Final Project for ML

About DatasetData

1. Airline: The name of the airline.
2. Date_of_Journey: The date of the journey
3. Source: The source from which the service begins.
4. Destination: The destination where the service ends.
5. Route: The route taken by the flight to reach the destination.
6. Dep_Time: The time when the journey starts from the source.
7. Arrival_Time: Time of arrival at the destination.
8. Duration: Total duration of the flight.
9. Total_Stops: Total stops between the source and destination.
10. Additional_Info: Additional information about the flight
11. Price(target): The price of the ticket

Importing Libraries

In [4]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

Loading Data

In [6]:
```
df = pd.read_excel('Data_Train.xlsx')
df_orig = df.copy()
df.head()
```

Out[6]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Du
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4

EDA

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          10683 non-null  object
 1   Date_of_Journey  10683 non-null  object
 2   Source           10683 non-null  object
 3   Destination      10683 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10683 non-null  object
 6   Arrival_Time     10683 non-null  object
 7   Duration         10683 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10683 non-null  object
 10  Price            10683 non-null  int64 
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

Cheacking Duplication

In [10]:
```
df.duplicated().sum()
```

Out[10]:
```
220
```

Drop Duplicates

In [12]:
```
df.drop_duplicates(inplace=True)
```

Cheacking Nulls

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
Airline           0
Date_of_Journey   0
Source            0
Destination       0
Route             1
Dep_Time          0
Arrival_Time      0
Duration          0
Total_Stops       1
Additional_Info   0
Price             0
dtype: int64
```

In [15]:
```
df.dropna(inplace=True)
```

Univariate Analysis

Airline Column

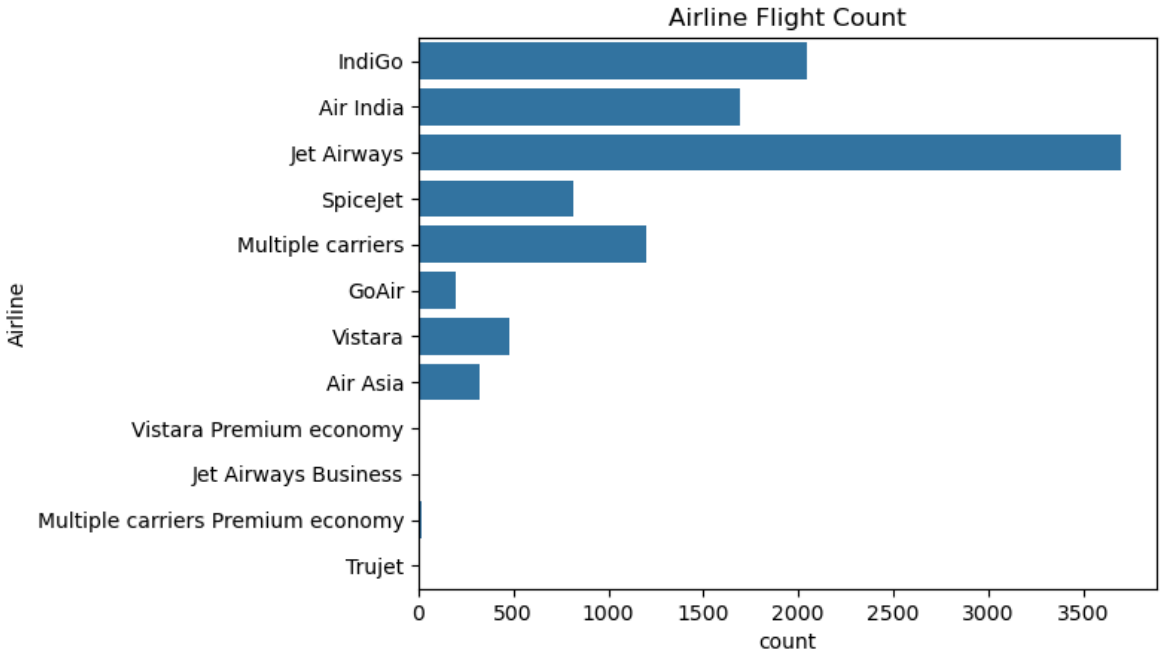
In [18]:
```
print(df['Airline'].nunique())
df['Airline'].value_counts()
```

```
12
```

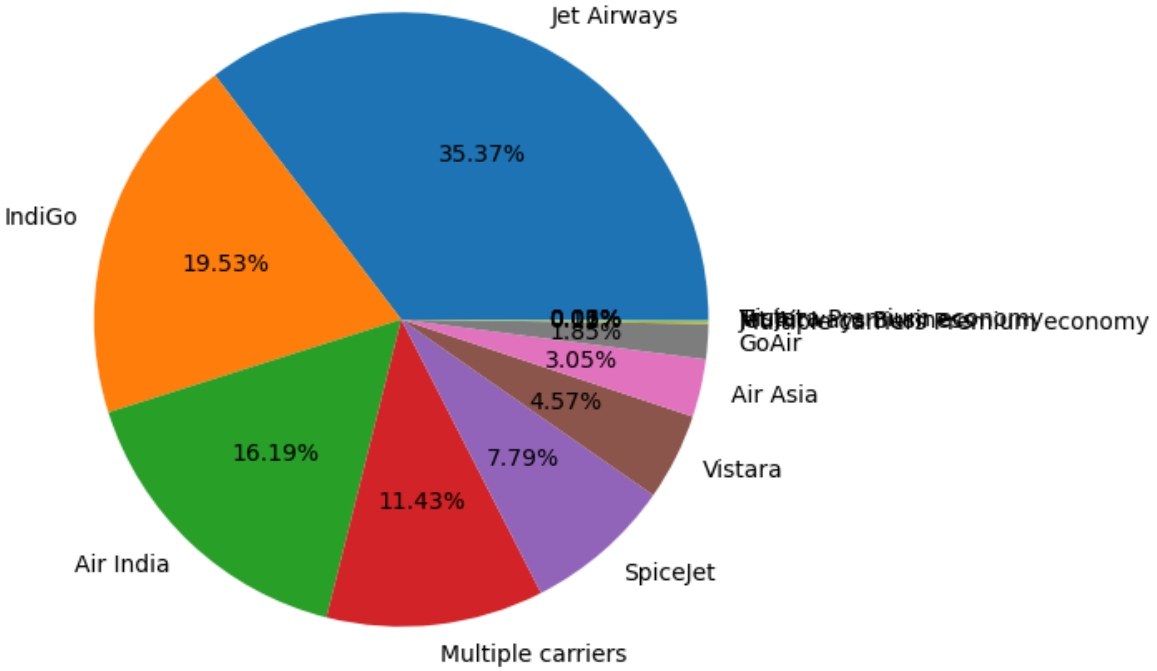
Out[18]:
```
Airline
Jet Airways                          3700
IndiGo                               2043
Air India                            1694
Multiple carriers                    1196
SpiceJet                              815
Vistara                               478
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: count, dtype: int64
```

In [19]:
```
sns.countplot(df['Airline'])
plt.ylabel('Airline')
plt.title('Airline Flight Count')
```

Out[19]:
```
Text(0.5, 1.0, 'Airline Flight Count')
```



In [20]:
```
plt.figure(figsize=(8,6))
plt.pie(df['Airline'].value_counts().values[:], labels=df['Airline'].value_count
```



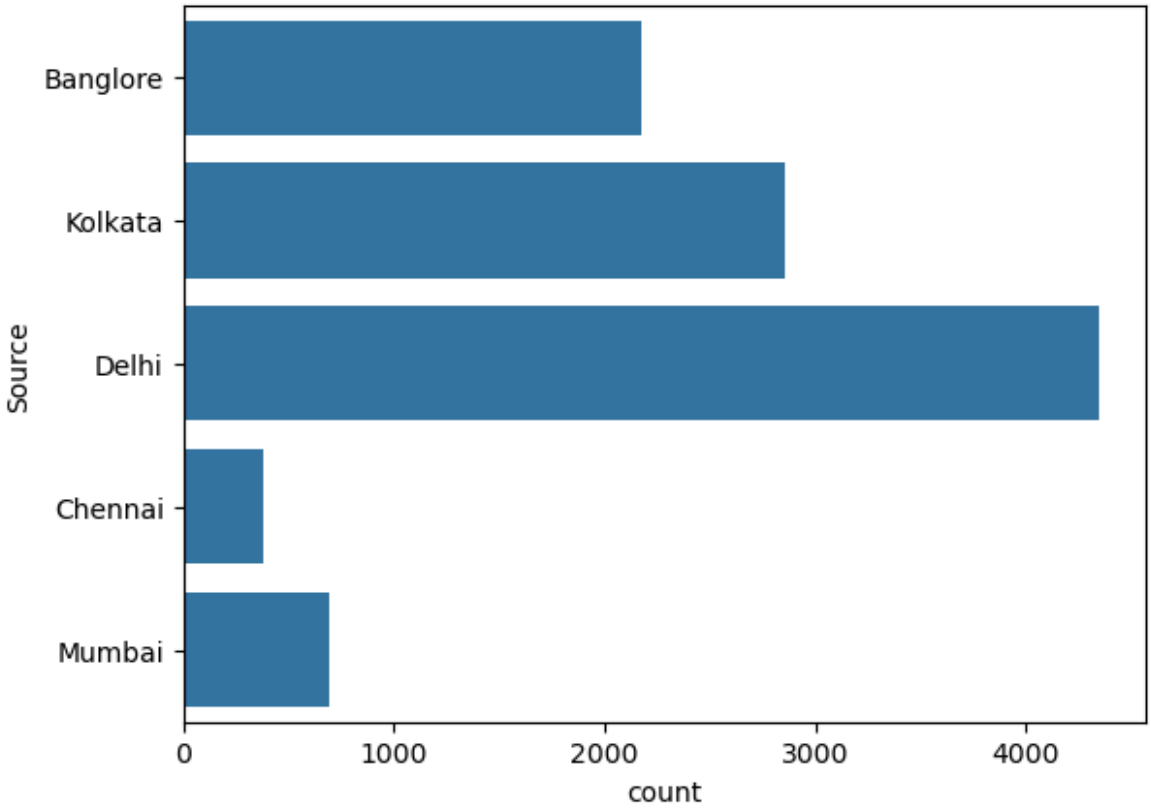
Source Column

In [22]:
```
df['Source'].value_counts()
```

Out[22]:
```
Source
Delhi       4345
Kolkata     2860
Banglore    2179
Mumbai       697
Chennai      381
Name: count, dtype: int64
```

In [23]:
```
sns.countplot(df['Source'])
```

Out[23]:
```
<Axes: xlabel='count', ylabel='Source'>
```



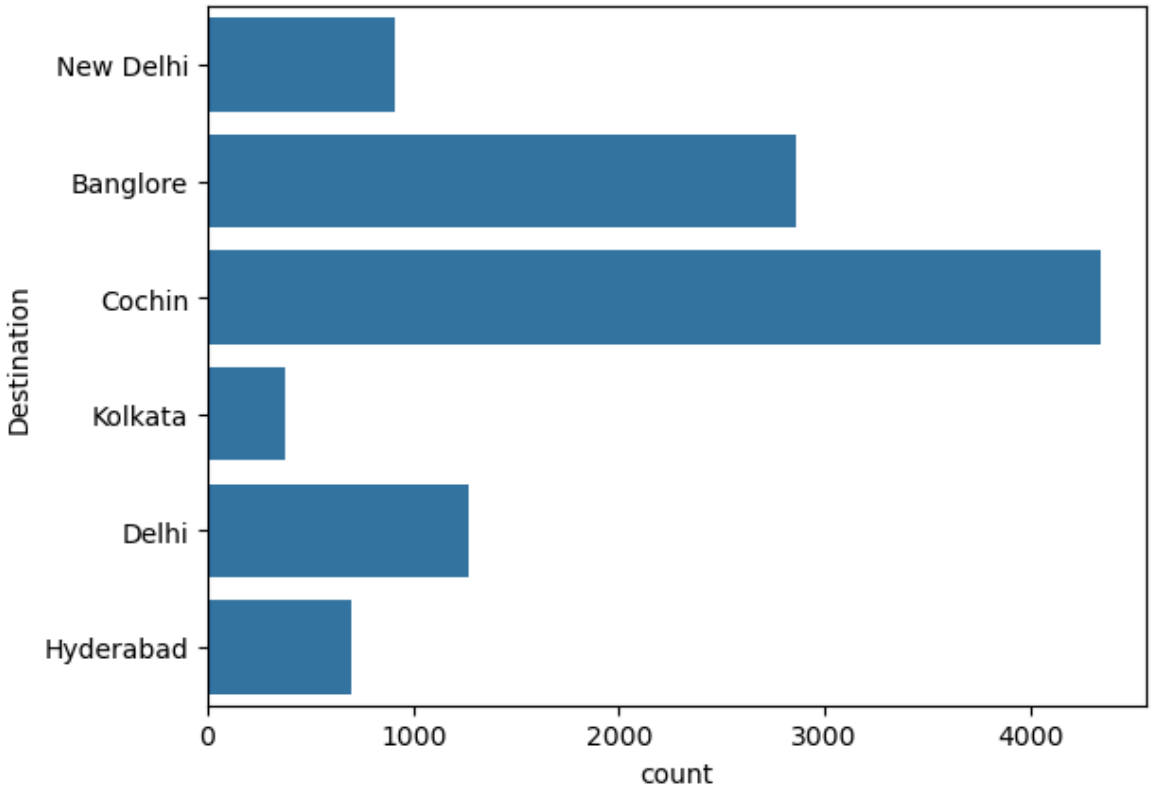
Destination Column

In [25]:
```
df['Destination'].value_counts()
```

Out[25]:
```
Destination
Cochin       4345
Banglore     2860
Delhi        1265
New Delhi     914
Hyderabad     697
Kolkata       381
Name: count, dtype: int64
```

In [26]:
```
sns.countplot(df['Destination'])
```

Out[26]:
```
<Axes: xlabel='count', ylabel='Destination'>
```



Total_Stops Column

In [28]:
```
df['Total_Stops'].value_counts()
```

Out[28]:
```
Total_Stops
1 stop      5625
non-stop    3475
2 stops     1318
3 stops       43
4 stops        1
Name: count, dtype: int64
```

Additional_Info Column

In [30]:
```
df['Additional_Info'].value_counts()
```

Out[30]:
```
Additional_Info
No info                         8182
In-flight meal not included     1926
No check-in baggage included     318
1 Long layover                    19
Change airports                    7
Business class                     4
No Info                            3
1 Short layover                    1
Red-eye flight                     1
2 Long layover                     1
Name: count, dtype: int64
```

we have a some values have the same meaning (No Info, No info) will make both like the same

Route Column

In [33]:
```
print(len(df['Route'].unique()))

df['Route'].unique()
```

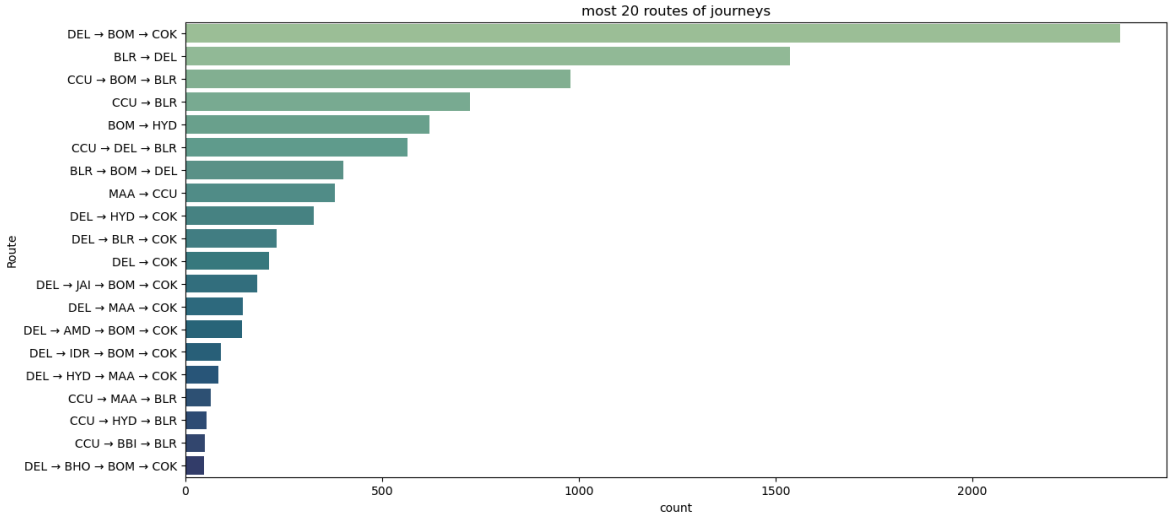
```
128
```

Out[33]:
```
array(['BLR → DEL', 'CCU → IXR → BBI → BLR', 'DEL → LKO → BOM → COK',
       'CCU → NAG → BLR', 'BLR → NAG → DEL', 'CCU → BLR',
       'BLR → BOM → DEL', 'DEL → BOM → COK', 'DEL → BLR → COK',
       'MAA → CCU', 'CCU → BOM → BLR', 'DEL → AMD → BOM → COK',
       'DEL → PNQ → COK', 'DEL → CCU → BOM → COK', 'BLR → COK → DEL',
       'DEL → IDR → BOM → COK', 'DEL → LKO → COK',
       'CCU → GAU → DEL → BLR', 'DEL → NAG → BOM → COK',
       'CCU → MAA → BLR', 'DEL → HYD → COK', 'CCU → HYD → BLR',
       'DEL → COK', 'CCU → DEL → BLR', 'BLR → BOM → AMD → DEL',
       'BOM → DEL → HYD', 'DEL → MAA → COK', 'BOM → HYD',
       'DEL → BHO → BOM → COK', 'DEL → JAI → BOM → COK',
       'DEL → ATQ → BOM → COK', 'DEL → JDH → BOM → COK',
       'CCU → BBI → BOM → BLR', 'BLR → MAA → DEL',
       'DEL → GOI → BOM → COK', 'DEL → BDQ → BOM → COK',
       'CCU → JAI → BOM → BLR', 'CCU → BBI → BLR', 'BLR → HYD → DEL',
       'DEL → TRV → COK', 'CCU → IXR → DEL → BLR',
       'DEL → IXU → BOM → COK', 'CCU → IXB → BLR',
       'BLR → BOM → JDH → DEL', 'DEL → UDR → BOM → COK',
       'DEL → HYD → MAA → COK', 'CCU → BOM → COK → BLR',
       'BLR → CCU → DEL', 'CCU → BOM → GOI → BLR',
       'DEL → RPR → NAG → BOM → COK', 'DEL → HYD → BOM → COK',
       'CCU → DEL → AMD → BLR', 'CCU → PNQ → BLR',
       'BLR → CCU → GAU → DEL', 'CCU → DEL → COK → BLR',
       'BLR → PNQ → DEL', 'BOM → JDH → DEL → HYD',
       'BLR → BOM → BHO → DEL', 'DEL → AMD → COK', 'BLR → LKO → DEL',
       'CCU → GAU → BLR', 'BOM → GOI → HYD', 'CCU → BOM → AMD → BLR',
       'CCU → BBI → IXR → DEL → BLR', 'DEL → DED → BOM → COK',
       'DEL → MAA → BOM → COK', 'BLR → AMD → DEL', 'BLR → VGA → DEL',
       'CCU → JAI → DEL → BLR', 'CCU → AMD → BLR',
       'CCU → VNS → DEL → BLR', 'BLR → BOM → IDR → DEL',
       'BLR → BBI → DEL', 'BLR → GOI → DEL', 'BOM → AMD → ISK → HYD',
       'BOM → DED → DEL → HYD', 'DEL → IXC → BOM → COK',
       'CCU → PAT → BLR', 'BLR → CCU → BBI → DEL',
       'CCU → BBI → HYD → BLR', 'BLR → BOM → NAG → DEL',
       'BLR → CCU → BBI → HYD → DEL', 'BLR → GAU → DEL',
       'BOM → BHO → DEL → HYD', 'BOM → JLR → HYD',
       'BLR → HYD → VGA → DEL', 'CCU → KNU → BLR',
       'CCU → BOM → PNQ → BLR', 'DEL → BBI → COK',
       'BLR → VGA → HYD → DEL', 'BOM → JDH → JAI → DEL → HYD',
       'DEL → GWL → IDR → BOM → COK', 'CCU → RPR → HYD → BLR',
       'CCU → VTZ → BLR', 'CCU → DEL → VGA → BLR',
       'BLR → BOM → IDR → GWL → DEL', 'CCU → DEL → COK → TRV → BLR',
       'BOM → COK → MAA → HYD', 'BOM → NDC → HYD', 'BLR → BDQ → DEL',
       'CCU → BOM → TRV → BLR', 'CCU → BOM → HBX → BLR',
       'BOM → BDQ → DEL → HYD', 'BOM → CCU → HYD',
       'BLR → TRV → COK → DEL', 'BLR → IDR → DEL',
       'CCU → IXZ → MAA → BLR', 'CCU → GAU → IMF → DEL → BLR',
       'BOM → GOI → PNQ → HYD', 'BOM → BLR → CCU → BBI → HYD',
       'BOM → MAA → HYD', 'BLR → BOM → UDR → DEL',
       'BOM → UDR → DEL → HYD', 'BLR → VGA → VTZ → DEL',
       'BLR → HBX → BOM → BHO → DEL', 'CCU → IXA → BLR',
       'BOM → RPR → VTZ → HYD', 'BLR → HBX → BOM → AMD → DEL',
       'BOM → IDR → DEL → HYD', 'BOM → BLR → HYD', 'BLR → STV → DEL',
       'CCU → IXB → DEL → BLR', 'BOM → JAI → DEL → HYD',
       'BOM → VNS → DEL → HYD', 'BLR → HBX → BOM → NAG → DEL',
       'BLR → BOM → IXC → DEL', 'BLR → CCU → BBI → HYD → VGA → DEL',
       'BOM → BBI → HYD'], dtype=object)
```

Most 20 routes

In [35]:
```
plt.figure(figsize=(15,7))
chart=sns.countplot(y=df['Route'], palette='crest', order = df['Route'].value_co
plt.title("most 20 routes of journeys")
plt.show()
```

```
C:\Users\zas\AppData\Local\Temp\ipykernel_7940\3057754910.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  chart=sns.countplot(y=df['Route'], palette='crest', order = df['Route'].value_c
ounts()[:20].index)
```



Date_of_Journey colunm

In [37]:
```
df['Date_of_Journey'].value_counts()
```

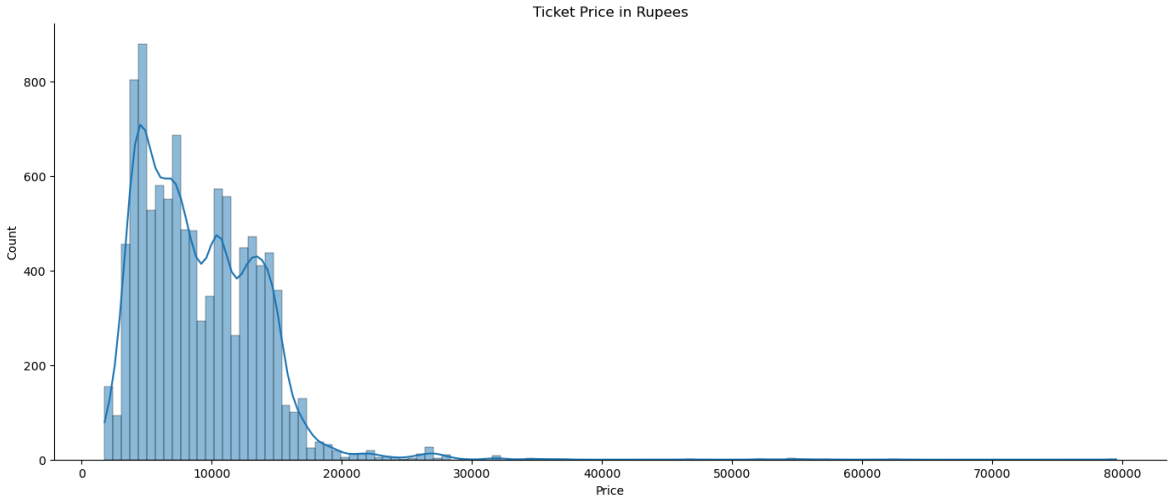
Out[37]:
```
Date_of_Journey
6/06/2019     490
18/05/2019    486
9/06/2019     485
12/06/2019    483
21/05/2019    482
9/05/2019     466
21/03/2019    412
15/05/2019    402
27/05/2019    369
27/06/2019    339
24/06/2019    330
1/06/2019     330
3/06/2019     326
15/06/2019    314
24/03/2019    314
6/03/2019     302
27/03/2019    290
24/05/2019    286
6/05/2019     281
1/05/2019     274
12/05/2019    259
1/04/2019     256
3/03/2019     217
9/03/2019     199
15/03/2019    162
18/03/2019    156
01/03/2019    151
12/03/2019    141
9/04/2019     125
3/04/2019     110
21/06/2019    109
18/06/2019    105
09/03/2019    100
6/04/2019     100
06/03/2019     95
27/04/2019     94
24/04/2019     92
03/03/2019     92
3/05/2019      90
15/04/2019     89
21/04/2019     82
18/04/2019     67
12/04/2019     63
1/03/2019      47
Name: count, dtype: int64
```

Price Column

In [39]:
```
df['Price'].describe()
```

Out[39]:
```
count    10462.000000
mean      9026.790289
std       4624.849541
min       1759.000000
25%       5224.000000
50%       8266.000000
75%      12344.750000
max      79512.000000
Name: Price, dtype: float64
```

In [40]:
```
ticket_price = sns.displot(x=df['Price'], kde=True)
plt.title('Ticket Price in Rupees')
ticket_price.fig.set_figwidth(16)
ticket_price.fig.set_figheight(6)
```



Bivariate Analysis

Which Airline Provide posistive information (feedbacks)?

In [43]:
```
df.groupby('Additional_Info')['Airline'].unique()
```

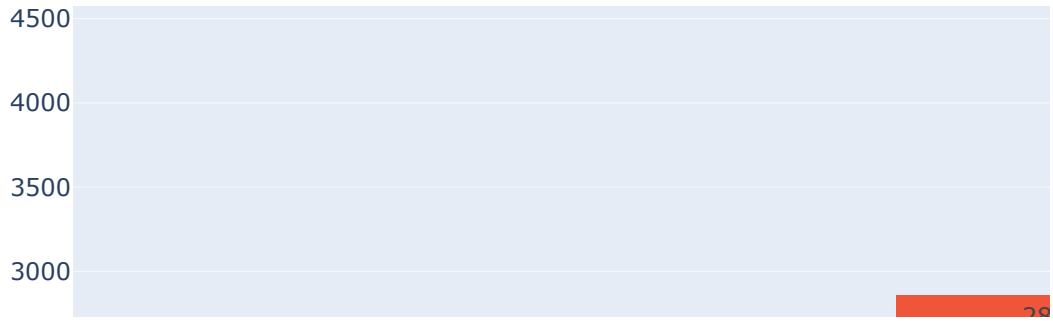
Out[43]:
```
Additional_Info
1 Long layover                                [Air India, Jet Airways, SpiceJe
t]
1 Short layover                                                      [Air Indi
a]
2 Long layover                                                       [Air Indi
a]
Business class                               [Jet Airways Business, Jet Airway
s]
Change airports                                                      [Air Indi
a]
In-flight meal not included                    [Jet Airways, Multiple carrier
s]
No Info                                                                [IndiG
o]
No check-in baggage included                                         [SpiceJe
t]
No info                          [IndiGo, Air India, Jet Airways, SpiceJet, Mu
l...
Red-eye flight                                                       [Air Asi
a]
Name: Airline, dtype: object
```

Is There A Relation Between The Source And The Destination ?

In [45]:
```
df.groupby('Source')['Destination'].unique()
```

Out[45]:
```
Source
Banglore    [New Delhi, Delhi]
Chennai              [Kolkata]
Delhi                 [Cochin]
Kolkata             [Banglore]
Mumbai             [Hyderabad]
Name: Destination, dtype: object
```

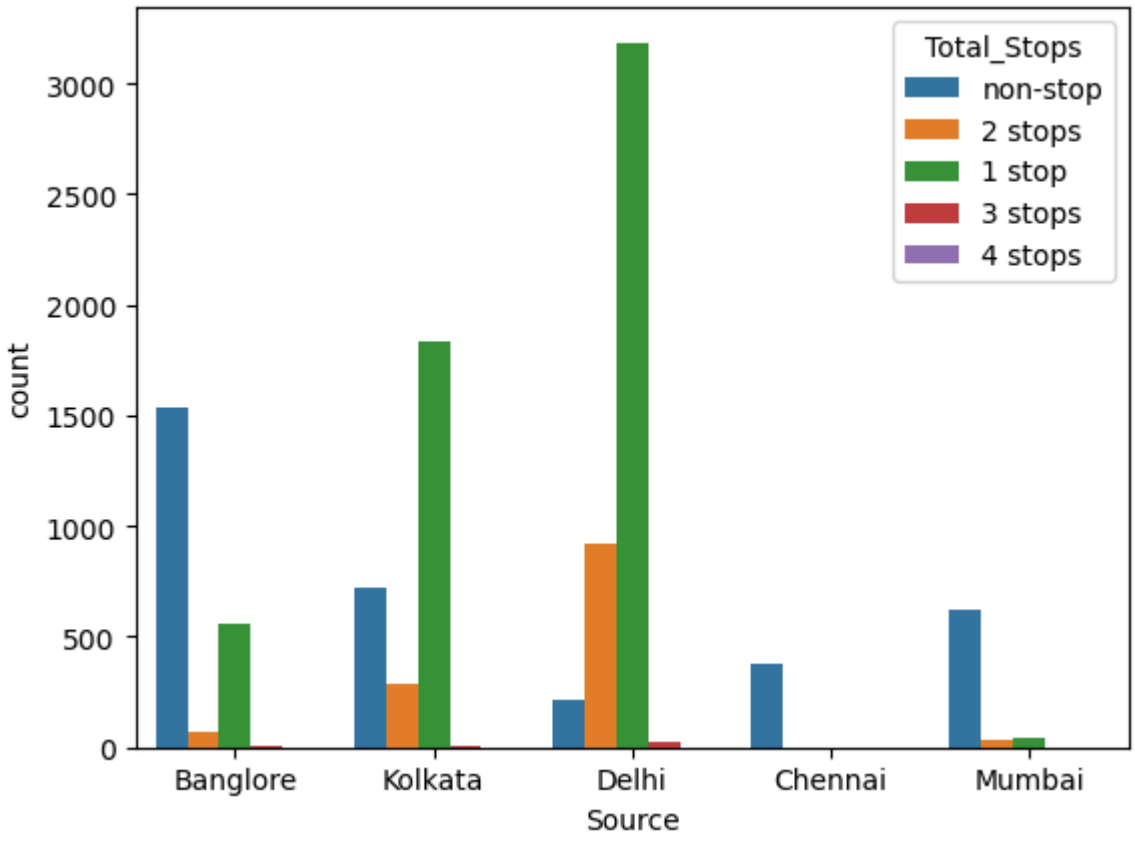
In [46]:
```
px.histogram(data_frame=df , x ="Destination" , color="Source"  , text_auto=True
```



What the Relation Between The Source And The Total_Stops ?

In [48]:
```
sns.countplot(x =df["Source"], hue=df["Total_Stops"])
```

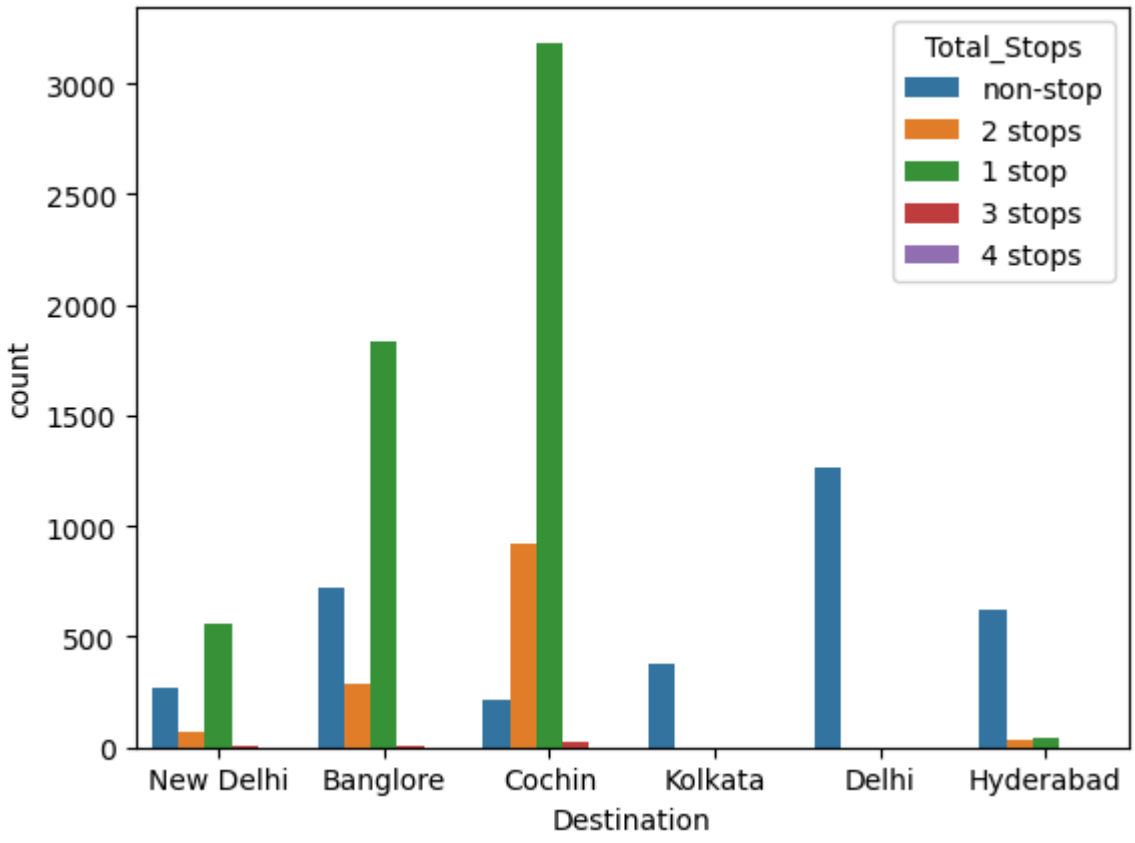
Out[48]:
```
<Axes: xlabel='Source', ylabel='count'>
```



What the Relation Between The Destination And The Total_Stops ?

In [50]:
```
sns.countplot(x =df["Destination"], hue=df["Total_Stops"])
```

Out[50]:
```
<Axes: xlabel='Destination', ylabel='count'>
```



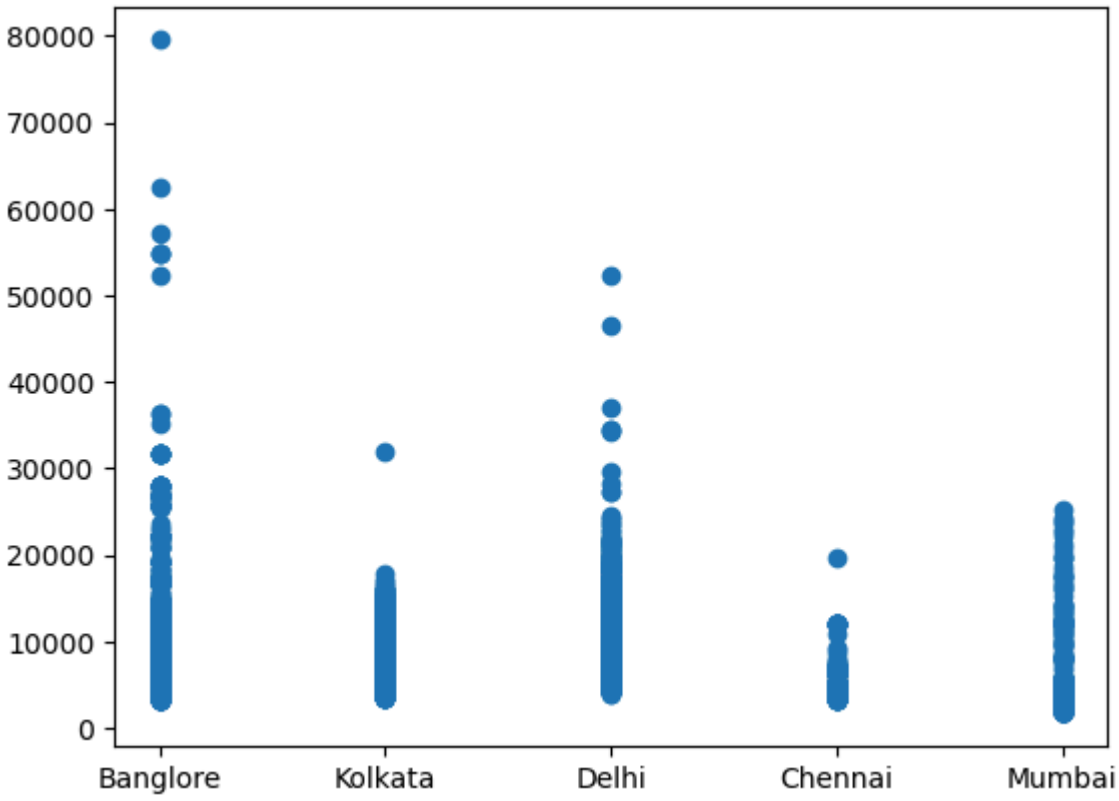
What the Effect of Source at Price?

In [52]:
```
df.groupby('Source')['Price'].mean()
```

Out[52]:
```
Source
Banglore     8022.872877
Chennai      4789.892388
Delhi       10461.600690
Kolkata      9143.083566
Mumbai       5059.708752
Name: Price, dtype: float64
```

In [53]:
```
plt.scatter(df['Source'], df['Price'])
```

Out[53]:
```
<matplotlib.collections.PathCollection at 0x1bed7eb5310>
```



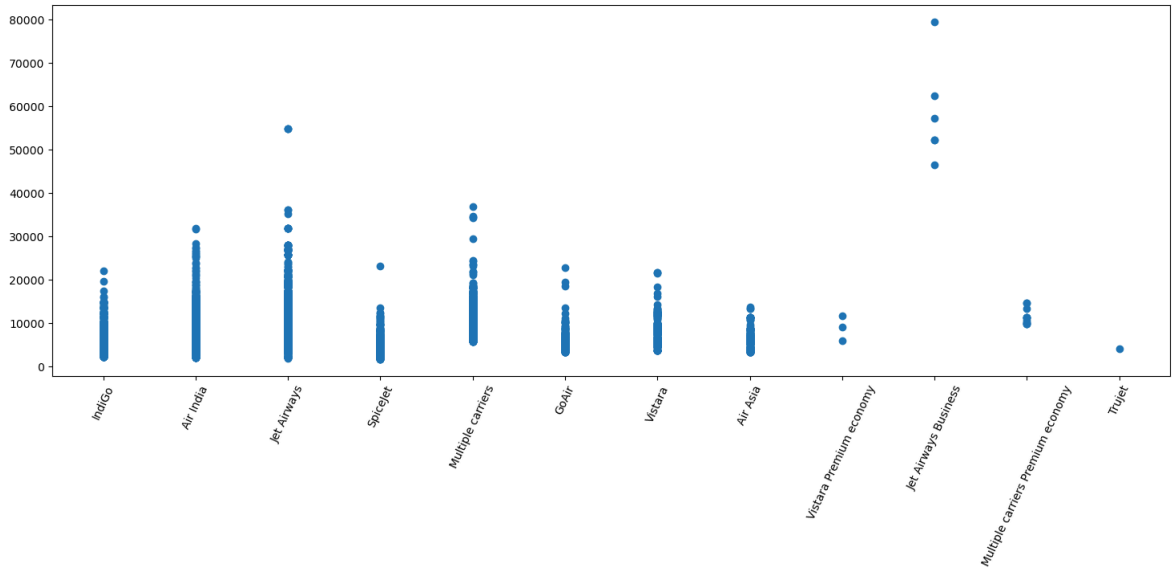
What the Effect of Airline at Price?

In [55]:
```
df.groupby('Airline')['Price'].mean()
```

Out[55]:
```
Airline
Air Asia                                5590.260188
Air India                               9556.608028
GoAir                                   5861.056701
IndiGo                                  5668.469897
Jet Airways                            11599.021081
Jet Airways Business                   58358.666667
Multiple carriers                      10902.678094
Multiple carriers Premium economy      11418.846154
SpiceJet                                4335.841718
Trujet                                  4140.000000
Vistara                                 7801.355649
Vistara Premium economy                 8962.333333
Name: Price, dtype: float64
```

In [56]:
```
plt.figure(figsize=(18,6))
plt.xticks(rotation=65)
plt.scatter(df['Airline'], df['Price'])
```

Out[56]:
```
<matplotlib.collections.PathCollection at 0x1bed7e8d610>
```



How Does The Total Stops Of The Flight Affect The Price?

In [58]:
```
df.groupby('Total_Stops')['Price'].mean()
```

Out[58]:
```
Total_Stops
1 stop      10594.123556
2 stops     12761.099393
3 stops     13260.674419
4 stops     17686.000000
non-stop     5018.506763
Name: Price, dtype: float64
```

Feature Engineering

In [60]:
```
df_tran = df.copy()
df.head()
```

Out[60]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Du
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4

In [61]:
```
# # Airline
df['Airline'].value_counts()
```

Out[61]:
```
Airline
Jet Airways                          3700
IndiGo                               2043
Air India                            1694
Multiple carriers                    1196
SpiceJet                              815
Vistara                               478
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: count, dtype: int64
```

In [62]:
```
df['Airline'] = df['Airline'].str.replace('Jet Airways Business', 'another_Airli
```

In [63]:
```
df['Airline'].value_counts()
```

Out[63]:
```
Airline
Jet Airways                          3700
IndiGo                               2043
Air India                            1694
Multiple carriers                    1196
SpiceJet                              815
Vistara                               478
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
another_Airline                        10
Name: count, dtype: int64
```

In [64]:
```
#  Additional_Info
# =====> replace No Info with No info
df['Additional_Info'] = df['Additional_Info'].str.replace('Info', 'info')
```

In [65]:
```
df['Additional_Info'].value_counts()
```

Out[65]:
```
Additional_Info
No info                         8185
In-flight meal not included     1926
No check-in baggage included     318
1 Long layover                    19
Change airports                    7
Business class                     4
1 Short layover                    1
Red-eye flight                     1
2 Long layover                     1
Name: count, dtype: int64
```

In [66]:
```
df['Additional_Info'] = df['Additional_Info'].str.replace('Change airports', 'ot
```

In [67]:
```
df['Additional_Info'].value_counts()
```

Out[67]:
```
Additional_Info
No info                         8185
In-flight meal not included     1926
No check-in baggage included     318
1 Long layover                    19
others                            14
Name: count, dtype: int64
```

In [68]:
```
#  total_stops
df['Total_Stops'].value_counts()
```

Out[68]:
```
Total_Stops
1 stop      5625
non-stop    3475
2 stops     1318
3 stops       43
4 stops        1
Name: count, dtype: int64
```

In [69]:
```
df['Total_Stops'] = df['Total_Stops'].str.replace('non-stop','0').str.replace('
```

In [70]:
```
df['Total_Stops'].value_counts()
```

Out[70]:
```
Total_Stops
1    5625
0    3475
2    1318
3      43
4       1
Name: count, dtype: int64
```

In [71]:
```
df['Total_Stops'].unique()
```

Out[71]:
```
array(['0', '2', '1', '3', '4'], dtype=object)
```

In [72]:
```
df['Total_Stops'] = df['Total_Stops'].apply(eval).astype('int64')
```

In [73]:
```
## 3. Duration
df['Duration'].head()
```

Out[73]:
```
0    2h 50m
1    7h 25m
2       19h
3    5h 25m
4    4h 45m
Name: Duration, dtype: object
```

In [74]:
```
# convert hours to minutes and delete 'h, m'.
df['Duration'] = df['Duration'].str.replace('h', '*60').str.replace(' ', '+').st
# convert the column to int data type
df['Duration'] = df['Duration'].apply(eval).astype('int64')
```

In [75]:
```
df['Duration'].head()
```

Out[75]:
```
0     170
1     445
2    1140
3     325
4     285
Name: Duration, dtype: int64
```

In [76]:
```
df.rename(columns={'Duration': 'Duration(m)'}, inplace=True)
```

In [77]:
```
# 4.'Date_of_Journey' Column
# Extract day, month, and year from 'Date_of_Journey' column
# 24/03/2019
df[['Day', 'Month', 'Year']] = df['Date_of_Journey'].str.split('/', expand=True)
```

In [78]:
```
# check values of days
df['Day'].value_counts()
```

Out[78]:
```
Day
9     1275
6     1173
27    1092
21    1085
24    1022
15     967
12     946
1      907
18     814
3      743
01     151
09     100
06      95
03      92
Name: count, dtype: int64
```

In [79]:
```
# delete 0 from the day number
df['Day'] = df['Day'].str.replace('0', '')
```

In [80]:
```
df['Day'].unique()
```

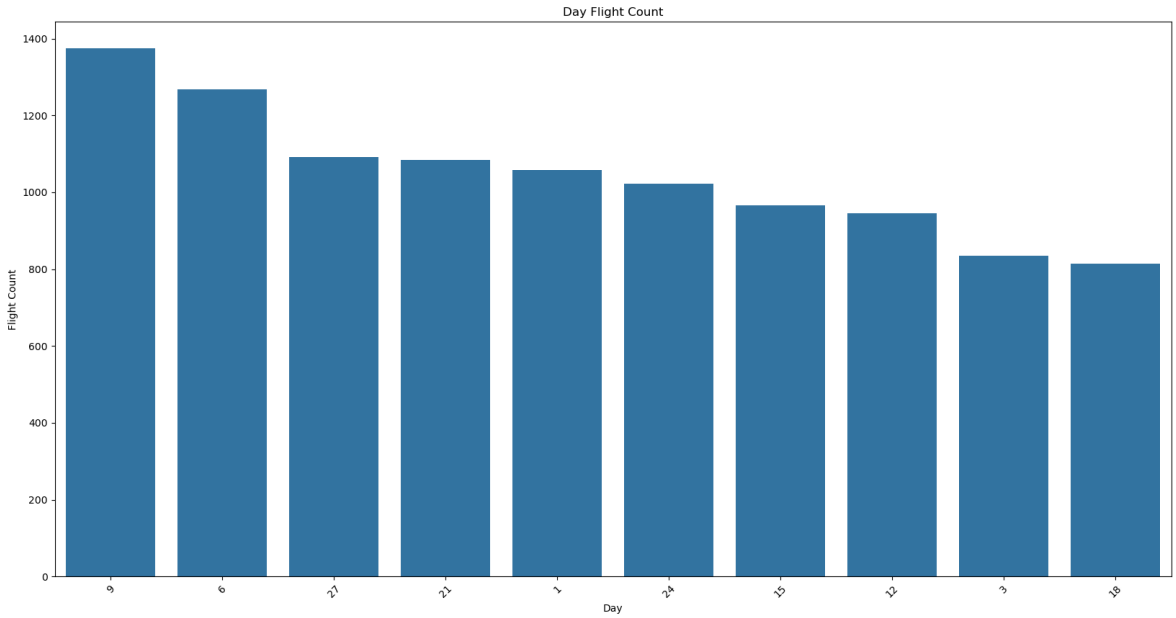
Out[80]:
```
array(['24', '1', '9', '12', '27', '18', '3', '15', '6', '21'],
      dtype=object)
```

In [81]:
```
df['Day'] = df['Day'].apply(eval).astype('int64')
```

In [82]:
```
df['Day'].unique()
```

Out[82]:
```
array([24,  1,  9, 12, 27, 18,  3, 15,  6, 21], dtype=int64)
```

In [83]:
```
# Plotting count Plot for Day flights
plt.figure(figsize=(20, 10))
sorted_df = df['Day'].value_counts().sort_values(ascending=False)
ax = sns.countplot(data=df, x='Day', order=sorted_df.index)

# Set labels and title for the plot
plt.xlabel('Day')
plt.ylabel('Flight Count')
plt.title('Day Flight Count')
plt.xticks(rotation=45)

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



In [84]:
```
# check values of Months
df['Month'].value_counts()
```

Out[84]:
```
Month
05    3395
06    3311
03    2678
04    1078
Name: count, dtype: int64
```

In [85]:
```
# delete 0 from the Month number
df['Month'] = df['Month'].str.replace('0', '')
```

In [86]:
```
df['Month'].unique()
```

Out[86]:
```
array(['3', '5', '6', '4'], dtype=object)
```

In [87]:
```
df['Month'] = df['Month'].apply(eval).astype('int64')
```

In [88]:
```
df['Month'].unique()
```

Out[88]:
```
array([3, 5, 6, 4], dtype=int64)
```

In [89]:
```
# # check values of Years
df['Year'].value_counts()
```

Out[89]:
```
Year
2019    10462
Name: count, dtype: int64
```

In [90]:
```
df.head()
```

Out[90]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Du
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	

In [91]:
```
df.shape
```

Out[91]: (10462, 14)

Dropping NonNeeded column

In [93]:
```
df.drop(columns=['Date_of_Journey', 'Route', 'Dep_Time', 'Arrival_Time','Year'],
```

Cheacking Duplications

In [95]:
```
df.duplicated().sum()
```

Out[95]: 793

In [96]:
```
df.drop_duplicates(inplace=True)
```

In [97]:
```
df.shape
```

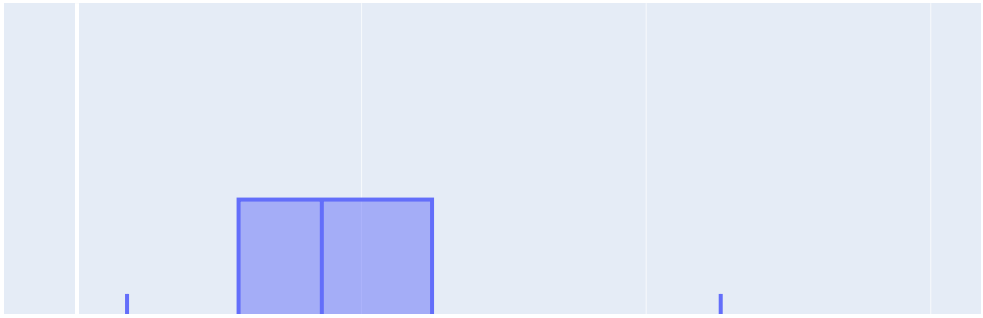
Out[97]: (9669, 9)

Outlires Handling

In [99]:
```
df['Price'].min()
```

Out[99]: 1759

In [100…
```
px.box(x=df['Price'])
```

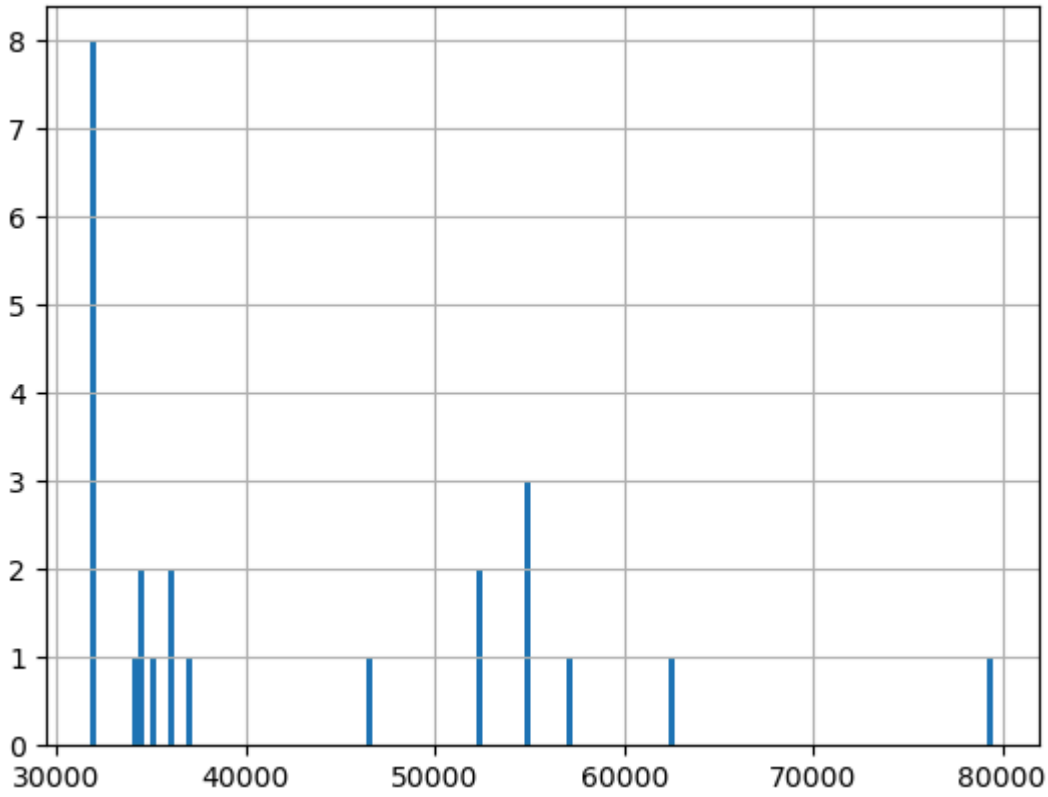


can make 30000 as cutoff

In [102…
```
print("The shape of outliers",df[df['Price'] > 30000].shape)
df['Price'][df['Price'] > 30000].hist(bins=150, )
```

```
The shape of outliers (24, 9)
```

Out[102… <Axes: >



In [103…

```
df[df['Price'] > 30000]
```

Out[103…

	Airline	Source	Destination	Duration(m)	Total_Stops	Additional_Info
396	Multiple carriers	Delhi	Cochin	385	1	No info
657	another_Airline	Banglore	New Delhi	300	1	No info
1478	Jet Airways	Banglore	New Delhi	365	1	No info
1629	Air India	Banglore	New Delhi	1545	2	No info
2099	Jet Airways	Banglore	New Delhi	305	1	No info
2618	Jet Airways	Banglore	New Delhi	375	1	No info
2693	Jet Airways	Banglore	New Delhi	365	1	No info
2924	another_Airline	Banglore	New Delhi	340	1	others
3700	Jet Airways	Banglore	New Delhi	640	1	1 Long layover
5013	Jet Airways	Banglore	New Delhi	695	1	1 Long layover
5372	another_Airline	Banglore	New Delhi	400	1	others
5439	Jet Airways	Banglore	New Delhi	365	1	No info
5662	Jet Airways	Banglore	New Delhi	890	1	No info
5719	Jet Airways	Banglore	New Delhi	435	1	No info
6576	Jet Airways	Banglore	New Delhi	890	1	1 Long layover
6991	Multiple carriers	Delhi	Cochin	760	1	No info
7351	another_Airline	Delhi	Cochin	500	2	No info
7617	Multiple carriers	Delhi	Cochin	630	2	No info
8598	Multiple carriers	Delhi	Cochin	630	2	No info
9019	Jet Airways	Banglore	New Delhi	755	1	1 Long layover
9715	another_Airline	Delhi	Cochin	500	2	No info
10052	Air India	Kolkata	Banglore	155	0	No info
10364	another_Airline	Banglore	New Delhi	280	1	others
10439	Jet Airways	Banglore	New Delhi	860	1	No info

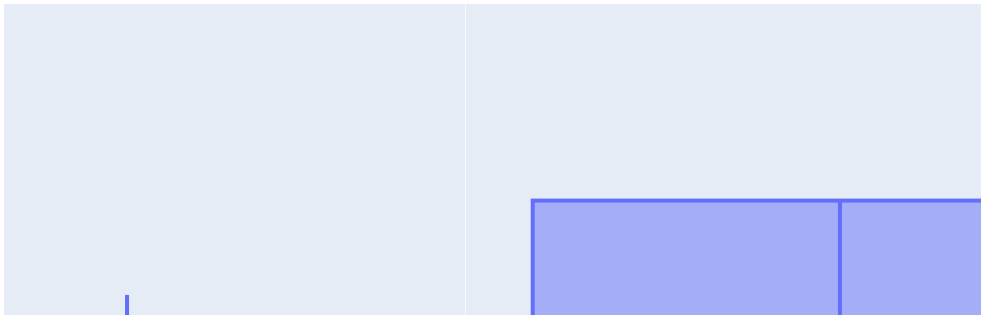
In [104…

```
q1 = 5678
q3 = 12476
min_out = q1 - (1.5 *(q3-q1))
max_out = q3 + (1.5 *(q3-q1))
min_out, max_out
## can make max_out = 23000
```

Out[104… (-4519.0, 22673.0)

In [105…
```
df_without_outlier = df[df['Price'] < 23000]
```

In [106…
```
px.box(x=df_without_outlier['Price'])
```



In [107…
```
df.head()
```

Out[107…

	Airline	Source	Destination	Duration(m)	Total_Stops	Additional_Info	Price	Da
0	IndiGo	Banglore	New Delhi	170	0	No info	3897	24
1	Air India	Kolkata	Banglore	445	2	No info	7662	
2	Jet Airways	Delhi	Cochin	1140	2	No info	13882	9
3	IndiGo	Kolkata	Banglore	325	1	No info	6218	12
4	IndiGo	Banglore	New Delhi	285	1	No info	13302	

In [108…
```
x= df.drop(columns=['Price'])
y = df['Price']
```

In [109…
```
x
```

Out[109…

	Airline	Source	Destination	Duration(m)	Total_Stops	Additional_Info	Day
0	IndiGo	Banglore	New Delhi	170	0	No info	24
1	Air India	Kolkata	Banglore	445	2	No info	1
2	Jet Airways	Delhi	Cochin	1140	2	No info	9
3	IndiGo	Kolkata	Banglore	325	1	No info	12
4	IndiGo	Banglore	New Delhi	285	1	No info	1
...	...	...	...	...	...	...	...
10677	SpiceJet	Banglore	Delhi	160	0	No check-in baggage included	21
10678	Air Asia	Kolkata	Banglore	150	0	No info	9
10679	Air India	Kolkata	Banglore	155	0	No info	27
10681	Vistara	Banglore	New Delhi	160	0	No info	1
10682	Air India	Delhi	Cochin	500	2	No info	9

9669 rows × 8 columns

In [110…

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder , StandardScaler , OrdinalEncode
from sklearn.impute import SimpleImputer , KNNImputer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import  SGDRegressor
from sklearn.model_selection import train_test_split , cross_validate

from sklearn.metrics import mean_absolute_error, mean_squared_error

from sklearn.ensemble import  RandomForestRegressor

from sklearn.model_selection import GridSearchCV
```

In [111…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9669 entries, 0 to 10682
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          9669 non-null   object
 1   Source           9669 non-null   object
 2   Destination      9669 non-null   object
 3   Duration(m)      9669 non-null   int64 
 4   Total_Stops      9669 non-null   int64 
 5   Additional_Info  9669 non-null   object
 6   Price            9669 non-null   int64 
 7   Day              9669 non-null   int64 
 8   Month            9669 non-null   int64 
dtypes: int64(5), object(4)
memory usage: 755.4+ KB
```

In [112…
```
df.head()
```

Out[112…

	Airline	Source	Destination	Duration(m)	Total_Stops	Additional_Info	Price	Da
0	IndiGo	Banglore	New Delhi	170	0	No info	3897	24
1	Air India	Kolkata	Banglore	445	2	No info	7662	
2	Jet Airways	Delhi	Cochin	1140	2	No info	13882	9
3	IndiGo	Kolkata	Banglore	325	1	No info	6218	12
4	IndiGo	Banglore	New Delhi	285	1	No info	13302	

In [113…
```
Num_Columns = x.select_dtypes(include="number")
Cat_Columns = x.select_dtypes(include="object_")
```

In [114…
```
Num_Columns.head()
```

Out[114…

	Duration(m)	Total_Stops	Day	Month
0	170	0	24	3
1	445	2	1	5
2	1140	2	9	6
3	325	1	12	5
4	285	1	1	3

In [115…
```
Cat_Columns.head()
```

Out[115…

	Airline	Source	Destination	Additional_Info
0	IndiGo	Banglore	New Delhi	No info
1	Air India	Kolkata	Banglore	No info
2	Jet Airways	Delhi	Cochin	No info
3	IndiGo	Kolkata	Banglore	No info
4	IndiGo	Banglore	New Delhi	No info

In [116…
```
Num_Steps = list()
Num_Steps.append(("Num_Imputer" , KNNImputer()))
Num_Steps.append(("Scaler" , StandardScaler()))
Num_Pipeline = Pipeline(steps=Num_Steps)
```

In [117…
```
Cat_Steps = list()
Cat_Steps.append(("Cat_Imputer" , SimpleImputer(strategy='most_frequent')))
Cat_Steps.append(("Cat_Encoder" , OrdinalEncoder()))
Cat_Pipeline = Pipeline(steps= Cat_Steps)
```

In [118…
```
Transformer = ColumnTransformer(transformers=[('Num' , Num_Pipeline , Num_Column
                                               ('Cat' , Cat_Pipeline , Cat_Column
```

In [119…
```
models = list()
models.append(("GDR" , SGDRegressor()))
models.append(("RF" ,  RandomForestRegressor()))
```

In [120…
```
for model in models:
    steps = list()
    steps.append(("Preprocessing" , Transformer))
    steps.append(model)
    pipeline = Pipeline(steps = steps)
    scores = cross_validate(pipeline , x , y , scoring="neg_mean_squared_error",
    print(model[0])
    print("Train_accuracy" , scores["train_score"].mean() )
    print("-" * 10)
    print("Test_accuracy" , scores["test_score"].mean())
    a = scores['test_score'].mean()
    a=a*-1
    print(a)
    print(np.sqrt(a))
    print("-" * 20)
    print("\n")
```

```
GDR
Train_accuracy -12757777.350196801
----------
Test_accuracy -12780187.834903227
12780187.834903227
3574.9388575055696
--------------------


RF
Train_accuracy -910813.4617045013
----------
Test_accuracy -3698657.593897085
3698657.593897085
1923.189432660518
--------------------
```

In [121…

```
models = list()
models.append(("GDR" , SGDRegressor(penalty=None, random_state=42, learning_rate
models.append(("RF" ,  RandomForestRegressor( n_estimators= 100, max_depth= 30,

for model in models:
    steps = list()
    steps.append(("Preprocessing" , Transformer))
    steps.append(model)
    pipeline = Pipeline(steps = steps)
    scores = cross_validate(pipeline , x , y , scoring="neg_mean_squared_error"
    print(model[0])
    print("Train_accuracy" , scores["train_score"].mean() )
    print("-" * 10)
    print("Test_accuracy" , scores["test_score"].mean())
    a = scores['test_score'].mean()
    a=a*-1
    print(a)
    import math
    print(np.sqrt(a))
    print("-" * 20)
    print("\n")
```

```
GDR
Train_accuracy -15145964.29150032
----------
Test_accuracy -15063010.282182151
15063010.282182151
3881.1094138380267
--------------------


RF
Train_accuracy -3548380.5533182197
----------
Test_accuracy -4393963.907492797
4393963.907492797
2096.1784054542677
--------------------
```

Tuning with Grid Search CV

In [123…
```
reg1 = SGDRegressor()
reg5 = RandomForestRegressor()
```

In [127…
```
steps = list()
steps.append(("Preprocessing" , Transformer))
steps.append(("RF", reg5))
pipeline = Pipeline(steps = steps)
```

In [124…
```
# GD
param1 = {}
param1['GDR__max_itr'] = [1500, 1000]
param1['GDR__learning_rate'] = ['invscaling', 'constant']
param1['GDR__eta0'] = [0.01, 0.001, 0.0001, 0.005]
param1['GDR__penalty'] = ['l2','l1', None]
param1['GDR__alpha'] = [0.01, 0.001, 0.0001, 0.005]

# RandomForest
param5 = {}
param5['RF__n_estimators'] = [10, 100]          # Number of trees in the forest
param5['RF__max_depth'] = [ 8, 15 , 25]                  # Maximum depth of each
```

In [125…
```
param5
```

Out[125…
```
{'RF__n_estimators': [10, 100], 'RF__max_depth': [8, 15, 25]}
```

In [129…
```
grid = GridSearchCV(pipeline, param5, cv=10, return_train_score=True, scoring='n
```

In [130…
```
# param1# grid.best_params_
grid.best_params_
```

Out[130…
```
{'RF__max_depth': 15, 'RF__n_estimators': 100}
```

In [131…
```
grid.best_score_ * -1
```

Out[131…
```
1866.2658572931646
```

In [132…
```
pd.DataFrame(grid.cv_results_)
```

Out[132…

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_RF__max_depth
0	0.249331	0.033209	0.018026	0.004678	8
1	1.764661	0.027800	0.041290	0.003842	8
2	0.345342	0.014790	0.016959	0.001345	15
3	3.096491	0.100400	0.064530	0.018475	15
4	0.411803	0.014032	0.017850	0.001038	25
5	3.616116	0.061055	0.068019	0.003126	25

6 rows × 32 columns

In [137…

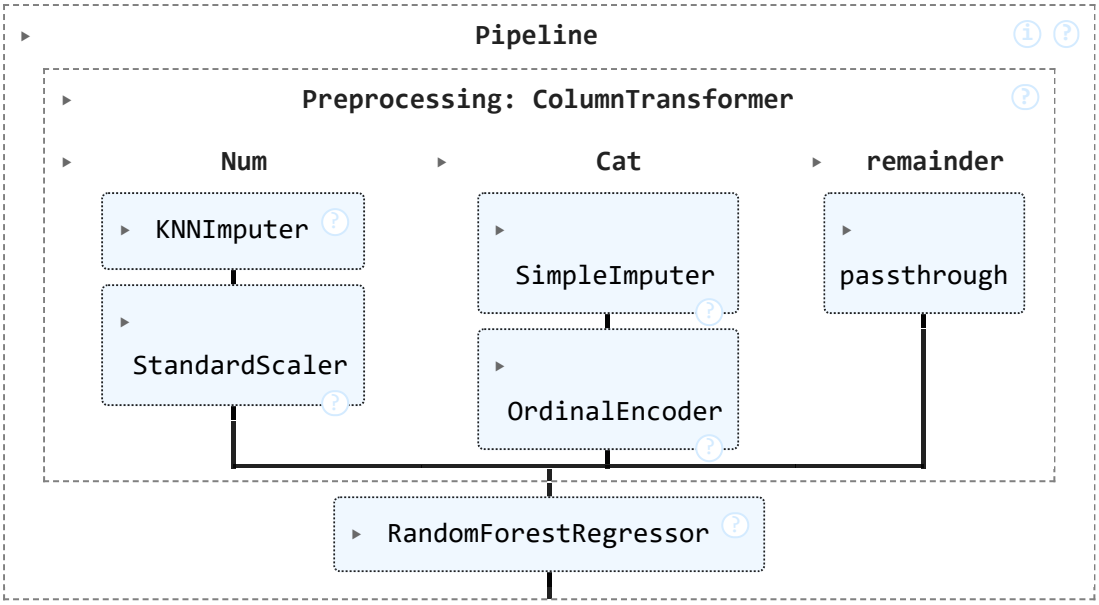
```
models4 = list()
models4.append(("RF1" ,  RandomForestRegressor( n_estimators= 10, max_depth= 25
models4.append(("RF2" ,  RandomForestRegressor( n_estimators= 100, max_depth= 15
models4.append(("RF3" ,  RandomForestRegressor( n_estimators= 10, max_depth= 15)
models4.append(("RF4" ,  RandomForestRegressor( n_estimators= 600, max_depth= 25
models4.append(("RF5" ,  RandomForestRegressor( n_estimators= 100, max_depth= 25

for model in models4:
    steps = list()
    steps.append(("Preprocessing" , Transformer))
    steps.append(model)
    pipeline = Pipeline(steps = steps)
    scores = cross_validate(pipeline , x , y , scoring="neg_mean_squared_error"
    print(model[0])
    print("Train_accuracy" , math.sqrt(scores["train_score"].mean()*-1 ))
    print("-" * 10)
    print("Test_accuracy" , math.sqrt(scores["test_score"].mean()*-1))

    print("-" * 20)
    print("\n")
```

```
RF1
Train_accuracy 1025.9172038046875
----------
Test_accuracy 1972.3489378709833
--------------------


RF2
Train_accuracy 1134.7785813696605
----------
Test_accuracy 1863.3757926255068
--------------------


RF3
Train_accuracy 1187.145592246644
----------
Test_accuracy 1900.5742433174623
--------------------


RF4
Train_accuracy 949.7719934318201
----------
Test_accuracy 1908.3757322845834
--------------------


RF5
Train_accuracy 955.7556121934961
----------
Test_accuracy 1918.1872293988017
--------------------
```

In [139…
```
steps = list()
steps.append(("Preprocessing" , Transformer))
steps.append(("RF2" ,  RandomForestRegressor( n_estimators= 100, max_depth= 15))
final_pipeline = Pipeline(steps = steps)
```

In [141…
```
final_pipeline.fit(x, y)
```

Out[141…



Saving Model & Deployment

In [143…

```
import joblib
joblib.dump(final_pipeline , "flight_model.pkl")
joblib.dump(x.columns,"Inputs.pkl")
```

Out[143…

```
['Inputs.pkl']
```

GUI Building

In [145…

```
%%writefile flight_app.py


import joblib
import pandas as pd
import numpy as np
import streamlit as st
import datetime

Model = joblib.load("flight_model.pkl")
Inputs = joblib.load("inputs.pkl")

def prediction(Airline, Source, Destination, Duration ,Total_Stops, Additional_I
    df = pd.DataFrame(columns=Inputs)
    df.at[0,"Airline"] = Airline
    df.at[0,"Source"] = Source
    df.at[0,"Destination"] = Destination
    df.at[0,"Duration(m)"] = Duration
    df.at[0,"Total_Stops"] = Total_Stops
    df.at[0,"Additional_Info  "] = Additional_Info
    df.at[0,"Day"] = Day
    df.at[0,"Month"] = Month


    result = Model.predict(df)[0]
    return result
```

```
def Main():
    Airline_list=['IndiGo', 'Air India', 'Jet Airways', 'SpiceJet','Multiple car
     'GoAir', 'Vistara', 'Air Asia', 'Vistara Premium economy', 'Jet Airways Bus

    Additional_Info_list = ['No info', 'In-flight meal not included',
       'No check-in baggage included', '1 Short layover', 'No Info',
       '1 Long layover', 'Change airports', 'Business class',
       'Red-eye flight', '2 Long layover']

    st.title("Flight Price Prediction")
    Airline = st.selectbox("Airline Name",Airline_list)
    Source = st.selectbox("Source",['Banglore', 'Kolkata', 'Delhi', 'Chennai', '
    Destination = st.selectbox("Destination",['New Delhi', 'Banglore', 'Cochin',

    Duration_time = st.text_input("Duration Time (e.g., 2h 50m)", value="2h 40m"
    h =  Duration_time.replace('h', '*60').replace(' ', '+').replace('m', '')
    Duration = eval(h)

    Total_Stops = st.slider("Flight Stops",min_value = 0.0, max_value = 4.0, ste
    Additional_Info = st.selectbox("Info About Flight",Additional_Info_list)


    d = st.date_input("Date_of_Journey", datetime.date(2019, 7, 6))
    Day = d.day
    Month = d.month


    h =  Duration_time.replace('h', '*60').replace(' ', '+').replace('m', '')

    Duration = eval(h)

    if st.button("Predict"):

        result = prediction(Airline, Source, Destination, Duration ,Total_Stops,
        st.text(f"The Flight price is: {result}")
Main()
```

```
Overwriting flight_app.py
```