airline-data-eda-and-prediction

June 27, 2024

[1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import networkx as nx
```

[2]:
```python
df = pd.read_csv('/kaggle/input/flight-price-data/flight_dataset.csv')
```

1 Data Cleaning

[3]:
```python
df.head()
```

[3]:
```
       Airline    Source Destination  Total_Stops  Price  Date  Month  Year  \
0       IndiGo  Banglore   New Delhi            0   3897    24      3  2019
1    Air India   Kolkata    Banglore            2   7662     1      5  2019
2  Jet Airways     Delhi      Cochin            2  13882     9      6  2019
3       IndiGo   Kolkata    Banglore            1   6218    12      5  2019
4       IndiGo  Banglore   New Delhi            1  13302     1      3  2019

   Dep_hours  Dep_min  Arrival_hours  Arrival_min  Duration_hours  \
0         22       20              1           10               2
1          5       50             13           15               7
2          9       25              4           25              19
3         18        5             23           30               5
4         16       50             21           35               4

   Duration_min
0            50
1            25
2             0
3            25
4            45
```

[4]:
```python
df.describe()
```

[4]:
```
         Total_Stops         Price          Date         Month     Year  \
count   10683.000000  10683.000000  10683.000000  10683.000000  10683.0
mean        0.824207   9087.064121     13.508378      4.708602   2019.0
std         0.675199   4611.359167      8.479277      1.164357      0.0
min         0.000000   1759.000000      1.000000      3.000000   2019.0
25%         0.000000   5277.000000      6.000000      3.000000   2019.0
50%         1.000000   8372.000000     12.000000      5.000000   2019.0
75%         1.000000  12373.000000     21.000000      6.000000   2019.0
max         4.000000  79512.000000     27.000000      6.000000   2019.0

           Dep_hours       Dep_min  Arrival_hours   Arrival_min  \
count   10683.000000  10683.000000   10683.000000  10683.000000
mean       12.490686     24.411214      13.348778     24.690630
std         5.748650     18.767980       6.859125     16.506036
min         0.000000      0.000000       0.000000      0.000000
25%         8.000000      5.000000       8.000000     10.000000
50%        11.000000     25.000000      14.000000     25.000000
75%        18.000000     40.000000      19.000000     35.000000
max        23.000000     55.000000      23.000000     55.000000

       Duration_hours  Duration_min
count    10683.000000  10683.000000
mean        10.246560     28.327249
std          8.494988     16.946113
min          1.000000      0.000000
25%          2.000000     15.000000
50%          8.000000     30.000000
75%         15.000000     45.000000
max         47.000000     55.000000
```

[5]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          10683 non-null  object
 1   Source           10683 non-null  object
 2   Destination      10683 non-null  object
 3   Total_Stops      10683 non-null  int64
 4   Price            10683 non-null  int64
 5   Date             10683 non-null  int64
 6   Month            10683 non-null  int64
 7   Year             10683 non-null  int64
 8   Dep_hours        10683 non-null  int64
 9   Dep_min          10683 non-null  int64
```

```
 10  Arrival_hours   10683 non-null  int64
 11  Arrival_min     10683 non-null  int64
 12  Duration_hours  10683 non-null  int64
 13  Duration_min    10683 non-null  int64
dtypes: int64(11), object(3)
memory usage: 1.1+ MB
```

[6]:
```
df.isna().sum()
```

[6]:
```
Airline           0
Source            0
Destination       0
Total_Stops       0
Price             0
Date              0
Month             0
Year              0
Dep_hours         0
Dep_min           0
Arrival_hours     0
Arrival_min       0
Duration_hours    0
Duration_min      0
dtype: int64
```

[7]:
```
df.duplicated().sum()
```

[7]:
```
222
```

[8]:
```
# Droping duplicated rows
df.drop_duplicates(inplace = True)
```

[9]:
```
df.shape
```

[9]:
```
(10461, 14)
```

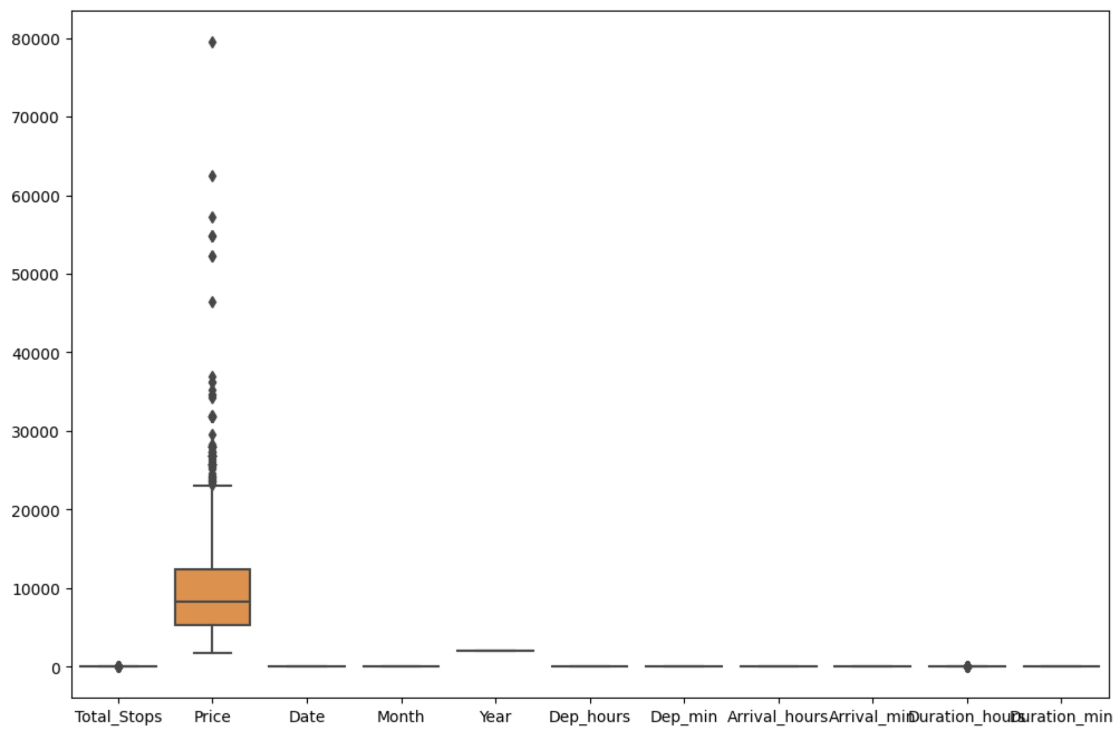
[10]:
```
df.head()
```

[10]:
```
       Airline    Source Destination  Total_Stops  Price  Date  Month  Year  \
0       IndiGo  Banglore   New Delhi            0   3897    24      3  2019
1    Air India   Kolkata    Banglore            2   7662     1      5  2019
2  Jet Airways     Delhi      Cochin            2  13882     9      6  2019
3       IndiGo   Kolkata    Banglore            1   6218    12      5  2019
4       IndiGo  Banglore   New Delhi            1  13302     1      3  2019

   Dep_hours  Dep_min  Arrival_hours  Arrival_min  Duration_hours  \
0         22       20              1           10               2
```

1	5	50	13	15	7
2	9	25	4	25	19
3	18	5	23	30	5
4	16	50	21	35	4

	Duration_min
0	50
1	25
2	0
3	25
4	45

[11]:
```
plt.figure(figsize=(12,8))
sns.boxplot(df)
plt.show()
```



[12]:
```
# Removing the outliers from price column

q1 = df['Price'].quantile(0.25)
q3 = df['Price'].quantile(0.75)

iqr = q3 - q1

ll = q1 - 1.5 * iqr
```

```
ul = q3 + 1.5 * iqr

outliers = (df['Price'] < ll ) |( df['Price'] > ul)
df = df[~outliers]
```

[13]:

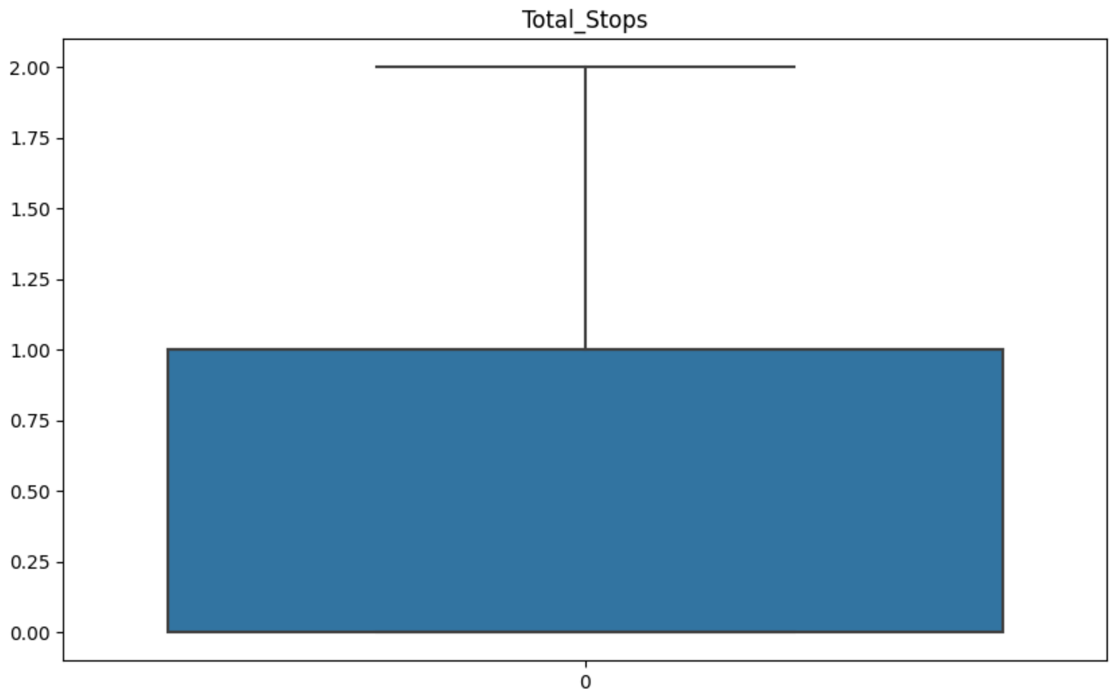
```
# Removing the outliers from Duration_hours column

q1 = df['Duration_hours'].quantile(0.25)
q3 = df['Duration_hours'].quantile(0.75)

iqr = q3 - q1

ll = q1 - 1.5 * iqr
ul = q3 + 1.5 * iqr

outliers = (df['Duration_hours'] < ll ) |( df['Duration_hours'] > ul)
df = df[~outliers]
```

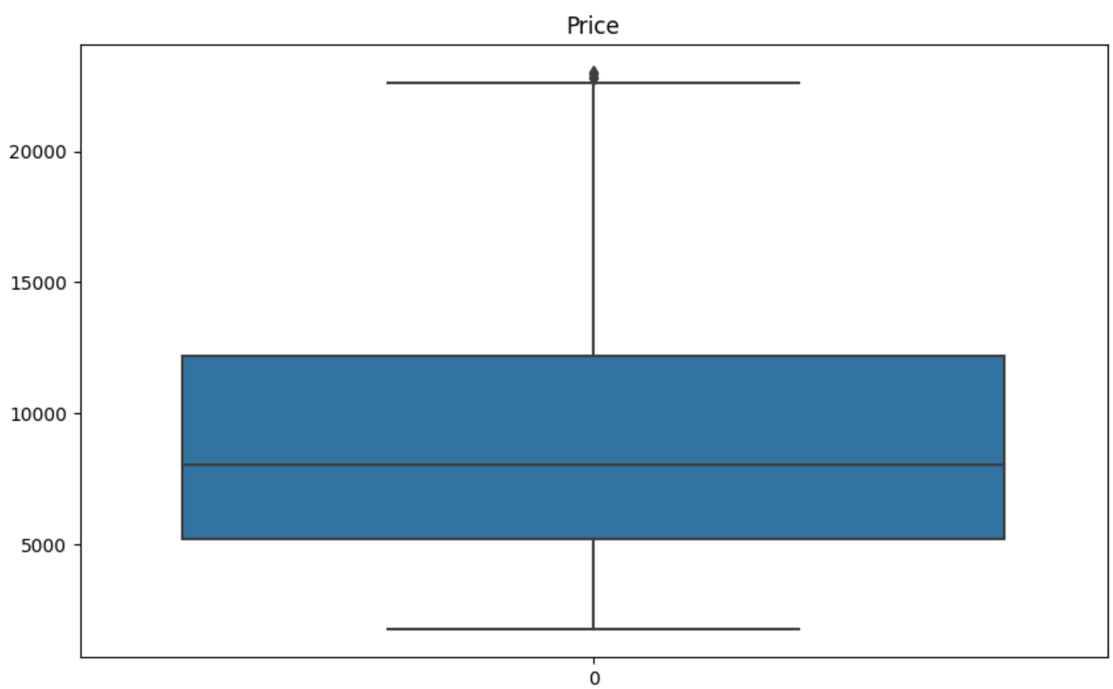
[14]:

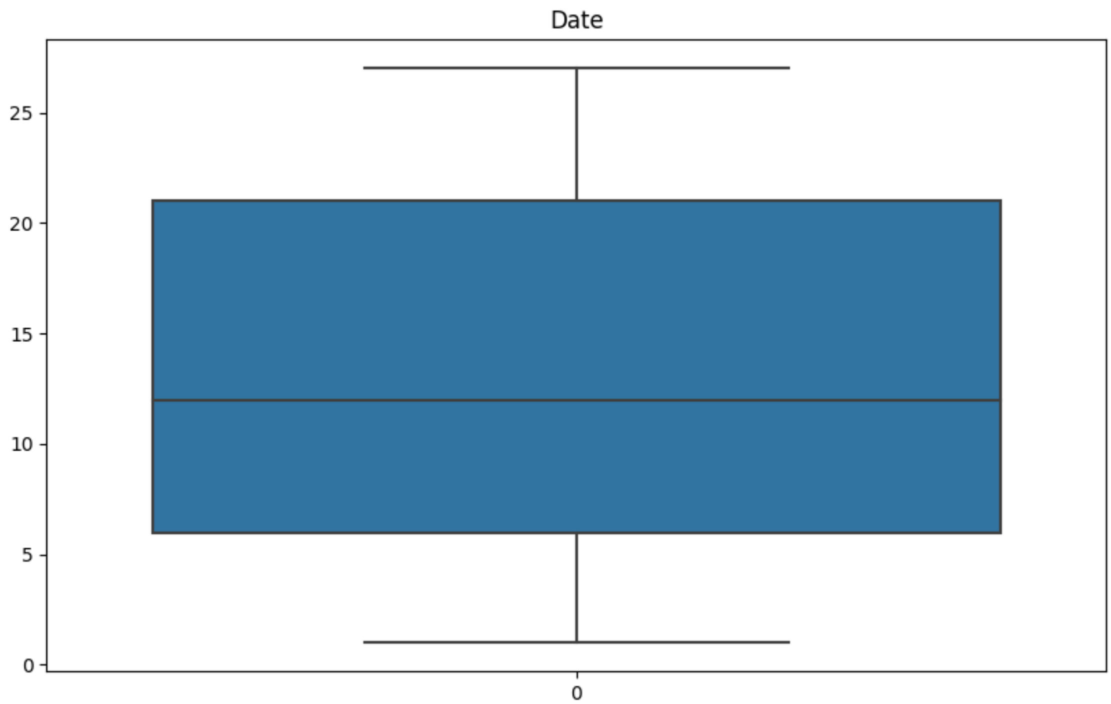
```
# Removing the outliers from Total_Stops column

q1 = df['Total_Stops'].quantile(0.25)
q3 = df['Total_Stops'].quantile(0.75)

iqr = q3 - q1

ll = q1 - 1.5 * iqr
ul = q3 + 1.5 * iqr

outliers = (df['Total_Stops'] < ll ) |( df['Total_Stops'] > ul)
df = df[~outliers]
```

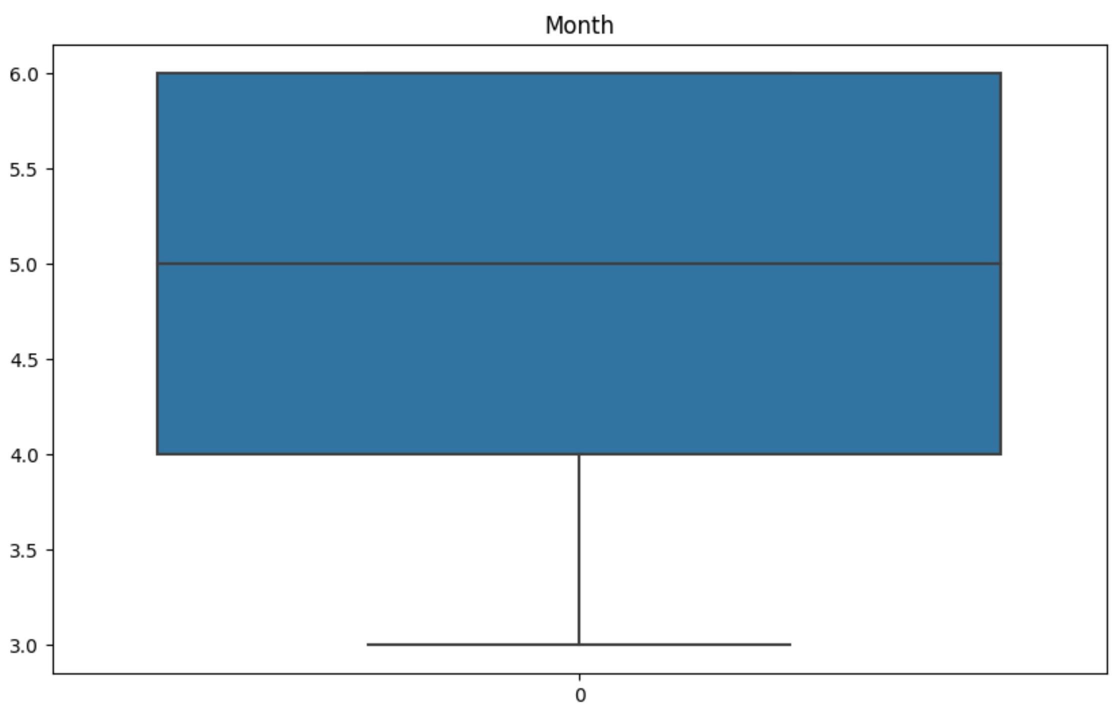
[15]:

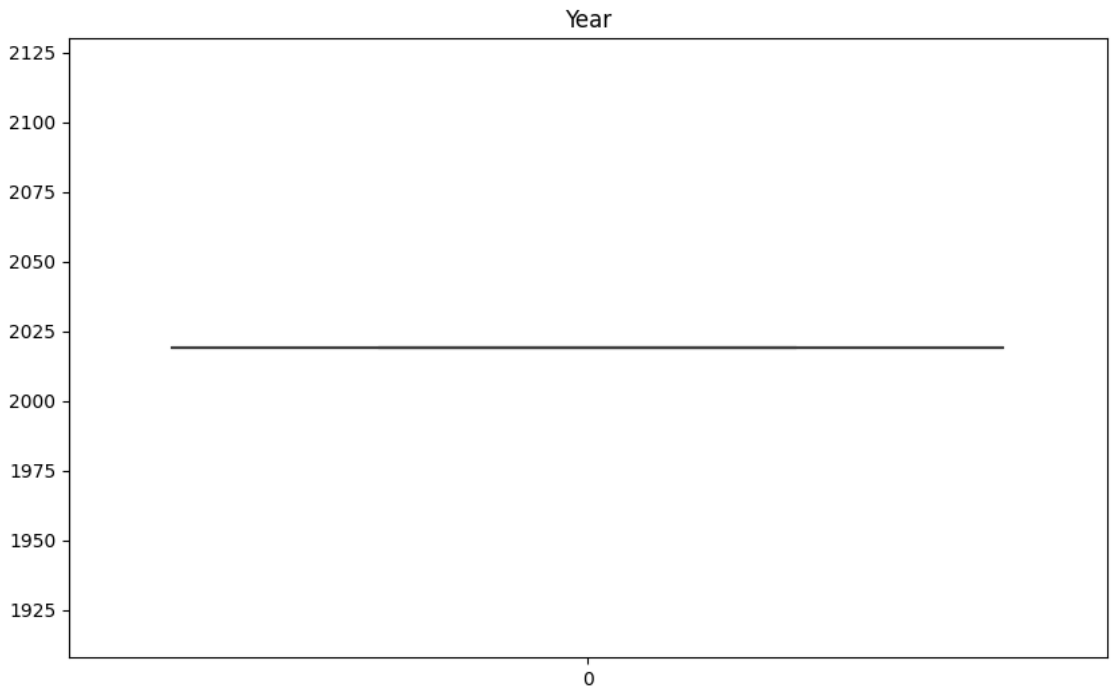
```
num_cols = df.select_dtypes(include=['int64'])
for i in num_cols.columns:
    plt.figure(figsize=(10,6))
    sns.boxplot(df[i])
    plt.title(i)
    plt.show()
```

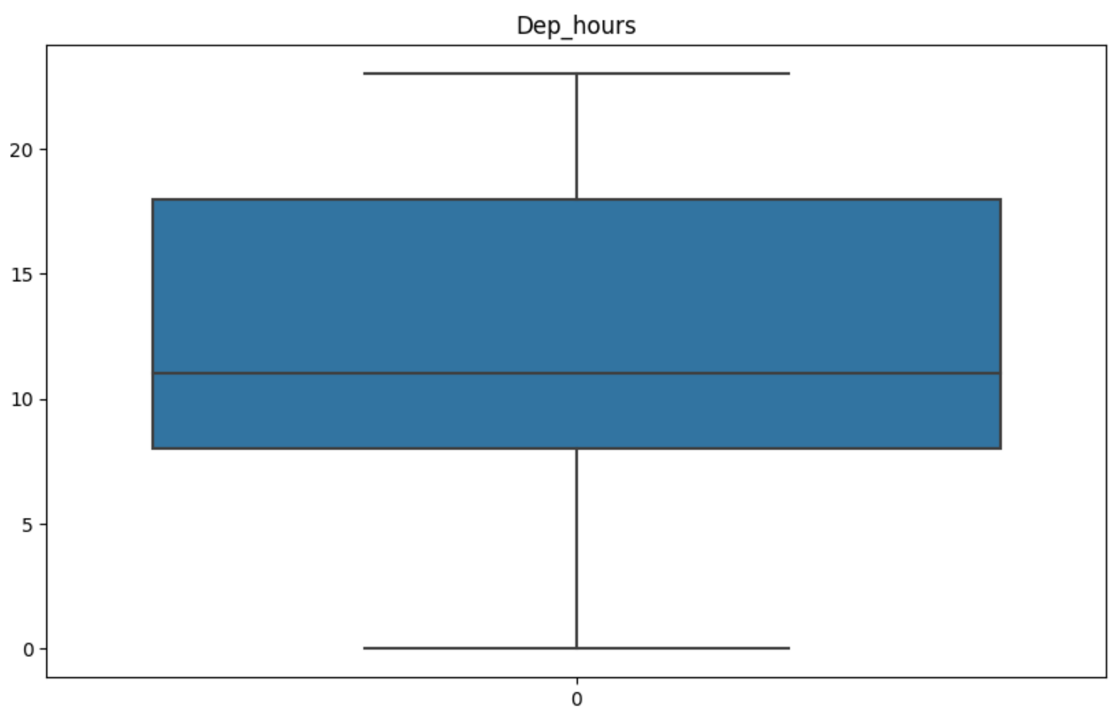

Total_Stops
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0

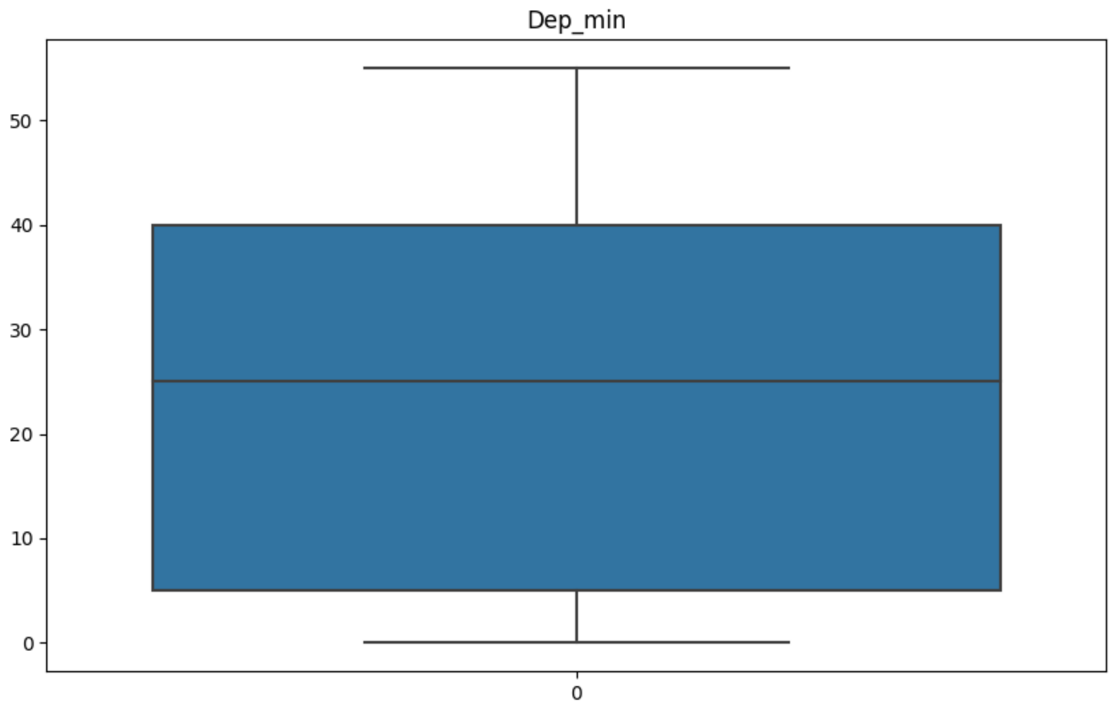

Price
20000
15000
10000
5000
0

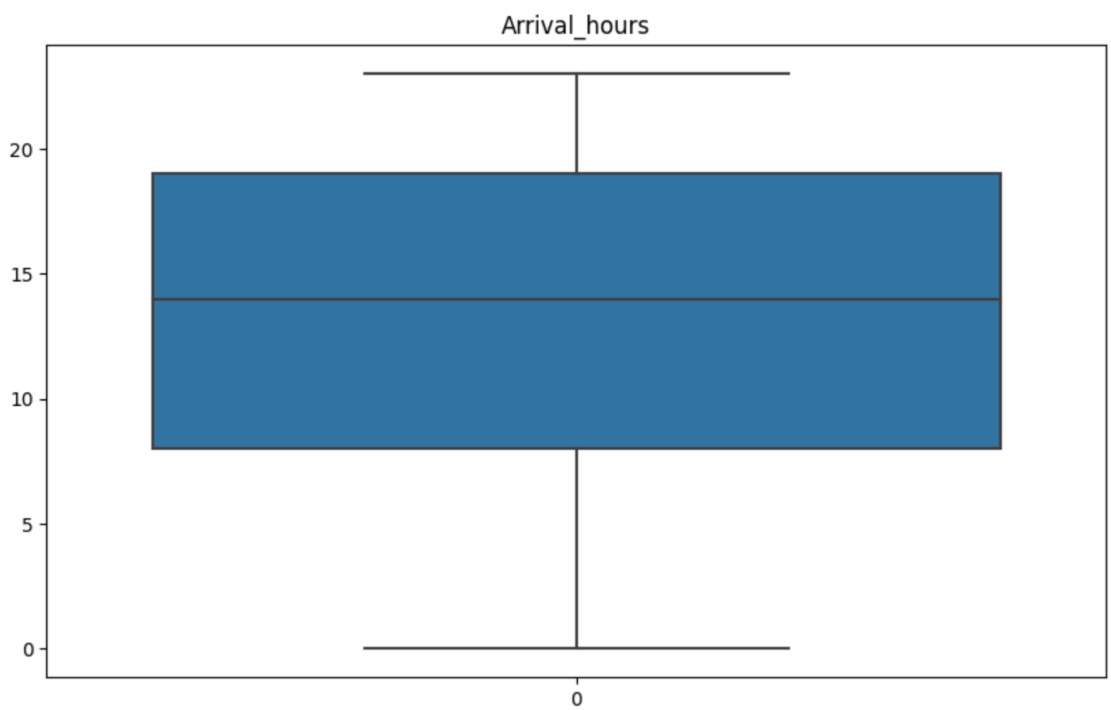

Date
25
20
15
10
5
0
0

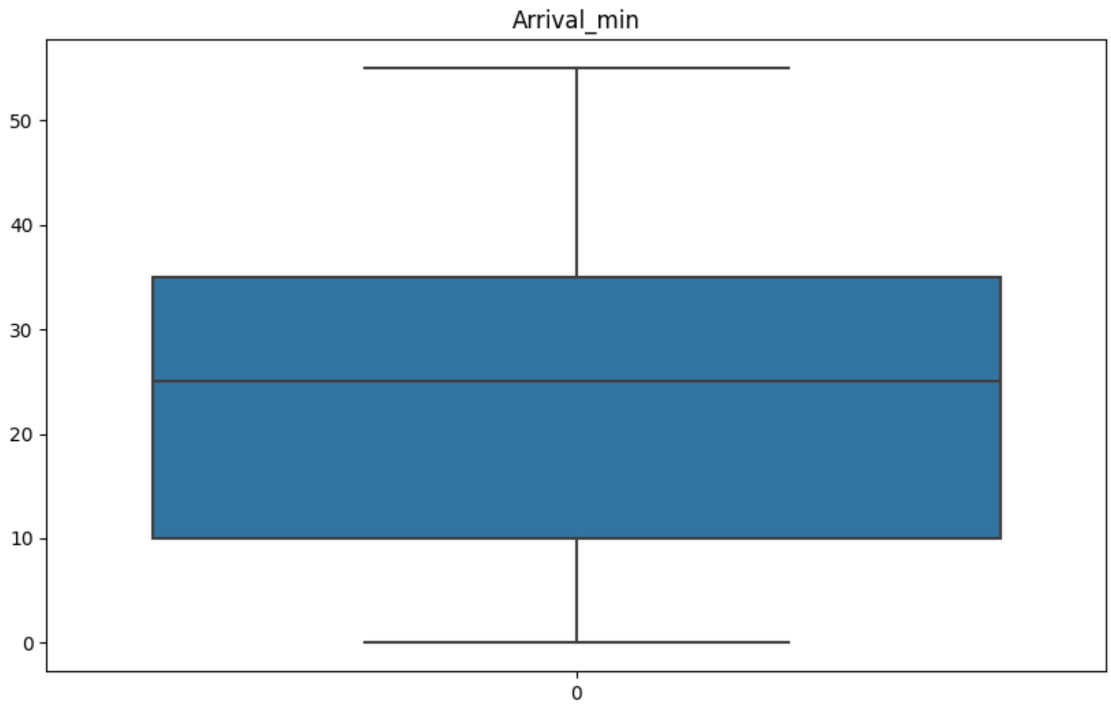

Month
6.0
5.5
5.0
4.5
4.0
3.5
3.0
0

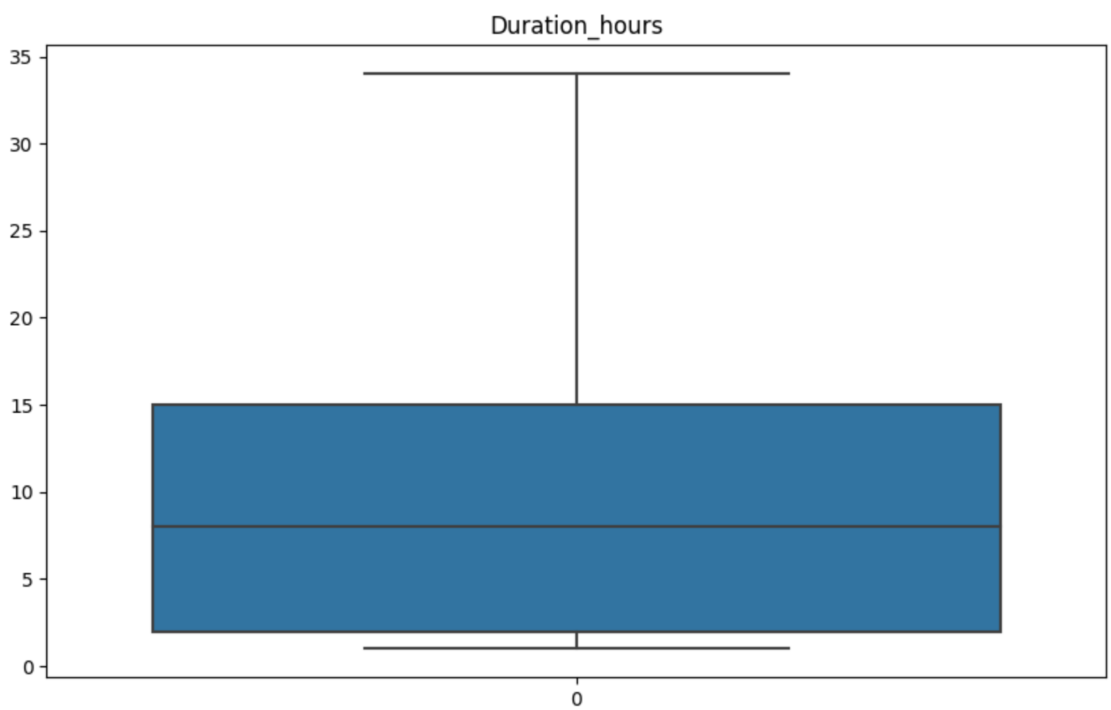

Year
2125
2100
2075
2050
2025
2000
1975
1950
1925
0

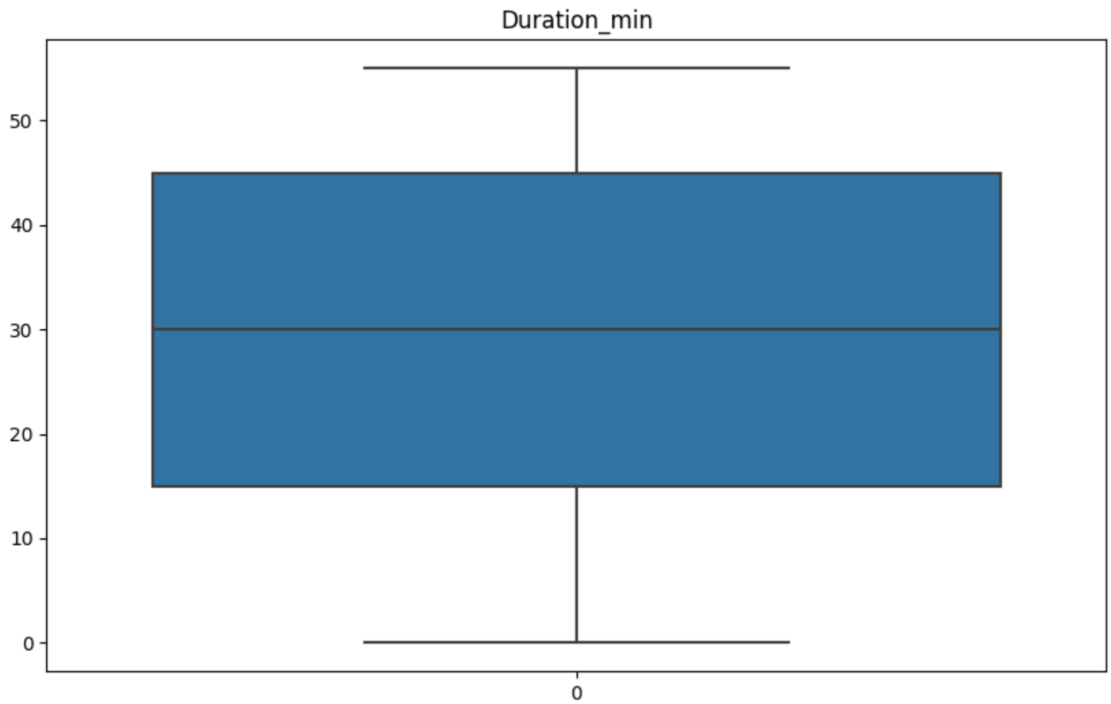

Dep_hours
20
15
10
5
0
0


Dep_min
50
40
30
20
10
0
0


Arrival_hours
20
15
10
5
0
0


Arrival_min
50
40
30
20
10
0
0

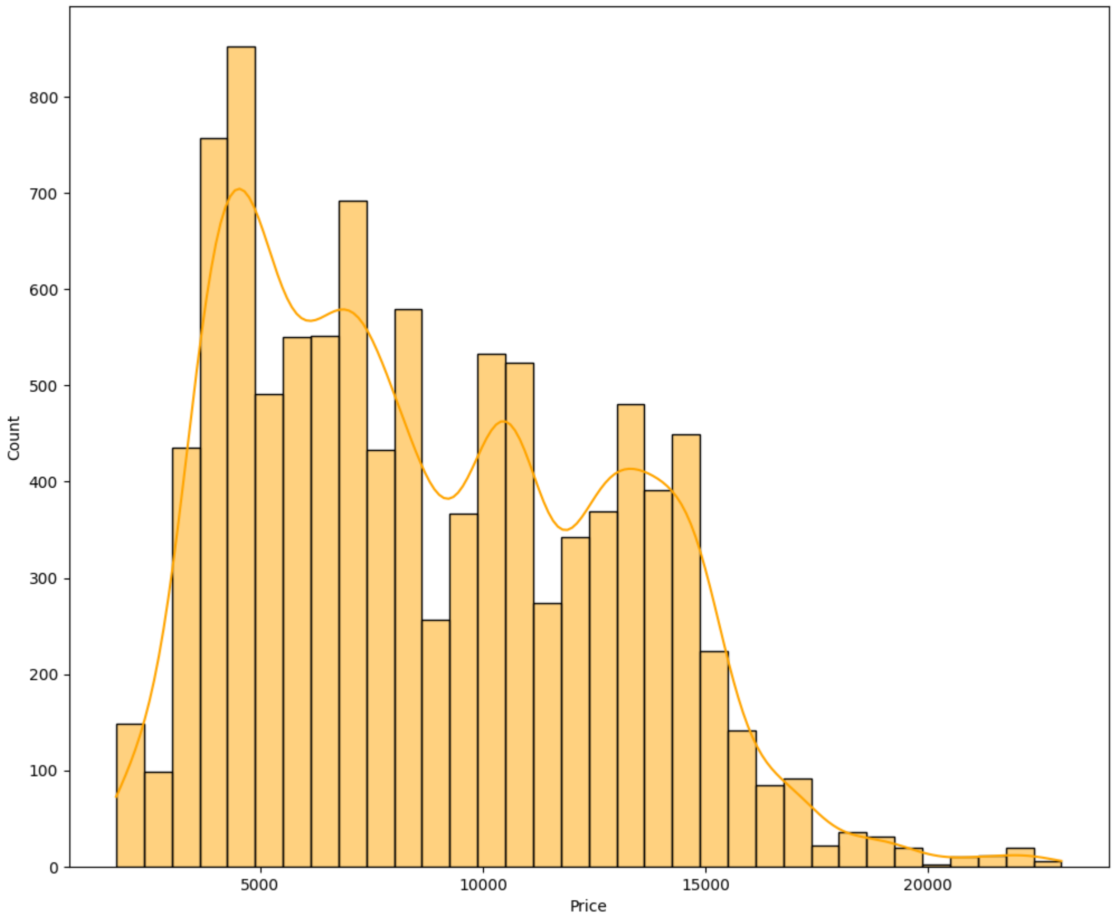

Duration_hours
35
30
25
20
15
10
5
0
0



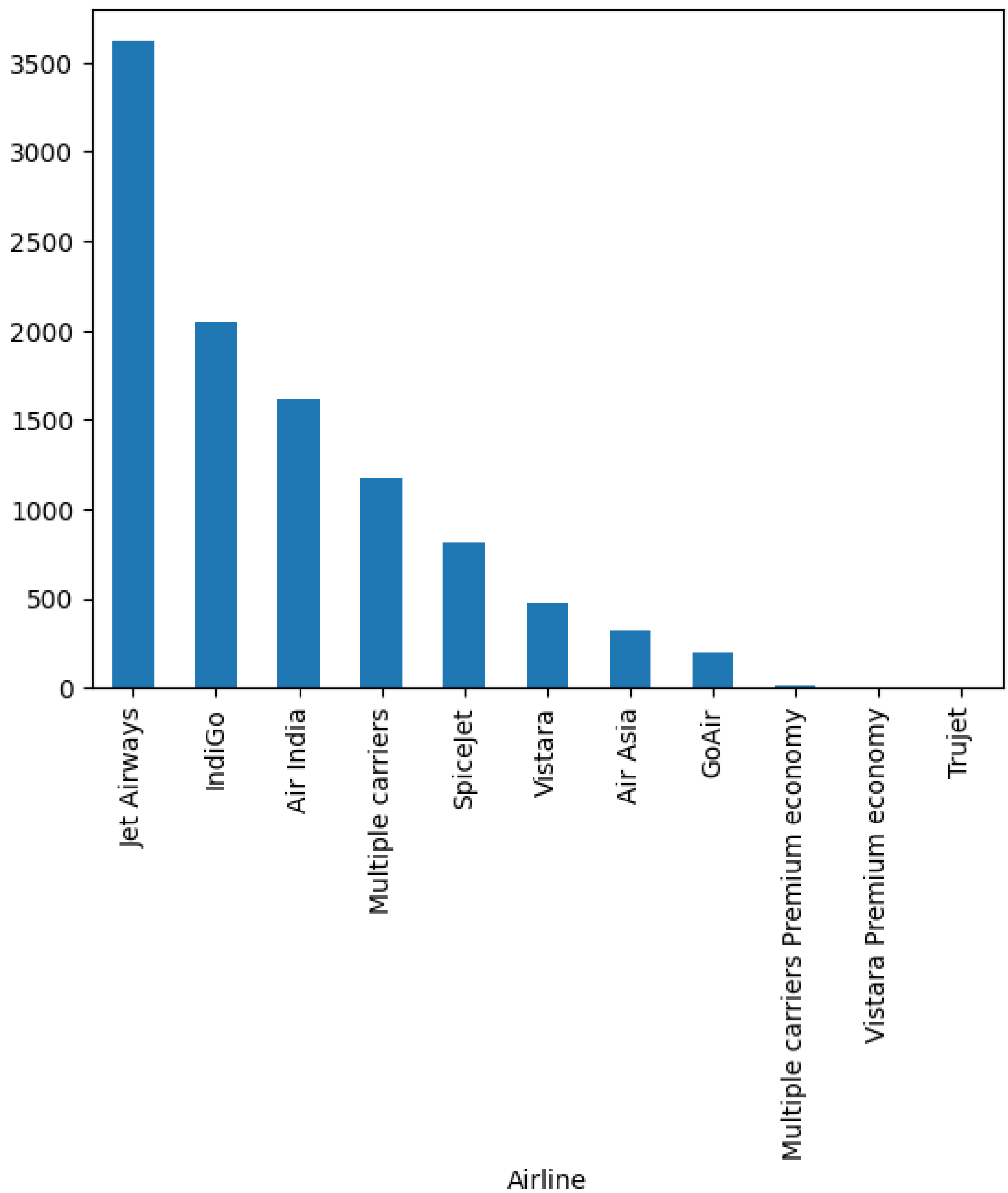
2 EDA

[16]:
```
# histogram for price column
plt.figure(figsize=(12,10))
sns.histplot(df['Price'], kde = True , color = 'orange')
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



[17]:
```
df['Airline'].value_counts().plot(kind = 'bar')
plt.show()
```



[18]:
```
# Droping the Year column as it has only one value(2019)
df.drop(columns = ['Year'], inplace = True)
```

[19]:
```
df.head()
```

[19]:
```
       Airline    Source Destination  Total_Stops  Price  Date  Month  \
0       IndiGo  Banglore   New Delhi            0   3897    24      3
1    Air India   Kolkata    Banglore            2   7662     1      5
2  Jet Airways     Delhi      Cochin            2  13882     9      6
```

```
3       IndiGo    Kolkata    Banglore            1   6218     12       5
4       IndiGo   Banglore   New Delhi            1  13302      1       3

   Dep_hours  Dep_min  Arrival_hours  Arrival_min  Duration_hours  \
0         22       20              1           10               2
1          5       50             13           15               7
2          9       25              4           25              19
3         18        5             23           30               5
4         16       50             21           35               4

   Duration_min
0            50
1            25
2             0
3            25
4            45
```

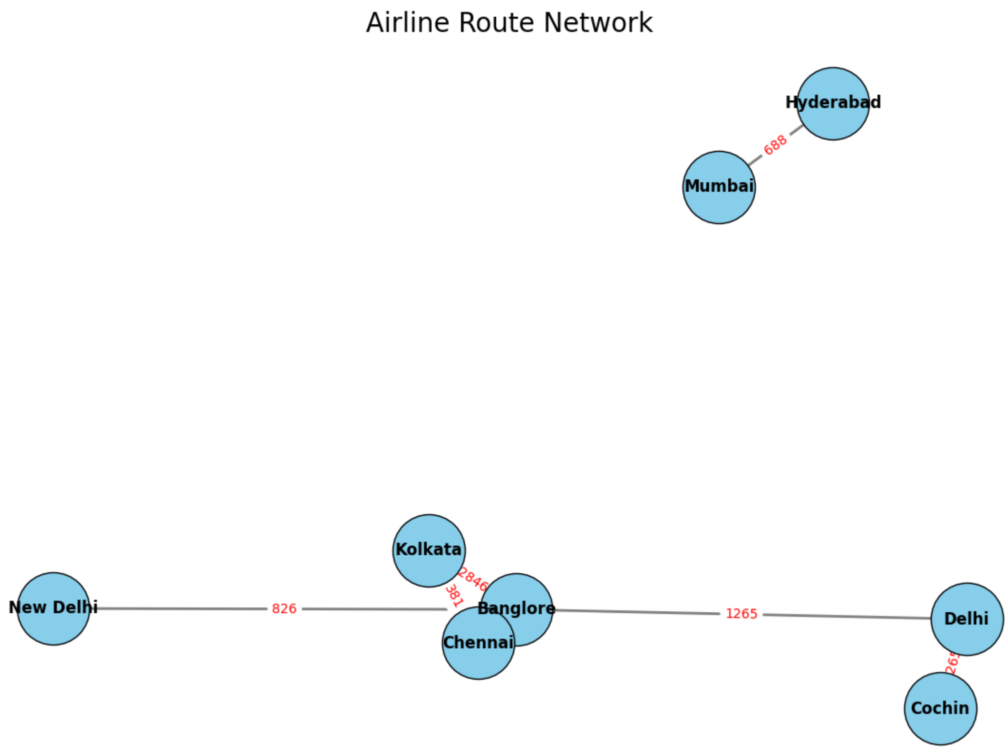
[20]:
```python
# Checking the most popular routes
route_freq = df.groupby(['Source','Destination']).size().
 ↪reset_index(name='count')
route_freq = route_freq.sort_values(by = 'count' , ascending = False)
```

[21]:
```python
route_freq.head()
```

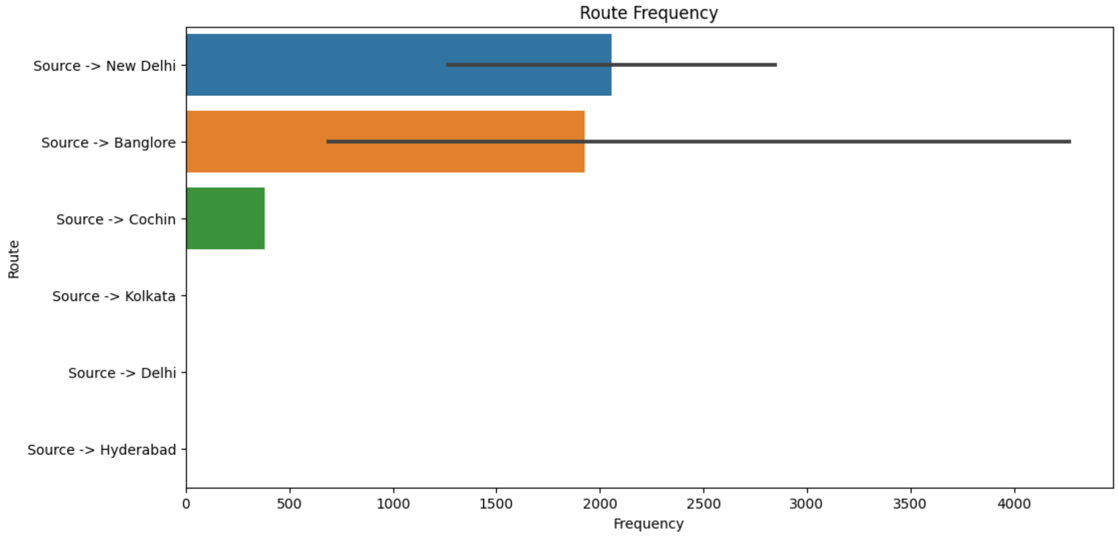
[21]:
```
     Source Destination  count
3     Delhi      Cochin   4265
4   Kolkata    Banglore   2846
0  Banglore       Delhi   1265
1  Banglore   New Delhi    826
5    Mumbai   Hyderabad    688
```

[22]:
```python
G = nx.from_pandas_edgelist(route_freq, 'Source', 'Destination', ['count'],␣
 ↪create_using=nx.DiGraph())
plt.figure(figsize=(15, 10))

pos = nx.spring_layout(G, k=0.5, iterations=50)
nx.draw_networkx_nodes(G, pos, node_size=3000, node_color='skyblue',␣
 ↪edgecolors='black')

nx.draw_networkx_edges(G, pos, arrowstyle='-|>', arrowsize=20,␣
 ↪edge_color='gray', width=2)

nx.draw_networkx_labels(G, pos, font_size=12, font_weight='bold')

edge_labels = nx.get_edge_attributes(G, 'count')
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=10,␣
 ↪font_color='red')
```

```
plt.title('Airline Route Network', size=20)

plt.axis('off')
plt.show()
```



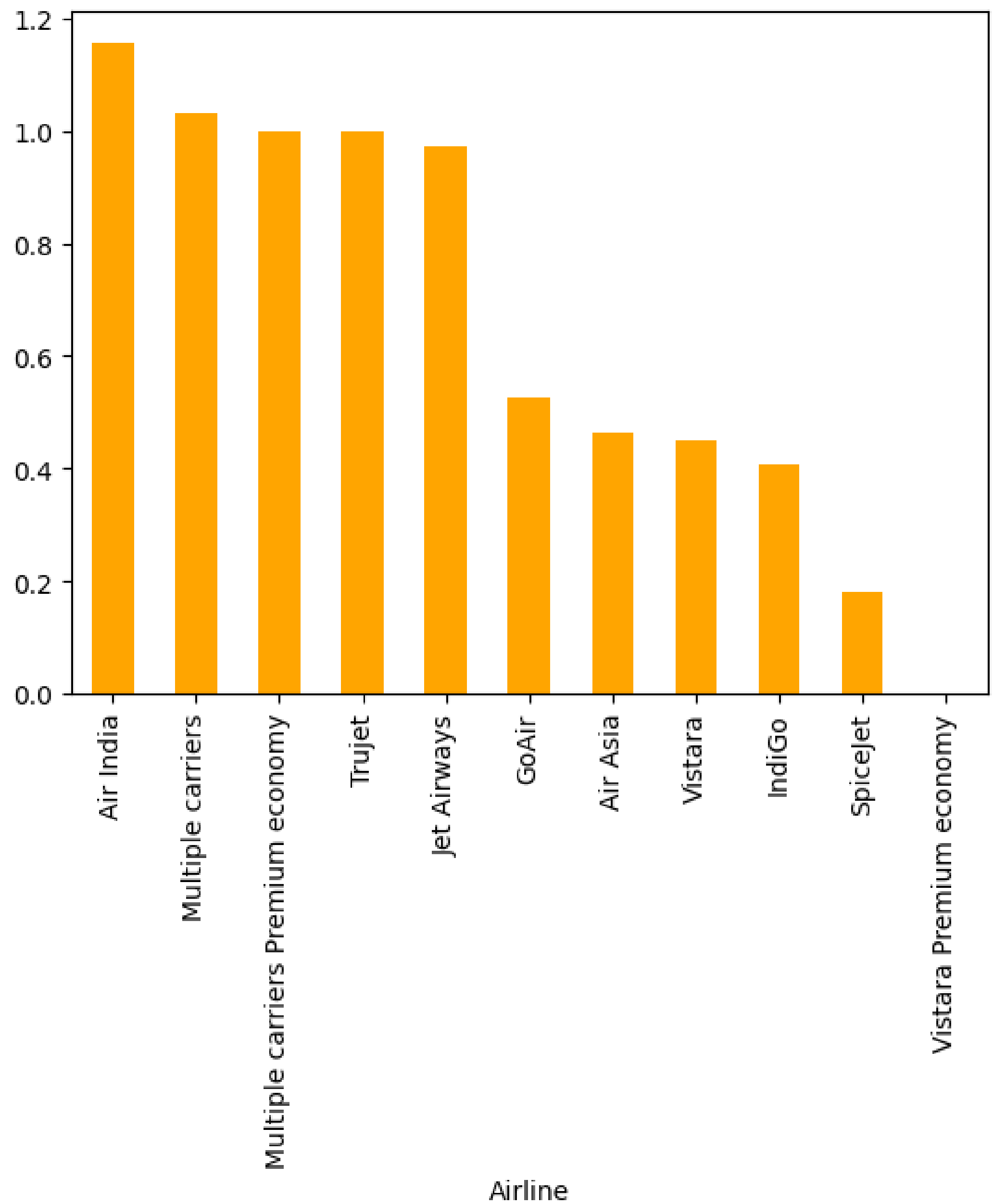
[23]:
```
plt.figure(figsize=(12, 6))
sns.barplot(data=route_freq, x='count', y='Source' + ' -> ' + df['Destination'])
plt.xlabel('Frequency')
plt.ylabel('Route')
plt.title('Route Frequency')
plt.show()
```



[24]:
```
df.head()
```

[24]:
```
       Airline    Source Destination  Total_Stops  Price  Date  Month  \
0       IndiGo  Banglore   New Delhi            0   3897    24      3
1    Air India   Kolkata    Banglore            2   7662     1      5
2  Jet Airways     Delhi      Cochin            2  13882     9      6
3       IndiGo   Kolkata    Banglore            1   6218    12      5
4       IndiGo  Banglore   New Delhi            1  13302     1      3

   Dep_hours  Dep_min  Arrival_hours  Arrival_min  Duration_hours  \
0         22       20              1           10               2
1          5       50             13           15               7
2          9       25              4           25              19
3         18        5             23           30               5
4         16       50             21           35               4

   Duration_min
0            50
1            25
2             0
3            25
4            45
```

[25]:
```
# Total Stops vs airline
df.groupby('Airline').Total_Stops.mean().sort_values(ascending = False).
 ↪plot(kind = 'bar', color ='orange')
plt.show()
```



[26]:
```python
# Total Stops vs source and destination
route_stops = df.groupby(['Source','Destination']).Total_Stops.mean().
 ↪reset_index().sort_values(by='Total_Stops',ascending=False)
```

[27]:
```python
route_stops
```

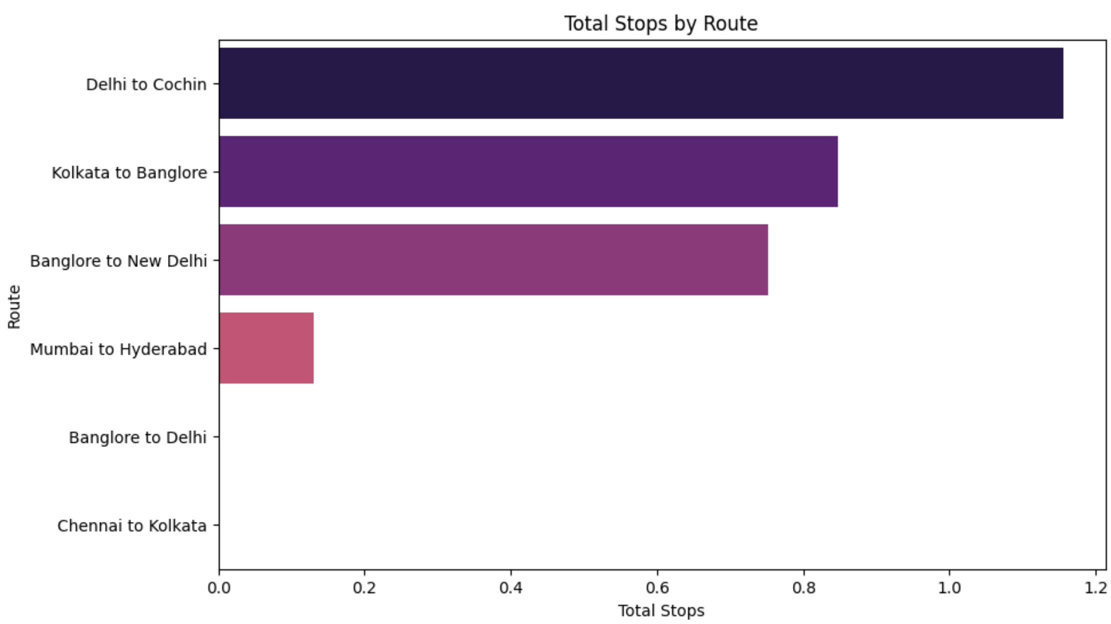
[27]:

	Source	Destination	Total_Stops
3	Delhi	Cochin	1.155451

```
4   Kolkata    Banglore     0.847505
1  Banglore   New Delhi     0.751816
5    Mumbai   Hyderabad     0.130814
0  Banglore       Delhi     0.000000
2   Chennai     Kolkata     0.000000
```

[28]:

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Total_Stops', y=route_stops['Source'] + ' to ' +
 ↪route_stops['Destination'], data=route_stops, palette='magma')

plt.xlabel('Total Stops')
plt.ylabel('Route')
plt.title('Total Stops by Route')

plt.show()
```



[29]:

```
df.head()
```

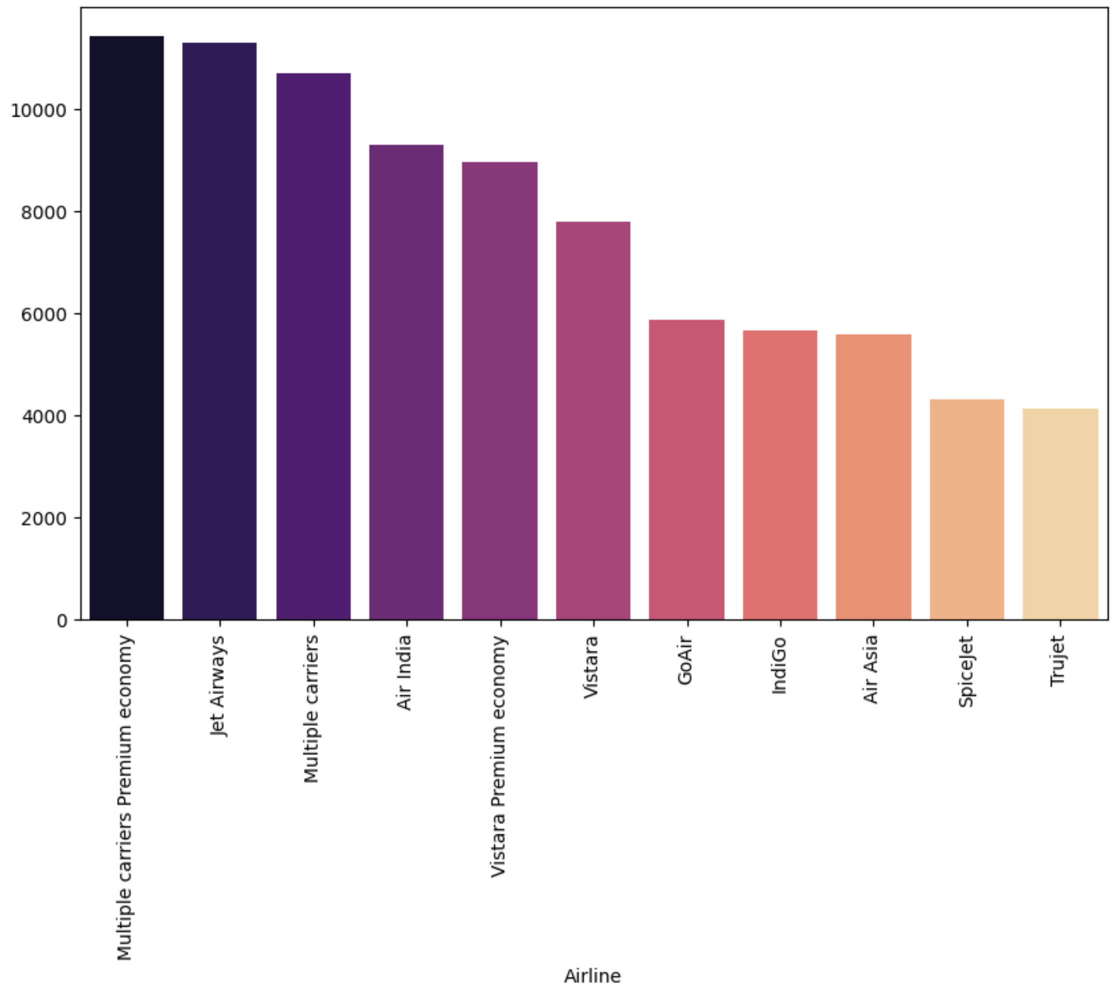
[29]:

```
       Airline    Source Destination  Total_Stops  Price  Date  Month  \
0       IndiGo  Banglore   New Delhi            0   3897    24      3
1    Air India   Kolkata    Banglore            2   7662     1      5
2  Jet Airways     Delhi      Cochin            2  13882     9      6
3       IndiGo   Kolkata    Banglore            1   6218    12      5
4       IndiGo  Banglore   New Delhi            1  13302     1      3

   Dep_hours  Dep_min  Arrival_hours  Arrival_min  Duration_hours  \
```

```
0          22        20                1            10                2
1           5        50               13            15                7
2           9        25                4            25               19
3          18         5               23            30                5
4          16        50               21            35                4

   Duration_min
0            50
1            25
2             0
3            25
4            45
```

[30]:
```
# Airline vs Price
airline_price = df.groupby('Airline').Price.mean().sort_values(ascending =␣
↪False)
```

[31]:
```
plt.figure(figsize=(10,6))
sns.barplot(x = airline_price.index , y = airline_price.values ,␣
↪palette='magma')

plt.xticks(rotation = 90)
plt.show()
```



[32]:
```
df.head()
```

[32]:
```
       Airline    Source Destination  Total_Stops  Price  Date  Month  \
0       IndiGo  Banglore   New Delhi            0   3897    24      3
1    Air India   Kolkata    Banglore            2   7662     1      5
2  Jet Airways     Delhi      Cochin            2  13882     9      6
3       IndiGo   Kolkata    Banglore            1   6218    12      5
4       IndiGo  Banglore   New Delhi            1  13302     1      3

   Dep_hours  Dep_min  Arrival_hours  Arrival_min  Duration_hours  \
0         22       20              1           10               2
1          5       50             13           15               7
2          9       25              4           25              19
3         18        5             23           30               5
4         16       50             21           35               4
```

```
   Duration_min
0            50
1            25
2             0
3            25
4            45
```

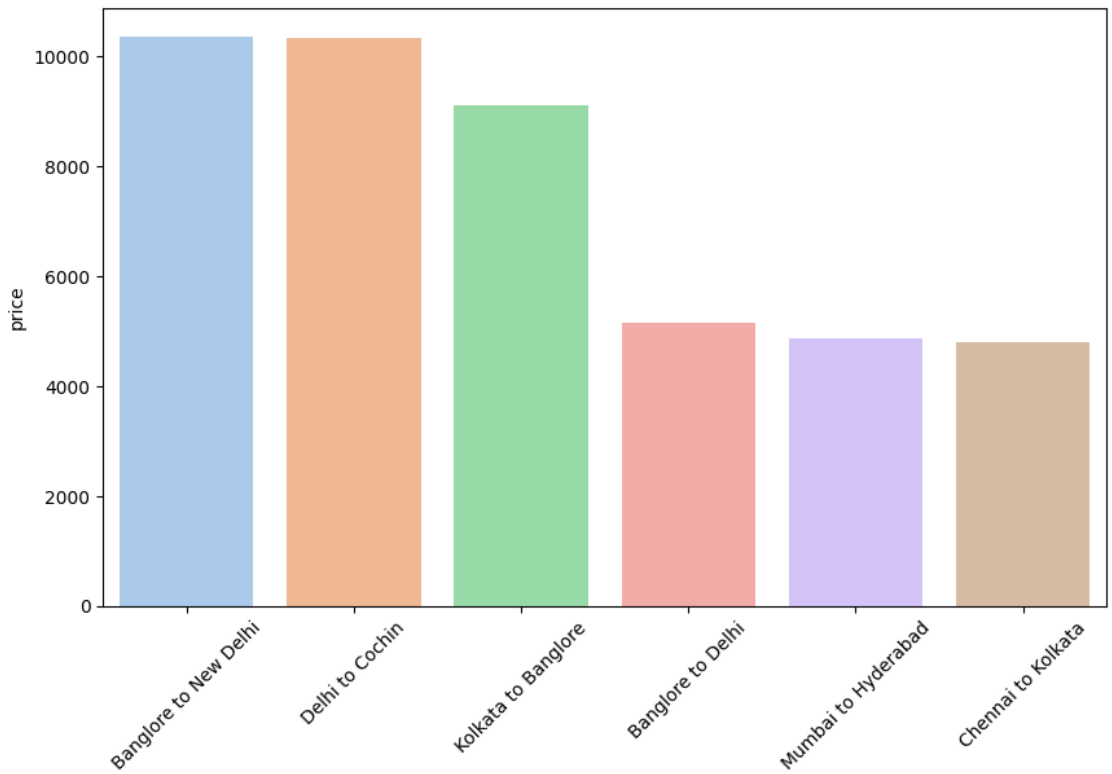
[33]:
```
# Routes vs Price
route_price = df.groupby(['Source','Destination']).Price.mean().
 ↪reset_index(name='price')
route_price = route_price.sort_values(by = 'price' , ascending = False)
```

[34]:
```
route_price
```

[34]:

	Source	Destination	price
1	Banglore	New Delhi	10374.016949
3	Delhi	Cochin	10350.422509
4	Kolkata	Banglore	9116.026001
0	Banglore	Delhi	5143.918577
5	Mumbai	Hyderabad	4865.125000
2	Chennai	Kolkata	4789.892388

[35]:
```
plt.figure(figsize=(10,6))
sns.barplot(x = route_price['Source'] +" to "+ route_price['Destination'], y =␣
 ↪route_price['price'] , palette='pastel')

plt.xticks(rotation = 45)
plt.show()
```



[36]:
```
# combining the Duration_hours and Duration_min column and converting it into␣
↪minutes for easier analysis
df['duration'] = (df['Duration_hours']*60) + df['Duration_min']
```

[37]:
```
df.drop(columns=['Duration_hours','Duration_min'], inplace = True)
```

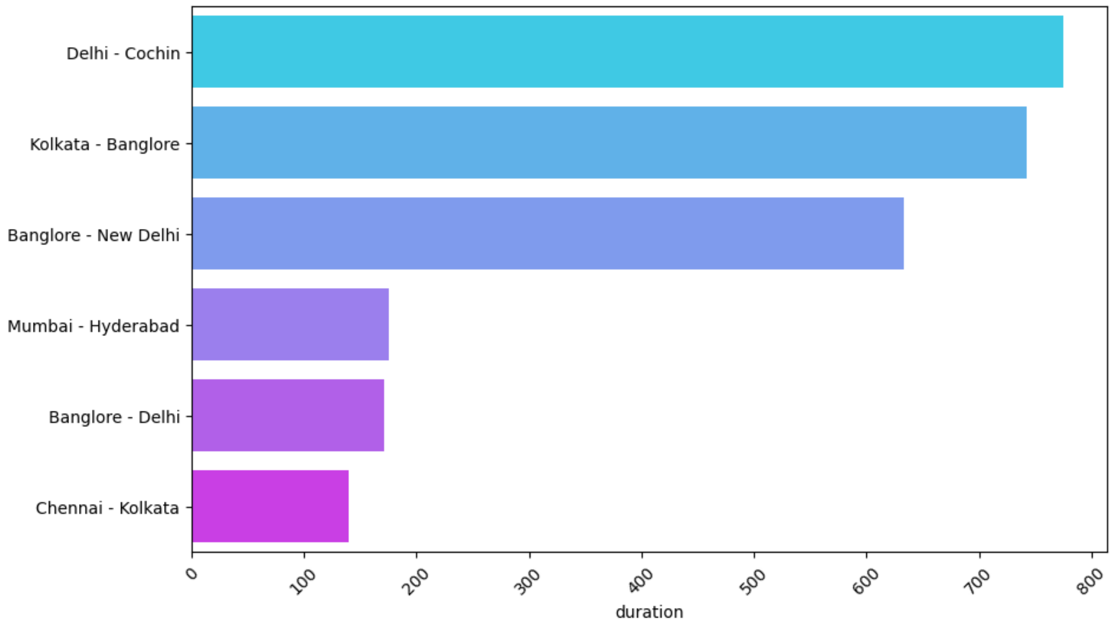
[38]:
```
# Routes vs duration
route_dur = df.groupby(['Source','Destination']).duration.mean().
↪reset_index(name='duration')
route_dur = route_dur.sort_values(by = 'duration' , ascending = False)
```

[39]:
```
route_dur
```

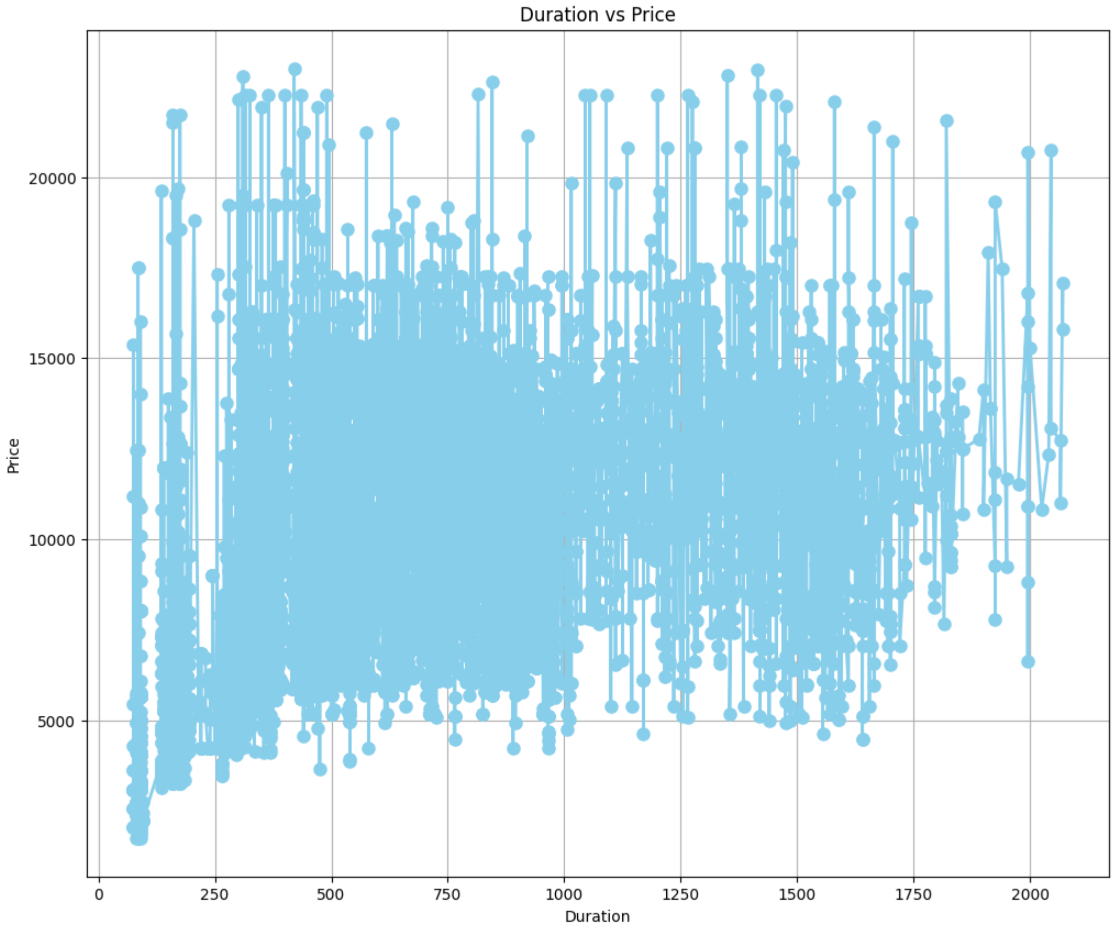
[39]:

	Source	Destination	duration
3	Delhi	Cochin	774.990621
4	Kolkata	Banglore	742.127547
1	Banglore	New Delhi	632.929782
5	Mumbai	Hyderabad	175.508721
0	Banglore	Delhi	171.695652
2	Chennai	Kolkata	139.619423

[40]:
```
plt.figure(figsize=(10,6))
sns.barplot(y = route_dur['Source'] +" - "+ route_dur['Destination'], x =␣
↪route_dur['duration'] , palette='cool')

plt.xticks(rotation = 45)
plt.show()
```



[41]:
```
df_sorted = df.sort_values(by='duration')

plt.figure(figsize=(12, 10))
plt.plot(df_sorted['duration'], df_sorted['Price'], marker='o',␣
↪color='skyblue', linewidth=2, markersize=8)

plt.xlabel('Duration')
plt.ylabel('Price')
plt.title('Duration vs Price')

plt.grid(True)
plt.show()
```



[42]:
```
df.head()
```

[42]:
```
       Airline    Source Destination  Total_Stops  Price  Date  Month  \
0       IndiGo  Banglore   New Delhi            0   3897    24      3
1    Air India   Kolkata    Banglore            2   7662     1      5
2  Jet Airways     Delhi      Cochin            2  13882     9      6
3       IndiGo   Kolkata    Banglore            1   6218    12      5
4       IndiGo  Banglore   New Delhi            1  13302     1      3

   Dep_hours  Dep_min  Arrival_hours  Arrival_min  duration
0         22       20              1           10       170
1          5       50             13           15       445
2          9       25              4           25      1140
3         18        5             23           30       325
4         16       50             21           35       285
```

[43]:
```
# month
df.Month.value_counts()
```

[43]:
```
Month
5    3379
6    3292
3    2525
4    1075
Name: count, dtype: int64
```

[44]:
```
df.Date.value_counts().sort_values(ascending = False)
```

[44]:
```
Date
9     1359
6     1250
27    1081
21    1075
24    1013
1      976
15     960
12     934
3      814
18     809
Name: count, dtype: int64
```

[45]:
```
df.Dep_hours.value_counts().sort_values(ascending = False)
```

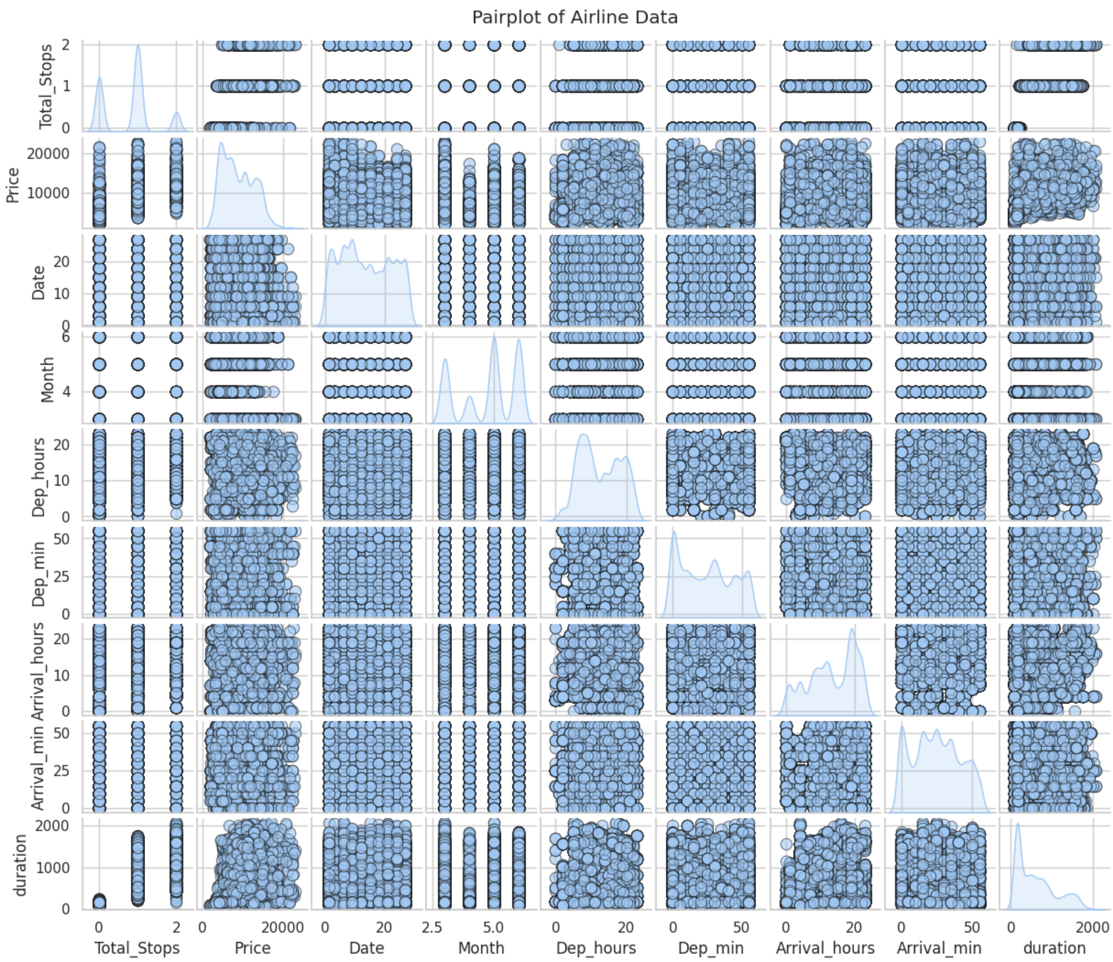
[45]:
```
Dep_hours
9     877
7     850
17    684
8     679
6     666
20    645
11    561
19    541
10    522
5     520
14    499
21    486
16    452
18    437
13    411
22    361
15    317
2     194
12    171
4     168
23    131
0      38
```

```
1      37
3      24
Name: count, dtype: int64
```

[46]:
```
sns.set(style="whitegrid", palette="pastel")

# Customize the pairplot
pairplot = sns.pairplot(df,
                        kind="scatter",
                        diag_kind="kde",
                        markers="o",
                        plot_kws={'alpha':0.6, 's':80, 'edgecolor':'k'},
                        diag_kws={'fill': True})

# Add titles and adjust the plot
pairplot.fig.suptitle("Pairplot of Airline Data", y=1.02)  # y=1.02 to move the␣
↪title a bit above the plot
pairplot.fig.set_size_inches(12, 10)

# Show the plot
plt.show()
```

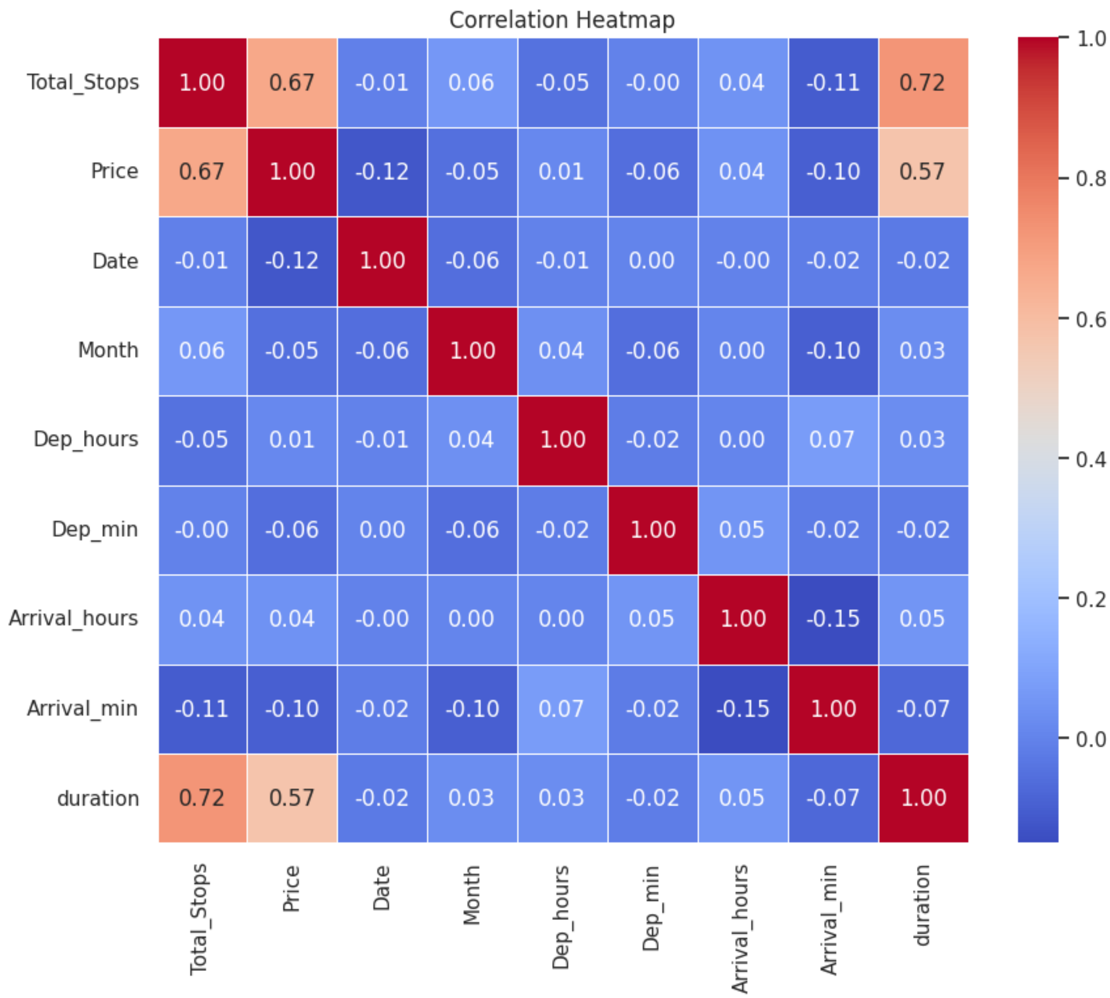
```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
```

```
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



[47]:
```
# heatmap for the dataset
plt.figure(figsize=(10, 8))
num_cols = df.select_dtypes('int64')
heatmap = sns.heatmap(num_cols.corr(), annot=True, cmap='coolwarm', fmt=".2f",␣
↪linewidths=.5)
```

```
heatmap.set_title('Correlation Heatmap')
plt.show()
```



[48]:

```
# price column is highly co related to duration and total stops and that makes␣
↪sense
```

3 Model building and evaluation

[49]:

```
df.head()
```

[49]:

```
       Airline    Source Destination  Total_Stops  Price  Date  Month  \
0       IndiGo  Banglore   New Delhi            0   3897    24      3
1    Air India   Kolkata    Banglore            2   7662     1      5
2  Jet Airways     Delhi      Cochin            2  13882     9      6
3       IndiGo   Kolkata    Banglore            1   6218    12      5
```

```
4       IndiGo  Banglore   New Delhi            1  13302     1       3

   Dep_hours  Dep_min  Arrival_hours  Arrival_min  duration
0         22       20              1           10       170
1          5       50             13           15       445
2          9       25              4           25      1140
3         18        5             23           30       325
4         16       50             21           35       285
```

[50]:
```python
from sklearn.preprocessing import OneHotEncoder,MinMaxScaler
```

[51]:
```python
# Normalization

def normalize_columns(df, columns):
    for col in columns:
        # Min-max normalization: (x - min) / (max - min)
        min_val = df[col].min()
        max_val = df[col].max()
        df[col] = (df[col] - min_val) / (max_val - min_val)

columns_to_normalize = ['Total_Stops', 'Date',
       'Month', 'Dep_hours', 'Dep_min', 'Arrival_hours', 'Arrival_min',
       'duration']

normalize_columns(df, columns_to_normalize)
```

[52]:
```python
# One-Hot_Encoding of categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns

df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
```

[53]:
```python
df.head()
```

[53]:
```
   Total_Stops  Price      Date     Month  Dep_hours   Dep_min  Arrival_hours  \
0          0.0   3897  0.884615  0.000000   0.956522  0.363636       0.043478
1          1.0   7662  0.000000  0.666667   0.217391  0.909091       0.565217
2          1.0  13882  0.307692  1.000000   0.391304  0.454545       0.173913
3          0.5   6218  0.423077  0.666667   0.782609  0.090909       1.000000
4          0.5  13302  0.000000  0.000000   0.695652  0.909091       0.913043

   Arrival_min  duration  Airline_Air India  …  \
0     0.181818  0.047619              False  …
1     0.272727  0.185464               True  …
2     0.454545  0.533835              False  …
3     0.545455  0.125313              False  …
4     0.636364  0.105263              False  …
```

```
   Airline_Vistara Premium economy  Source_Chennai  Source_Delhi  \
0                            False           False         False
1                            False           False         False
2                            False           False          True
3                            False           False         False
4                            False           False         False

   Source_Kolkata  Source_Mumbai  Destination_Cochin  Destination_Delhi  \
0           False          False               False              False
1            True          False               False              False
2           False          False                True              False
3            True          False               False              False
4           False          False               False              False

   Destination_Hyderabad  Destination_Kolkata  Destination_New Delhi
0                  False                False                   True
1                  False                False                  False
2                  False                False                  False
3                  False                False                  False
4                  False                False                   True

[5 rows x 28 columns]
```

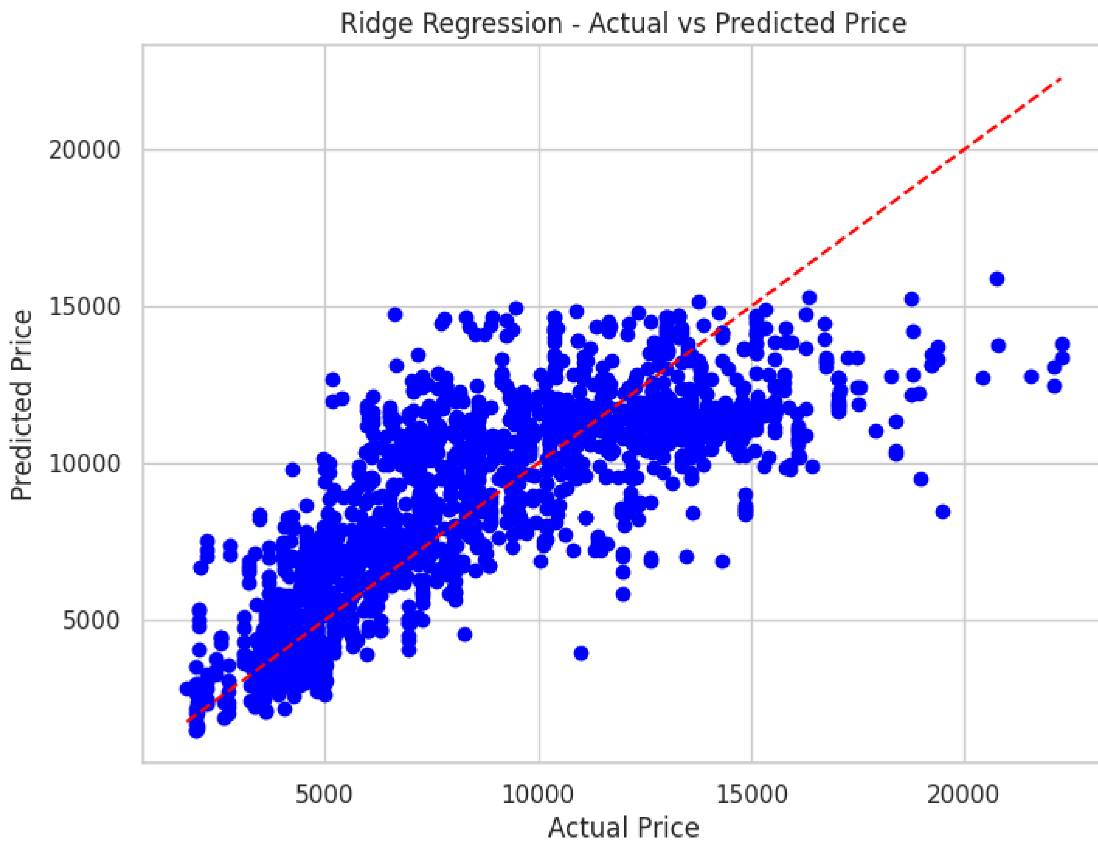
[54]:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error,␣
 ↪median_absolute_error, r2_score, explained_variance_score

X = df.drop(columns=['Price'])
y = df['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

models = {
    'Linear Regression': LinearRegression(),
    'Ridge Regression': Ridge(),
    'Lasso Regression': Lasso(),
    'Decision Tree Regressor': DecisionTreeRegressor(random_state=42),
    'Random Forest Regressor': RandomForestRegressor(random_state=42),
    'Gradient Boosting Regressor': GradientBoostingRegressor(random_state=42)
}

for model_name, model in models.items():
```

```
    print(f"Training {model_name}...")
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    medae = median_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    evs = explained_variance_score(y_test, y_pred)

    print(f"\n{model_name} Results:")
    print(f"Mean Squared Error (MSE): {mse:.2f}")
    print(f"Mean Absolute Error (MAE): {mae:.2f}")
    print(f"Median Absolute Error: {medae:.2f}")
    print(f"R^2 Score: {r2:.2f}")
    print(f"Explained Variance Score: {evs:.2f}")
    print("\n")

    plt.figure(figsize=(8, 6))
    plt.scatter(y_test, y_pred, color='blue')
    plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
  ↪linestyle='--', color='red')
    plt.xlabel('Actual Price')
    plt.ylabel('Predicted Price')
    plt.title(f'{model_name} - Actual vs Predicted Price')
    plt.grid(True)
    plt.show()
```

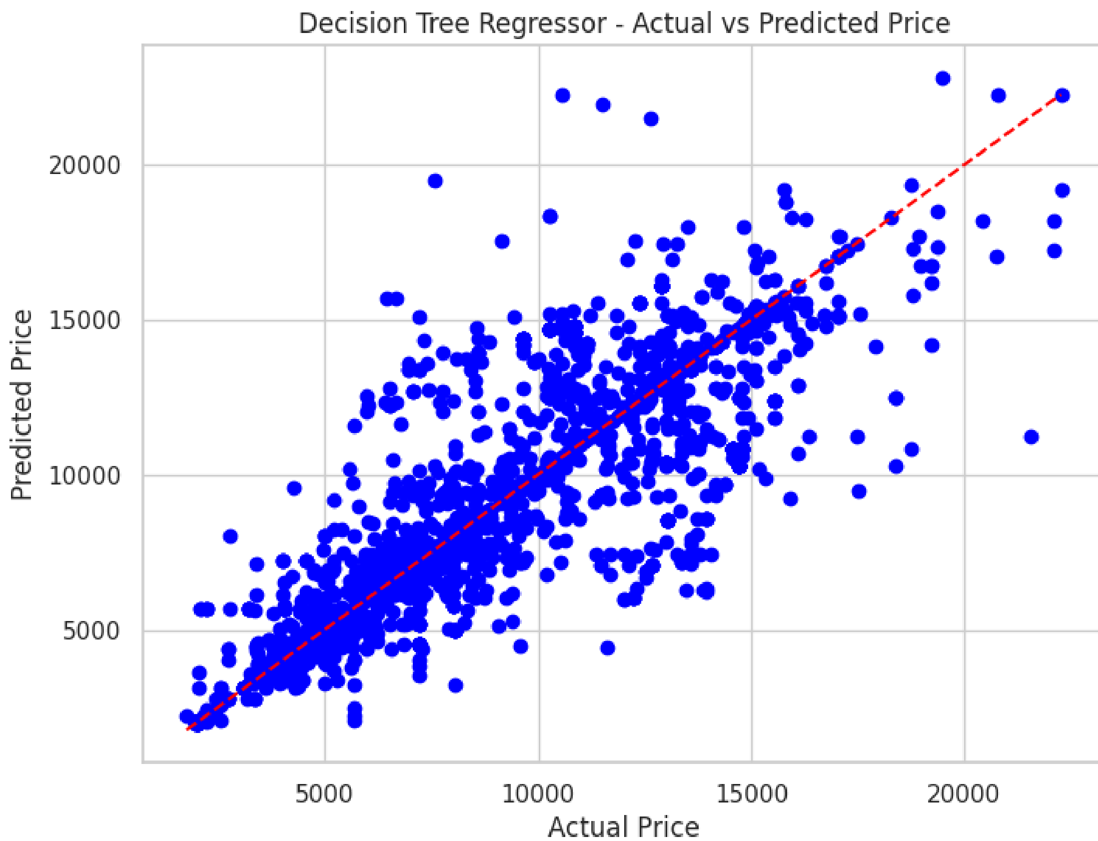
```
Training Linear Regression…

Linear Regression Results:
Mean Squared Error (MSE): 5684185.75
Mean Absolute Error (MAE): 1793.79
Median Absolute Error: 1313.00
R^2 Score: 0.64
Explained Variance Score: 0.64
```


```
Training Ridge Regression…

Ridge Regression Results:
Mean Squared Error (MSE): 5672244.66
Mean Absolute Error (MAE): 1792.40
Median Absolute Error: 1311.73
R^2 Score: 0.64
Explained Variance Score: 0.64
```



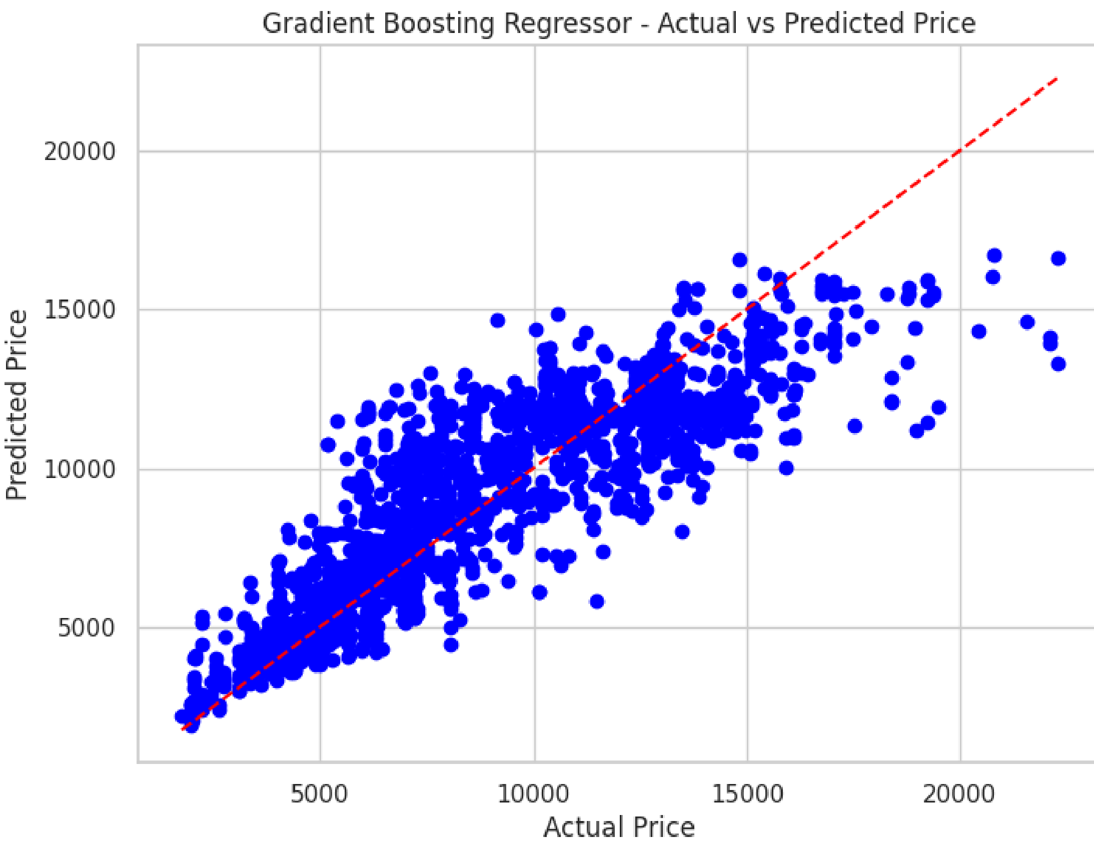

```
Training Lasso Regression…

Lasso Regression Results:
Mean Squared Error (MSE): 5668318.33
Mean Absolute Error (MAE): 1791.05
Median Absolute Error: 1302.68
R^2 Score: 0.64
Explained Variance Score: 0.64
```


```
Training Decision Tree Regressor…

Decision Tree Regressor Results:
Mean Squared Error (MSE): 4774660.00
Mean Absolute Error (MAE): 1317.87
Median Absolute Error: 525.00
R^2 Score: 0.70
Explained Variance Score: 0.70
```




```
Training Random Forest Regressor…

Random Forest Regressor Results:
Mean Squared Error (MSE): 3122732.64
Mean Absolute Error (MAE): 1148.94
Median Absolute Error: 613.32
R^2 Score: 0.80
Explained Variance Score: 0.80
```


```
Training Gradient Boosting Regressor…

Gradient Boosting Regressor Results:
Mean Squared Error (MSE): 3586350.75
Mean Absolute Error (MAE): 1427.26
Median Absolute Error: 1079.79
R^2 Score: 0.77
Explained Variance Score: 0.77
```


Gradient Boosting Regressor - Actual vs Predicted Price
Predicted Price
Actual Price
20000
15000
10000
5000
5000
10000
15000
20000