Project Name:- Flight Price Prediction Project using Machine Learning.

Importing Libraries

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

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score
from math import sqrt
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV

from prettytable import PrettyTable
```

Loading Dataset

In [1]:
```
import pandas as pd
df=pd.read_csv("flightprice.csv")
df.head()
```

Out[1]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302

Exploratory Data Analysis (EDA)

Now here we will be looking at the kind of columns our dataset has.

In [8]:
```
df.columns
```

Out[8]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price'],
      dtype='object')
```

Here we can get more information about our dataset

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10682 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          10682 non-null  object
 1   Date_of_Journey  10682 non-null  object
 2   Source           10682 non-null  object
 3   Destination      10682 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10682 non-null  object
 6   Arrival_Time     10682 non-null  object
 7   Duration         10682 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10682 non-null  object
 10  Price            10682 non-null  int64
dtypes: int64(1), object(10)
memory usage: 1001.4+ KB
```

To know more about the dataset

In [10]:
```
df.describe()
```

Out[10]:

	Price
count	10682.000000
mean	9087.214567
std	4611.548810
min	1759.000000
25%	5277.000000
50%	8372.000000
75%	12373.000000
max	79512.000000

Now while using the IsNull function we will gonna see the number of null values in our dataset

In [11]:
```
df.isnull().head()
```

Out[11]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False

Now while using the IsNull function and sum function we will gonna see the number of null values in our dataset

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
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 [13]:
```
df.dropna(inplace = True)
```

Duplicate values

In [14]:
```
df[df.duplicated()].head()
```

Out[14]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
683	Jet Airways	1/06/2019	Delhi	Cochin	DEL ? NAG ? BOM ? COK	14:35	04:25 02 Jun	13h 50m	2 stops	No info	13376
1061	Air India	21/05/2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	22:00	19:15 22 May	21h 15m	2 stops	No info	10231
1348	Air India	18/05/2019	Delhi	Cochin	DEL ? HYD ? BOM ? COK	17:15	19:15 19 May	26h	2 stops	No info	12392
1418	Jet Airways	6/06/2019	Delhi	Cochin	DEL ? JAI ? BOM ? COK	05:30	04:25 07 Jun	22h 55m	2 stops	In-flight meal not included	10368
1674	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	18:25	21:20	2h 55m	non-stop	No info	7303

Here we will be removing those repeated values from the dataset and keeping the in-place attribute to be true so that there will be no changes.

In [16]:
```
df.drop_duplicates(keep='first',inplace=True)
df.head()
```

Out[16]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302

In [17]:
```
df.shape
```

Out[17]:
```
(10462, 11)
```

Checking the Additional_info column and having the count of unique types of values.

In [19]:
```
df["Additional_Info"].value_counts()
```

Out[19]:
```
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
2 Long layover                     1
1 Short layover                    1
Red-eye flight                     1
Name: Additional_Info, dtype: int64
```

Checking the different Airlines

In [20]:
```
df["Airline"].unique()
```

Out[20]:
```
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)
```

Checking the different Airline Routes

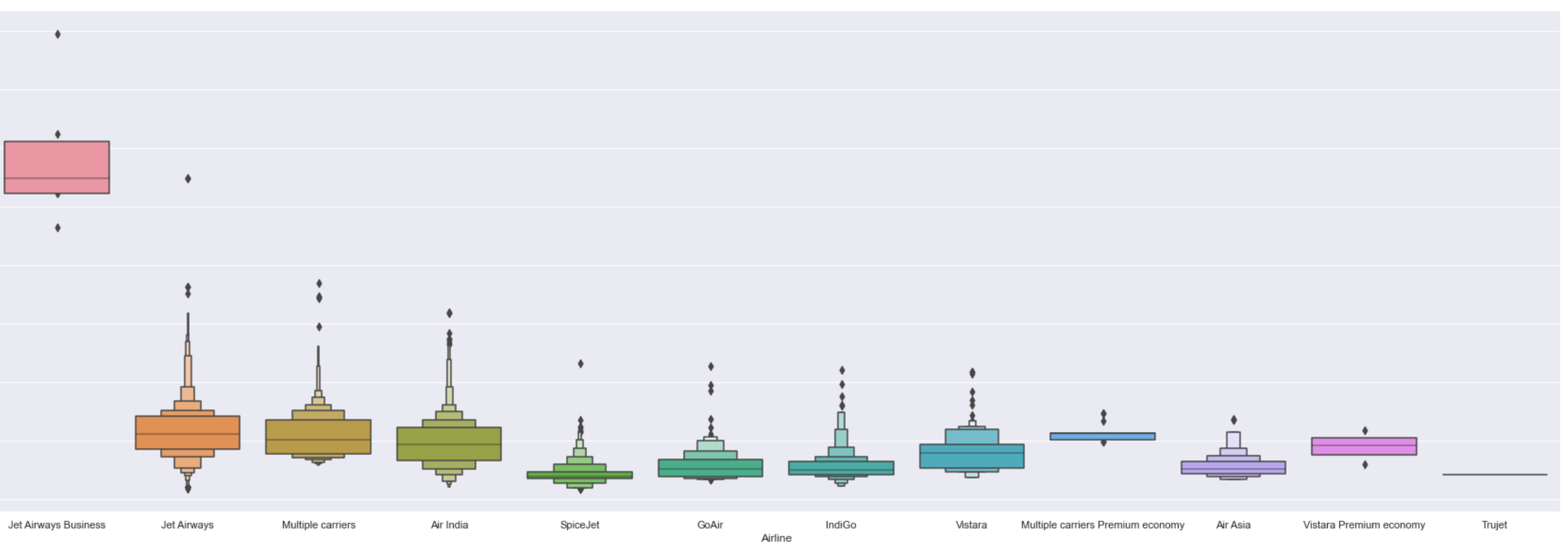
In [21]:
```
df["Route"].unique()
```

Out[21]:
```
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)
```

Data Visualization

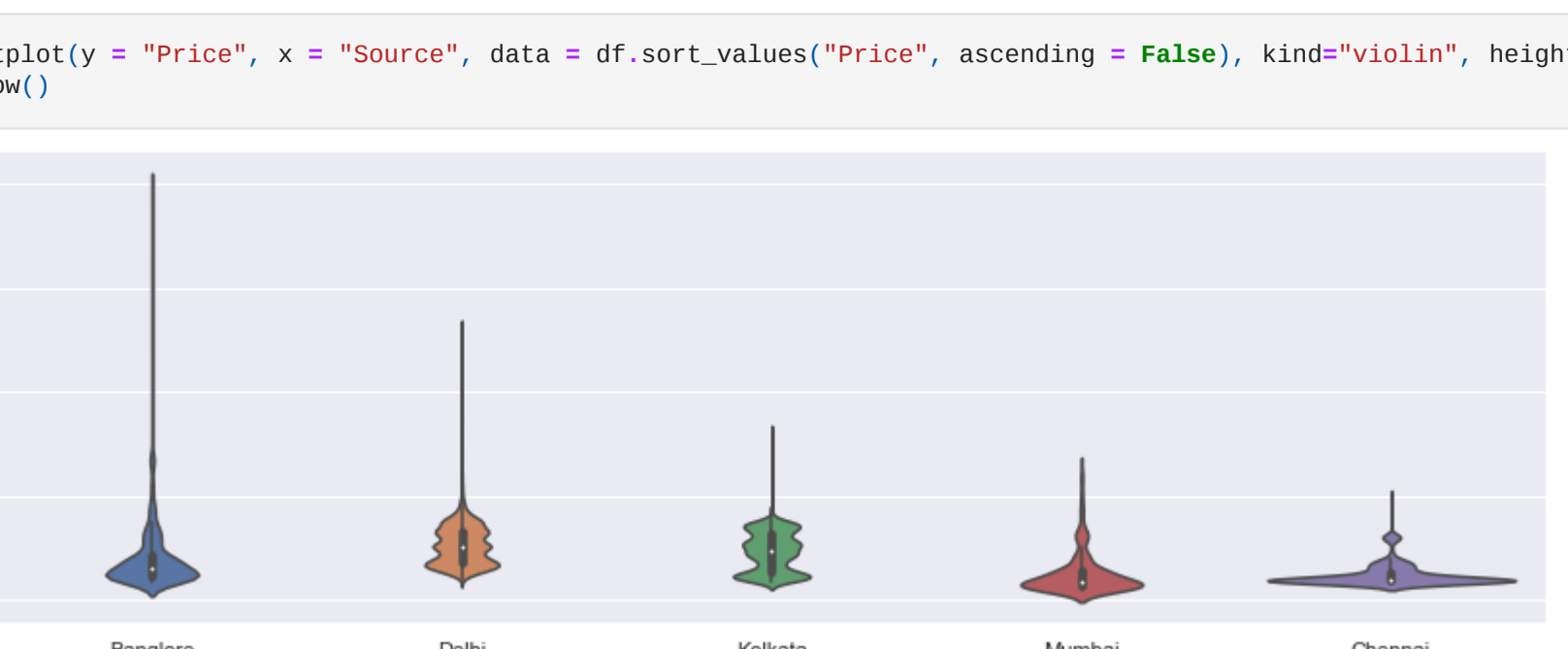
Plotting Price vs Airline plot

In [22]:
```
sns.catplot(y = "Price", x = "Airline", data = df.sort_values("Price", ascending = False), kind="boxen", height = 8, aspect = 3)
plt.show()
```



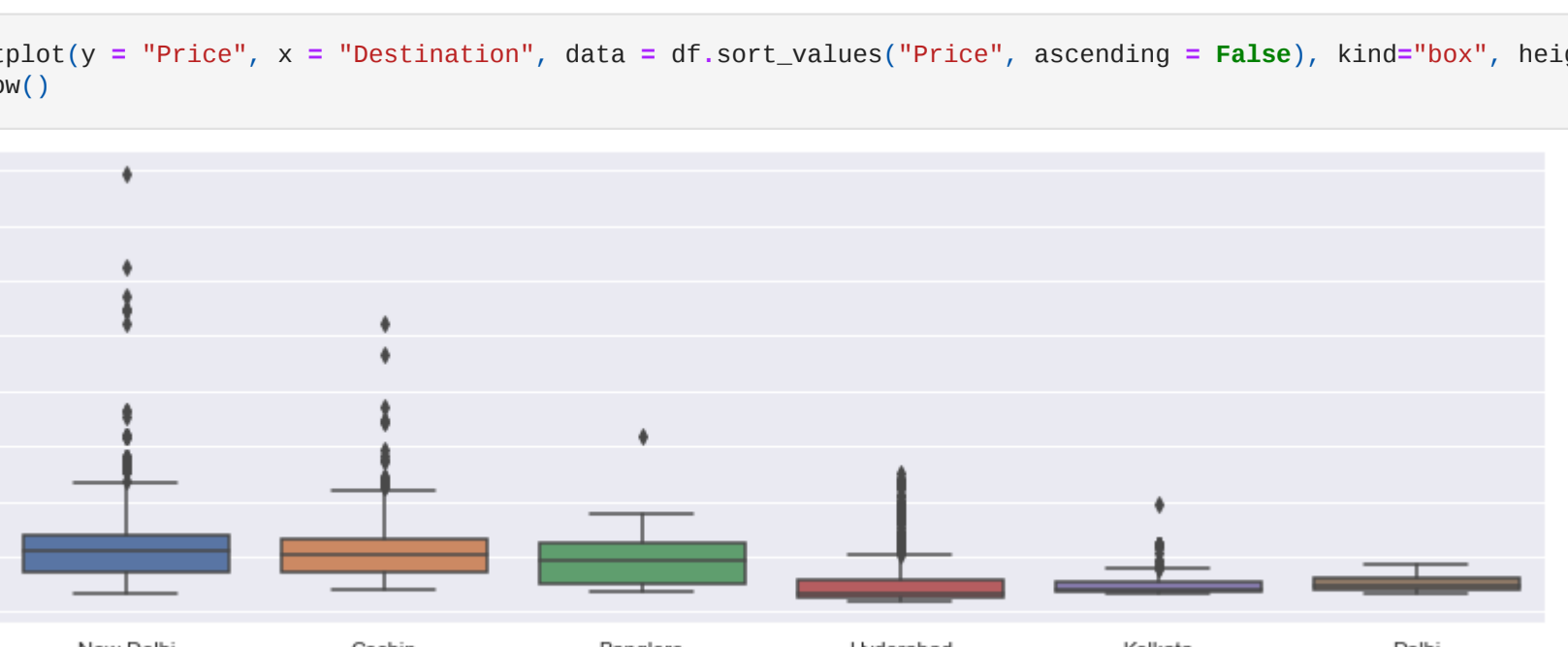
Plotting Violin plot for Price vs Source

In [23]:
```
sns.catplot(y = "Price", x = "Source", data = df.sort_values("Price", ascending = False), kind="violin", height = 4, aspect = 3)
plt.show()
```



Plotting Box plot for Price vs Destination

In [24]:
```
sns.catplot(y = "Price", x = "Destination", data = df.sort_values("Price", ascending = False), kind="box", height = 4, aspect = 3)
plt.show()
```



Let's see our processed data first

In [25]:
```
df.head()
```

Out[25]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302

Here first we are dividing the features and labels and then converting the hours in minutes.

In [ ]:
```
df['Duration'] = df['Duration'].str.replace("h", '*60').str.replace(' ','+').str.replace('m','*1').apply(eval)
df['Duration'] = df['Duration'].str.replace("h", '*60').str.replace(' ','+').str.replace('m','*1').apply(eval)
```

Date_of_Journey: Here we are organizing the format of the date of journey in our dataset for better preprocessing in the model stage.

In [ ]:
```
df["Journey_day"] = df['Date_of_Journey'].str.split('/').str[0].astype(int)
df["Journey_month"] = df['Date_of_Journey'].str.split('/').str[1].astype(int)
df.drop(["Date_of_Journey"], axis = 1, inplace = True)
```

Dep_Time: Here we are converting departure time into hours and minutes

In [ ]:
```
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)
```

Arrival_Time: Similarly we are converting the arrival time into hours and minutes.

In [ ]:
```
train_df["Arrival_hour"] = pd.to_datetime(train_df.Arrival_Time).dt.hour
train_df["Arrival_min"] = pd.to_datetime(train_df.Arrival_Time).dt.minute
train_df.drop(["Arrival_Time"], axis = 1, inplace = True)
```

Now after final preprocessing let's see our dataset

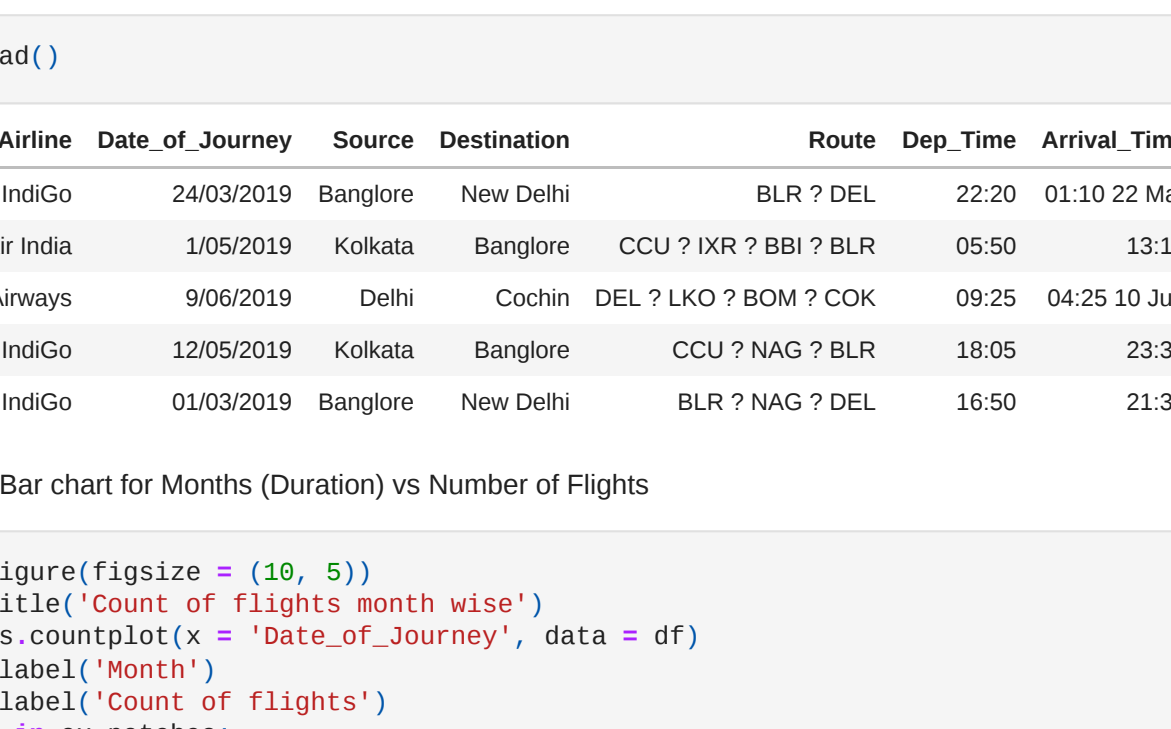
In [26]:
```
df.head()
```

Out[26]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302

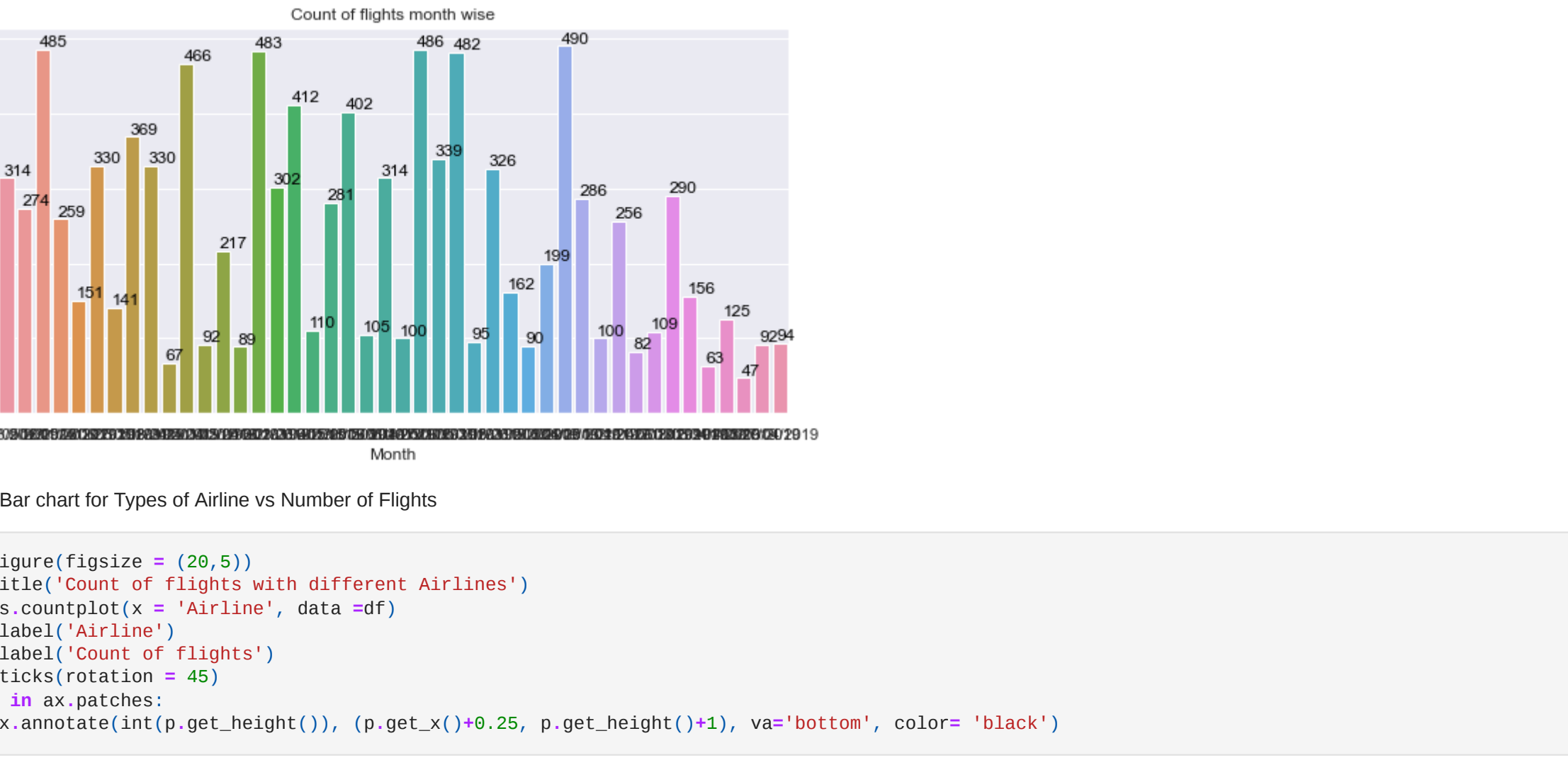
Plotting Bar chart for Months (Duration) vs Number of Flights

In [28]:
```
plt.figure(figsize = (10, 5))
plt.title('Count of flights month wise')
ax=sns.countplot(x = 'Date_of_Journey', data = df)
plt.xlabel('Month')
plt.ylabel('Count of flights')
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va='bottom', color= 'black')
```



Plotting Bar chart for Types of Airline vs Number of Flights

In [29]:
```
plt.figure(figsize = (20,5))
plt.title('Count of flights with different Airlines')
ax=sns.countplot(x = 'Airline', data =df)
plt.xlabel('Airline')
plt.ylabel('Count of flights')
plt.xticks(rotation = 45)
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va='bottom', color= 'black')
```



Plotting Ticket Prices VS Airlines

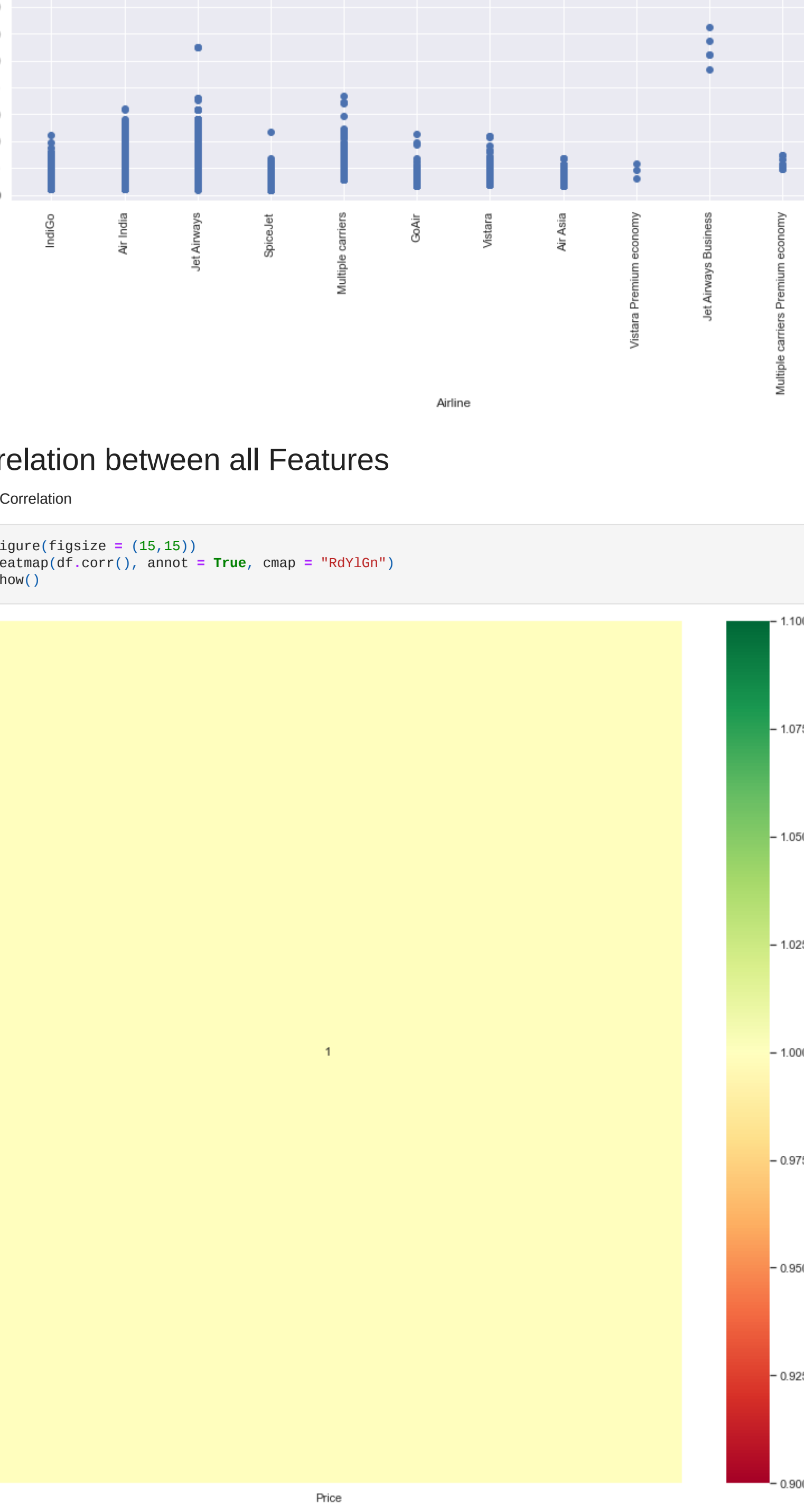
In [30]:
```
plt.figure(figsize = (15,4))
plt.title('Price VS Airlines')
plt.scatter(df['Airline'], df['Price'])
plt.xticks
plt.xlabel('Airline')
plt.ylabel('Price of ticket')
plt.xticks(rotation = 90)
```

Out[30]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```

Correlation between all Features

Plotting Correlation

In [31]:
```
plt.figure(figsize = (15,15))
sns.heatmap(df.corr(), annot = True, cmap = "RdYlGn")
plt.show()
```



Dropping the Price column as it is of no use

In [ ]:
```
data = df.drop(["Price"], axis=1)
```

In [35]:
```
train_categorical_data = df.select_dtypes(exclude=['int64', 'float','int32'])
train_numerical_data = df.select_dtypes(include=['int64', 'float','int32'])

test_categorical_data = df.select_dtypes(exclude=['int64', 'float','int32','int32'])
test_numerical_data  = df.select_dtypes(include=['int64', 'float','int32'])
train_categorical_data.head()
```

Out[35]:

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

abel Encode and Hot Encode for Categorical Columns

In [37]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
train_categorical_data = train_categorical_data.apply(LabelEncoder().fit_transform)
test_categorical_data = test_categorical_data.apply(LabelEncoder().fit_transform)
train_categorical_data.head()
```

Out[37]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
0	3	24	0	5	18	211	233	240	4	8
1	1	6	3	0	84	31	906	336	1	8
2	4	43	2	1	118	70	413	106	1	8
3	3	10	3	0	91	164	1324	311	0	8
4	3	0	0	5	29	149	1237	303	0	8

Concatenating both Categorical Data and Numerical Data

In [38]:
```
X = pd.concat([train_categorical_data, train_numerical_data], axis=1)
y = df['Price']
test_set = pd.concat([test_categorical_data, test_numerical_data], axis=1)
X.head()
```

Out[38]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	3	24	0	5	18	211	233	240	4	8	3897
1	1	6	3	0	84	31	906	336	1	8	7662
2	4	43	2	1	118	70	413	106	1	8	13882
3	3	10	3	0	91	164	1324	311	0	8	6218
4	3	0	0	5	29	149	1237	303	0	8	13302

In [39]:
```
y.head()
```

Out[39]:
```
0     3897
1     7662
2    13882
3     6218
4    13302
Name: Price, dtype: int64
```

Conclusion

So as we saw that we have done a complete EDA process, getting data insights, feature engineering, and data visualization as well so after all these steps one can go for the prediction using machine learning model-making steps.

I hope this Project helped you to understand Data Analysis, Data Preparation, and Model building approaches in a much simpler way.

THANK YOU

In [ ]: