____Airfare Price Prediction_____

Importing basic libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
sns.set()
```

Importing dataset

In [3]:

```
df = pd.read_excel(r'C:\Users\Hp\Desktop\Data_Train.xlsx
```

set max coulmns to None so we can see all columns from dataset

In [4]:

```
pd.set_option('display.max_columns', None)
```

Checking the basic information of dataset

In [5]:

```
df.head()
```

Out[5]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	P
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13

In [6]:

```
df.columns
```

Out[6]:

```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price'],
      dtype='object')
```

In [7]:

```
df.shape
```

Out[7]:

```
(10683, 11)
```

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
```

In [9]:

```
df["Duration"].value_counts()
```

Out[9]:

```
2h 50m     550
1h 30m     386
2h 45m     337
2h 55m     337
2h 35m     329
          ...
31h 30m      1
30h 25m      1
42h 5m       1
4h 10m       1
47h 40m      1
Name: Duration, Length: 368, dtype: int64
```

In [10]:

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

Out[10]:

```
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 [11]:

```
df.dropna(inplace=True)
```

In [12]:

```
df.shape
```

Out[12]:

```
(10682, 11)
```

In [13]:

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

Out[13]:

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Additional_Info    0
Price              0
dtype: int64
```

In [14]:

```
df["Additional_Info"].unique()
```

Out[14]:

```
array(['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'], dtype=object)
```

checking the unique values in Route counts

In [15]:

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

Out[15]:

```
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)
```

In [16]:

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

Out[16]:

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

There is only one value in Total_Stops & Route so dropping null value from dataset

In [17]:

```
df.dropna(inplace = True)
```

In [18]:

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

Out[18]:

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Additional_Info    0
Price              0
dtype: int64
```

In [19]:

```
df.head()
```

Out[19]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	P
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13

Now extracting day values and month values from Date_of_Journey

In [20]:

```
df["Journey_day"] = pd.to_datetime(df.Date_of_Journey, format="%d/%m/%Y").dt.day
```

In [21]:

```
df["Journey_month"] = pd.to_datetime(df["Date_of_Journey"], format = "%d/%m/%Y").dt.month
```

In [22]:

```
df.head(2)
```

Out[22]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Pri
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	38
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	76

In [23]:

```
df.drop(["Date_of_Journey"], axis = 1, inplace = True)
```

In [24]:

```
df["Dep_hour"] = pd.to_datetime(df["Dep_Time"]).dt.hour

df["Dep_min"] = pd.to_datetime(df["Dep_Time"]).dt.minute

df.drop(["Dep_Time"], axis = 1, inplace = True)
```

In [25]:

```
df.head()
```

Out[25]:

	Airline	Source	Destination	Route	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_n
0	IndiGo	Banglore	New Delhi	BLR → DEL	01:10 22 Mar	2h 50m	non-stop	No info	3897	24	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	13:15	7h 25m	2 stops	No info	7662	1	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	04:25 10 Jun	19h	2 stops	No info	13882	9	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	23:30	5h 25m	1 stop	No info	6218	12	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	21:35	4h 45m	1 stop	No info	13302	1	

similarly extracting hour and minutes from arrival time

In [26]:

```
# Extracting Hours
df["Arrival_hour"] = pd.to_datetime(df.Arrival_Time).dt.hour

# Extracting Minutes
df["Arrival_min"] = pd.to_datetime(df.Arrival_Time).dt.minute

# Now we can drop Arrival_Time as it is of no use
df.drop(["Arrival_Time"], axis = 1, inplace = True)
```

In [27]:

```
df.head()
```

Out[27]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hc
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	5	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	9	6	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	12	5	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	1	3	

for checking the duration values

In [28]:

```
duration = list(df["Duration"])
duration
```

Out[28]:

```
['2h 50m',
 '7h 25m',
 '19h',
 '5h 25m',
 '4h 45m',
 '2h 25m',
 '15h 30m',
 '21h 5m',
 '25h 30m',
 '7h 50m',
 '13h 15m',
 '2h 35m',
 '2h 15m',
 '12h 10m',
 '2h 35m',
 '26h 35m',
 '4h 30m',
```

using loop to check duration if it contains only hour mins as suffixes

In [29]:

```
for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration contains only hour or mins
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"   # Adds 0 minute
        else:
            duration[i] = "0h " + duration[i]           # Adds 0 hour
```

In [30]:

```
duration
```

Out[30]:

```
['2h 50m',
 '7h 25m',
 '19h 0m',
 '5h 25m',
 '4h 45m',
 '2h 25m',
 '15h 30m',
 '21h 5m',
 '25h 30m',
 '7h 50m',
 '13h 15m',
 '2h 35m',
 '2h 15m',
 '12h 10m',
 '2h 35m',
 '26h 35m',
 '4h 30m',
```

In [31]:

```
df.head(1)
```

Out[31]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hou
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	2

In [32]:

```
i = "6h 15m"
int(i.split(sep = "m")[0].split()[-1])
```

Out[32]:

```
15
```

In [ ]:

```
for i in range(len(duration)):
    v=duration[i].split()
    print(v[0])
```

In [34]:

```
duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))    # Extract hours from duration
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))   # Extracts only minutes from duration
```

In [35]:

```
# Adding duration_hours and duration_mins list to train_data dataframe
df["Duration_hours"] = duration_hours
df["Duration_mins"] = duration_mins
```

In [36]:

```
df.drop(["Duration"], axis = 1, inplace = True)
```

In [37]:

```
df.head(2)
```

Out[37]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_mi
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	3897	24	3	22	2
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	7662	1	5	5	5

Now handling the Data in Categorical manner

In [38]:

```
df["Airline"].value_counts()
```

Out[38]:

```
Jet Airways                          3849
IndiGo                               2053
Air India                            1751
Multiple carriers                    1196
SpiceJet                              818
Vistara                               479
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
```

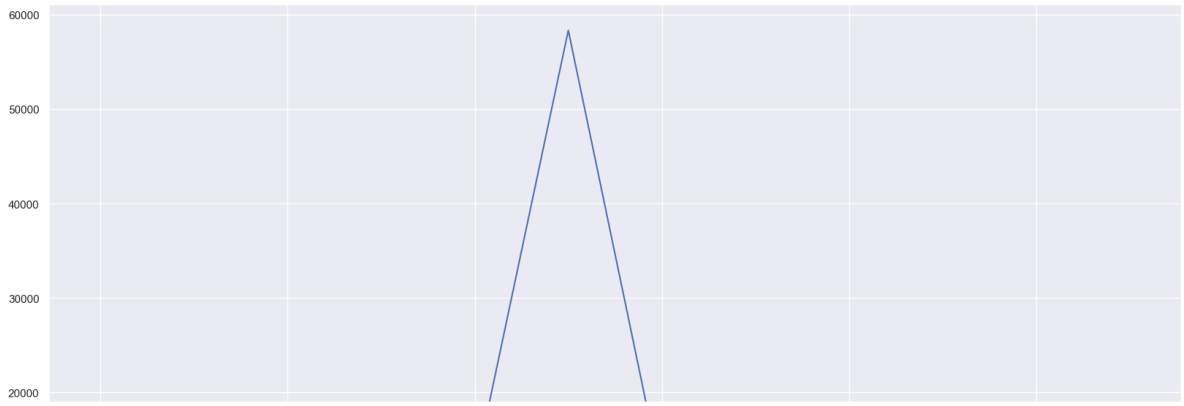
checking price according to Airline

In [39]:

```
df.groupby('Airline')['Price'].mean().plot(figsize=(20,10))
```

Out[39]:

```
<AxesSubplot:xlabel='Airline'>
```



In [40]:

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

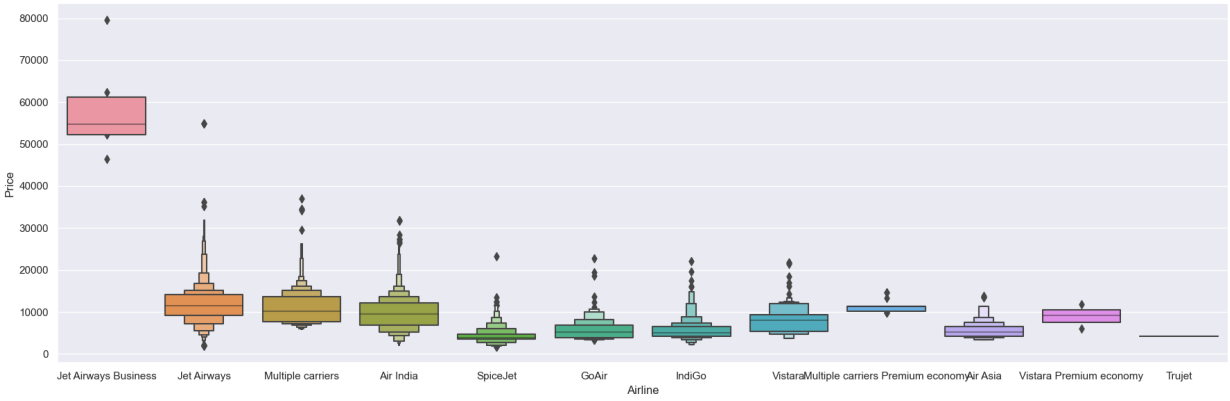
Out[40]:

```
Airline
Air Asia                                5590.260188
Air India                               9612.427756
GoAir                                   5861.056701
IndiGo                                  5673.682903
Jet Airways                            11643.923357
Jet Airways Business                   58358.666667
Multiple carriers                      10902.678094
Multiple carriers Premium economy      11418.846154
SpiceJet                                4338.284841
Trujet                                  4140.000000
Vistara                                 7796.348643
Vistara Premium economy                 8962.333333
Name: Price, dtype: float64
```

average price according to Airline

In [41]:

```
sns.catplot(y = "Price", x = "Airline", data = df.sort_values("Price", ascending = False), kind="boxen", height = 
plt.show()
```



Appplying OneHotEncoding for addressing categorical data

In [42]:

```
df["Airline"].unique()
```

Out[42]:

```
array(['IndiGo', 'Air India', 'Jet Airways', 'SpiceJet',
       'Multiple carriers', 'GoAir', 'Vistara', 'Air Asia',
       'Vistara Premium economy', 'Jet Airways Business',
       'Multiple carriers Premium economy', 'Trujet'], dtype=object)
```

In [43]:

```
Airline = df[["Airline"]]

Airline = pd.get_dummies(Airline, drop_first= True)

Airline.head()
```

Out[43]:

	Airline_Air India	Airline_GoAir	Airline_IndiGo	Airline_Jet Airways	Airline_Jet Airways Business	Airline_Multiple carriers	Airline_Multiple carriers Premium economy	Airline_SpiceJet	Airline_
0	0	0	1	0	0	0	0	0	
1	1	0	0	0	0	0	0	0	
2	0	0	0	1	0	0	0	0	
3	0	0	1	0	0	0	0	0	
4	0	0	1	0	0	0	0	0	

In [44]:

```
df["Source"].value_counts()
```

Out[44]:

```
Delhi       4536
Kolkata     2871
Banglore    2197
Mumbai       697
Chennai      381
Name: Source, dtype: int64
```

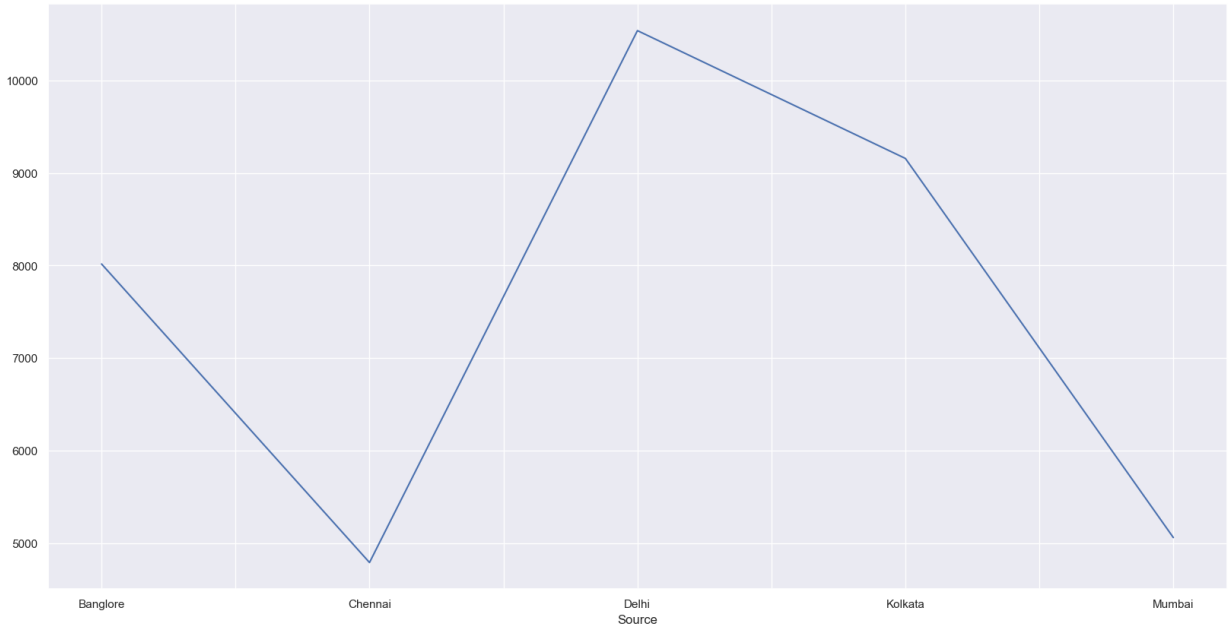
average price according to source

In [45]:

```
df.groupby('Source')['Price'].mean().plot(figsize=(20,10))
```

Out[45]:

```
<AxesSubplot:xlabel='Source'>
```



In [46]:

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

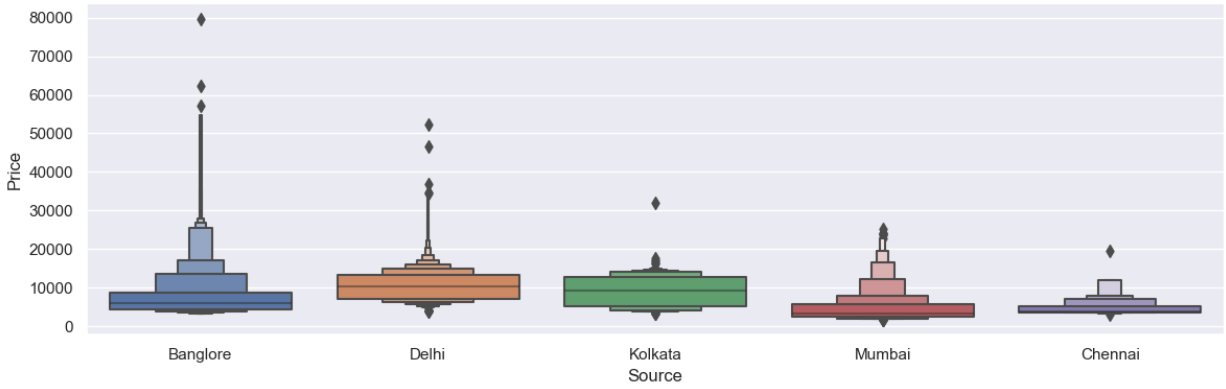
Out[46]:

```
Source
Banglore     8017.464269
Chennai      4789.892388
Delhi       10540.113536
Kolkata      9158.389411
Mumbai       5059.708752
Name: Price, dtype: float64
```

Source and Price

In [47]:

```
sns.catplot(y = "Price", x = "Source", data = df.sort_values("Price", ascending = False), kind="boxen", height = 4
plt.show()
```



In [48]:

```
Source = df[["Source"]]

Source = pd.get_dummies(Source, drop_first= True)

Source.head()
```

Out[48]:

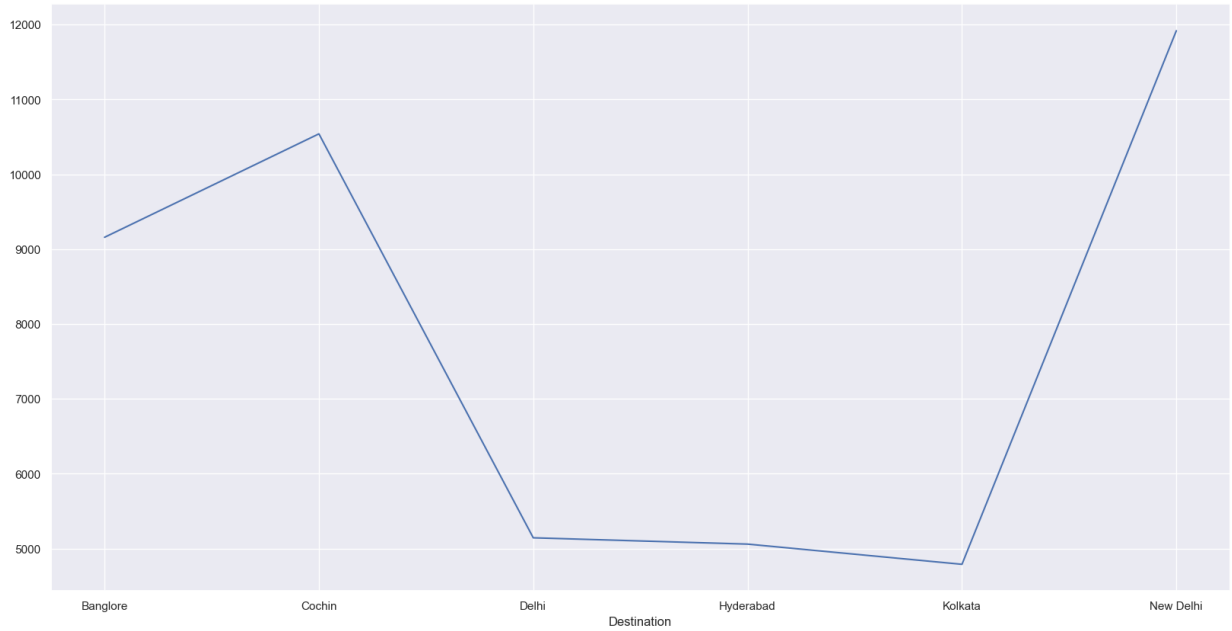
	Source_Chennai	Source_Delhi	Source_Kolkata	Source_Mumbai
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	0

In [49]:

```
df.groupby('Destination')['Price'].mean().plot(figsize=(20,10))
```

Out[49]:

```
<AxesSubplot:xlabel='Destination'>
```



In [50]:

```
df["Destination"].value_counts()
```

Out[50]:

```
Cochin       4536
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: Destination, dtype: int64
```

In [51]:

```
Destination = df[["Destination"]]

Destination = pd.get_dummies(Destination, drop_first = True)

Destination.head()
```

Out[51]:

	Destination_Cochin	Destination_Delhi	Destination_Hyderabad	Destination_Kolkata	Destination_New Delhi
0	0	0	0	0	1
1	0	0	0	0	0
2	1	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	1

In [52]:

```
df["Route"]
```

Out[52]:

```
0                    BLR → DEL
1            CCU → IXR → BBI → BLR
2            DEL → LKO → BOM → COK
3            CCU → NAG → BLR
4            BLR → NAG → DEL
                 ...
10678                CCU → BLR
10679                CCU → BLR
10680                BLR → DEL
10681                BLR → DEL
10682        DEL → GOI → BOM → COK
Name: Route, Length: 10682, dtype: object
```

In [53]:

```
df["Additional_Info"].value_counts()
```

Out[53]:

```
No info                         8344
In-flight meal not included     1982
No check-in baggage included     320
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: Additional_Info, dtype: int64
```

Dropping Route and Additional_Info columns because of insignificance

In [54]:

```
df.drop(["Route", "Additional_Info"], axis = 1, inplace = True)
```

In [55]:

```
df["Total_Stops"].value_counts()
```

Out[55]:

```
1 stop      5625
non-stop    3491
2 stops     1520
3 stops       45
4 stops        1
Name: Total_Stops, dtype: int64
```

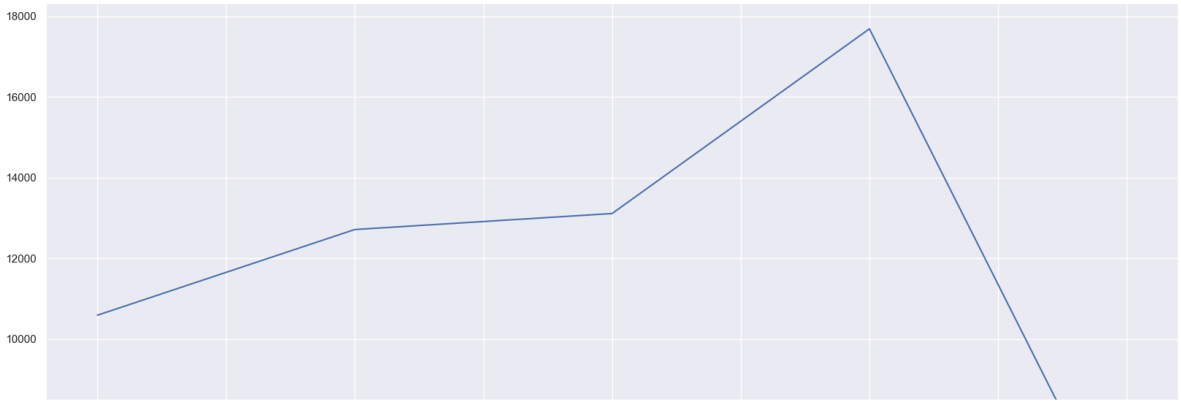
relation between Total_Stops and Price

In [56]:

```
df.groupby('Total_Stops')['Price'].mean().plot(figsize=(20,10))
```

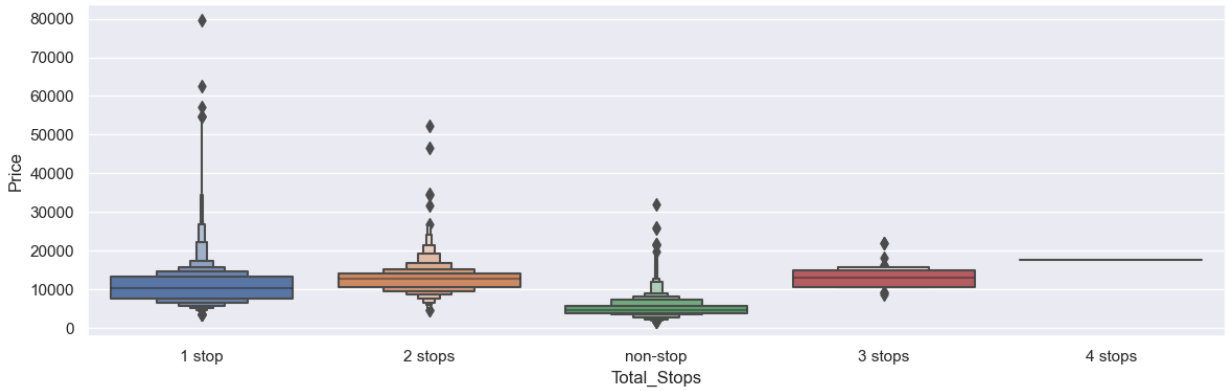
Out[56]:

```
<AxesSubplot:xlabel='Total_Stops'>
```



In [57]:

```
sns.catplot(y = "Price", x = "Total_Stops", data = df.sort_values("Price", ascending = False), kind="boxen", heigh
plt.show()
```



Replacing the categorical value in Total_stops with numeric value

In [58]:

```
df.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4 stops": 4}, inplace = True)
```

In [59]:

```
df.head()
```

Out[59]:

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arriva
0	IndiGo	Banglore	New Delhi	0	3897	24	3	22	20	1	
1	Air India	Kolkata	Banglore	2	7662	1	5	5	50	13	
2	Jet Airways	Delhi	Cochin	2	13882	9	6	9	25	4	
3	IndiGo	Kolkata	Banglore	1	6218	12	5	18	5	23	
4	IndiGo	Banglore	New Delhi	1	13302	1	3	16	50	21	

Now merging all encoded data with orignal dataset

In [60]:

```
df = pd.concat([df, Airline, Source, Destination], axis = 1)
```

In [61]:

```
df.head()
```

Out[61]:

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arriva
0	IndiGo	Banglore	New Delhi	0	3897	24	3	22	20	1	
1	Air India	Kolkata	Banglore	2	7662	1	5	5	50	13	
2	Jet Airways	Delhi	Cochin	2	13882	9	6	9	25	4	
3	IndiGo	Kolkata	Banglore	1	6218	12	5	18	5	23	
4	IndiGo	Banglore	New Delhi	1	13302	1	3	16	50	21	

Dropping Categorical columns from dataset

In [62]:

```
df.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)
```

In [63]:

```
df.head()
```

Out[63]:

	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duratior
0	0	3897	24	3	22	20	1	10	2	
1	2	7662	1	5	5	50	13	15	7	
2	2	13882	9	6	9	25	4	25	19	
3	1	6218	12	5	18	5	23	30	5	
4	1	13302	1	3	16	50	21	35	4	

In [64]:

```
df.shape
```

Out[64]:

```
(10682, 30)
```

In [65]:

```
df.shape
```

Out[65]:

```
(10682, 30)
```

Check all columns from dataset

In [66]:

```
df.columns
```

Out[66]:

```
Index(['Total_Stops', 'Price', 'Journey_day', 'Journey_month', 'Dep_hour',
       'Dep_min', 'Arrival_hour', 'Arrival_min', 'Duration_hours',
       'Duration_mins', 'Airline_Air India', 'Airline_GoAir', 'Airline_IndiGo',
       'Airline_Jet Airways', 'Airline_Jet Airways Business',
       'Airline_Multiple carriers',
       'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',
       'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium economy',
       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
       'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
       'Destination_Kolkata', 'Destination_New Delhi'],
      dtype='object')
```

Dividing Independent and Dependent variables apart

In [67]:

```
X = df.drop('Price',axis=1)
y = df.Price
```

In [68]:

```
X.head()
```

Out[68]:

	Total_Stops	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins
0	0	24	3	22	20	1	10	2	50
1	2	1	5	5	50	13	15	7	25
2	2	9	6	9	25	4	25	19	0
3	1	12	5	18	5	23	30	5	25
4	1	1	3	16	50	21	35	4	45

Finds correlation between Independent and dependent attributes

In [69]:

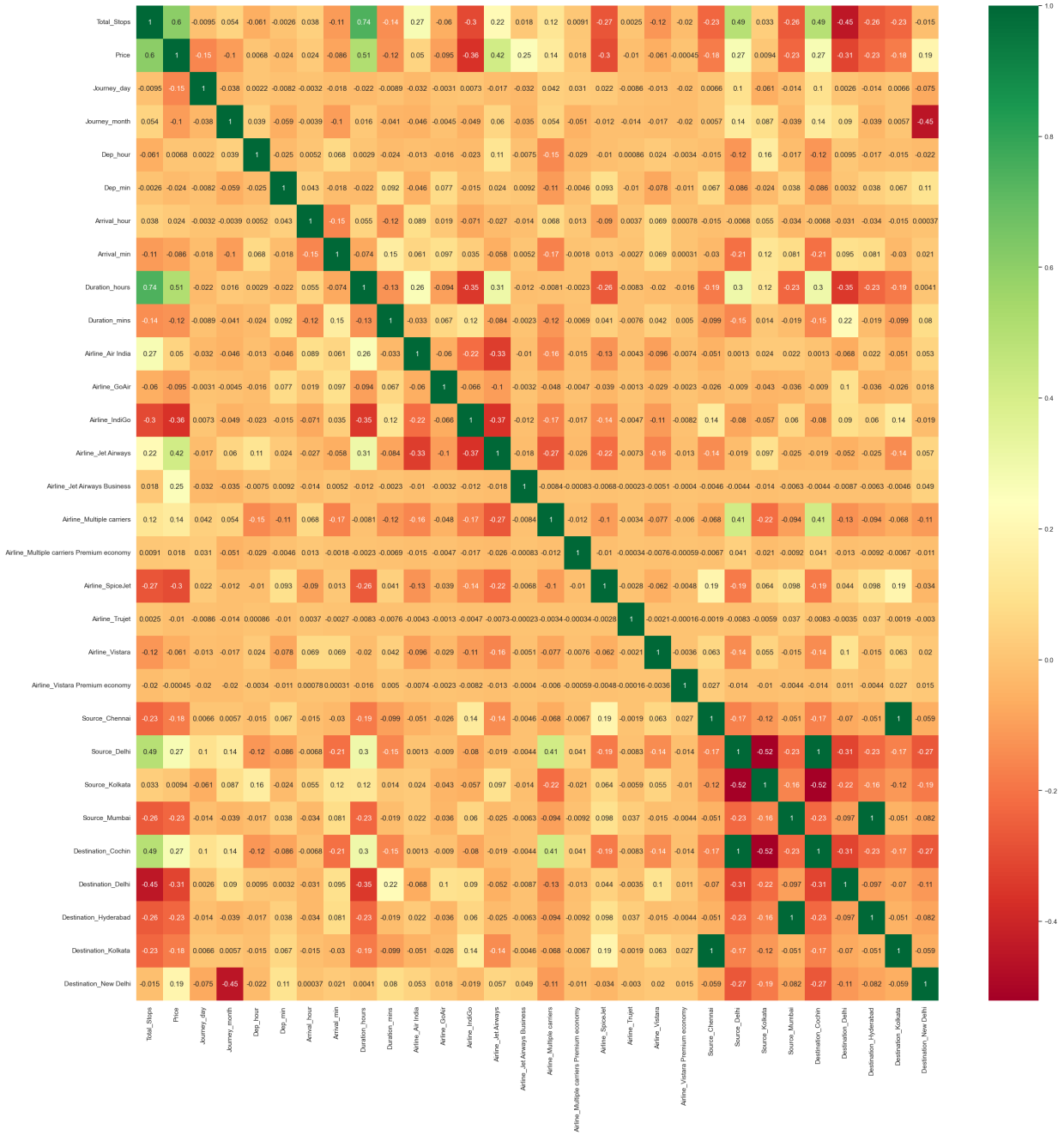
```
df.corr()
```

Out[69]:

	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min
Total_Stops	1.000000	0.603897	-0.009451	0.054383	-0.061476	-0.002618	0.038140	-0.106940
Price	0.603897	1.000000	-0.153774	-0.103643	0.006799	-0.024458	0.024244	-0.086155
Journey_day	-0.009451	-0.153774	1.000000	-0.038359	0.002170	-0.008170	-0.003245	-0.017510
Journey_month	0.054383	-0.103643	-0.038359	1.000000	0.039127	-0.059267	-0.003927	-0.100626
Dep_hour	-0.061476	0.006799	0.002170	0.039127	1.000000	-0.024745	0.005180	0.067911
Dep_min	-0.002618	-0.024458	-0.008170	-0.059267	-0.024745	1.000000	0.043122	-0.017597
Arrival_hour	0.038140	0.024244	-0.003245	-0.003927	0.005180	0.043122	1.000000	-0.154363
Arrival_min	-0.106940	-0.086155	-0.017510	-0.100626	0.067911	-0.017597	-0.154363	1.000000
Duration_hours	0.739916	0.508778	-0.022059	0.016141	0.002869	-0.022104	0.055276	-0.074450
Duration_mins	-0.136706	-0.124855	-0.008940	-0.040897	-0.023707	0.092485	-0.118309	0.151628
Airline_Air India	0.271094	0.050432	-0.032490	-0.045981	-0.012879	-0.045688	0.088872	0.061231
Airline_GoAir	-0.060110	-0.095151	-0.003122	-0.004494	-0.016373	0.076751	0.018526	0.096839
Airline_IndiGo	-0.302991	-0.361070	0.007281	-0.048504	-0.023395	-0.014714	-0.071491	0.035124
Airline_Jet Airways	0.215063	0.416124	-0.017304	0.059735	0.113942	0.024455	-0.027377	-0.057698
Airline_Jet Airways Business	0.017876	0.253303	-0.031713	-0.034787	-0.007524	0.009168	-0.014456	0.005232
Airline_Multiple carriers	0.118399	0.139793	0.042163	0.053685	-0.149992	-0.109370	0.067930	-0.167455
Airline_Multiple carriers Premium economy	0.009089	0.017650	0.030839	-0.051222	-0.028672	-0.004624	0.013491	-0.001786
Airline_SpiceJet	-0.274351	-0.296565	0.022154	-0.011977	-0.010451	0.092634	-0.090058	0.012543
Airline_Trujet	0.002519	-0.010381	-0.008569	-0.014199	0.000857	-0.010007	0.003739	-0.002750
Airline_Vistara	-0.120447	-0.060654	-0.013169	-0.017252	0.023906	-0.077903	0.068834	0.069422
Airline_Vistara Premium economy	-0.020459	-0.000454	-0.020115	-0.019797	-0.003375	-0.011380	0.000776	0.000314
Source_Chennai	-0.234758	-0.179223	0.006611	0.005650	-0.014846	0.067110	-0.014795	-0.030493
Source_Delhi	0.490170	0.270676	0.100088	0.139222	-0.118780	-0.085534	-0.006790	-0.209882
Source_Kolkata	0.032761	0.009358	-0.060558	0.087177	0.155471	-0.024238	0.054693	0.118573
Source_Mumbai	-0.260752	-0.230755	-0.014030	-0.039352	-0.017292	0.037705	-0.033512	0.081196
Destination_Cochin	0.490170	0.270676	0.100088	0.139222	-0.118780	-0.085534	-0.006790	-0.209882
Destination_Delhi	-0.447390	-0.313417	0.002632	0.090490	0.009469	0.003200	-0.030867	0.095250
Destination_Hyderabad	-0.260752	-0.230755	-0.014030	-0.039352	-0.017292	0.037705	-0.033512	0.081196
Destination_Kolkata	-0.234758	-0.179223	0.006611	0.005650	-0.014846	0.067110	-0.014795	-0.030493
Destination_New Delhi	-0.015302	0.189777	-0.075254	-0.453685	-0.022138	0.107129	0.000366	0.021271

In [70]:

```
plt.figure(figsize = (30,30))
sns.heatmap(df.corr(), annot = True, cmap = "RdYlGn")
plt.show()
```



Using ExtraTreesRegressor

In [71]:

```
from sklearn.ensemble import ExtraTreesRegressor
selection = ExtraTreesRegressor()
selection.fit(X, y)
```

Out[71]:

```
ExtraTreesRegressor()
```

In [72]:

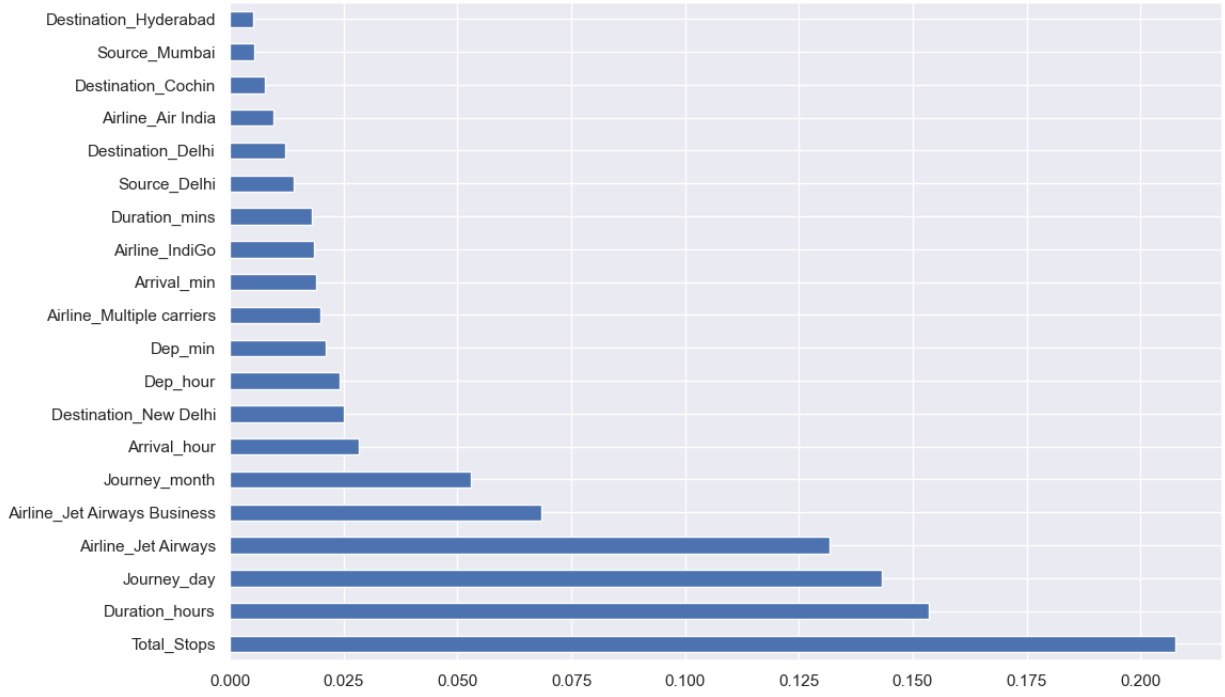
```
print(selection.feature_importances_)
```

```
[2.07674796e-01 1.43141929e-01 5.30034357e-02 2.41823151e-02
 2.10498328e-02 2.83856043e-02 1.89871735e-02 1.53459640e-01
 1.80287775e-02 9.45339797e-03 1.85246951e-03 1.84372044e-02
 1.31767711e-01 6.84898242e-02 1.99412177e-02 8.39515789e-04
 2.57296089e-03 7.93188753e-05 4.90124914e-03 8.82473110e-05
 4.16920037e-04 1.39635168e-02 3.44026615e-03 5.34352178e-03
 7.64653303e-03 1.21996778e-02 5.14345007e-03 5.56361960e-04
 2.49531321e-02]
```

visualizing the feature importances

In [73]:

```
plt.figure(figsize = (12,8))
feat_importances = pd.Series(selection.feature_importances_, index=X.columns)
feat_importances.nlargest(20).plot(kind='barh')
plt.show()
```



splitting out training and testing data

In [74]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

Applying Linear regression-ML

In [75]:

```
from sklearn.linear_model import LinearRegression
```

In [76]:

```
model_li = LinearRegression()
model_li.fit(X_train,y_train)
```

Out[76]:

```
LinearRegression()
```

In [77]:

```
model_li.score(X_train,y_train)
```

Out[77]:

```
0.6240840020468166
```

In [78]:

```
model_li.score(X_test,y_test)
```

Out[78]:

```
0.6195943729070101
```

Trying all diffrent regression algorithms, getting out the testing score

In [79]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
#from xgboost import XGBRFRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.svm import SVR
```

In [80]:

```
#model = [DecisionTreeRegressor,SVR,RandomForestRegressor,KNeighborsRegressor,AdaBoostRegressor,****XGBRFRegressor
model = [DecisionTreeRegressor,SVR,RandomForestRegressor,KNeighborsRegressor,AdaBoostRegressor]
for mod in model:
    reg = mod()
    reg = reg.fit(X_train,y_train)
    print(mod , 'accuracy',reg.score(X_test,y_test))
```

```
<class 'sklearn.tree._classes.DecisionTreeRegressor'> accuracy 0.7324262018350121
<class 'sklearn.svm._classes.SVR'> accuracy -0.00041646312498344606
<class 'sklearn.ensemble._forest.RandomForestRegressor'> accuracy 0.7958682034371062
<class 'sklearn.neighbors._regression.KNeighborsRegressor'> accuracy 0.5743709506218349
<class 'sklearn.ensemble._weight_boosting.AdaBoostRegressor'> accuracy 0.523404723489113
```

In [81]:

```
X_train.shape,X_test.shape
```

Out[81]:

```
((8545, 29), (2137, 29))
```

Now applying Kflod and cross validation technique

In [82]:

```
from sklearn.model_selection import KFold,cross_val_score
```

In [83]:

```
models = []
models.append(('KNN', KNeighborsRegressor()))
models.append(('CART', DecisionTreeRegressor()))
models.append(('RF', RandomForestRegressor()))
models.append(('SVM', SVR()))
models.append(('AdaBoost', AdaBoostRegressor()))
#models.append(('XGB', XGBRFRegressor()))


results = []
names = []
for name,model in models:
    kfold = KFold(n_splits=5)
    cv_result =cross_val_score(model,X_train,y_train,cv=kfold)
    names.append(name)
    results.append(cv_result)
for i in range(len(names)):
    print(names[i],results[i].mean())
```

```
KNN 0.5414479722960382
CART 0.6954960880642737
RF 0.8027036132570086
SVM -0.002338885960318793
AdaBoost 0.3704592892754233
```

Using RandomForest Regressor algorithm-ML

In [84]:

```
from sklearn.ensemble import RandomForestRegressor
reg_rf = RandomForestRegressor()
reg_rf.fit(X_train, y_train)
```

Out[84]:

```
RandomForestRegressor()
```

In [85]:

```
y_pred = reg_rf.predict(X_test)
```

In [86]:

```
reg_rf.score(X_train, y_train)
```

Out[86]:

```
0.9539990527465142
```

In [87]:

```
reg_rf.score(X_test, y_test)
```

Out[87]:

```
0.7965834544776972
```

Performing Hyper-prameter tuning using RandomizedSearchCV

In [88]:

```
from sklearn.model_selection import RandomizedSearchCV
```

create list for all possible parameter

In [89]:

```
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
min_samples_split = [2, 5, 10, 15, 100]
min_samples_leaf = [1, 2, 5, 10]
```

In [90]:

```
print("n esti" , n_estimators )
print("max_features" , max_features )
print("max_depth" , max_depth )
print("min_samples_split" , min_samples_split )
print("min_samples_leaf" , min_samples_leaf )
```

```
n esti [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200]
max_features ['auto', 'sqrt']
max_depth [5, 10, 15, 20, 25, 30]
min_samples_split [2, 5, 10, 15, 100]
min_samples_leaf [1, 2, 5, 10]
```

In [91]:

```
12*2*6*5*4
```

Out[91]:

```
2880
```

In [92]:

```
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}
```

Random search of parameters, using 5 fold cross validation and search across 100 different combinations

In [93]:

```
rf_random = RandomizedSearchCV(estimator = reg_rf,
                               param_distributions = random_grid,
                               scoring='neg_mean_squared_error',
                               n_iter = 10, cv = 5,
                               verbose=2,
                               random_state=42, n_jobs = 1)
```

In [94]:

```
rf_random.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=
900; total time=   5.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=
900; total time=   5.3s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=
900; total time=   5.4s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=
900; total time=   5.4s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=
900; total time=   5.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators
=1100; total time=   8.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators
=1100; total time=   8.5s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators
=1100; total time=   8.8s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators
=1100: total time=   8.8s
```

Best parameters list

In [95]:

```
rf_random.best_params_
```

Out[95]:

```
{'n_estimators': 700,
 'min_samples_split': 15,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 20}
```

In [96]:

```
from sklearn.ensemble import RandomForestRegressor
reg_rf = RandomForestRegressor(n_estimators= 700,
 min_samples_split= 15,
 min_samples_leaf= 1,
 max_features='auto',
 max_depth= 20)
reg_rf.fit(X_train, y_train)
```

Out[96]:

```
RandomForestRegressor(max_depth=20, min_samples_split=15, n_estimators=700)
```

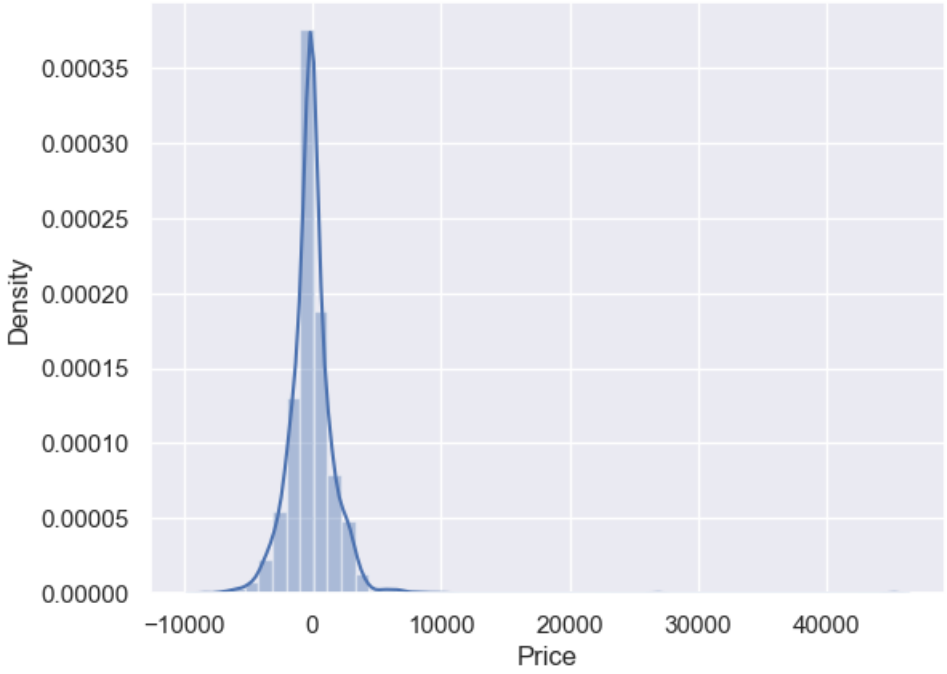
In [97]:

```
y_pred=reg_rf.predict(X_test)
```

comparing y_test and y_pred value using distplot

In [98]:

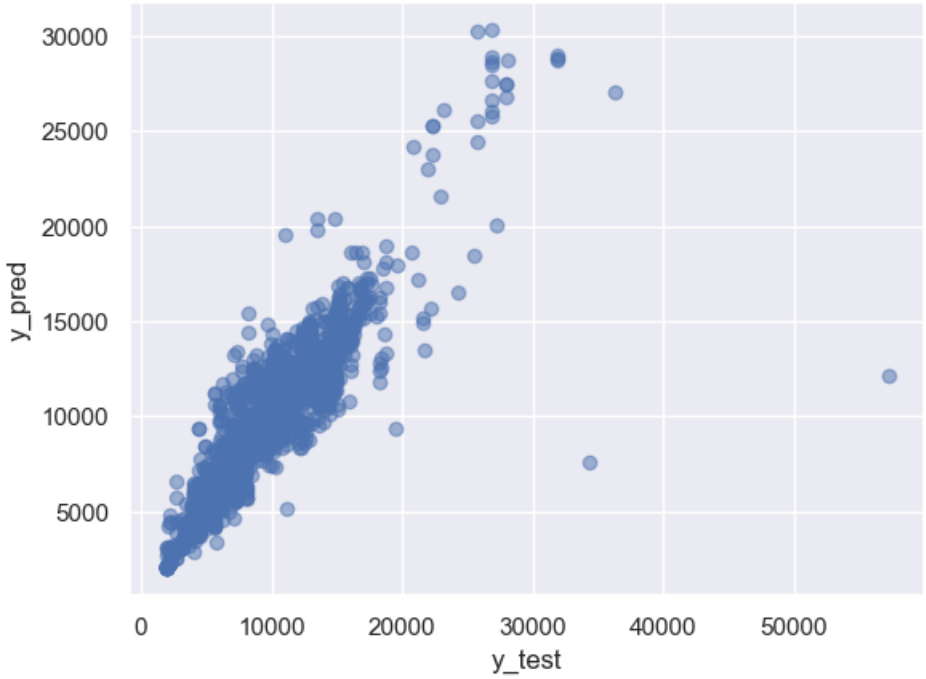
```
sns.distplot(y_test-y_pred)
plt.show()
```



plotting scatter plot

In [99]:

```
plt.scatter(y_test, y_pred, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```



Evaluating the model

In [100]:

```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

In [101]:

```
mean_absolute_error(y_test, y_pred)
```

Out[101]:

```
1163.4456744331417
```

In [102]:

```
mean_squared_error(y_test, y_pred)
```

Out[102]:

```
4039893.6339230947
```

In [103]:

```
r2_score(y_test, y_pred)
```

Out[103]:

```
0.8126387588833022
```

In [104]:

```
y_pred_train=reg_rf.predict(X_train)
```

In [105]:

```
r2_score(y_train,y_pred_train)
```

Out[105]:

```
0.8958131221128793
```