```
# flight fare prediction project
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
```

```
data=pd.read_excel(r"C:\Users\91967\Desktop\downloads\
flightfile.xlsx")
# The main advantage of using excel file and pandas library here is
that while loading data
#pandas allows or gives us a lot of parameters which are very helful
in loading the data
#most common among them is sheet_name which allows us to read a
particular sheet if our file has multiple sheets
#in a single excel file.
```

```
data
```

```
           Airline Date_of_Journey    Source Destination  \
0           IndiGo      24/03/2019  Banglore   New Delhi
1        Air India       1/05/2019   Kolkata    Banglore
2      Jet Airways       9/06/2019     Delhi      Cochin
3           IndiGo      12/05/2019   Kolkata    Banglore
4           IndiGo      01/03/2019  Banglore   New Delhi
...            ...             ...       ...         ...
10678     Air Asia       9/04/2019   Kolkata    Banglore
10679    Air India      27/04/2019   Kolkata    Banglore
10680  Jet Airways      27/04/2019  Banglore       Delhi
10681      Vistara      01/03/2019  Banglore   New Delhi
10682    Air India       9/05/2019     Delhi      Cochin

                       Route Dep_Time  Arrival_Time Duration
Total_Stops  \
0                  BLR → DEL    22:20  01:10 22 Mar   2h 50m    non-
stop
1      CCU → IXR → BBI → BLR    05:50         13:15   7h 25m     2
stops
2      DEL → LKO → BOM → COK    09:25  04:25 10 Jun      19h     2
stops
3            CCU → NAG → BLR    18:05         23:30   5h 25m      1
stop
4            BLR → NAG → DEL    16:50         21:35   4h 45m      1
stop
...                      ...      ...           ...      ...         .
..
10678              CCU → BLR    19:55         22:25   2h 30m    non-
stop
10679              CCU → BLR    20:45         23:20   2h 35m    non-
stop
10680              BLR → DEL    08:20         11:20       3h    non-
stop
```

```
10681              BLR → DEL    11:30         14:10   2h 40m    non-
stop
10682  DEL → GOI → BOM → COK    10:55         19:15   8h 20m     2
stops

      Additional_Info  Price
0             No info   3897
1             No info   7662
2             No info  13882
3             No info   6218
4             No info  13302
...               ...    ...
10678         No info   4107
10679         No info   4145
10680         No info   7229
10681         No info  12648
10682         No info  11753

[10683 rows x 11 columns]
```

```
data.shape
```

```
(10683, 11)
```

1. After loading the dataset we need to apply a lot of things on that.
2. First of all we will start with the very basic steps which is preprocessing i.e.

   (A) Checking the format of the data.

   (B) Check for the null values in the data which also hold a good portion of the data.

   (C) Check for the solution of the null values i.e. whether to delete them or to replace them with any other values.

   (D) Describing the data which can give us statistical analysis.

1. In the above data we can see that Price(column) is the only dependent column in our dataset rest columns are the indepedent ones.
2. From above data we came to understand that we have to do a lot of feature engineering in our data i.e. we have a lot of columns that holds string values and we have to convert them into machine learning understandable form.

```
# if or dataset contains hidden columns then we have to use
# pd.set_option("display.max_columns",None)
```

```
data.head()
```

```
       Airline Date_of_Journey    Source Destination                  
Route  \
0       IndiGo      24/03/2019  Banglore   New Delhi              BLR 
→ DEL   
1    Air India       1/05/2019   Kolkata    Banglore  CCU → IXR → BBI 
→ BLR   
2  Jet Airways       9/06/2019     Delhi      Cochin  DEL → LKO → BOM 
→ COK   
```

```
3       IndiGo      12/05/2019   Kolkata    Banglore        CCU → NAG
→ BLR
4       IndiGo      01/03/2019  Banglore   New Delhi        BLR → NAG
→ DEL

  Dep_Time  Arrival_Time Duration Total_Stops Additional_Info  Price
0    22:20  01:10 22 Mar   2h 50m    non-stop         No info   3897
1    05:50         13:15   7h 25m     2 stops         No info   7662
2    09:25  04:25 10 Jun      19h     2 stops         No info  13882
3    18:05         23:30   5h 25m      1 stop         No info   6218
4    16:50         21:35   4h 45m      1 stop         No info  13302
```

```
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
```

As now we are able to see that our data contains a lot of string values so we have to convert them into machine learning form and that process is called as preprocessing or feature engineering.

```
data.isnull().sum()
```

```
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
```

```
data.shape
```

```
(10683, 11)
```

Now we will drop null values in our data as we have checked before that there are very less number of null values we have So instead of finding an alternative solution to the null values we will just simply drop them .

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

As we have used the dropna command so we will again check the shape of our data to see what changes in our data comes after deleting the null values and if the effect is not good we can also revert the dropna command. We can do that by using the below command. data = data.reset_index(drop=True)

```
data.shape
```

```
(10682, 11)
```

As we are able to see that there is no such a big difference so we will keep them deleted

```
EDA
```

As we have done preprocessing earlier above so now we will do EDA which is exploratory data analysis i.e. With performing this we make or data readable for machine learning as we are seeing that or data has a lot of string values . So we will convert these values into machine readable form.

```
data.head()
```

```
       Airline Date_of_Journey    Source Destination                  
Route  \
0       IndiGo      24/03/2019  Banglore   New Delhi              BLR 
→ DEL   
1    Air India       1/05/2019   Kolkata    Banglore  CCU → IXR → BBI 
→ BLR   
2  Jet Airways       9/06/2019     Delhi      Cochin  DEL → LKO → BOM 
→ COK   
3       IndiGo      12/05/2019   Kolkata    Banglore        CCU → NAG 
→ BLR   
4       IndiGo      01/03/2019  Banglore   New Delhi        BLR → NAG 
→ DEL   

  Dep_Time  Arrival_Time Duration Total_Stops Additional_Info  Price  
0    22:20  01:10 22 Mar   2h 50m    non-stop         No info   3897  
1    05:50         13:15   7h 25m     2 stops         No info   7662  
2    09:25  04:25 10 Jun      19h     2 stops         No info  13882  
3    18:05         23:30   5h 25m      1 stop         No info   6218  
4    16:50         21:35   4h 45m      1 stop         No info  13302  
```

As now we will see that column"date_of_journey" is not readable as it is in string form so we will convert this into machine readable form.

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

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

As we have tried to make the data of Date_of_Journey column to be readable for machine learning so we will now see the changes we have made in our data.

1. dt.month will only extract the month from the date whereas dt.day extract the day from the date.

```
data.head()
```

```
       Airline Date_of_Journey    Source Destination
Route  \
0       IndiGo      24/03/2019  Banglore   New Delhi              BLR
→ DEL
1    Air India       1/05/2019   Kolkata    Banglore  CCU → IXR → BBI
→ BLR
2  Jet Airways       9/06/2019     Delhi      Cochin  DEL → LKO → BOM
→ COK
3       IndiGo      12/05/2019   Kolkata    Banglore        CCU → NAG
→ BLR
4       IndiGo      01/03/2019  Banglore   New Delhi        BLR → NAG
→ DEL

  Dep_Time  Arrival_Time Duration Total_Stops Additional_Info
Price  \
0    22:20  01:10 22 Mar   2h 50m    non-stop         No info   3897

1    05:50         13:15   7h 25m     2 stops         No info   7662

2    09:25  04:25 10 Jun      19h     2 stops         No info  13882

3    18:05         23:30   5h 25m      1 stop         No info   6218

4    16:50         21:35   4h 45m      1 stop         No info  13302


   journey_day  journey_month
0           24              3
1            1              5
2            9              6
3           12              5
4            1              3
```

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

```
data.head()
```

```
       Airline    Source Destination                    Route 
Dep_Time  \
0       IndiGo  Banglore   New Delhi              BLR → DEL    22:20   

1    Air India   Kolkata    Banglore  CCU → IXR → BBI → BLR    05:50   

2  Jet Airways     Delhi      Cochin  DEL → LKO → BOM → COK    09:25   

3       IndiGo   Kolkata    Banglore        CCU → NAG → BLR    18:05   

4       IndiGo  Banglore   New Delhi        BLR → NAG → DEL    16:50   


   Arrival_Time Duration Total_Stops Additional_Info  Price 
journey_day  \
0  01:10 22 Mar   2h 50m    non-stop         No info   3897         
24   
1         13:15   7h 25m     2 stops         No info   7662          
1   
2  04:25 10 Jun      19h     2 stops         No info  13882          
9   
3         23:30   5h 25m      1 stop         No info   6218         
12   
4         21:35   4h 45m      1 stop         No info  13302          
1   

   journey_month  
0              3  
1              5  
2              6  
3              5  
4              3  
```

Same like the Date_of_Journey column that we have dropped we can also use the same for the Dep_time and we can extract the hour and minute of thr departure time.

```
data["dep_hour"]=pd.to_datetime(data.Dep_Time).dt.hour

data["dep_min"]=pd.to_datetime(data.Dep_Time).dt.minute

data.head()
```

```
       Airline    Source Destination                    Route 
Dep_Time  \
0       IndiGo  Banglore   New Delhi              BLR → DEL    22:20   

1    Air India   Kolkata    Banglore  CCU → IXR → BBI → BLR    05:50   

2  Jet Airways     Delhi      Cochin  DEL → LKO → BOM → COK    09:25   
```

```
3       IndiGo   Kolkata    Banglore        CCU → NAG → BLR    18:05

4       IndiGo  Banglore   New Delhi        BLR → NAG → DEL    16:50


   Arrival_Time Duration Total_Stops Additional_Info  Price
journey_day  \
0  01:10 22 Mar   2h 50m    non-stop         No info   3897
24
1         13:15   7h 25m     2 stops         No info   7662
1
2  04:25 10 Jun      19h     2 stops         No info  13882
9
3         23:30   5h 25m      1 stop         No info   6218
12
4         21:35   4h 45m      1 stop         No info  13302
1

   journey_month  dep_hour  dep_min
0              3        22       20
1              5         5       50
2              6         9       25
3              5        18        5
4              3        16       50
```

As now we have also transformed the column into machine readable form so now we will delete that column.

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

```
data.head()
```

```
       Airline    Source Destination                  Route
Arrival_Time  \
0       IndiGo  Banglore   New Delhi              BLR → DEL  01:10 22
Mar
1    Air India   Kolkata    Banglore  CCU → IXR → BBI → BLR
13:15
2  Jet Airways     Delhi      Cochin  DEL → LKO → BOM → COK  04:25 10
Jun
3       IndiGo   Kolkata    Banglore        CCU → NAG → BLR
23:30
4       IndiGo  Banglore   New Delhi        BLR → NAG → DEL
21:35

  Duration Total_Stops Additional_Info  Price  journey_day
journey_month  \
0   2h 50m    non-stop         No info   3897           24
3
1   7h 25m     2 stops         No info   7662            1
```

```
5
2      19h     2 stops          No info  13882            9
6
3   5h 25m      1 stop          No info   6218           12
5
4   4h 45m      1 stop          No info  13302            1
3

   dep_hour  dep_min
0        22       20
1         5       50
2         9       25
3        18        5
4        16       50
```

```python
# Now we will perform the same process on column Arrival_Time

data["arrival_hour"]=pd.to_datetime(data.Arrival_Time).dt.hour

data["arrival_minute"]=pd.to_datetime(data.Arrival_Time).dt.minute

data.head()
```

```
       Airline    Source Destination                  Route  
Arrival_Time  \
0       IndiGo  Banglore   New Delhi              BLR → DEL  01:10 22
Mar
1    Air India   Kolkata    Banglore  CCU → IXR → BBI → BLR
13:15
2  Jet Airways     Delhi      Cochin  DEL → LKO → BOM → COK  04:25 10
Jun
3       IndiGo   Kolkata    Banglore        CCU → NAG → BLR
23:30
4       IndiGo  Banglore   New Delhi        BLR → NAG → DEL
21:35

  Duration Total_Stops Additional_Info  Price  journey_day
journey_month  \
0   2h 50m    non-stop         No info   3897           24
3
1   7h 25m     2 stops         No info   7662            1
5
2      19h     2 stops         No info  13882            9
6
3   5h 25m      1 stop         No info   6218           12
5
4   4h 45m      1 stop         No info  13302            1
3

   dep_hour  dep_min  arrival_hour  arrival_minute
0        22       20             1              10
```

```
1         5       50            13              15
2         9       25             4              25
3        18        5            23              30
4        16       50            21              35
```

```
data.drop(["Arrival_Time"],axis=1,inplace=True)
```

```
data.head()
```

```
       Airline    Source Destination                  Route 
Duration  \
0       IndiGo  Banglore   New Delhi              BLR → DEL   2h 50m

1    Air India   Kolkata    Banglore  CCU → IXR → BBI → BLR   7h 25m

2  Jet Airways     Delhi      Cochin  DEL → LKO → BOM → COK      19h

3       IndiGo   Kolkata    Banglore        CCU → NAG → BLR   5h 25m

4       IndiGo  Banglore   New Delhi        BLR → NAG → DEL   4h 45m


  Total_Stops Additional_Info  Price  journey_day  journey_month 
dep_hour  \
0    non-stop         No info   3897           24              3 
22
1     2 stops         No info   7662            1              5 
5
2     2 stops         No info  13882            9              6 
9
3      1 stop         No info   6218           12              5 
18
4      1 stop         No info  13302            1              3 
16

   dep_min  arrival_hour  arrival_minute
0       20             1              10
1       50            13              15
2       25             4              25
3        5            23              30
4       50            21              35
```

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

```
No info                         8344
In-flight meal not included     1982
No check-in baggage included     320
1 Long layover                    19
Change airports                    7
Business class                     4
No Info                            3
```

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

Duration is the time in which a passenger travelled from one place to another. Also we have the duration column in categorical form but it is not in the pure categorical form i.e. it can be spliteed into two parts by using length function and we can create two columns from "Duration" column also as we have done earlier.

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

for i in range(len(duration)):
    if len(duration[i].split()) != 2:
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"
        else:
            duration[i] = "0h " + duration[i]
```

```
----------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in
get_loc(self, key, method, tolerance)
   3360             try:
-> 3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:

~\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in
pandas._libs.index.IndexEngine.get_loc()

~\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in
pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Duration'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_10876/1413142309.py in <module>
----> 1 duration = list(data["Duration"])
      2 
      3 for i in range(len(duration)):
      4     if len(duration[i].split()) != 2:
      5         if "h" in duration[i]:

~\Anaconda3\lib\site-packages\pandas\core\frame.py in
__getitem__(self, key)
   3456             if self.columns.nlevels > 1:
   3457                 return self._getitem_multilevel(key)
-> 3458             indexer = self.columns.get_loc(key)
   3459             if is_integer(indexer):
   3460                 indexer = [indexer]

~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in
get_loc(self, key, method, tolerance)
   3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:
-> 3363                 raise KeyError(key) from err
   3364 
   3365         if is_scalar(key) and isna(key) and not self.hasnans:

KeyError: 'Duration'
```

```
duration_hour = []
duration_min = []
for i in range(len(duration)):
    duration_hour.append(int(duration[i].split(sep = "h")[0]))
    duration_min.append(int(duration[i].split(sep = "m")[0].split()[-
1]))
```

```
data["Duration_hours"]=duration_hour
data["Duration_mins"]=duration_min
```

```
data
```

```
           Airline    Source Destination                  Route
Total_Stops  \
0            IndiGo  Banglore   New Delhi              BLR → DEL
non-stop   
1         Air India   Kolkata    Banglore  CCU → IXR → BBI → BLR     2
stops   
2       Jet Airways     Delhi      Cochin  DEL → LKO → BOM → COK     2
stops   
3            IndiGo   Kolkata    Banglore        CCU → NAG → BLR      1
stop   
4            IndiGo  Banglore   New Delhi        BLR → NAG → DEL      1
stop   
...             ...       ...         ...                    ...
```

```
...
10678     Air Asia   Kolkata    Banglore              CCU → BLR
non-stop
10679    Air India   Kolkata    Banglore              CCU → BLR
non-stop
10680  Jet Airways  Banglore       Delhi              BLR → DEL
non-stop
10681      Vistara  Banglore   New Delhi              BLR → DEL
non-stop
10682    Air India     Delhi      Cochin  DEL → GOI → BOM → COK     2
stops

      Additional_Info  Price  journey_day  journey_month  dep_hour
dep_min  \
0             No info   3897           24              3        22
20
1             No info   7662            1              5         5
50
2             No info  13882            9              6         9
25
3             No info   6218           12              5        18
5
4             No info  13302            1              3        16
50
...               ...    ...          ...            ...       ...
...
10678         No info   4107            9              4        19
55
10679         No info   4145           27              4        20
45
10680         No info   7229           27              4         8
20
10681         No info  12648            1              3        11
30
10682         No info  11753            9              5        10
55

       arrival_hour  arrival_minute  Duration_hours  Duration_mins
0                 1              10               2             50
1                13              15               7             25
2                 4              25              19              0
3                23              30               5             25
4                21              35               4             45
...             ...             ...             ...            ...
10678            22              25               2             30
10679            23              20               2             35
10680            11              20               3              0
10681            14              10               2             40
10682            19              15               8             20
```

```
[10682 rows x 15 columns]
```

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

```
----------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_10876/2316766621.py in <module>
----> 1 data.drop(["Duration"], axis = 1, inplace = True)

~\Anaconda3\lib\site-packages\pandas\util\_decorators.py in
wrapper(*args, **kwargs)
    309                     stacklevel=stacklevel,
    310                 )
--> 311             return func(*args, **kwargs)
    312 
    313         return wrapper

~\Anaconda3\lib\site-packages\pandas\core\frame.py in drop(self,
labels, axis, index, columns, level, inplace, errors)
   4904                 weight  1.0     0.8
   4905         """
-> 4906         return super().drop(
   4907             labels=labels,
   4908             axis=axis,

~\Anaconda3\lib\site-packages\pandas\core\generic.py in drop(self,
labels, axis, index, columns, level, inplace, errors)
   4148         for axis, labels in axes.items():
   4149             if labels is not None:
-> 4150                 obj = obj._drop_axis(labels, axis,
level=level, errors=errors)
   4151 
   4152         if inplace:

~\Anaconda3\lib\site-packages\pandas\core\generic.py in
_drop_axis(self, labels, axis, level, errors)
   4183                 new_axis = axis.drop(labels, level=level,
errors=errors)
   4184             else:
-> 4185                 new_axis = axis.drop(labels, errors=errors)
   4186             result = self.reindex(**{axis_name: new_axis})
   4187 

~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in
drop(self, labels, errors)
   6015         if mask.any():
   6016             if errors != "ignore":
```

```
-> 6017                 raise KeyError(f"{labels[mask]} not found in
axis")
   6018             indexer = indexer[~mask]
   6019         return self.delete(indexer)

KeyError: "['Duration'] not found in axis"
```

```
data.head()
```

```
       Airline    Source Destination                  Route
Total_Stops  \
0       IndiGo  Banglore   New Delhi              BLR → DEL    non-
stop
1    Air India   Kolkata    Banglore  CCU → IXR → BBI → BLR     2
stops
2  Jet Airways     Delhi      Cochin  DEL → LKO → BOM → COK     2
stops
3       IndiGo   Kolkata    Banglore        CCU → NAG → BLR      1
stop
4       IndiGo  Banglore   New Delhi        BLR → NAG → DEL      1
stop

  Additional_Info  Price  journey_day  journey_month  dep_hour
dep_min  \
0         No info   3897           24              3        22
20
1         No info   7662            1              5         5
50
2         No info  13882            9              6         9
25
3         No info   6218           12              5        18
5
4         No info  13302            1              3        16
50

   arrival_hour  arrival_minute  Duration_hours  Duration_mins
0             1              10               2             50
1            13              15               7             25
2             4              25              19              0
3            23              30               5             25
4            21              35               4             45
```

```
data.drop(["Duration_hour","Duration_min"], axis=1 , inplace=True)
```

```
----------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_10876/1461846534.py in <module>
----> 1 data.drop(["Duration_hour","Duration_min"], axis=1 ,
inplace=True)
```

```
~\Anaconda3\lib\site-packages\pandas\util\_decorators.py in
wrapper(*args, **kwargs)
    309                     stacklevel=stacklevel,
    310                 )
--> 311             return func(*args, **kwargs)
    312 
    313         return wrapper

~\Anaconda3\lib\site-packages\pandas\core\frame.py in drop(self,
labels, axis, index, columns, level, inplace, errors)
   4904                 weight  1.0     0.8
   4905         """
-> 4906         return super().drop(
   4907             labels=labels,
   4908             axis=axis,

~\Anaconda3\lib\site-packages\pandas\core\generic.py in drop(self,
labels, axis, index, columns, level, inplace, errors)
   4148         for axis, labels in axes.items():
   4149             if labels is not None:
-> 4150                 obj = obj._drop_axis(labels, axis,
level=level, errors=errors)
   4151 
   4152         if inplace:

~\Anaconda3\lib\site-packages\pandas\core\generic.py in
_drop_axis(self, labels, axis, level, errors)
   4183                 new_axis = axis.drop(labels, level=level,
errors=errors)
   4184             else:
-> 4185                 new_axis = axis.drop(labels, errors=errors)
   4186             result = self.reindex(**{axis_name: new_axis})
   4187 

~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in
drop(self, labels, errors)
   6015         if mask.any():
   6016             if errors != "ignore":
-> 6017                 raise KeyError(f"{labels[mask]} not found in
axis")
   6018             indexer = indexer[~mask]
   6019         return self.delete(indexer)

KeyError: "['Duration_hour' 'Duration_min'] not found in axis"
```

```
data.head()
```

```
       Airline    Source Destination                  Route
Total_Stops  \
0       IndiGo  Banglore   New Delhi              BLR → DEL    non-
stop
1    Air India   Kolkata    Banglore  CCU → IXR → BBI → BLR     2
stops
2  Jet Airways     Delhi      Cochin  DEL → LKO → BOM → COK     2
stops
3       IndiGo   Kolkata    Banglore        CCU → NAG → BLR      1
stop
4       IndiGo  Banglore   New Delhi        BLR → NAG → DEL      1
stop

  Additional_Info  Price  journey_day  journey_month  dep_hour
dep_min  \
0         No info   3897           24              3        22
20
1         No info   7662            1              5         5
50
2         No info  13882            9              6         9
25
3         No info   6218           12              5        18
5
4         No info  13302            1              3        16
50

   arrival_hour  arrival_minute  Duration_hours  Duration_mins
0             1              10               2             50
1            13              15               7             25
2             4              25              19              0
3            23              30               5             25
4            21              35               4             45
```

I have run the above code twice that's why it is showing errors as once i have deleted those columns and again if i want to fetch them then it is not possible.

Now we will try to handle categorical data that we hsve on our dataset now. categorical data is of 2 types :

1. Nominal data and Ordinal data

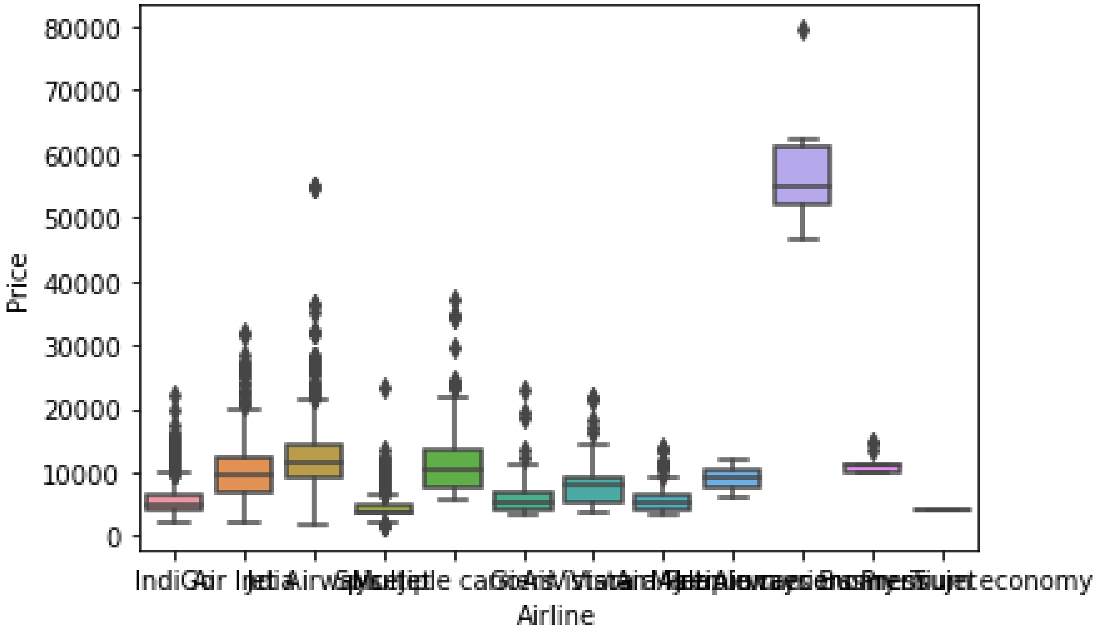
Now we have the column "Airline" and now we will try to perform operations on it

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

```
Jet Airways                          3849
IndiGo                               2053
Air India                            1751
Multiple carriers                    1196
SpiceJet                              818
Vistara                               479
Air Asia                              319
GoAir                                 194
```

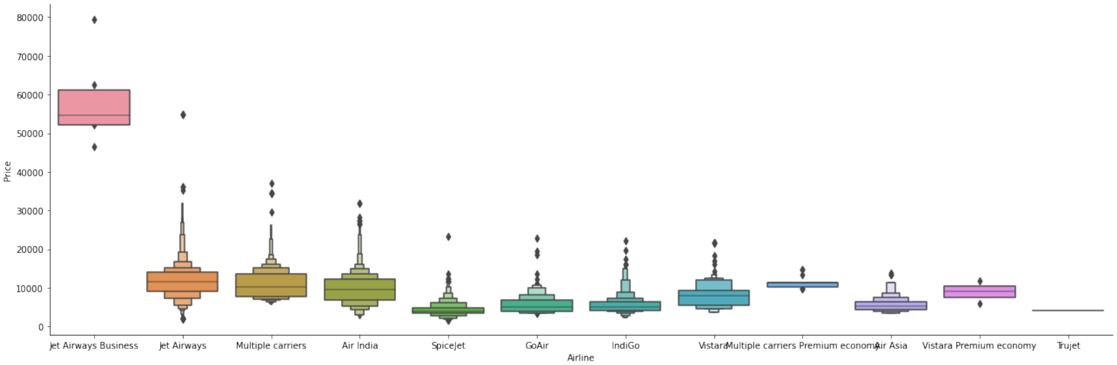
```
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
```

```
sb.boxplot(x="Airline",y="Price",data=data,width=0.8,saturation=0.75)
plt.show()
```



As we are able to see that with this plot we are not able to see or understand things properly so we will use here catplot

```
sb.catplot(y = "Price", x = "Airline", data =
data.sort_values("Price", ascending = False), kind="boxen", height =
6,aspect=3)
plt.show()
```



As our airline column is an example of nominal data so we will use onehot encoding technique and try to convert it into machine learning form

```
Airline = data[["Airline"]]

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

# to check the above command we will use
Airline.head()
```

```
   Airline_Air India  Airline_GoAir  Airline_IndiGo  Airline_Jet
Airways  \
0                  0              0               1
0
1                  1              0               0
0
2                  0              0               0
1
3                  0              0               1
0
4                  0              0               1
0

   Airline_Jet Airways Business  Airline_Multiple carriers  \
0                             0                          0
1                             0                          0
2                             0                          0
3                             0                          0
4                             0                          0

   Airline_Multiple carriers Premium economy  Airline_SpiceJet  \
0                                          0                 0
1                                          0                 0
2                                          0                 0
3                                          0                 0
4                                          0                 0

   Airline_Trujet  Airline_Vistara  Airline_Vistara Premium economy
0               0                0                                0
1               0                0                                0
2               0                0                                0
3               0                0                                0
4               0                0                                0
```

In the above cell the value is 1 where the same airline matches with the column value

```
data.head()
```

```
       Airline    Source Destination                  Route
Total_Stops  \
0       IndiGo  Banglore   New Delhi              BLR → DEL    non-
stop
1    Air India   Kolkata    Banglore  CCU → IXR → BBI → BLR     2
stops
```

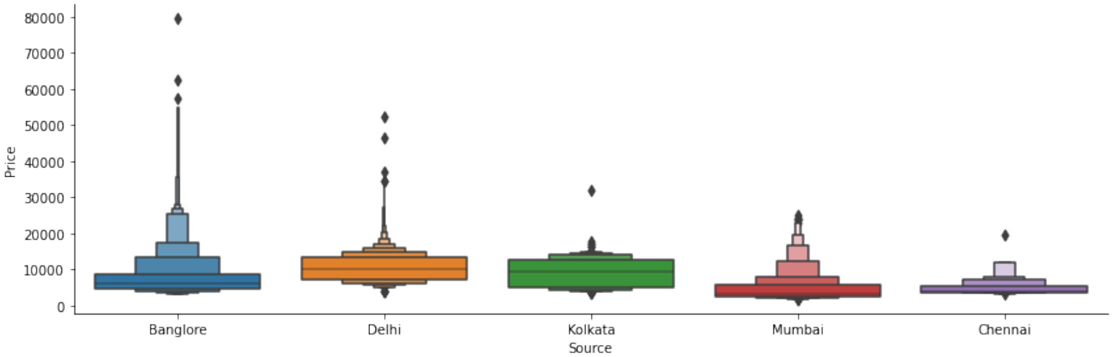
```
2  Jet Airways     Delhi      Cochin  DEL → LKO → BOM → COK     2
stops
3      IndiGo   Kolkata    Banglore        CCU → NAG → BLR      1
stop
4      IndiGo  Banglore   New Delhi        BLR → NAG → DEL      1
stop

  Additional_Info  Price  journey_day  journey_month  dep_hour
dep_min  \
0         No info   3897           24              3        22
20
1         No info   7662            1              5         5
50
2         No info  13882            9              6         9
25
3         No info   6218           12              5        18
5
4         No info  13302            1              3        16
50

   arrival_hour  arrival_minute  Duration_hours  Duration_mins
0             1              10               2             50
1            13              15               7             25
2             4              25              19              0
3            23              30               5             25
4            21              35               4             45
```

Now we will do the same with the "Source" column

```
sb.catplot(y = "Price", x = "Source", data = data.sort_values("Price",
ascending = False), kind="boxen", height = 4, aspect = 3)
plt.show()
```



```
Source = data[["Source"]]

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

Source.head()
```

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

Now we will perform same for "Destination" column

```
Destination = data[["Destination"]]

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

Destination.head()
```

```
   Destination_Cochin  Destination_Delhi  Destination_Hyderabad  \
0                   0                  0                      0
1                   0                  0                      0
2                   1                  0                      0
3                   0                  0                      0
4                   0                  0                      0

   Destination_Kolkata  Destination_New Delhi
0                    0                      1
1                    0                      0
2                    0                      0
3                    0                      0
4                    0                      1
```

```
# now we will drop extra columns which are of less use or no use or
have a single information or have repeatitive info such
# as additional info and routes
data.drop(["Route", "Additional_Info"], axis = 1, inplace = True)

data.head()
```

```
       Airline    Source Destination Total_Stops  Price
journey_day  \
0       IndiGo  Banglore   New Delhi    non-stop   3897           24

1    Air India   Kolkata    Banglore     2 stops   7662            1

2  Jet Airways     Delhi      Cochin     2 stops  13882            9

3       IndiGo   Kolkata    Banglore      1 stop   6218           12

4       IndiGo  Banglore   New Delhi      1 stop  13302            1


   journey_month  dep_hour  dep_min  arrival_hour  arrival_minute  \
0              3        22       20             1              10
```

```
1              5         5       50            13               15
2              6         9       25             4               25
3              5        18        5            23               30
4              3        16       50            21               35

   Duration_hours  Duration_mins
0               2             50
1               7             25
2              19              0
3               5             25
4               4             45
```

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

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

As we can see that the column have only 5 values so we can convert it into by using dictionary and allot them different values which are understandable for machine learning.

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

```
# now we will put all the values in our data or the columns we have
created using .
data1 = pd.concat([data, Airline, Source, Destination], axis = 1)
```

```
# And we will drop all those columns which still contains categorical
values
data1.drop(["Airline", "Source", "Destination"], axis = 1, inplace =
True)
data1.head()
```

```
----------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_10876/1936457592.py in <module>
      1 # And we will drop all those columns which still contains
categorical values
----> 2 data1.drop(["Airline", "Source", "Destination"], axis = 1,
inplace = True)
      3 data1.head()

~\Anaconda3\lib\site-packages\pandas\util\_decorators.py in
wrapper(*args, **kwargs)
```

```
    309                     stacklevel=stacklevel,
    310                 )
--> 311             return func(*args, **kwargs)
    312 
    313         return wrapper

~\Anaconda3\lib\site-packages\pandas\core\frame.py in drop(self,
labels, axis, index, columns, level, inplace, errors)
   4904                 weight  1.0     0.8
   4905         """
-> 4906         return super().drop(
   4907             labels=labels,
   4908             axis=axis,

~\Anaconda3\lib\site-packages\pandas\core\generic.py in drop(self,
labels, axis, index, columns, level, inplace, errors)
   4148         for axis, labels in axes.items():
   4149             if labels is not None:
-> 4150                 obj = obj._drop_axis(labels, axis,
level=level, errors=errors)
   4151 
   4152         if inplace:

~\Anaconda3\lib\site-packages\pandas\core\generic.py in
_drop_axis(self, labels, axis, level, errors)
   4212                 labels_missing = (axis.get_indexer_for(labels)
== -1).any()
   4213                 if errors == "raise" and labels_missing:
-> 4214                     raise KeyError(f"{labels} not found in
axis")
   4215 
   4216             slicer = [slice(None)] * self.ndim

KeyError: "['Airline' 'Source' 'Destination'] not found in axis"
```

As now we can see that during the time we have uploaded our data we have only 11 columns and now we have 30 columns which makes or data more readable for machine learning

Now i have taken 2 datasets of same type i.e. which have same number of rows in the starting and only the difference is that of 1 column which is Price column. I have done that to prevent data leakage because if we keep the same data and delete column Price then that will not be good for our model fitting and same data can effect our model fitting. so i have taken 2 datasets.

```
test_data=pd.read_excel(r"C:\Users\91967\Desktop\downloads\
test_dataset.xlsx")
```

```
test_data.head()
```

```
             Airline Date_of_Journey    Source Destination
Route  \
0        Jet Airways       6/06/2019     Delhi      Cochin  DEL → BOM
→ COK
1             IndiGo      12/05/2019   Kolkata    Banglore  CCU → MAA
→ BLR
2        Jet Airways      21/05/2019     Delhi      Cochin  DEL → BOM
→ COK
3  Multiple carriers      21/05/2019     Delhi      Cochin  DEL → BOM
→ COK
4           Air Asia      24/06/2019  Banglore       Delhi        BLR
→ DEL

  Dep_Time  Arrival_Time Duration Total_Stops
Additional_Info
0    17:30  04:25 07 Jun  10h 55m      1 stop                     No
info
1    06:20         10:20       4h      1 stop                     No
info
2    19:15  19:00 22 May  23h 45m      1 stop  In-flight meal not
included
3    08:00         21:00      13h      1 stop                     No
info
4    23:55  02:45 25 Jun   2h 50m    non-stop                     No
info
```

```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2671 entries, 0 to 2670
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          2671 non-null   object
 1   Date_of_Journey  2671 non-null   object
 2   Source           2671 non-null   object
 3   Destination      2671 non-null   object
 4   Route            2671 non-null   object
 5   Dep_Time         2671 non-null   object
 6   Arrival_Time     2671 non-null   object
 7   Duration         2671 non-null   object
 8   Total_Stops      2671 non-null   object
 9   Additional_Info  2671 non-null   object
dtypes: object(10)
memory usage: 208.8+ KB
```

Now we will see that Price column is missing here so that we can predict price after the completion of our model.now we will perform all the functions we have done in the above data and we can do it in a single attempt to make our path clear to train our model.

```
# Date_of_Journey
test_data["Journey_day"] = pd.to_datetime(test_data.Date_of_Journey,
format="%d/%m/%Y").dt.day
test_data["Journey_month"] =
pd.to_datetime(test_data["Date_of_Journey"], format =
"%d/%m/%Y").dt.month
test_data.drop(["Date_of_Journey"], axis = 1, inplace = True)
test_data["Dep_hour"] = pd.to_datetime(test_data["Dep_Time"]).dt.hour
test_data["Dep_min"] = pd.to_datetime(test_data["Dep_Time"]).dt.minute
test_data.drop(["Dep_Time"], axis = 1, inplace = True)
test_data["Arrival_hour"] =
pd.to_datetime(test_data.Arrival_Time).dt.hour
test_data["Arrival_min"] =
pd.to_datetime(test_data.Arrival_Time).dt.minute
test_data.drop(["Arrival_Time"], axis = 1, inplace = True)
duration = list(test_data["Duration"])
for i in range(len(duration)):
    if len(duration[i].split()) != 2:
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"
        else:
            duration[i] = "0h " + duration[i]
duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-
1]))
test_data["Duration_hours"] = duration_hours
test_data["Duration_mins"] = duration_mins
test_data.drop(["Duration"], axis = 1, inplace = True)
#Categorical Data
print("Airline")
print("-"*75)
print(test_data["Airline"].value_counts())
Airline = pd.get_dummies(test_data["Airline"], drop_first= True)
print()
print("Source")
print("-"*75)
print(test_data["Source"].value_counts())
Source = pd.get_dummies(test_data["Source"], drop_first= True)
print()
print("Destination")
print("-"*75)
print(test_data["Destination"].value_counts())
Destination = pd.get_dummies(test_data["Destination"], drop_first =
True)
test_data.drop(["Route", "Additional_Info"], axis = 1, inplace = True)
test_data.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3
stops": 3, "4 stops": 4}, inplace = True)
```

```
data_test = pd.concat([test_data, Airline, Source, Destination], axis
= 1)
data_test.drop(["Airline", "Source", "Destination"], axis = 1, inplace
= True)
print()
print()
print("Shape of test data : ", data_test.shape)
```

```
Airline
----------------------------------------------------------------------
-----
Jet Airways                          897
IndiGo                               511
Air India                            440
Multiple carriers                    347
SpiceJet                             208
Vistara                              129
Air Asia                              86
GoAir                                 46
Multiple carriers Premium economy      3
Vistara Premium economy                2
Jet Airways Business                   2
Name: Airline, dtype: int64

Source
----------------------------------------------------------------------
-----
Delhi       1145
Kolkata      710
Banglore     555
Mumbai       186
Chennai       75
Name: Source, dtype: int64

Destination
----------------------------------------------------------------------
-----
Cochin       1145
Banglore      710
Delhi         317
New Delhi     238
Hyderabad     186
Kolkata        75
Name: Destination, dtype: int64


Shape of test data :  (2671, 28)
```

```
# i have run the above code twice that's why it is showing this error
```

```
data_test.head()
```

```
   Total_Stops  Journey_day  Journey_month  Dep_hour  Dep_min
Arrival_hour  \
0            1            6              6        17       30
4
1            1           12              5         6       20
10
2            1           21              5        19       15
19
3            1           21              5         8        0
21
4            0           24              6        23       55
2

   Arrival_min  Duration_hours  Duration_mins  Air India  ...  \
0           25              10             55          0  ...
1           20               4              0          0  ...
2            0              23             45          0  ...
3            0              13              0          0  ...
4           45               2             50          0  ...

   Vistara Premium economy  Chennai  Delhi  Kolkata  Mumbai  Cochin
Delhi  \
0                        0        0      1        0       0       1
0
1                        0        0      0        1       0       0
0
2                        0        0      1        0       0       1
0
3                        0        0      1        0       0       1
0
4                        0        0      0        0       0       0
1

   Hyderabad  Kolkata  New Delhi
0          0        0          0
1          0        0          0
2          0        0          0
3          0        0          0
4          0        0          0

[5 rows x 28 columns]
```

Now we will move towards feature selection technique

```
data_test.head()
```

```
   Total_Stops  Journey_day  Journey_month  Dep_hour  Dep_min
Arrival_hour  \
0            1            6              6        17       30
```

```
4
1            1           12               5         6        20
10
2            1           21               5        19        15
19
3            1           21               5         8         0
21
4            0           24               6        23        55
2

   Arrival_min  Duration_hours  Duration_mins  Air India  ...  \
0           25              10             55          0  ...
1           20               4              0          0  ...
2            0              23             45          0  ...
3            0              13              0          0  ...
4           45               2             50          0  ...

   Vistara Premium economy  Chennai  Delhi  Kolkata  Mumbai  Cochin
Delhi  \
0                        0        0      1        0       0       1
0
1                        0        0      0        1       0       0
0
2                        0        0      1        0       0       1
0
3                        0        0      1        0       0       1
0
4                        0        0      0        0       0       0
1

   Hyderabad  Kolkata  New Delhi
0          0        0          0
1          0        0          0
2          0        0          0
3          0        0          0
4          0        0          0

[5 rows x 28 columns]
```

```
data_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2671 entries, 0 to 2670
Data columns (total 28 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   Total_Stops                        2671 non-null   int64
 1   Journey_day                        2671 non-null   int64
 2   Journey_month                      2671 non-null   int64
 3   Dep_hour                           2671 non-null   int64
 4   Dep_min                               2671 non-null   int64
 5   Arrival_hour                          2671 non-null   int64
 6   Arrival_min                           2671 non-null   int64
 7   Duration_hours                        2671 non-null   int64
 8   Duration_mins                         2671 non-null   int64
 9   Air India                             2671 non-null   uint8
 10  GoAir                                 2671 non-null   uint8
 11  IndiGo                                2671 non-null   uint8
 12  Jet Airways                           2671 non-null   uint8
 13  Jet Airways Business                  2671 non-null   uint8
 14  Multiple carriers                     2671 non-null   uint8
 15  Multiple carriers Premium economy     2671 non-null   uint8
 16  SpiceJet                              2671 non-null   uint8
 17  Vistara                               2671 non-null   uint8
 18  Vistara Premium economy               2671 non-null   uint8
 19  Chennai                               2671 non-null   uint8
 20  Delhi                                 2671 non-null   uint8
 21  Kolkata                               2671 non-null   uint8
 22  Mumbai                                2671 non-null   uint8
 23  Cochin                                2671 non-null   uint8
 24  Delhi                                 2671 non-null   uint8
 25  Hyderabad                             2671 non-null   uint8
 26  Kolkata                               2671 non-null   uint8
 27  New Delhi                             2671 non-null   uint8
dtypes: int64(9), uint8(19)
memory usage: 237.5 KB
```

```
# now we will move towards feature selection
```

```
data_test.shape
```

```
(2671, 28)
```

```
data.columns
```

```
Index(['Total_Stops', 'Price', 'journey_day', 'journey_month',
'dep_hour',
       'dep_min', 'arrival_hour', 'arrival_minute', 'Duration_hours',
       'Duration_mins', 'Airline_Air India', 'Airline_GoAir',
'Airline_IndiGo',
       'Airline_Jet Airways', 'Airline_Jet Airways Business',
       'Airline_Multiple carriers',
       'Airline_Multiple carriers Premium economy',
'Airline_SpiceJet',
       'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium
economy',
       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata',
'Source_Mumbai',
       'Destination_Cochin', 'Destination_Delhi',
'Destination_Hyderabad',
       'Destination_Kolkata', 'Destination_New Delhi'],
      dtype='object')
```

```
X=data1.loc[:,['Total_Stops', 'Price', 'journey_day', 'journey_month',
'dep_hour',
       'dep_min', 'arrival_hour', 'arrival_minute', 'Duration_hours',
       'Duration_mins', 'Airline_Air India', 'Airline_GoAir',
'Airline_IndiGo',
       'Airline_Jet Airways', 'Airline_Jet Airways Business',
       'Airline_Multiple carriers',
       'Airline_Multiple carriers Premium economy',
'Airline_SpiceJet',
       'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium
economy',
       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata',
'Source_Mumbai',
       'Destination_Cochin', 'Destination_Delhi',
'Destination_Hyderabad',
       'Destination_Kolkata', 'Destination_New Delhi']]
X.head()
```

```
   Total_Stops  Price  journey_day  journey_month  dep_hour
dep_min  \
0            0   3897           24              3        22       20

1            2   7662            1              5         5       50

2            2  13882            9              6         9       25

3            1   6218           12              5        18        5

4            1  13302            1              3        16       50


   arrival_hour  arrival_minute  Duration_hours  Duration_mins  ...  \
0             1              10               2             50  ...
1            13              15               7             25  ...
2             4              25              19              0  ...
3            23              30               5             25  ...
4            21              35               4             45  ...

   Airline_Vistara Premium economy  Source_Chennai  Source_Delhi  \
0                                0               0             0
1                                0               0             0
2                                0               0             1
3                                0               0             0
4                                0               0             0

   Source_Kolkata  Source_Mumbai  Destination_Cochin
Destination_Delhi  \
0               0              0                   0
0
1               1              0                   0
0
2               0              0                   1
0
3               1              0                   0
0
4               0              0                   0
0

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

[5 rows x 30 columns]
```

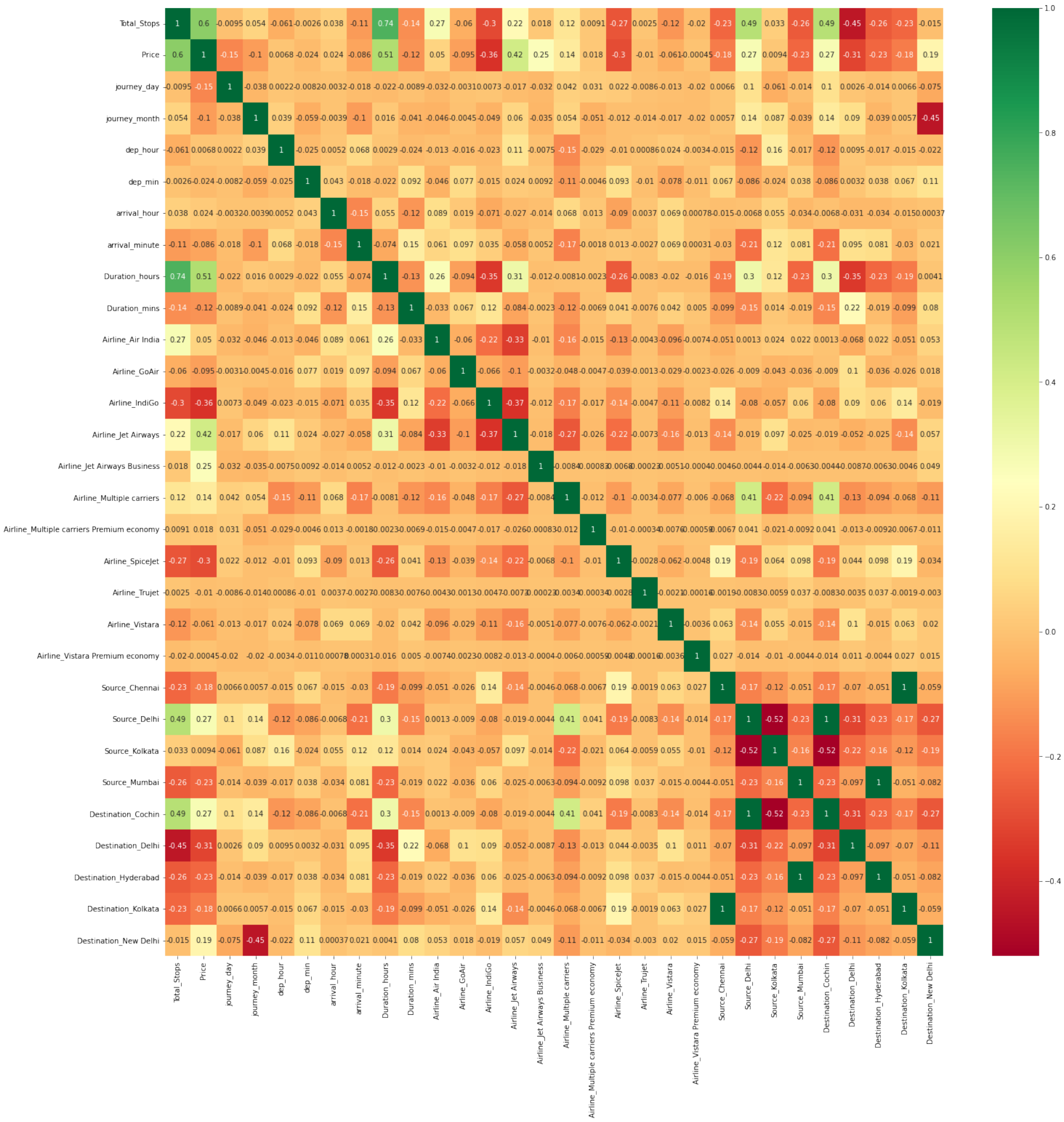
```
Y=data1.iloc[:,1]
Y.head()
```

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

```
# to seee the correlation between dependent and independent variables
we will see heatmap

plt.figure(figsize = (24,24))
sb.heatmap(data.corr(), annot = True,cmap = "RdYlGn")

plt.show()
```



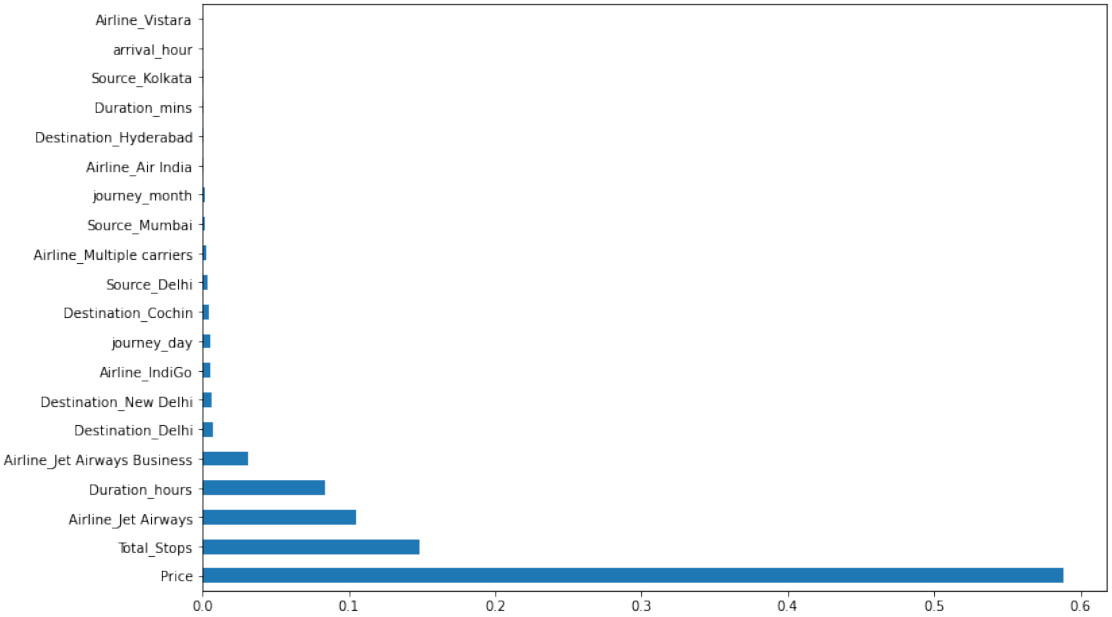
```
from sklearn.ensemble import ExtraTreesRegressor
selection = ExtraTreesRegressor()
selection.fit(X, Y)

ExtraTreesRegressor()

print(selection.feature_importances_)

[1.48681057e-01 5.88053495e-01 5.25817445e-03 1.79716616e-03
 1.63657975e-04 1.05213281e-04 3.21939376e-04 1.19375887e-04
 8.34573596e-02 4.23565890e-04 1.03173281e-03 8.96448145e-06
 5.82278394e-03 1.04905948e-01 3.16884543e-02 2.62364636e-03
 7.16660893e-06 1.33985824e-04 1.37128576e-07 2.69933328e-04
 4.50193483e-07 2.83841270e-05 3.43676874e-03 4.22897457e-04
 2.07043392e-03 4.79294159e-03 6.88738370e-03 6.29304991e-04
 2.22909920e-05 6.83538653e-03]
```

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



```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
0.2, random_state = 42)
```

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

```
RandomForestRegressor()
```

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

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

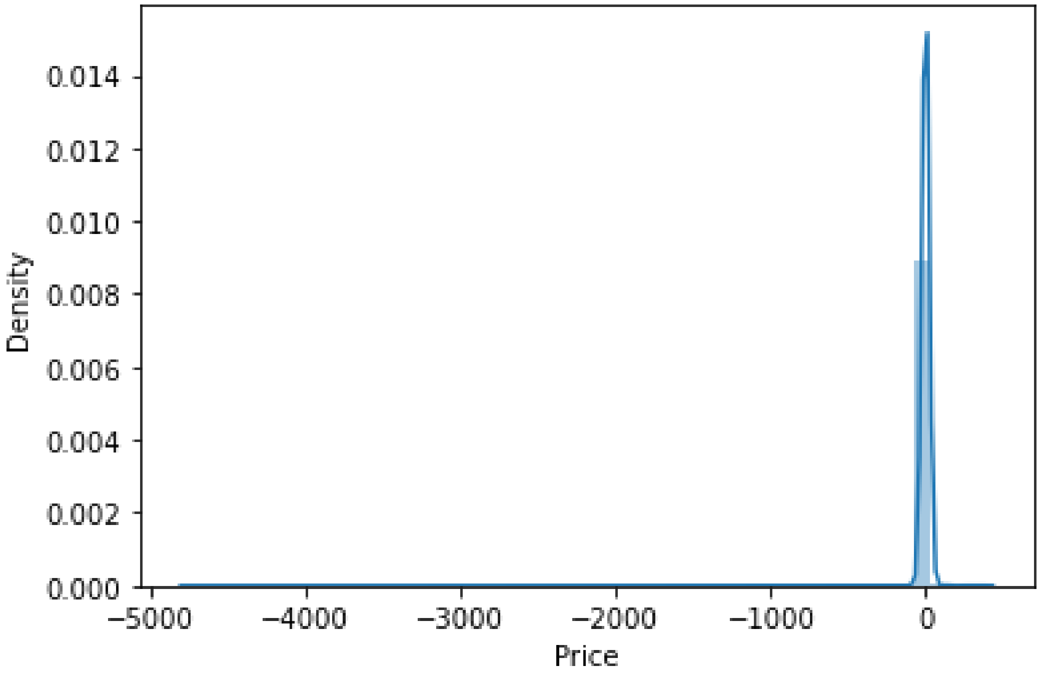
```
0.9994517008264339
```

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

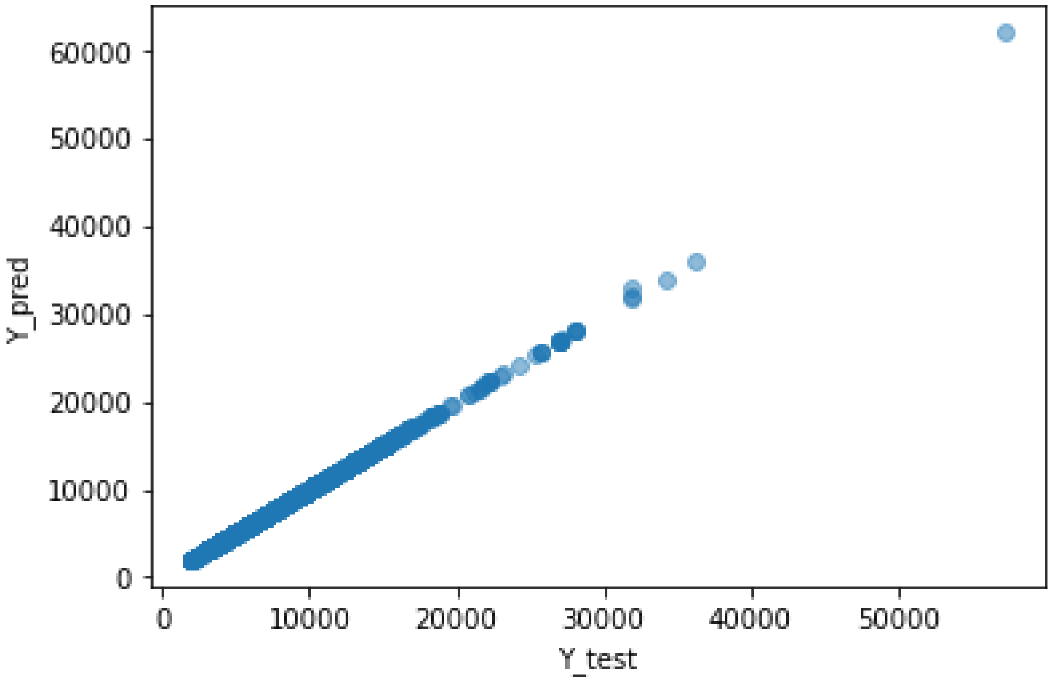
```
0.999478044094519
```

```
sb.distplot(Y_test-Y_pred)
plt.show()
```

```
C:\Users\91967\Anaconda3\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)
```



```
plt.scatter(Y_test, Y_pred, alpha = 0.5)
plt.xlabel("Y_test")
plt.ylabel("Y_pred")
plt.show()
```



```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(Y_test, Y_pred))
print('MSE:', metrics.mean_squared_error(Y_test, Y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(Y_test, Y_pred)))
```

```
MAE: 4.71979878334115
MSE: 11254.44263270942
RMSE: 106.08695788224593
```

```
2090.5509/(max(Y)-min(Y))
```

```
0.026887077025966846
```

```
metrics.r2_score(Y_test, Y_pred)
```

```
0.999478044094519
```

now we will do hyperparameter tuning

```
from sklearn.model_selection import RandomizedSearchCV


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

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

rf_random = RandomizedSearchCV(estimator = reg_rf, param_distributions
= random_grid,scoring='neg_mean_squared_error', n_iter = 10, cv = 5,
verbose=2)

rf_random.fit(X_train,Y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=1000; total time=  39.8s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=1000; total time=  36.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=1000; total time=  37.1s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=1000; total time=  37.6s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=1000; total time=  35.4s
```

```
[CV] END max_depth=10, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=200; total time=   7.1s
[CV] END max_depth=10, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=200; total time=   6.2s
[CV] END max_depth=10, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=200; total time=   6.7s
[CV] END max_depth=10, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=200; total time=   7.6s
[CV] END max_depth=10, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=200; total time=   6.4s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=10,
min_samples_split=2, n_estimators=500; total time=   4.1s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=10,
min_samples_split=2, n_estimators=500; total time=   3.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=10,
min_samples_split=2, n_estimators=500; total time=   3.7s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=10,
min_samples_split=2, n_estimators=500; total time=   3.9s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=10,
min_samples_split=2, n_estimators=500; total time=   4.1s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=10,
min_samples_split=10, n_estimators=700; total time=   5.7s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=10,
min_samples_split=10, n_estimators=700; total time=   5.4s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=10,
min_samples_split=10, n_estimators=700; total time=   6.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=10,
min_samples_split=10, n_estimators=700; total time=   5.4s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=10,
min_samples_split=10, n_estimators=700; total time=   5.4s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=2, n_estimators=700; total time=  13.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=2, n_estimators=700; total time=  10.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=2, n_estimators=700; total time=   9.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=2, n_estimators=700; total time=  10.4s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=2, n_estimators=700; total time=  10.3s
[CV] END max_depth=25, max_features=auto, min_samples_leaf=5,
min_samples_split=10, n_estimators=300; total time=  10.4s
[CV] END max_depth=25, max_features=auto, min_samples_leaf=5,
min_samples_split=10, n_estimators=300; total time=  11.2s
[CV] END max_depth=25, max_features=auto, min_samples_leaf=5,
min_samples_split=10, n_estimators=300; total time=  11.0s
[CV] END max_depth=25, max_features=auto, min_samples_leaf=5,
min_samples_split=10, n_estimators=300; total time=  10.1s
[CV] END max_depth=25, max_features=auto, min_samples_leaf=5,
min_samples_split=10, n_estimators=300; total time=  10.7s
```

```
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=1000; total time=  14.2s
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=1000; total time=  13.6s
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=1000; total time=  12.6s
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=1000; total time=  12.3s
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=1000; total time=  13.2s
[CV] END max_depth=5, max_features=auto, min_samples_leaf=1,
min_samples_split=10, n_estimators=200; total time=   3.4s
[CV] END max_depth=5, max_features=auto, min_samples_leaf=1,
min_samples_split=10, n_estimators=200; total time=   3.7s
[CV] END max_depth=5, max_features=auto, min_samples_leaf=1,
min_samples_split=10, n_estimators=200; total time=   3.4s
[CV] END max_depth=5, max_features=auto, min_samples_leaf=1,
min_samples_split=10, n_estimators=200; total time=   3.3s
[CV] END max_depth=5, max_features=auto, min_samples_leaf=1,
min_samples_split=10, n_estimators=200; total time=   3.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=5,
min_samples_split=10, n_estimators=400; total time=   4.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=5,
min_samples_split=10, n_estimators=400; total time=   4.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=5,
min_samples_split=10, n_estimators=400; total time=   4.6s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=5,
min_samples_split=10, n_estimators=400; total time=   3.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=5,
min_samples_split=10, n_estimators=400; total time=   4.9s
[CV] END max_depth=30, max_features=auto, min_samples_leaf=2,
min_samples_split=100, n_estimators=1200; total time=  32.3s
[CV] END max_depth=30, max_features=auto, min_samples_leaf=2,
min_samples_split=100, n_estimators=1200; total time=  30.2s
[CV] END max_depth=30, max_features=auto, min_samples_leaf=2,
min_samples_split=100, n_estimators=1200; total time=  30.9s
[CV] END max_depth=30, max_features=auto, min_samples_leaf=2,
min_samples_split=100, n_estimators=1200; total time=  27.4s
[CV] END max_depth=30, max_features=auto, min_samples_leaf=2,
min_samples_split=100, n_estimators=1200; total time=  31.4s

RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(),
                   param_distributions={'max_depth': [5, 10, 15, 20,
25, 30],
                                        'max_features': ['auto',
'sqrt'],
                                        'min_samples_leaf': [1, 2, 5,
10],
                                        'min_samples_split': [2, 5,
10, 15,
                                                              100],
                                        'n_estimators': [100, 200,
300, 400,
                                                         500, 600,
700, 800,
                                                         900, 1000,
1100,
                                                         1200]},
                   scoring='neg_mean_squared_error', verbose=2)
```

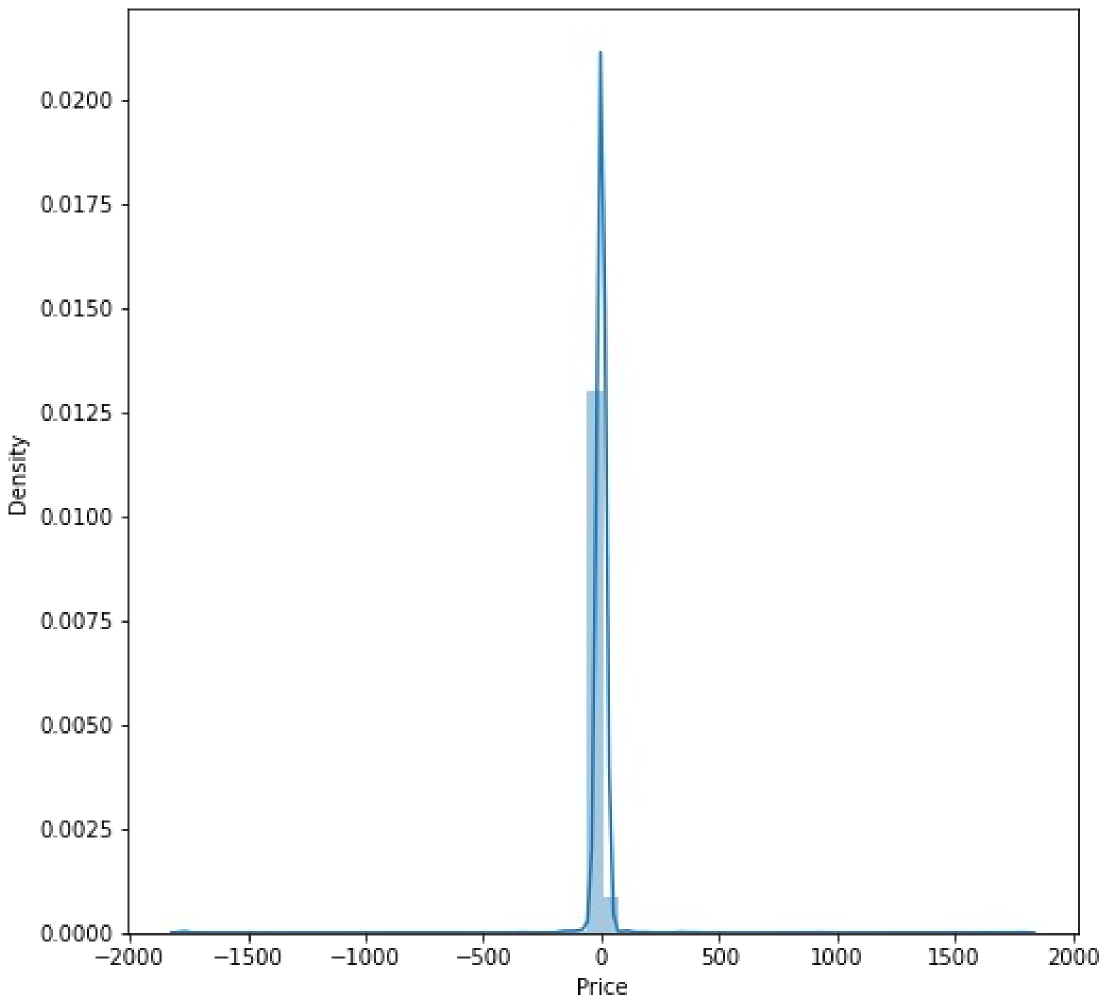
```
rf_random.best_params_
```

```
{'n_estimators': 200,
 'min_samples_split': 15,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 10}
```

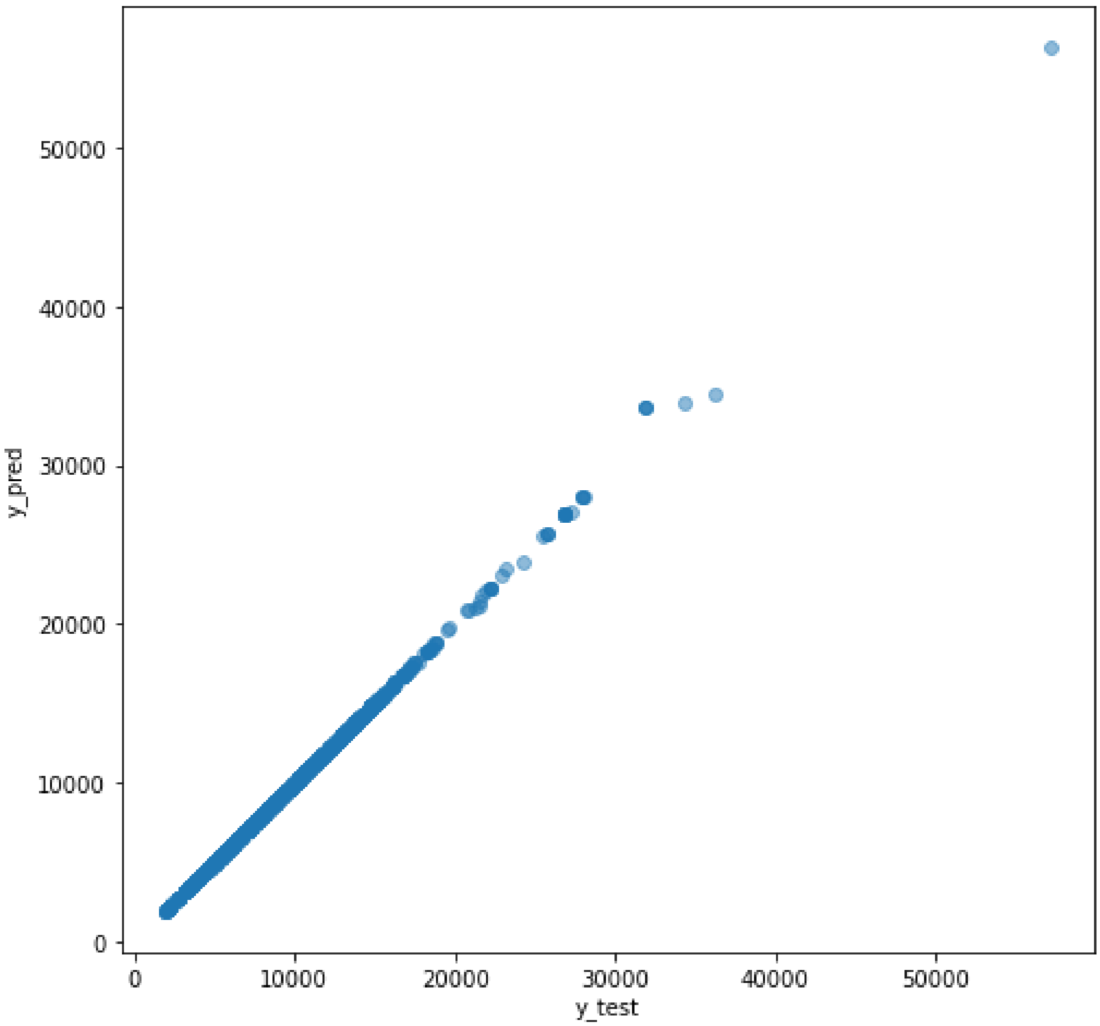
```
prediction = rf_random.predict(X_test)
```

```
plt.figure(figsize = (8,8))
sb.distplot(Y_test-prediction)
plt.show()
```

```
C:\Users\91967\Anaconda3\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)
```



```
plt.figure(figsize = (8,8))
plt.scatter(Y_test, prediction, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```



```
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: 8.28493916299273
MSE: 6701.432227674737
RMSE: 81.86227597419179
```

Now we will save our model so that we can use it again