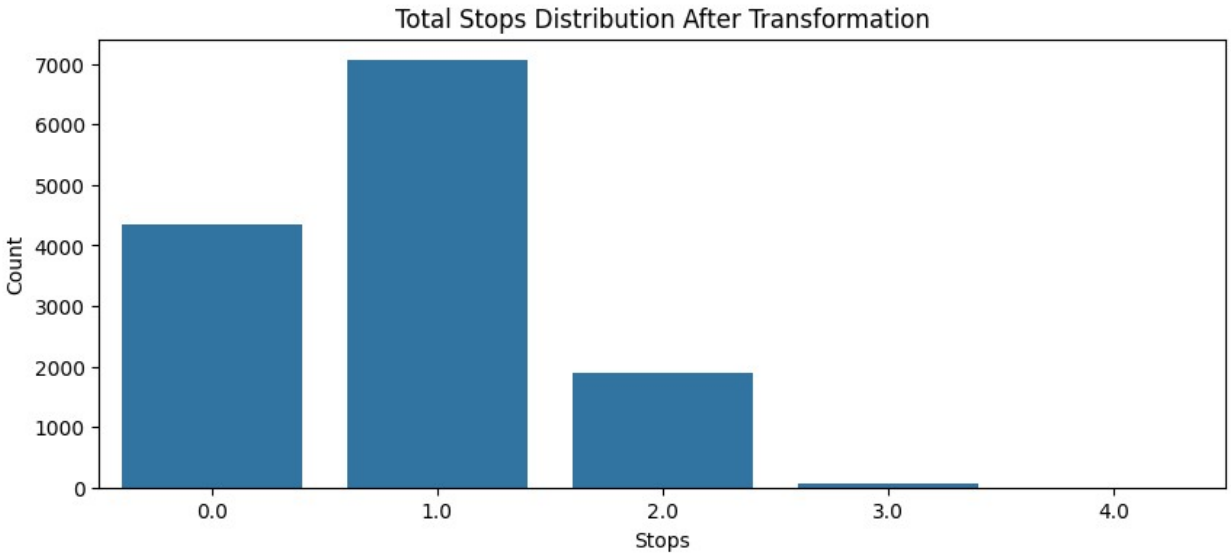
Plot for Flight Distribution After Conversion to Minutes

```
plt.figure(figsize=(10,4))
sns.histplot(df['Duration_minutes'], bins=50)
plt.title("Flight Duration in Minutes (Numeric)")
plt.xlabel("Duration (Minutes)")
plt.ylabel("Count")
plt.show()
```



Plot for Total Stops Distribution after Encoding

```
plt.figure(figsize=(10,4))
sns.countplot(x='Total_Stops', data=df)
plt.title("Total Stops Distribution After Transformation")
plt.xlabel("Stops")
plt.ylabel("Count")
plt.show()
```




```
import pandas as pd
price_bins = [0, 6000, 12000, float('inf')]
price_labels = ['Low', 'Medium', 'High']
y_class = pd.cut(y, bins=price_bins, labels=price_labels, right=False)

print("Target variable converted to classes:")
print(y_class.value_counts())
print("\\n" + "="*40 + "\\n")

# Step 2: Import the classifier and split the data for validation
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# We split the preprocessed training data (X_train_df) into a new
training set and a validation set.
# This allows us to evaluate the model before using the final test
data.
X_train, X_val, y_train_class, y_val_class = train_test_split(
    X_train_df, y_class, test_size=0.2, random_state=42,
stratify=y_class # stratify ensures balanced classes
)

# Step 3: Initialize and train the KNN Classifier
knn_classifier = KNeighborsClassifier(n_neighbors=5) # We'll start
with K=5
knn_classifier.fit(X_train, y_train_class)

print("KNN Classifier model trained successfully!")
```

```
Target variable converted to classes:
Price
Medium    4620
```

```
Low       3167
High      2896
Name: count, dtype: int64
\n=======================================\n
KNN Classifier model trained successfully!
```

```
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

# Make predictions on the VALIDATION set (X_val), not the training set
y_pred_val = knn_classifier.predict(X_val)

print("--- Accuracy Score ---")
print(f"Accuracy: {accuracy_score(y_val_class, y_pred_val):.2f}\\n")

# 2. Confusion Matrix: Shows correct vs. incorrect predictions for
each class
print("--- Confusion Matrix ---")
print(confusion_matrix(y_val_class, y_pred_val))
print("\\n")

# 3. Classification Report: Provides precision, recall, and F1-score
print("--- Classification Report ---")
print(classification_report(y_val_class, y_pred_val))
```

```
--- Accuracy Score ---
Accuracy: 0.86\n
--- Confusion Matrix ---
[[485   7  87]
 [  2 564  68]
 [ 71  66 787]]
\n
--- Classification Report ---
              precision    recall  f1-score   support

        High       0.87      0.84      0.85       579
         Low       0.89      0.89      0.89       634
      Medium       0.84      0.85      0.84       924

    accuracy                           0.86      2137
   macro avg       0.86      0.86      0.86      2137
weighted avg       0.86      0.86      0.86      2137
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import numpy as np
import matplotlib.pyplot as plt

# We will test K values from 1 to 39
k_range = range(1, 40)
```

```
error_rate = []

for i in k_range:
    # Use KNeighborsClassifier
    knn = KNeighborsClassifier(n_neighbors=i)

    # Fit the model using the CORRECT training data and CLASSIFIED
labels
    knn.fit(X_train, y_train_class)

    # Make predictions on the VALIDATION set
    pred_i = knn.predict(X_val)

    # Calculate the error rate (1 - accuracy) on the validation set
    # We compare the predictions (pred_i) with the actual validation
labels (y_val_class)
    error_rate.append(1 - accuracy_score(y_val_class, pred_i))

# --- Plotting the error rate ---
plt.figure(figsize=(12, 6))
plt.plot(k_range, error_rate, color='blue', linestyle='dashed',
marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value for Classification')
plt.xlabel('K Value')
plt.ylabel('Error Rate (1 - Accuracy)')
plt.show()
```