By: Nitish Adhikari

Contact : nitishbuzzpro@gmail.com (mailto:nitishbuzzpro@gmail.com)

Linkdin: https://www.linkedin.com/in/nitish-adhikari-6b2350248 (https://www.linkedin.com/in/nitish-adhikari-6b2350248)

Project: Predict Fare of the Airline

In [2]:

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

In [3]:

```
train_data = pd.read_excel('Data_Train.xlsx')
```

In [4]:

```
train_data.head()
```

Out[4]:

	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 [5]:

```
train_data.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 [6]:

```
train_data.isnull().sum()
```

Out[6]:

```
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 [7]:

```
train_data.shape
```

Out[7]:

```
(10683, 11)
```

In [8]:

```
train_data[train_data['Total_Stops'].isnull()]
```

Out[8]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
9039	Air India	6/05/2019	Delhi	Cochin	NaN	09:45	09:25 07 May	23h 40m	NaN	No info	7480

In [9]:

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

In [10]:

```
train_data.isnull().sum()
```

Out[10]:

```
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 [11]:

```
data=train_data.copy()
```

In [12]:

```
data.head(2)
```

Out[12]:

	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

In [13]:

```
data.dtypes
```

Out[13]:

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

In [14]:

```
def change_into_datetime(col):
    data[col]= pd.to_datetime(data[col])
```

In [15]:

```
data.columns
```

Out[15]:

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

In [16]:

```
for feature in ['Date_of_Journey', 'Dep_Time', 'Arrival_Time']:
    change_into_datetime(feature)
```

```
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '24/03/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '24/06/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '27/05/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '18/04/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '24/04/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '15/04/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '21/03/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '15/05/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '18/06/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '15/06/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '18/05/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '27/06/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '21/05/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '15/03/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '24/05/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '21/04/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '21/06/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '27/03/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '18/03/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '27/04/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
```

In [17]:

```
data.dtypes
```

Out[17]:

```
Airline                    object
Date_of_Journey    datetime64[ns]
Source                     object
Destination                object
Route                      object
Dep_Time           datetime64[ns]
Arrival_Time       datetime64[ns]
Duration                   object
Total_Stops                object
Additional_Info            object
Price                       int64
dtype: object
```

In [18]:

```
data['Date_of_Journey'].min()
```

Out[18]:

```
Timestamp('2019-01-03 00:00:00')
```

In [19]:

```
data['Date_of_Journey'].max()
```

Out[19]:

```
Timestamp('2019-12-06 00:00:00')
```

In [20]:

```
data['journey_day']= data['Date_of_Journey'].dt.day
```

In [21]:

```
data['journey_month']= data['Date_of_Journey'].dt.month
```

In [22]:

```
data['journey_year']= data['Date_of_Journey'].dt.year
```

In [23]:

```
data.head(2)
```

Out[23]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_m
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2022-12-23 22:20:00	2022-03-22 01:10:00	2h 50m	non-stop	No info	3897	24	
1	Air India	2019-01-05	Kolkata	Banglore	CCU → IXR → BBI → BLR	2022-12-23 05:50:00	2022-12-23 13:15:00	7h 25m	2 stops	No info	7662	5	

In [24]:

```
data.drop('Date_of_Journey',axis=1, inplace=True)
```

In [25]:

```
data.head()
```

Out[25]:

	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	journey_ye
0	IndiGo	Banglore	New Delhi	BLR → DEL	2022-12-23 22:20:00	2022-03-22 01:10:00	2h 50m	non-stop	No info	3897	24	3	20
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2022-12-23 05:50:00	2022-12-23 13:15:00	7h 25m	2 stops	No info	7662	5	1	20
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2022-12-23 09:25:00	2022-06-10 04:25:00	19h	2 stops	No info	13882	6	9	20
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	2022-12-23 18:05:00	2022-12-23 23:30:00	5h 25m	1 stop	No info	6218	5	12	20
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	2022-12-23 16:50:00	2022-12-23 21:35:00	4h 45m	1 stop	No info	13302	3	1	20

In [26]:

```
def extract_hour_min(df,col):
    df[col+'_hour']=df[col].dt.hour
    df[col+'_min']=df[col].dt.minute
    df.drop(col,axis=1,inplace=True)
    return df.head(2)
```

In [27]:

```
extract_hour_min(data,'Dep_Time')
```

Out[27]:

	Airline	Source	Destination	Route	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	journey_year	Dep_Time
0	IndiGo	Banglore	New Delhi	BLR → DEL	2022-03-22 01:10:00	2h 50m	non-stop	No info	3897	24	3	2019	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2022-12-23 13:15:00	7h 25m	2 stops	No info	7662	5	1	2019	

In [28]:

```
extract_hour_min(data,'Arrival_Time')
```

Out[28]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	journey_year	Dep_Time_hour	Dep_T
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	2019	22	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	5	1	2019	5	

Perform Data Analysis

In [29]:

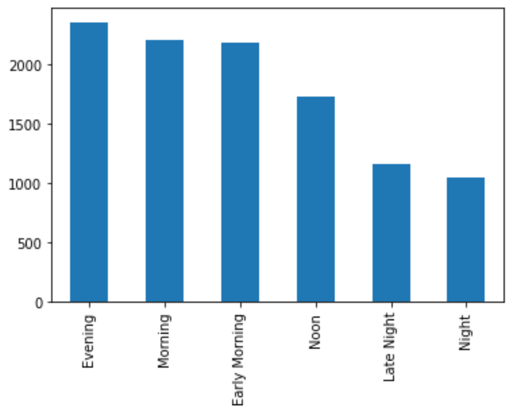
```
def flight_dep_time(x):

    if (x>4) and (x<8):
        return 'Early Morning'

    elif (x>8) and (x<=12):
        return 'Morning'

    elif (x>12) and (x<=16):
        return 'Noon'

    elif (x>16) and (x<=20):
        return 'Evening'

    elif (x>20) and (x<=24):
        return 'Night'

    else:
        return 'Late Night'
```

In [30]:

```
data['Dep_Time_hour'].apply(flight_dep_time).value_counts().plot(kind='bar')
```

Out[30]:

```
<AxesSubplot:>
```



In [31]:

```
data.head(10)
```

Out[31]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	journey_year	Dep_Time_hour	Dep
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	2019	22	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	5	1	2019	5	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	6	9	2019	9	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	5	12	2019	18	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	3	1	2019	16	
5	SpiceJet	Kolkata	Banglore	CCU → BLR	2h 25m	non-stop	No info	3873	24	6	2019	9	
6	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	15h 30m	1 stop	In-flight meal not included	11087	3	12	2019	18	
7	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	21h 5m	1 stop	No info	22270	3	1	2019	8	
8	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	25h 30m	1 stop	In-flight meal not included	11087	3	12	2019	8	
9	Multiple carriers	Delhi	Cochin	DEL → BOM → COK	7h 50m	1 stop	No info	8625	27	5	2019	11	

In [32]:

```
def preprocess_duration(x):
    if 'h' not in x:
        x='0h '+x
    elif 'm' not in x:
        x=x+ ' 0m'
    return x
```

In [33]:

```
data['Duration']=data['Duration'].apply(preprocess_duration)
data['Duration']
```

Out[33]:

```
0         2h 50m
1         7h 25m
2         19h 0m
3         5h 25m
4         4h 45m
           ...
10678     2h 30m
10679     2h 35m
10680      3h 0m
10681     2h 40m
10682     8h 20m
Name: Duration, Length: 10682, dtype: object
```

In [34]:

```
data['Duration'][0].split(' ')
```

Out[34]:

```
['2h', '50m']
```

In [35]:

```
int(data['Duration'][0].split(' ')[0][0:-1])
```

Out[35]:

```
2
```

In [36]:

```
int(data['Duration'][0].split(' ')[1][0:-1])
```

Out[36]:

```
50
```

In [37]:

```
data['Duration_hours']=data['Duration'].apply(lambda x:int(x.split(' ')[0][0:-1]))
```

In [38]:

```
data['Duration_mins']=data['Duration'].apply(lambda x:int(x.split(' ')[1][0:-1]))
```

In [39]:

```
data.head(3)
```

Out[39]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	journey_year	Dep_Time_hour	Dep_
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	2019	22	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	5	1	2019	5	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h 0m	2 stops	No info	13882	6	9	2019	9	

1. Analyse wether Duration impacts on price or not

2. Which city has maximum final destination of flights

In [40]:

```
data['Duration_total_mins']=data['Duration'].str.replace('h','*60').str.replace(' ','+').str.replace('m','*1').apply(eval)
```

In [41]:

```
data.head(2)
```

Out[41]:

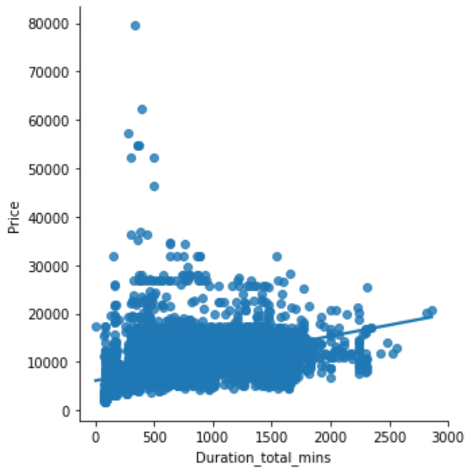
	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	journey_year	Dep_Time_hour	Dep_T
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	2019	22	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	5	1	2019	5	

In [42]:

```
sns.lmplot(x='Duration_total_mins', y='Price',data=data)
```

Out[42]:

```
<seaborn.axisgrid.FacetGrid at 0x1839722eb60>
```



In [43]:

```
data['Destination'].unique()
```

Out[43]:

```
array(['New Delhi', 'Banglore', 'Cochin', 'Kolkata', 'Delhi', 'Hyderabad'],
      dtype=object)
```

In [44]:

```
data['Destination'].value_counts()
```

Out[44]:

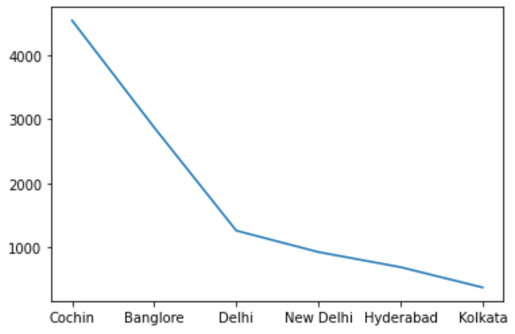
```
Cochin       4536
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: Destination, dtype: int64
```

In [45]:

```
data['Destination'].value_counts().plot()
```

Out[45]:

```
<AxesSubplot:>
```

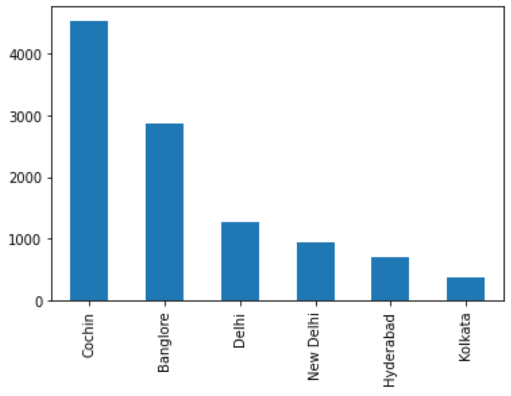


In [46]:

```
data['Destination'].value_counts().plot(kind='bar')
```

Out[46]:

```
<AxesSubplot:>
```

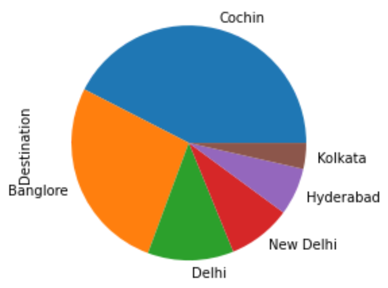


In [47]:

```
data['Destination'].value_counts().plot(kind='pie')
```

Out[47]:

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



Problem Statement : On which route jet Airways is extremely used?

In [48]:

```
data['Route']
```

Out[48]:

```
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 [49]:

```
data['Airline']
```

Out[49]:

```
0             IndiGo
1          Air India
2        Jet Airways
3             IndiGo
4             IndiGo
             ...
10678       Air Asia
10679      Air India
10680    Jet Airways
10681        Vistara
10682      Air India
Name: Airline, Length: 10682, dtype: object
```

In [50]:

```
data['Airline']=='Jet Airways'
```

Out[50]:

```
0        False
1        False
2         True
3        False
4        False
         ...
10678    False
10679    False
10680     True
10681    False
10682    False
Name: Airline, Length: 10682, dtype: bool
```

In [51]:

```
data[data['Airline']=='Jet Airways']
```

Out[51]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	journey_year	Dep_Time_hour	Dep_Time_min
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h 0m	2 stops	No info	13882	6	9	2019	9	25
6	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	15h 30m	1 stop	In-flight meal not included	11087	3	12	2019	18	55
7	Jet Airways	Banglore	New Delhi	BLR → BOM	21h 5m	1 stop	No info	22270	3	1	2019	8	0

In [52]:

```
data[data['Airline']=='Jet Airways'].groupby('Route').size().sort_values(ascending=False)
```

Out[52]:

```
Route
CCU → BOM → BLR          930
DEL → BOM → COK          875
BLR → BOM → DEL          385
BLR → DEL                382
CCU → DEL → BLR          300
BOM → HYD                207
DEL → JAI → BOM → COK    207
DEL → AMD → BOM → COK    141
DEL → IDR → BOM → COK     86
DEL → NAG → BOM → COK     61
DEL → ATQ → BOM → COK     38
DEL → COK                 34
DEL → BHO → BOM → COK     29
DEL → BDQ → BOM → COK     28
DEL → LKO → BOM → COK     25
DEL → JDH → BOM → COK     23
CCU → GAU → BLR           22
DEL → MAA → BOM → COK     16
DEL → IXC → BOM → COK     13
BLR → MAA → DEL           10
BLR → BDQ → DEL            8
DEL → UDR → BOM → COK      7
BOM → DEL → HYD            5
CCU → BOM → PNQ → BLR      4
BLR → BOM → JDH → DEL      3
DEL → DED → BOM → COK      2
BOM → BDQ → DEL → HYD      2
DEL → CCU → BOM → COK      1
BOM → VNS → DEL → HYD      1
BOM → UDR → DEL → HYD      1
BOM → JDH → DEL → HYD      1
BOM → IDR → DEL → HYD      1
BOM → DED → DEL → HYD      1
dtype: int64
```

Problem Statements:

1. On which route Jet Airways is extremely used?
2. Airline vs Price Analysis?

Box Plot:

Q1,Q2,Q3,Q4

IQR : Inter Quartile Range = Q3 - Q1

Max = Q3 + 1.5IQR

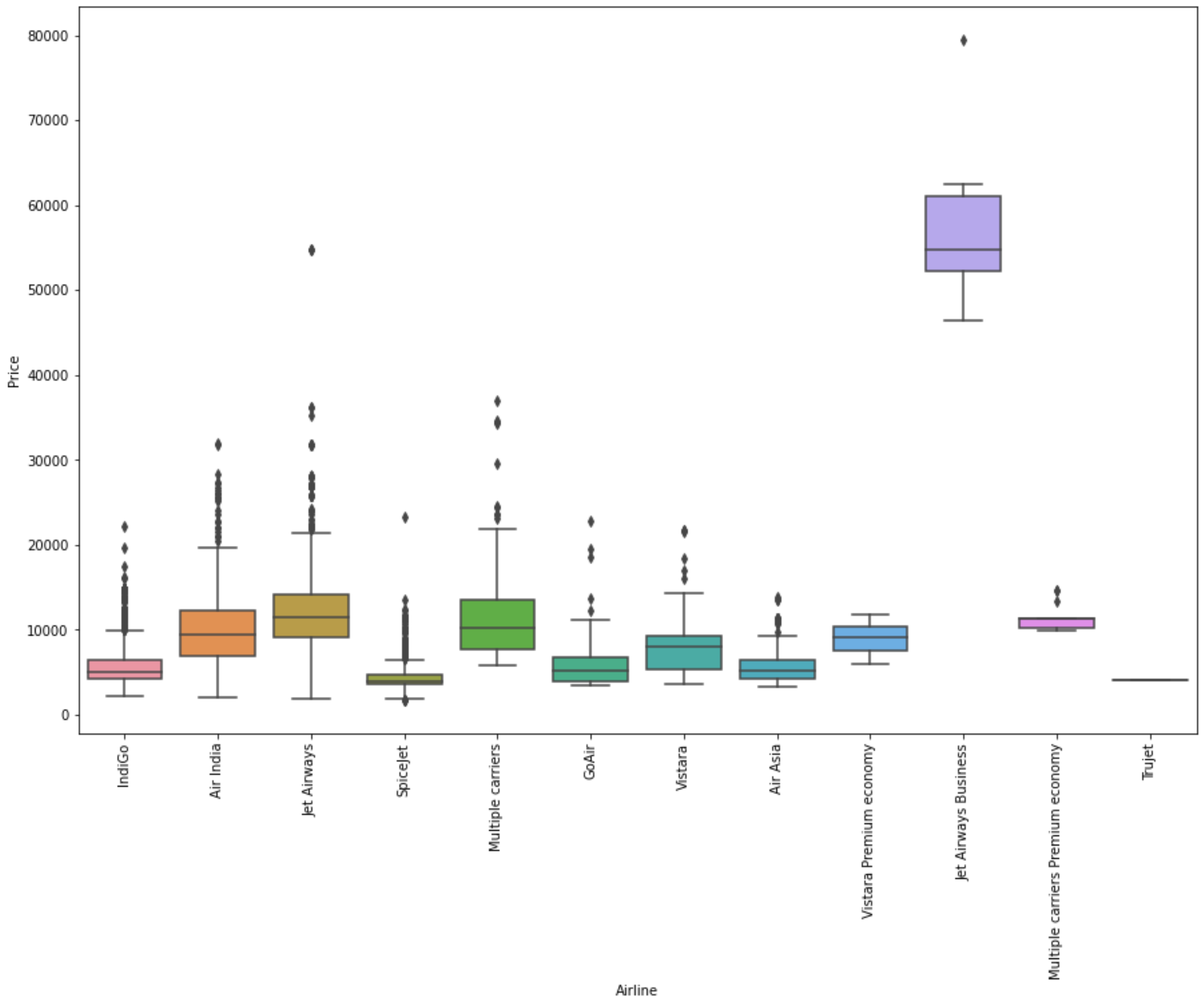
Min = Q3 - 1.5IQR

In [53]:

```
plt.figure(figsize=(15,10))
sns.boxplot(y='Price', x='Airline',data=data)
plt.xticks(rotation='vertical')
```

Out[53]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
 [Text(0, 0, 'IndiGo'),
  Text(1, 0, 'Air India'),
  Text(2, 0, 'Jet Airways'),
  Text(3, 0, 'SpiceJet'),
  Text(4, 0, 'Multiple carriers'),
  Text(5, 0, 'GoAir'),
  Text(6, 0, 'Vistara'),
  Text(7, 0, 'Air Asia'),
  Text(8, 0, 'Vistara Premium economy'),
  Text(9, 0, 'Jet Airways Business'),
  Text(10, 0, 'Multiple carriers Premium economy'),
  Text(11, 0, 'Trujet')])
```

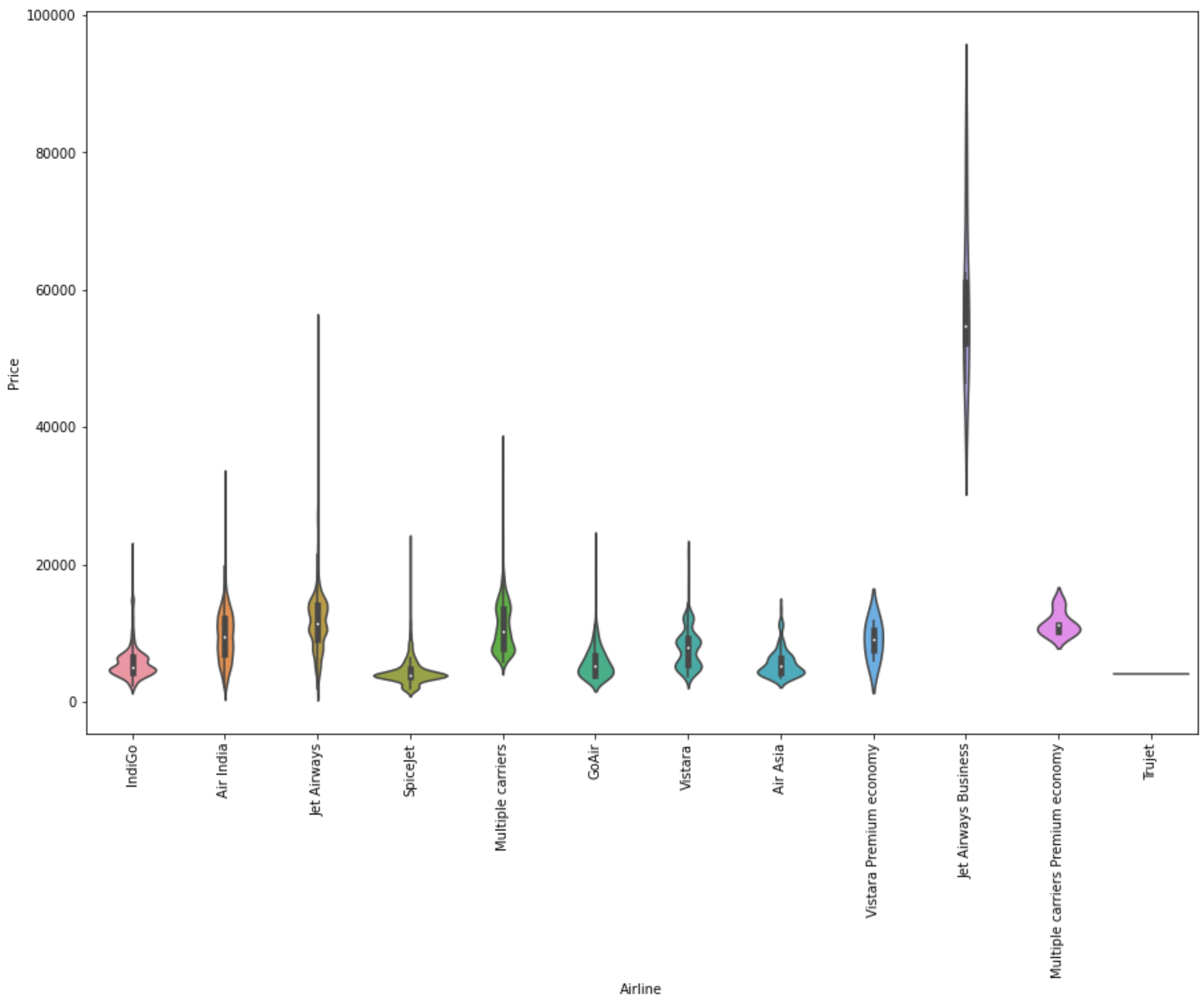


In [54]:

```
plt.figure(figsize=(15,10))
sns.violinplot(y='Price', x='Airline', data=data)
plt.xticks(rotation='vertical')
```

Out[54]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
 [Text(0, 0, 'IndiGo'),
  Text(1, 0, 'Air India'),
  Text(2, 0, 'Jet Airways'),
  Text(3, 0, 'SpiceJet'),
  Text(4, 0, 'Multiple carriers'),
  Text(5, 0, 'GoAir'),
  Text(6, 0, 'Vistara'),
  Text(7, 0, 'Air Asia'),
  Text(8, 0, 'Vistara Premium economy'),
  Text(9, 0, 'Jet Airways Business'),
  Text(10, 0, 'Multiple carriers Premium economy'),
  Text(11, 0, 'Trujet')])
```



In [ ]:

```

```

Apply one hot Encoding on data(feature Encoding)

In [55]:

```
data.head(2)
```

Out[55]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	journey_year	Dep_Time_hour	Dep_T
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	2019	22	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	5	1	2019	5	

In [56]:

```
np.round(data['Additional_Info'].value_counts()/len(data)*100)
```

Out[56]:

```
No info                         78.0
In-flight meal not included     19.0
No check-in baggage included     3.0
1 Long layover                   0.0
Change airports                  0.0
Business class                   0.0
No Info                          0.0
1 Short layover                  0.0
Red-eye flight                   0.0
2 Long layover                   0.0
Name: Additional_Info, dtype: float64
```

In [57]:

```
cat_col = [col for col in data.columns if data[col].dtype=='object']
cat_col
```

Out[57]:

```
['Airline',
 'Source',
 'Destination',
 'Route',
 'Duration',
 'Total_Stops',
 'Additional_Info']
```

In [58]:

```
num_col = [col for col in data.columns if data[col].dtype!='object']
num_col
```

Out[58]:

```
['Price',
 'journey_day',
 'journey_month',
 'journey_year',
 'Dep_Time_hour',
 'Dep_Time_min',
 'Arrival_Time_hour',
 'Arrival_Time_min',
 'Duration_hours',
 'Duration_mins',
 'Duration_total_mins']
```

In [59]:

```
data['Source']
```

Out[59]:

```
0         Banglore
1          Kolkata
2            Delhi
3          Kolkata
4         Banglore
           ...
10678      Kolkata
10679      Kolkata
10680     Banglore
10681     Banglore
10682        Delhi
Name: Source, Length: 10682, dtype: object
```

In [60]:

```
data['Source'].unique()
```

Out[60]:

```
array(['Banglore', 'Kolkata', 'Delhi', 'Chennai', 'Mumbai'], dtype=object)
```

In [61]:

```
data['Source'].apply(lambda x: 1 if x=='Bangalore' else 0)
```

Out[61]:

```
0        0
1        0
2        0
3        0
4        0
        ..
10678    0
10679    0
10680    0
10681    0
10682    0
Name: Source, Length: 10682, dtype: int64
```

In [62]:

```
for category in data['Source'].unique():
    data['Source_'+category]=data['Source'].apply(lambda x: 1 if x==category else 0)
```

In [63]:

```
data.head(5)
```

Out[63]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	...	Arrival_Time_hour	Arrival_Time_
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	...	1	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	5	1	...	13	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h 0m	2 stops	No info	13882	6	9	...	4	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	5	12	...	23	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	3	1	...	21	

5 rows × 23 columns

PERFORM TARGET GUIDED ENCODING ON DATA PERFORM MANUAL ENCODING ON DATA

In [64]:

```
airlines = data.groupby(['Airline'])['Price'].mean().sort_values().index
airlines
```

Out[64]:

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

In [65]:

```
dict1 = {key:index for index, key in enumerate(airlines,0)}
dict1
```

Out[65]:

```
{'Trujet': 0,
 'SpiceJet': 1,
 'Air Asia': 2,
 'IndiGo': 3,
 'GoAir': 4,
 'Vistara': 5,
 'Vistara Premium economy': 6,
 'Air India': 7,
 'Multiple carriers': 8,
 'Multiple carriers Premium economy': 9,
 'Jet Airways': 10,
 'Jet Airways Business': 11}
```

In [66]:

```
data['Airline']=data['Airline'].map(dict1)
data['Airline']
```

Out[66]:

```
0         3
1         7
2        10
3         3
4         3
         ..
10678     2
10679     7
10680    10
10681     5
10682     7
Name: Airline, Length: 10682, dtype: int64
```

In [67]:

```
data.head(2)
```

Out[67]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	...	Arrival_Time_hour	Arrival_Time_n
0	3	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	...	1	
1	7	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	5	1	...	13	

2 rows × 23 columns

In [68]:

```
data['Destination'].unique()
```

Out[68]:

```
array(['New Delhi', 'Banglore', 'Cochin', 'Kolkata', 'Delhi', 'Hyderabad'],
      dtype=object)
```

In [69]:

```
data['Destination'].replace('New Delhi','Delhi', inplace=True)
```

In [70]:

```
data['Destination'].unique()
```

Out[70]:

```
array(['Delhi', 'Banglore', 'Cochin', 'Kolkata', 'Hyderabad'],
      dtype=object)
```

In [71]:

```
dest=data.groupby(['Destination'])['Price'].mean().sort_values().index
dest
```

Out[71]:

```
Index(['Kolkata', 'Hyderabad', 'Delhi', 'Banglore', 'Cochin'], dtype='object', name='Destination')
```

In [72]:

```
dict2 = {key:index for index, key in enumerate(dest,0)}
dict2
```

Out[72]:

```
{'Kolkata': 0, 'Hyderabad': 1, 'Delhi': 2, 'Banglore': 3, 'Cochin': 4}
```

In [73]:

```
data['Destination']=data['Destination'].map(dict2)
data['Destination']
```

Out[73]:

```
0        2
1        3
2        4
3        3
4        2
        ..
10678    3
10679    3
10680    2
10681    2
10682    4
Name: Destination, Length: 10682, dtype: int64
```

In [74]:

```
data.head(2)
```

Out[74]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	...	Arrival_Time_hour	Arrival_Time_n
0	3	Banglore	2	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	...	1	
1	7	Kolkata	3	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	5	1	...	13	

2 rows × 23 columns

In [75]:

```
stops={'non-stop':0, '2 stops':2, '1 stop':1, '3 stops':3, '4 stops':4}
```

In [76]:

```
data['Total_Stops'] = data['Total_Stops'].map(stops)
data['Total_Stops']
```

Out[76]:

```
0        0
1        2
2        2
3        1
4        1
        ..
10678    0
10679    0
10680    0
10681    0
10682    2
Name: Total_Stops, Length: 10682, dtype: int64
```

Perform outlier Detection

How to deal with Outlier

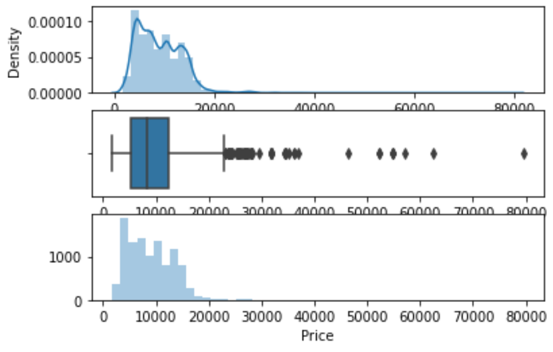
In [77]:

```
def plot(df,col):
    fig,(ax1,ax2,ax3)=plt.subplots(3,1)
    sns.distplot(df[col],ax=ax1)
    sns.boxplot(df[col],ax=ax2, orient='h')
    sns.distplot(df[col],ax=ax3,kde=False)
```

In [78]:

```
plot(data,'Price')
```

```
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and pass
ing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [79]:

```
data['Price'] = np.where(data['Price']>=35000,data['Price'].median(),data['Price'])
data['Price']
```

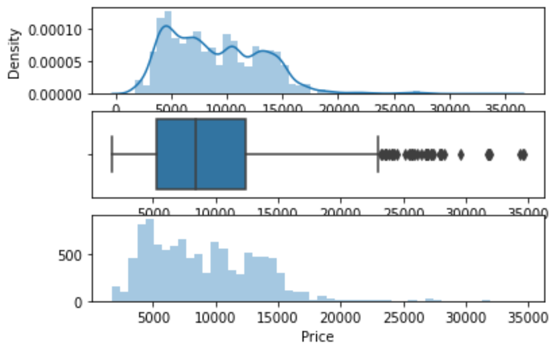
Out[79]:

```
0         3897.0
1         7662.0
2        13882.0
3         6218.0
4        13302.0
          ...
10678     4107.0
10679     4145.0
10680     7229.0
10681    12648.0
10682    11753.0
Name: Price, Length: 10682, dtype: float64
```

In [80]:

```
plot(data,'Price')
```

```
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and pass
ing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



perform feature selection

In [81]:

```
data.head(2)
```

Out[81]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	...	Arrival_Time_hour	Arrival_Time_
0	3	Banglore	2	BLR → DEL	2h 50m	0	No info	3897.0	24	3	...	1	
1	7	Kolkata	3	CCU → IXR → BBI → BLR	7h 25m	2	No info	7662.0	5	1	...	13	

2 rows × 23 columns

In [82]:

```
data.drop(columns=['Source','Route','Duration','Additional_Info'],axis=1,inplace=True)
```

In [83]:

```
data.head()
```

Out[83]:

	Airline	Destination	Total_Stops	Price	journey_day	journey_month	journey_year	Dep_Time_hour	Dep_Time_min	Arrival_Time_hour	Arrival_Time
0	3	2	0	3897.0	24	3	2019	22	20	1	
1	7	3	2	7662.0	5	1	2019	5	50	13	
2	10	4	2	13882.0	6	9	2019	9	25	4	
3	3	3	1	6218.0	5	12	2019	18	5	23	
4	3	2	1	13302.0	3	1	2019	16	50	21	

In [84]:

```
data.dtypes
```

Out[84]:

```
Airline                 int64
Destination             int64
Total_Stops             int64
Price                 float64
journey_day             int64
journey_month           int64
journey_year            int64
Dep_Time_hour           int64
Dep_Time_min            int64
Arrival_Time_hour       int64
Arrival_Time_min        int64
Duration_hours          int64
Duration_mins           int64
Duration_total_mins     int64
Source_Banglore         int64
Source_Kolkata          int64
Source_Delhi            int64
Source_Chennai          int64
Source_Mumbai           int64
dtype: object
```

In [85]:

```
from sklearn.feature_selection import mutual_info_regression
```

In [86]:

```
X = data.drop(['Price'],axis=1)
```

In [87]:

```
y = data['Price']
```

In [88]:

```
X.dtypes
```

Out[88]:

```
Airline                int64
Destination            int64
Total_Stops            int64
journey_day            int64
journey_month          int64
journey_year           int64
Dep_Time_hour          int64
Dep_Time_min           int64
Arrival_Time_hour      int64
Arrival_Time_min       int64
Duration_hours         int64
Duration_mins          int64
Duration_total_mins    int64
Source_Banglore        int64
Source_Kolkata         int64
Source_Delhi           int64
Source_Chennai         int64
Source_Mumbai          int64
dtype: object
```

In [89]:

```
mutual_info_regression(X,y)
```

Out[89]:

```
array([0.98057995, 1.00208693, 0.79719403, 0.19594542, 0.24717497,
       0.        , 0.33590966, 0.26134499, 0.40691194, 0.34270387,
       0.47209452, 0.33431404, 0.49864368, 0.38230312, 0.45671337,
       0.51552379, 0.13992276, 0.19093383])
```

In [90]:

```
imp=pd.DataFrame(mutual_info_regression(X,y),index=X.columns)
imp
```

Out[90]:

	0
Airline	0.975911
Destination	1.007568
Total_Stops	0.788004
journey_day	0.198397
journey_month	0.241038
journey_year	0.000000
Dep_Time_hour	0.345107
Dep_Time_min	0.252794
Arrival_Time_hour	0.394846
Arrival_Time_min	0.339066
Duration_hours	0.470798
Duration_mins	0.342943
Duration_total_mins	0.490521
Source_Banglore	0.393344
Source_Kolkata	0.455512
Source_Delhi	0.530347
Source_Chennai	0.136811
Source_Mumbai	0.189745

In [91]:

```
imp.columns=['Importance']
```

In [92]:

```
imp.sort_values(by='Importance', ascending=False)
```

Out[92]:

	Importance
Destination	1.007568
Airline	0.975911
Total_Stops	0.788004
Source_Delhi	0.530347
Duration_total_mins	0.490521
Duration_hours	0.470798
Source_Kolkata	0.455512
Arrival_Time_hour	0.394846
Source_Banglore	0.393344
Dep_Time_hour	0.345107
Duration_mins	0.342943
Arrival_Time_min	0.339066
Dep_Time_min	0.252794
journey_month	0.241038
journey_day	0.198397
Source_Mumbai	0.189745
Source_Chennai	0.136811
journey_year	0.000000

Buil ML model

Save ML model

In [93]:

```
from sklearn.model_selection import train_test_split
```

In [94]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

In [95]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [96]:

```
ml_model=RandomForestRegressor()
```

In [97]:

```
model= ml_model.fit(X_train,y_train)
```

In [98]:

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

Out[98]:

```
array([16738.29,  6123.54,  8826.77, ...,  3525.02,  6373.7 ,  7348.4 ])
```

In [99]:

```
y_pred.shape
```

Out[99]:

```
(2671,)
```

In [ ]:

```

```

In [100]:

```
!pip install pickle
import pickle
```

```
ERROR: Could not find a version that satisfies the requirement pickle (from versions: none)
ERROR: No matching distribution found for pickle
WARNING: There was an error checking the latest version of pip.
```

In [101]:

```
file=open('rf_random.pk1','wb')
```

In [102]:

```
pickle.dump(model,file)
```

In [103]:

```
model=open('rf_random.pk1','rb')
```

In [104]:

```
forest=pickle.load(model)
```

In [105]:

```
forest.predict(X_test)
```

Out[105]:

```
array([16738.29,  6123.54,  8826.77, ...,  3525.02,  6373.7 ,  7348.4 ])
```

define your Evaluation metric

MAPE : Mean Absolute Percentage Error

In [106]:

```
def mape(y_true,y_pred):
    y_true,y_pred = np.array(y_true), np.array(y_pred)

    return np.mean(np.abs((y_true-y_pred)/y_true))*100
```

In [107]:

```
mape(y_test,forest.predict(X_test))
```

Out[107]:

```
13.151881638962118
```

Automate ML pipeline

In [108]:

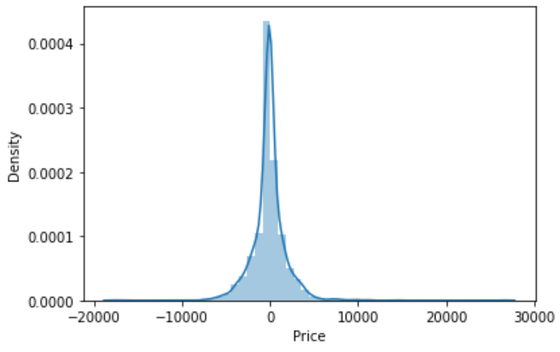
```
def predict(ml_model):
    model = ml_model.fit(X_train, y_train)
    print(f'Training score : {model.score(X_train,y_train)}')
    y_prediction = model.predict(X_test)
    print(f'Prediction : {y_prediction}')
    print('\n')

    from sklearn import metrics
    print(f'r2_Score: {metrics.r2_score(y_test,y_prediction)}')
    print(f'MSE: {metrics.mean_squared_error(y_test,y_prediction)}')
    print(f'MAE: {metrics.mean_absolute_error(y_test,y_prediction)}')
    print(f'RMSE: {np.sqrt(metrics.mean_absolute_error(y_test,y_prediction))}')
    print(f'MAPE: {mape(y_test,y_prediction)}')
    sns.distplot(y_test-y_prediction)
```

In [109]:

```
predict(RandomForestRegressor())
```

```
Training score : 0.9503690636224871
Prediction : [16801.08  5914.5   8910.58 ...  3526.11  6201.73  7424.8 ]


r2_Score: 0.8085150317203675
MSE: 3727752.704415473
MAE: 1183.3962163698754
RMSE: 34.400526396697416
MAPE: 13.300970521472177
```

```
C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Hypertune ML model

In [110]:

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

In [111]:

```
reg_rf = RandomForestRegressor()
```

In [112]:

```
np.linspace(start=1000,stop=1200,num=6)
```

Out[112]:

```
array([1000., 1040., 1080., 1120., 1160., 1200.])
```

In [113]:

```
[int(x) for x in np.linspace(start=1000,stop=1200,num=6)]
```

Out[113]:

```
[1000, 1040, 1080, 1120, 1160, 1200]
```

In [114]:

```
[int(x) for x in np.linspace(start=5,stop=10,num=4)]
```

Out[114]:

```
[5, 6, 8, 10]
```

In [115]:

```
n_estimators=[int(x) for x in np.linspace(start=1000,stop=1200,num=6)]

max_depth=[int(x) for x in np.linspace(start=5,stop=10,num=4)]

min_samples_split=[5,10,15,100]

max_features=['auto', 'sqrt']
```

In [116]:

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

In [117]:

```
random_grid
```

Out[117]:

```
{'n_estimators': [1000, 1040, 1080, 1120, 1160, 1200],
 'max_depth': [5, 6, 8, 10],
 'max_features': ['auto', 'sqrt'],
 'min_samples_split': [5, 10, 15, 100]}
```

In [118]:

```
rf_random = RandomizedSearchCV(reg_rf, param_distributions=random_grid, cv=3, verbose=2)
rf_random
```

Out[118]:

```
▸        RandomizedSearchCV
▸ estimator: RandomForestRegressor
      ▸ RandomForestRegressor
```

In [119]:

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

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
[CV] END max_depth=5, max_features=sqrt, min_samples_split=15, n_estimators=1080; total time=   4.1s
[CV] END max_depth=5, max_features=sqrt, min_samples_split=15, n_estimators=1080; total time=   3.8s
[CV] END max_depth=5, max_features=sqrt, min_samples_split=15, n_estimators=1080; total time=   3.4s
[CV] END max_depth=8, max_features=sqrt, min_samples_split=100, n_estimators=1040; total time=   4.0s
[CV] END max_depth=8, max_features=sqrt, min_samples_split=100, n_estimators=1040; total time=   4.0s
[CV] END max_depth=8, max_features=sqrt, min_samples_split=100, n_estimators=1040; total time=   4.0s
[CV] END max_depth=10, max_features=sqrt, min_samples_split=10, n_estimators=1160; total time=   5.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_split=10, n_estimators=1160; total time=   5.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_split=10, n_estimators=1160; total time=   7.0s

C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarni
ng: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly
set `max_features=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTre
esRegressors.
  warn(

[CV] END max_depth=10, max_features=auto, min_samples_split=5, n_estimators=1120; total time=  16.3s

C:\Users\DELL PC\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarni
```

In [120]:

```
rf_random.best_params_
```

Out[120]:

```
{'n_estimators': 1120,
 'min_samples_split': 5,
 'max_features': 'auto',
 'max_depth': 10}
```

In [149]:

```
pred2=rf_random.predict(X_test)
pred2
```

Out[149]:

```
array([16462.45193682,  6013.95263068,  8648.72612041, ...,
        4220.96176186,  7816.20670524,  7441.34789807])
```

In [150]:

```
from sklearn import metrics
metrics.r2_score(y_test,pred2)
```

Out[150]:

```
0.8224163716668245
```