In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing dataset

1.Since data is in form of excel file we have to use pandas read_excel to load the data
2.After loading it is important to check null values in a column or a row
3.If it is present then following can be done,
    a.Filling NaN values with mean, median and mode using fillna() method
    b.If Less missing values, we can drop it as well

In [5]:
```
train_data=pd.read_csv('Data_Train.csv')
```

In [6]:
```
train_data.head()
```

Out[6]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total
0	IndiGo	24-03-2019	Banglore	New Delhi	BLR → DEL	22:20	22-03-2021 01:10	2h 50m	n
1	Air India	01-05-2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	
2	Jet Airways	09-06-2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	10-06-2021 04:25	19h	
3	IndiGo	12-05-2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	
4	IndiGo	01-03-2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	

In [7]:
```
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 [8]:
```
train_data.isnull().sum()
```

Out[8]:
```
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
```

as less missing values,I can directly drop these

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]:
```
train_data.dtypes
```

Out[11]:
```
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 [ ]:
```

```

From description we can see that Date_of_Journey is a object data type,

Therefore, we have to convert this datatype into timestamp so as to use this column properly for prediction,bcz our
model will not be able to understand Theses string values,it just under stand Time-stamp
For this we require pandas to_datetime to convert object data type to da tetime dtype.

dt.day method will extract only day of that date
dt.month method will extract only month of that date

In [12]:
```
def change_into_datetime(col):
    train_data[col]=pd.to_datetime(train_data[col])
```

In [13]:
```
train_data.columns
```

Out[13]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price'],
      dtype='object')
```

In [14]:
```
for i in ['Date_of_Journey','Dep_Time', 'Arrival_Time']:
    change_into_datetime(i)
```

In [15]:
```
train_data.dtypes
```

Out[15]:
```
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 [ ]:
```

```

In [ ]:
```

```

In [16]:
```
train_data['Journey_day']=train_data['Date_of_Journey'].dt.day
```

In [17]:
```
train_data['Journey_month']=train_data['Date_of_Journey'].dt.month
```

In [18]:
```
train_data.head()
```

Out[18]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2021-09-16 22:20:00	2021-03-22 01:10:00	2h 50m	n
1	Air India	2019-01-05	Kolkata	Banglore	CCU → IXR → BBI → BLR	2021-09-16 05:50:00	2021-09-16 13:15:00	7h 25m	
2	Jet Airways	2019-09-06	Delhi	Cochin	DEL → LKO → BOM → COK	2021-09-16 09:25:00	2021-10-06 04:25:00	19h	
3	IndiGo	2019-12-05	Kolkata	Banglore	CCU → NAG → BLR	2021-09-16 18:05:00	2021-09-16 23:30:00	5h 25m	
4	IndiGo	2019-01-03	Banglore	New Delhi	BLR → NAG → DEL	2021-09-16 16:50:00	2021-09-16 21:35:00	4h 45m	

In [ ]:
```
```

In [19]:
```
## Since we have converted Date_of_Journey column into integers, Now we can drop
train_data.drop('Date_of_Journey', axis=1, inplace=True)
```

In [ ]:
```
```

In [ ]:
```
```

In [20]:
```
train_data.head()
```

Out[20]:

	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional
0	IndiGo	Banglore	New Delhi	BLR → DEL	2021-09-16 22:20:00	2021-03-22 01:10:00	2h 50m	non-stop	N
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2021-09-16 05:50:00	2021-09-16 13:15:00	7h 25m	2 stops	N
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2021-09-16 09:25:00	2021-10-06 04:25:00	19h	2 stops	N
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	2021-09-16 18:05:00	2021-09-16 23:30:00	5h 25m	1 stop	N
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	2021-09-16 16:50:00	2021-09-16 21:35:00	4h 45m	1 stop	N

In [ ]:
```

```

In [21]:
```
def extract_hour(df,col):
    df[col+"_hour"]=df[col].dt.hour
```

In [22]:
```
def extract_min(df,col):
    df[col+"_minute"]=df[col].dt.minute
```

In [23]:
```
def drop_column(df,col):
    df.drop(col,axis=1,inplace=True)
```

In [ ]:
```

```

In [24]:
```
# Departure time is when a plane leaves the gate.
# Similar to Date_of_Journey we can extract values from Dep_Time
extract_hour(train_data,'Dep_Time')
```

In [25]:
```
# Extracting Minutes
extract_min(train_data,'Dep_Time')
```

In [26]:
```
# Now we can drop Dep_Time as it is of no use
drop_column(train_data,'Dep_Time')
```

In [27]:
```
train_data.head()
```

Out[27]:

	Airline	Source	Destination	Route	Arrival_Time	Duration	Total_Stops	Additional_Info	Pric
0	IndiGo	Banglore	New Delhi	BLR → DEL	2021-03-22 01:10:00	2h 50m	non-stop	No info	389
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2021-09-16 13:15:00	7h 25m	2 stops	No info	766
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2021-10-06 04:25:00	19h	2 stops	No info	1388
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	2021-09-16 23:30:00	5h 25m	1 stop	No info	621
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	2021-09-16 21:35:00	4h 45m	1 stop	No info	1330

In [ ]:
```
```

In [28]:
```
# Arrival time is when the plane pulls up to the gate.
# Similar to Date_of_Journey we can extract values from Arrival_Time

# Extracting Hours
extract_hour(train_data,'Arrival_Time')

# Extracting minutes
extract_min(train_data,'Arrival_Time')

# Now we can drop Arrival_Time as it is of no use
drop_column(train_data,'Arrival_Time')
```

In [29]:
```
train_data.head()
```

Out[29]:

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

In [ ]:
```

```

In [30]:
```
'2h 50m'.split(' ')
```

Out[30]:
```
['2h', '50m']
```

In [ ]:
```

```

Lets Apply pre-processing on duration column,Separate Duration hours and minute from duration

In [31]:
```
duration=list(train_data['Duration'])

for i in range(len(duration)):
    if len(duration[i].split(' '))==2:
        pass
    else:
        if 'h' in duration[i]:                    # Check if duration contains onl
            duration[i]=duration[i] + ' 0m'       # Adds 0 minute
        else:
            duration[i]='0h '+ duration[i]        # if duration contains only secc
```

In [32]:
```
train_data['Duration']=duration
```

In [33]:
```
train_data.head()
```

Out[33]:

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

In [34]:
```
'2h 50m'.split(' ')[1][0:-1]
```

Out[34]:
```
'50'
```

In [ ]:

```
```

In [35]:

```
def hour(x):
    return x.split(' ')[0][0:-1]
```

In [36]:

```
def min(x):
    return x.split(' ')[1][0:-1]
```

In [37]:

```
train_data['Duration_hours']=train_data['Duration'].apply(hour)
train_data['Duration_mins']=train_data['Duration'].apply(min)
```

In [38]:

```
train_data.head()
```

Out[38]:

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

In [39]:

```
train_data.drop('Duration',axis=1,inplace=True)
```

In [40]:
```
train_data.head()
```

Out[40]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Price	Journey_day	Journe
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	3897	24	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	7662	5	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info	13882	6	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info	6218	5	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info	13302	3	

In [41]:
```
train_data.dtypes
```

Out[41]:
```
Airline                object
Source                 object
Destination            object
Route                  object
Total_Stops            object
Additional_Info        object
Price                   int64
Journey_day             int64
Journey_month           int64
Dep_Time_hour           int64
Dep_Time_minute         int64
Arrival_Time_hour       int64
Arrival_Time_minute     int64
Duration_hours         object
Duration_mins          object
dtype: object
```

In [42]:
```
train_data['Duration_hours']=train_data['Duration_hours'].astype(int)
train_data['Duration_mins']=train_data['Duration_mins'].astype(int)
```

In [43]:
```
train_data.dtypes
```

Out[43]:
```
Airline                object
Source                 object
Destination            object
Route                  object
Total_Stops            object
Additional_Info        object
Price                   int64
Journey_day             int64
Journey_month           int64
Dep_Time_hour           int64
Dep_Time_minute         int64
Arrival_Time_hour       int64
Arrival_Time_minute     int64
Duration_hours          int32
Duration_mins           int32
dtype: object
```

In [44]:
```
train_data.head()
```

Out[44]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Price	Journey_day	Journe
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	3897	24	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	7662	5	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info	13882	6	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info	6218	5	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info	13302	3	

In [45]:
```
train_data.dtypes
```

Out[45]:
```
Airline               object
Source                object
Destination           object
Route                 object
Total_Stops           object
Additional_Info       object
Price                  int64
Journey_day            int64
Journey_month          int64
Dep_Time_hour          int64
Dep_Time_minute        int64
Arrival_Time_hour      int64
Arrival_Time_minute    int64
Duration_hours         int32
Duration_mins          int32
dtype: object
```

In [46]:
```
cat_col=[col for col in train_data.columns if train_data[col].dtype=='O']
cat_col
```

Out[46]:
```
['Airline', 'Source', 'Destination', 'Route', 'Total_Stops', 'Additional_Info']
```

In [47]:
```
cont_col=[col for col in train_data.columns if train_data[col].dtype!='O']
cont_col
```

Out[47]:
```
['Price',
 'Journey_day',
 'Journey_month',
 'Dep_Time_hour',
 'Dep_Time_minute',
 'Arrival_Time_hour',
 'Arrival_Time_minute',
 'Duration_hours',
 'Duration_mins']
```

Handling Categorical Data

We are using 2 main Encoding Techniques to convert Categorical data into some numerical format

```
Nominal data --> data are not in any order --> OneHotEncoder is used in
 this case
Ordinal data --> data are in order -->       LabelEncoder is used in thi
s case
```

In [48]:
```
categorical=train_data[cat_col]
categorical.head()
```

Out[48]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info

In [49]:
```
categorical['Airline'].value_counts()
```

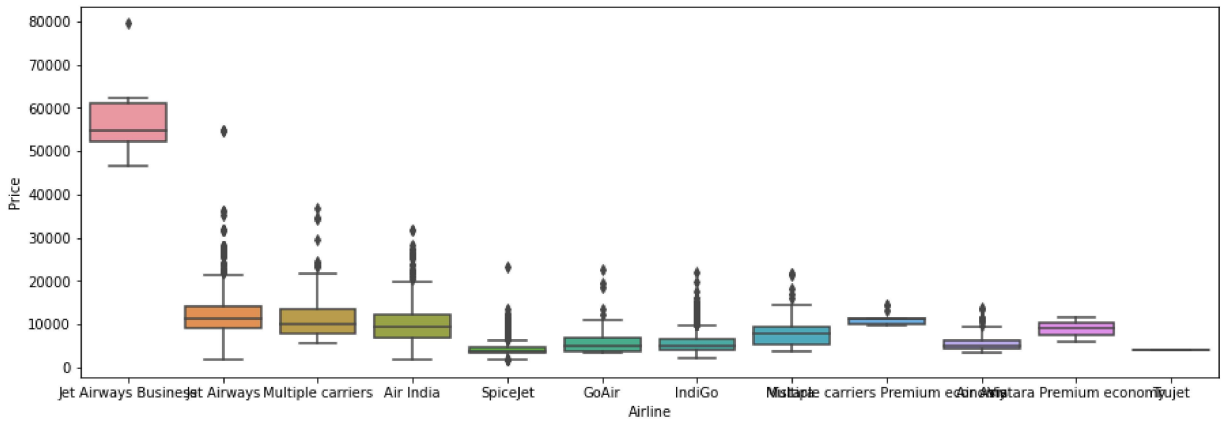
Out[49]:
```
Jet Airways                          3849
IndiGo                               2053
Air India                            1751
Multiple carriers                    1196
SpiceJet                              818
Vistara                               479
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
```

In [ ]:
```

```

Airline vs Price Analysis

In [50]:
```
plt.figure(figsize=(15,5))
sns.boxplot(y='Price',x='Airline',data=train_data.sort_values('Price',ascending=F
```

Out[50]: <AxesSubplot:xlabel='Airline', ylabel='Price'>



In [ ]:

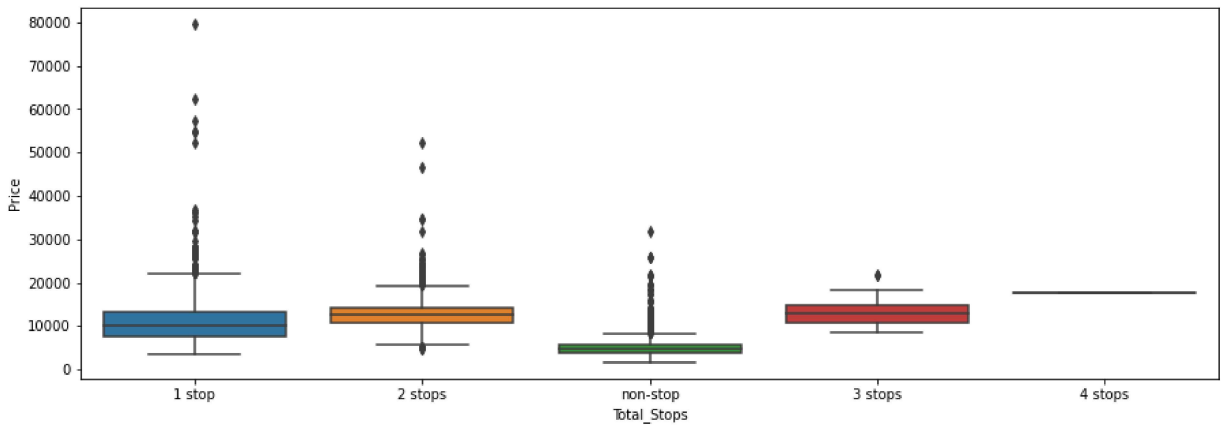
Conclusion--> From graph we can see that Jet Airways Business have the highest Price., Apart from the first Airline almost all are having similar median

In [ ]:

Perform Total_Stops vs Price Analysis

In [51]:
```
plt.figure(figsize=(15,5))
sns.boxplot(y='Price',x='Total_Stops',data=train_data.sort_values('Price',ascendi
```

Out[51]: <AxesSubplot:xlabel='Total_Stops', ylabel='Price'>



In [52]:
```
len(categorical['Airline'].unique())
```

Out[52]: 12

In [53]:
```
# As Airline is Nominal Categorical data we will perform OneHotEncoding
Airline=pd.get_dummies(categorical['Airline'], drop_first=True)
Airline.head()
```

Out[53]:

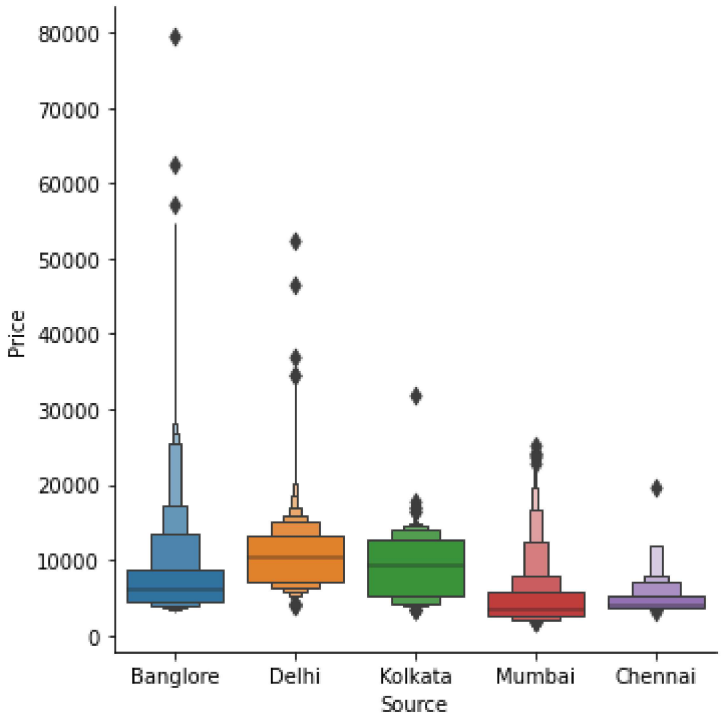
	Air India	GoAir	IndiGo	Jet Airways	Jet Airways Business	Multiple carriers	Multiple carriers Premium economy	SpiceJet	Trujet	Vistara	Vistara Premium economy
0	0	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0	0

In [54]:
```
categorical['Source'].value_counts()
```

Out[54]:
```
Delhi       4536
Kolkata     2871
Banglore    2197
Mumbai       697
Chennai      381
Name: Source, dtype: int64
```

In [55]:
```
# Source vs Price

plt.figure(figsize=(15,5))
sns.catplot(y='Price',x='Source',data=train_data.sort_values('Price',ascending=Fa
```

Out[55]:
```
<seaborn.axisgrid.FacetGrid at 0x20bf99eeb20>

<Figure size 1080x360 with 0 Axes>
```



In [56]:
```
# As Source is Nominal Categorical data we will perform OneHotEncoding


Source=pd.get_dummies(categorical['Source'], drop_first=True)
Source.head()
```

Out[56]:

	Chennai	Delhi	Kolkata	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 [57]:
```
categorical['Destination'].value_counts()
```

Out[57]:
```
Cochin       4536
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: Destination, dtype: int64
```

In [58]:
```
# As Destination is Nominal Categorical data we will perform OneHotEncoding

Destination=pd.get_dummies(categorical['Destination'], drop_first=True)
Destination.head()
```

Out[58]:

	Cochin	Delhi	Hyderabad	Kolkata	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 [59]:
```
categorical['Route']
```

Out[59]:
```
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 [60]:
```
categorical['Route_1']=categorical['Route'].str.split('→').str[0]
categorical['Route_2']=categorical['Route'].str.split('→').str[1]
categorical['Route_3']=categorical['Route'].str.split('→').str[2]
categorical['Route_4']=categorical['Route'].str.split('→').str[3]
categorical['Route_5']=categorical['Route'].str.split('→').str[4]
```

```
<ipython-input-60-103bd018a128>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  categorical['Route_1']=categorical['Route'].str.split('→').str[0]
<ipython-input-60-103bd018a128>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  categorical['Route_2']=categorical['Route'].str.split('→').str[1]
<ipython-input-60-103bd018a128>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  categorical['Route_3']=categorical['Route'].str.split('→').str[2]
<ipython-input-60-103bd018a128>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  categorical['Route_4']=categorical['Route'].str.split('→').str[3]
<ipython-input-60-103bd018a128>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  categorical['Route_5']=categorical['Route'].str.split('→').str[4]
```

In [61]:
```
categorical.head()
```

Out[61]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Route_1	Route_2	Route_3
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	BLR	DEL	NaN
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	CCU	IXR	BBI
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info	DEL	LKO	BOM
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info	CCU	NAG	BLR
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info	BLR	NAG	DEL

In [64]:
```
import warnings
from warnings import filterwarnings
filterwarnings('ignore')
```

In [65]:
```
categorical['Route_1'].fillna('None',inplace=True)
categorical['Route_2'].fillna('None',inplace=True)
categorical['Route_3'].fillna('None',inplace=True)
categorical['Route_4'].fillna('None',inplace=True)
categorical['Route_5'].fillna('None',inplace=True)
```

In [66]:
```
categorical.head()
```

Out[66]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Route_1	Route_2	Route_3
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	BLR	DEL	None
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	CCU	IXR	BBI
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info	DEL	LKO	BOM
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info	CCU	NAG	BLR
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info	BLR	NAG	DEL

In [67]:
```
#now extract how many categories in each cat_feature
for feature in categorical.columns:
    print('{} has total {} categories \n'.format(feature,len(categorical[feature]
```

```
Airline has total 12 categories

Source has total 5 categories

Destination has total 6 categories

Route has total 128 categories

Total_Stops has total 5 categories

Additional_Info has total 10 categories

Route_1 has total 5 categories

Route_2 has total 45 categories

Route_3 has total 30 categories

Route_4 has total 14 categories

Route_5 has total 6 categories
```

In [68]:
```
### as we will see we have lots of features in Route , one hot encoding will not
```

In [69]:
```
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
```

In [70]:
```
categorical.columns
```

Out[70]:
```
Index(['Airline', 'Source', 'Destination', 'Route', 'Total_Stops',
       'Additional_Info', 'Route_1', 'Route_2', 'Route_3', 'Route_4',
       'Route_5'],
      dtype='object')
```

In [71]:
```
for i in ['Route_1', 'Route_2', 'Route_3', 'Route_4','Route_5']:
    categorical[i]=encoder.fit_transform(categorical[i])
```

In [72]:
```
categorical.head()
```

Out[72]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Route_1	Route_2	Route_3
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	0	13	29
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	2	25	1
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info	3	32	4
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info	2	34	3
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info	0	34	8

In [73]:
```
# Additional_Info contains almost 80% no_info,so we can drop this column
# we can drop Route as well as we have pre-process that column

drop_column(categorical,'Route')
drop_column(categorical,'Additional_Info')
```

In [74]:
```
categorical.head()
```

Out[74]:

	Airline	Source	Destination	Total_Stops	Route_1	Route_2	Route_3	Route_4	Route_5
0	IndiGo	Banglore	New Delhi	non-stop	0	13	29	13	5
1	Air India	Kolkata	Banglore	2 stops	2	25	1	3	5
2	Jet Airways	Delhi	Cochin	2 stops	3	32	4	5	5
3	IndiGo	Kolkata	Banglore	1 stop	2	34	3	13	5
4	IndiGo	Banglore	New Delhi	1 stop	0	34	8	13	5

In [75]:
```
categorical['Total_Stops'].value_counts()
```

Out[75]:
```
1 stop      5625
non-stop    3491
2 stops     1520
3 stops       45
4 stops        1
Name: Total_Stops, dtype: int64
```

In [76]:
```
categorical['Total_Stops'].unique()
```

Out[76]:
```
array(['non-stop', '2 stops', '1 stop', '3 stops', '4 stops'],
      dtype=object)
```

In [77]:
```
# As this is case of Ordinal Categorical type we perform LabelEncoder
# Here Values are assigned with corresponding key

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

In [78]:
```
categorical['Total_Stops']=categorical['Total_Stops'].map(dict)
```

In [79]:
```
categorical.head()
```

Out[79]:

	Airline	Source	Destination	Total_Stops	Route_1	Route_2	Route_3	Route_4	Route_5
0	IndiGo	Banglore	New Delhi	0	0	13	29	13	5
1	Air India	Kolkata	Banglore	2	2	25	1	3	5
2	Jet Airways	Delhi	Cochin	2	3	32	4	5	5
3	IndiGo	Kolkata	Banglore	1	2	34	3	13	5
4	IndiGo	Banglore	New Delhi	1	0	34	8	13	5

In [80]:
```
train_data[cont_col]
```

Out[80]:

	Price	Journey_day	Journey_month	Dep_Time_hour	Dep_Time_minute	Arrival_Time_hour
0	3897	24	3	22	20	1
1	7662	5	1	5	50	13
2	13882	6	9	9	25	4
3	6218	5	12	18	5	23
4	13302	3	1	16	50	21
...	...	...	...	...	...	...
10678	4107	4	9	19	55	22
10679	4145	27	4	20	45	23
10680	7229	27	4	8	20	11
10681	12648	3	1	11	30	14
10682	11753	5	9	10	55	19

10682 rows × 9 columns

In [81]:
```
# Concatenate dataframe --> categorical + Airline + Source + Destination

data_train=pd.concat([categorical,Airline,Source,Destination,train_data[cont_col]
data_train.head()
```

Out[81]:

	Airline	Source	Destination	Total_Stops	Route_1	Route_2	Route_3	Route_4	Route_5	A Indi
0	IndiGo	Banglore	New Delhi	0	0	13	29	13	5	
1	Air India	Kolkata	Banglore	2	2	25	1	3	5	
2	Jet Airways	Delhi	Cochin	2	3	32	4	5	5	
3	IndiGo	Kolkata	Banglore	1	2	34	3	13	5	
4	IndiGo	Banglore	New Delhi	1	0	34	8	13	5	

5 rows × 38 columns

In [82]:
```
drop_column(data_train,'Airline')
drop_column(data_train,'Source')
drop_column(data_train,'Destination')
```

In [83]:
```
data_train.head()
```

Out[83]:

	Total_Stops	Route_1	Route_2	Route_3	Route_4	Route_5	Air India	GoAir	IndiGo	Jet Airways	...
0	0	0	13	29	13	5	0	0	1	0	...
1	2	2	25	1	3	5	1	0	0	0	...
2	2	3	32	4	5	5	0	0	0	1	...
3	1	2	34	3	13	5	0	0	1	0	...
4	1	0	34	8	13	5	0	0	1	0	...

5 rows × 35 columns

In [ ]:
```

```

In [84]:
```
pd.set_option('display.max_columns',35)
```

In [85]:
```
data_train.head()
```

Out[85]:

	Total_Stops	Route_1	Route_2	Route_3	Route_4	Route_5	Air India	GoAir	IndiGo	Jet Airways	Ai Bus
0	0	0	13	29	13	5	0	0	1	0	
1	2	2	25	1	3	5	1	0	0	0	
2	2	3	32	4	5	5	0	0	0	1	
3	1	2	34	3	13	5	0	0	1	0	
4	1	0	34	8	13	5	0	0	1	0	

In [86]:
```
data_train.columns
```

Out[86]:
```
Index(['Total_Stops', 'Route_1', 'Route_2', 'Route_3', 'Route_4', 'Route_5',
       'Air India', 'GoAir', 'IndiGo', 'Jet Airways', 'Jet Airways Business',
       'Multiple carriers', 'Multiple carriers Premium economy', 'SpiceJet',
       'Trujet', 'Vistara', 'Vistara Premium economy', 'Chennai', 'Delhi',
       'Kolkata', 'Mumbai', 'Cochin', 'Delhi', 'Hyderabad', 'Kolkata',
       'New Delhi', 'Price', 'Journey_day', 'Journey_month', 'Dep_Time_hour',
       'Dep_Time_minute', 'Arrival_Time_hour', 'Arrival_Time_minute',
       'Duration_hours', 'Duration_mins'],
      dtype='object')
```

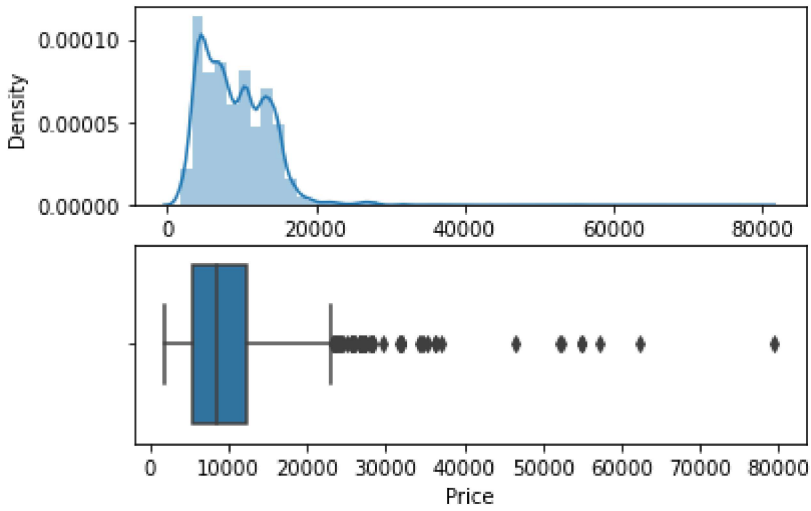
In [ ]:
```

```

outlier detection

In [87]:
```
def plot(df,col):
    fig,(ax1,ax2)=plt.subplots(2,1)
    sns.distplot(df[col],ax=ax1)
    sns.boxplot(df[col],ax=ax2)
```

In [88]:
```
plt.figure(figsize=(30,20))
plot(data_train,'Price')
```

```
<Figure size 2160x1440 with 0 Axes>
```



In [ ]:

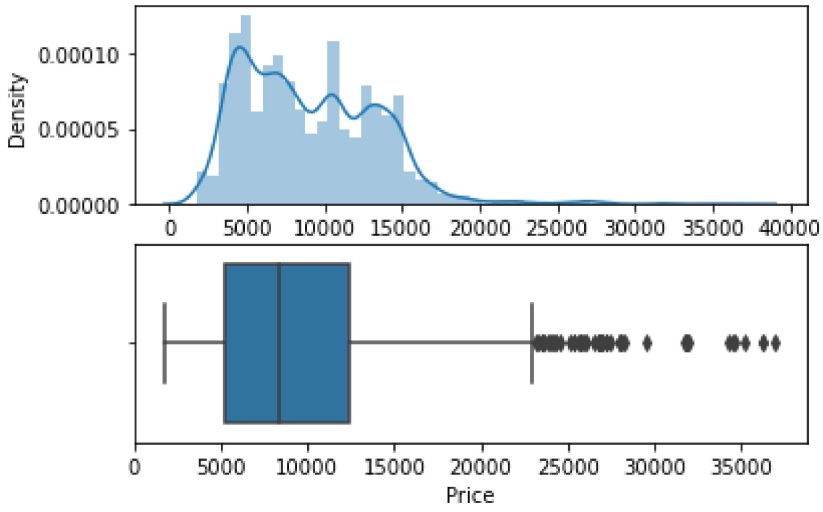
In [ ]:

dealing with Outliers

In [89]:
```
data_train['Price']=np.where(data_train['Price']>=40000,data_train['Price'].media
```

In [90]:
```
plt.figure(figsize=(30,20))
plot(data_train,'Price')
```

```
<Figure size 2160x1440 with 0 Axes>
```



In [ ]:

In [91]:
```
### separate your independent & dependent data
```

In [92]:
```
X=data_train.drop('Price',axis=1)
X.head()
```

Out[92]:

	Total_Stops	Route_1	Route_2	Route_3	Route_4	Route_5	Air India	GoAir	IndiGo	Jet Airways	Ai Bus
0	0	0	13	29	13	5	0	0	1	0	
1	2	2	25	1	3	5	1	0	0	0	
2	2	3	32	4	5	5	0	0	0	1	
3	1	2	34	3	13	5	0	0	1	0	
4	1	0	34	8	13	5	0	0	1	0	

In [ ]:

In [93]:
```
y=data_train['Price']
y
```

Out[93]:
```
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 [94]:
```
##type(X)
```

In [95]:
```
##type(y)
```

In [96]:
```
##X.isnull().sum()
```

In [97]:
```
##y.isnull().sum()
```

In [98]:
```
#### as now we dont have any missing value in data, we can definitely go ahead wi
```

In [ ]:

Feature Selection

Finding out the best feature which will contribute and have good relation with target variable.

Why to apply Feature Selection?

To select important features to get rid of curse of dimensionality ie..to get rid of duplicate features

In [99]:
```
###np.array(X)
```

In [100]:
```
##np.array(y)
```

I wanted to find mutual information scores or matrix to get to know about the relationship between all features.

Feature Selection using Information Gain,

In [101]:
```
from sklearn.feature_selection import mutual_info_classif
```

In [103]:
```
mutual_info_classif(X,y)
```

Out[103]:
```
array([2.16624865e+00, 2.09184250e+00, 2.77496221e+00, 2.32402390e+00,
       1.52348762e+00, 7.35083602e-01, 7.52228471e-01, 1.01426798e-01,
       6.46385448e-01, 9.04196356e-01, 1.38454990e-03, 5.64863998e-01,
       1.18164379e-03, 3.15359696e-01, 6.80766677e-03, 2.38183968e-01,
       6.42272987e-03, 1.67725520e-01, 1.57602639e+00, 8.81644318e-01,
       2.92239561e-01, 1.52222670e+00, 3.98749975e-01, 2.94814042e-01,
       1.73266498e-01, 3.86681136e-01, 1.08583146e+00, 8.74390662e-01,
       1.45202595e+00, 1.22109350e+00, 1.86829982e+00, 1.54637493e+00,
       1.77813619e+00, 1.07158169e+00])
```

In [105]:
```
###mutual_info_classif(np.array(X),np.array(y))
```

In [106]:
```
X.dtypes
```

Out[106]:
```
Total_Stops                          int64
Route_1                              int32
Route_2                              int32
Route_3                              int32
Route_4                              int32
Route_5                              int32
Air India                            uint8
GoAir                                uint8
IndiGo                               uint8
Jet Airways                          uint8
Jet Airways Business                 uint8
Multiple carriers                    uint8
Multiple carriers Premium economy    uint8
SpiceJet                             uint8
Trujet                               uint8
Vistara                              uint8
Vistara Premium economy              uint8
Chennai                              uint8
Delhi                                uint8
Kolkata                              uint8
Mumbai                               uint8
Cochin                               uint8
Delhi                                uint8
Hyderabad                            uint8
Kolkata                              uint8
New Delhi                            uint8
Journey_day                          int64
Journey_month                        int64
Dep_Time_hour                        int64
Dep_Time_minute                      int64
Arrival_Time_hour                    int64
Arrival_Time_minute                  int64
Duration_hours                       int32
Duration_mins                        int32
dtype: object
```

In [108]:
```
mutual_info_classif(X,y)
```

Out[108]:
```
array([2.10420694, 2.02288503, 2.76702036, 2.30495152, 1.46031667,
       0.69379895, 0.7743615 , 0.09794173, 0.67639837, 0.9112161 ,
       0.01903731, 0.59863894, 0.0057815 , 0.31567583, 0.00823585,
       0.22964172, 0.        , 0.16111551, 1.55830179, 0.87712674,
       0.29763784, 1.54765299, 0.41085078, 0.28681834, 0.17035392,
       0.38600422, 1.10288431, 0.86152576, 1.4202642 , 1.23995323,
       1.82083961, 1.52753584, 1.76681439, 1.09880196])
```

In [109]:
```
imp=pd.DataFrame(mutual_info_classif(X,y),index=X.columns)
imp
```

Out[109]:

	0
Total_Stops	2.181263
Route_1	2.050280
Route_2	2.761046
Route_3	2.290830
Route_4	1.470175
Route_5	0.723556
Air India	0.751284
GoAir	0.098168
IndiGo	0.669521
Jet Airways	0.916764
Jet Airways Business	0.000000
Multiple carriers	0.573889
Multiple carriers Premium economy	0.007555
SpiceJet	0.328995
Trujet	0.000000
Vistara	0.212400
Vistara Premium economy	0.000000
Chennai	0.157762
Delhi	1.507579
Kolkata	0.875130
Mumbai	0.297960
Cochin	1.557589
Delhi	0.401512
Hyderabad	0.299550
Kolkata	0.169371
New Delhi	0.378301
Journey_day	1.079159
Journey_month	0.862234
Dep_Time_hour	1.428549
Dep_Time_minute	1.224485
Arrival_Time_hour	1.822581
Arrival_Time_minute	1.542449
Duration_hours	1.795299
Duration_mins	1.057212

In [110]:
```
imp.columns=['importance']
imp.sort_values(by='importance',ascending=False)
```

Out[110]:

	importance
Route_2	2.761046
Route_3	2.290830
Total_Stops	2.181263
Route_1	2.050280
Arrival_Time_hour	1.822581
Duration_hours	1.795299
Cochin	1.557589
Arrival_Time_minute	1.542449
Delhi	1.507579
Route_4	1.470175
Dep_Time_hour	1.428549
Dep_Time_minute	1.224485
Journey_day	1.079159
Duration_mins	1.057212
Jet Airways	0.916764
Kolkata	0.875130
Journey_month	0.862234
Air India	0.751284
Route_5	0.723556
IndiGo	0.669521
Multiple carriers	0.573889
Delhi	0.401512
New Delhi	0.378301
SpiceJet	0.328995
Hyderabad	0.299550
Mumbai	0.297960
Vistara	0.212400
Kolkata	0.169371
Chennai	0.157762
GoAir	0.098168
Multiple carriers Premium economy	0.007555
Jet Airways Business	0.000000
Vistara Premium economy	0.000000
Trujet	0.000000

In [ ]:
```
```

split dataset into train & test

In [111]:
```
from sklearn.model_selection import train_test_split
```

In [112]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

In [ ]:
```
```

In [125]:
```
from sklearn import metrics
##dump your model using pickle so that we will re-use
import pickle
def predict(ml_model,dump):
    model=ml_model.fit(X_train,y_train)
    print('Training score : {}'.format(model.score(X_train,y_train)))
    y_prediction=model.predict(X_test)
    print('predictions are: \n {}'.format(y_prediction))
    print('\n')
    r2_score=metrics.r2_score(y_test,y_prediction)
    print('r2 score: {}'.format(r2_score))
    print('MAE:',metrics.mean_absolute_error(y_test,y_prediction))
    print('MSE:',metrics.mean_squared_error(y_test,y_prediction))
    print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_prediction)))
    sns.distplot(y_test-y_prediction)

    if dump==1:
        ##dump your model using pickle so that we will re-use
        file=open('D:\Rounith/model.pkl','wb')
        pickle.dump(model,file)
```

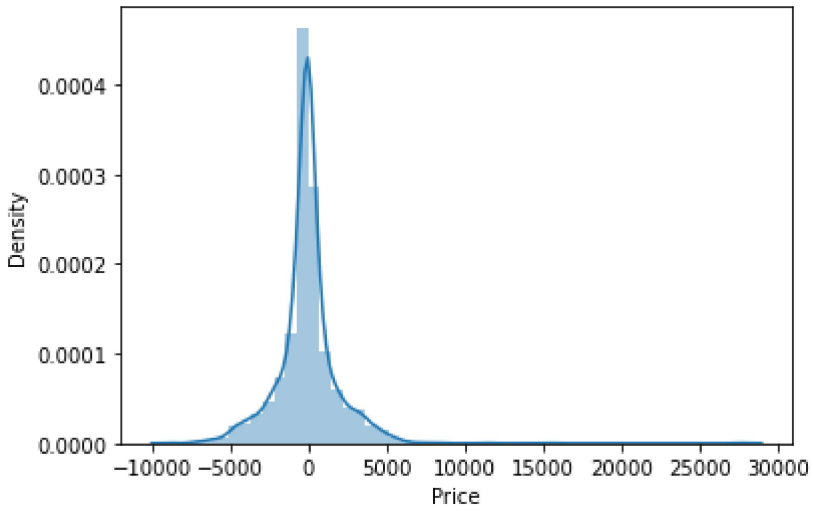
In [ ]:
```
```

import randomforest class

In [126]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [127]:
```
predict(RandomForestRegressor(),1)
```

```
Training score : 0.9556600854797384
predictions are:
 [ 6126.59       12863.75716667 10723.63       ...  7002.58
 15054.80333333 13627.2182619 ]


r2 score: 0.8041451152065427
MAE: 1181.0056999696608
MSE: 3785640.3836101824
RMSE: 1945.6722189542056
```



In [ ]:

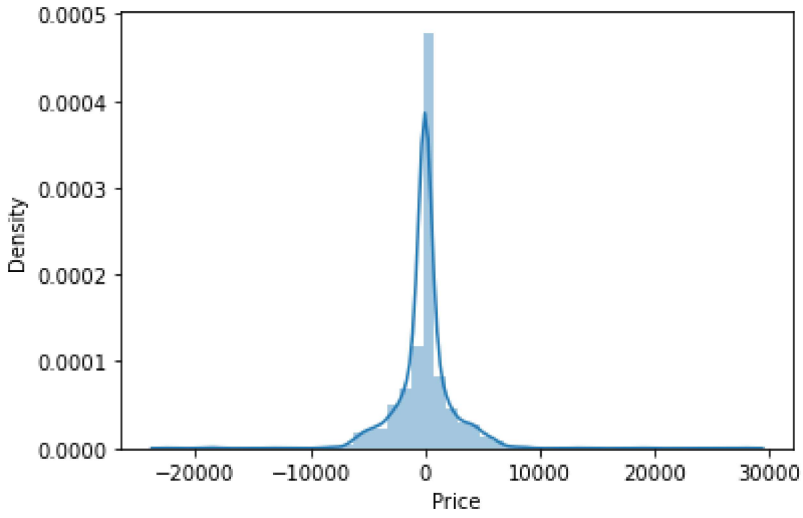
play with multiple Algorithms

In [128]:
```
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
```

In [ ]:

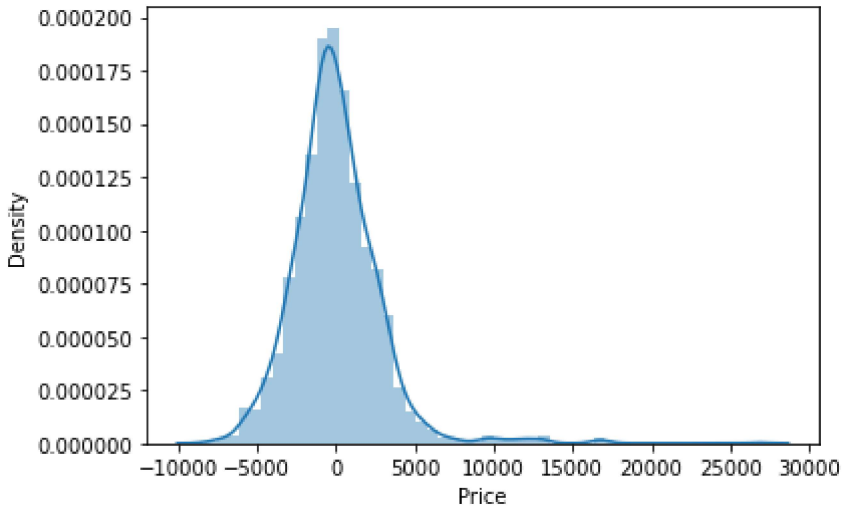
In [129]:
```
predict(DecisionTreeRegressor(),0)
```

```
Training score : 0.9689537258948039
predictions are:
 [ 6093. 13759.  9794. ... 13885. 15129. 12819.]


r2 score: 0.6870793526426198
MAE: 1373.4253080642645
MSE: 6048381.385793817
RMSE: 2459.3457231129214
```



In [130]:
```
predict(LinearRegression(),0)
```

```
Training score : 0.618480388062921
predictions are:
 [ 6997.52539965 10971.51788072 11782.16021287 ...  5226.84165045
 15149.01072668 14718.25246529]


r2 score: 0.6041088652729711
MAE: 1956.7866378850092
MSE: 7652101.548125215
RMSE: 2766.243219264209
```



In [144]:
```
reg_rf = RandomForestRegressor()
```

Hyperparameter Tuning

1.Choose following method for hyperparameter tuning
    a.RandomizedSearchCV --> Fast way to Hypertune model
    b.GridSearchCV--> Slow way to hypertune my model

2.Assign hyperparameters in form of dictionary
3.Fit the model
4.Check best paramters and best score

In [139]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [140]:
```
# Number of trees in random forest
n_estimators=[int(x) for x in np.linspace(start=100,stop=1200,num=6)]

# Number of features to consider at every split
max_features=['auto','sqrt']

# Maximum number of levels in tree
max_depth=[int(x) for x in np.linspace(5,30,num=4)]

# Minimum number of samples required to split a node
min_samples_split=[5,10,15,100]
```

In [141]:
```
# Create the random grid

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

In [142]:
```
random_grid
```

Out[142]:
```
{'n_estimators': [100, 320, 540, 760, 980, 1200],
 'max_features': ['auto', 'sqrt'],
 'max_depth': [5, 13, 21, 30],
 'min_samples_split': [5, 10, 15, 100]}
```

In [145]:
```
# Random search of parameters, using 3 fold cross validation

rf_random=RandomizedSearchCV(estimator=reg_rf,param_distributions=random_grid,cv=
```

In [146]:
```
rf_random.fit(X_train,y_train)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

Out[146]:
```
RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(), n_jobs=-1,
                   param_distributions={'max_depth': [5, 13, 21, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_split': [5, 10, 15, 100],
                                        'n_estimators': [100, 320, 540, 760,
                                                         980, 1200]},
                   verbose=2)
```

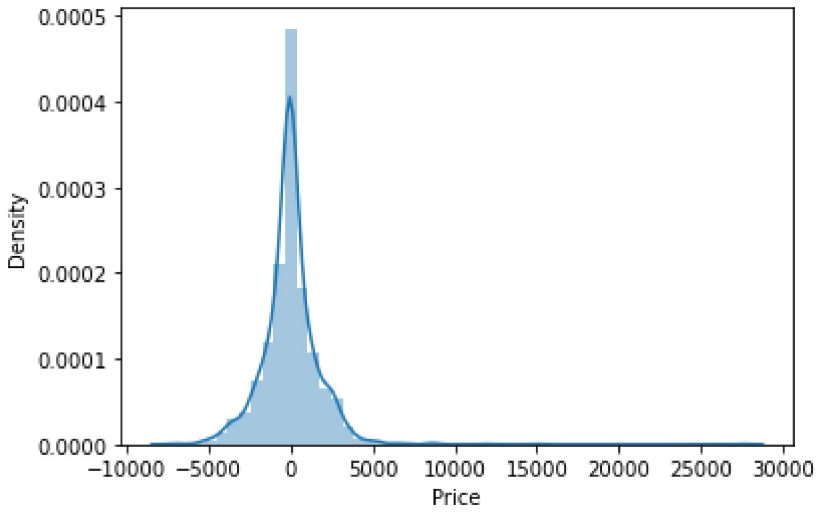
In [147]:
```
rf_random.best_params_
```

Out[147]:
```
{'n_estimators': 320,
 'min_samples_split': 10,
 'max_features': 'auto',
 'max_depth': 13}
```

In [148]:
```
prediction=rf_random.predict(X_test)
```

In [149]:
```
sns.distplot(y_test-prediction)
```

Out[149]:
```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```



In [150]:
```
metrics.r2_score(y_test,prediction)
```

Out[150]:
```
0.8411538669119496
```

In [151]:
```
print('MAE',metrics.mean_absolute_error(y_test,prediction))
print('MSE',metrics.mean_squared_error(y_test,prediction))
print('RMSE',np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE 1121.8618413315507
MSE 3070305.5317338174
RMSE 1752.2287327098072
```

In [ ]:
```

```

Save the model to reuse it again

In [152]:
```
!pip install pickle
```

```
ERROR: Could not find a version that satisfies the requirement pickle (from ver
sions: none)
ERROR: No matching distribution found for pickle
```

In [153]:
```
import pickle
```

In [154]:
```
# open a file, where you want to store the data
file=open('rf_random.pkl','wb')
```

In [155]:
```
# dump information to that file
pickle.dump(rf_random,file)
```

In [156]:
```
model=open('rf_random.pkl','rb')
forest=pickle.load(model)
```

In [157]:
```
y_prediction=forest.predict(X_test)
```

In [158]:
```
y_prediction
```

Out[158]:
```
array([ 6461.14354316, 12560.21076477, 10281.04823968, ...,
        7227.53439089, 15218.9952264 , 13519.28845402])
```

In [159]:
```
metrics.r2_score(y_test,y_prediction)
```

Out[159]:
```
0.8411538669119496
```