Airfare Price Prediction

Statement of Issue It can be hard to guess airline ticket rates, we might see a fare today, find out the price of the same flight tomorrow, it's going to be a different story. We may have heard travelers sometimes complain that the costs of airline fares are too volatile. As data scientists, we can show that something can be expected provided the correct data.

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

Importing dataset

```
df = pd.read_excel("Python_Flight_Fare_d.xlsx")
```

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

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

```
df.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
```

basic information of dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          10683 non-null  object
 1   Date_of_Journey  10683 non-null  object
 2   Source           10683 non-null  object
 3   Destination      10683 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10683 non-null  object
 6   Arrival_Time     10683 non-null  object
 7   Duration         10683 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10683 non-null  object
 10  Price            10683 non-null  int64 
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

value counts of Duration column

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

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

count of null values in dataset column

```
df.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
```

unique values in Route counts

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

```
array(['BLR → DEL', 'CCU → IXR → BBI → BLR', 'DEL → LKO → BOM → COK',
       'CCU → NAG → BLR', 'BLR → NAG → DEL', 'CCU → BLR',
       'BLR → BOM → DEL', 'DEL → BOM → COK', 'DEL → BLR → COK',
       'MAA → CCU', 'CCU → BOM → BLR', 'DEL → AMD → BOM → COK',
       'DEL → PNQ → COK', 'DEL → CCU → BOM → COK', 'BLR → COK → DEL',
       'DEL → IDR → BOM → COK', 'DEL → LKO → COK',
       'CCU → GAU → DEL → BLR', 'DEL → NAG → BOM → COK',
       'CCU → MAA → BLR', 'DEL → HYD → COK', 'CCU → HYD → BLR',
       'DEL → COK', 'CCU → DEL → BLR', 'BLR → BOM → AMD → DEL',
       'BOM → DEL → HYD', 'DEL → MAA → COK', 'BOM → HYD',
       'DEL → BHO → BOM → COK', 'DEL → JAI → BOM → COK',
       'DEL → ATQ → BOM → COK', 'DEL → JDH → BOM → COK',
       'CCU → BBI → BOM → BLR', 'BLR → MAA → DEL',
       'DEL → GOI → BOM → COK', 'DEL → BDQ → BOM → COK',
       'CCU → JAI → BOM → BLR', 'CCU → BBI → BLR', 'BLR → HYD → DEL',
       'DEL → TRV → COK', 'CCU → IXR → DEL → BLR',
       'DEL → IXU → BOM → COK', 'CCU → IXB → BLR',
       'BLR → BOM → JDH → DEL', 'DEL → UDR → BOM → COK',
       'DEL → HYD → MAA → COK', 'CCU → BOM → COK → BLR',
       'BLR → CCU → DEL', 'CCU → BOM → GOI → BLR',
       'DEL → RPR → NAG → BOM → COK', 'DEL → HYD → BOM → COK',
       'CCU → DEL → AMD → BLR', 'CCU → PNQ → BLR',
       'BLR → CCU → GAU → DEL', 'CCU → DEL → COK → BLR',
       'BLR → PNQ → DEL', 'BOM → JDH → DEL → HYD',
       'BLR → BOM → BHO → DEL', 'DEL → AMD → COK', 'BLR → LKO → DEL',
       'CCU → GAU → BLR', 'BOM → GOI → HYD', 'CCU → BOM → AMD → BLR',
       'CCU → BBI → IXR → DEL → BLR', 'DEL → DED → BOM → COK',
       'DEL → MAA → BOM → COK', 'BLR → AMD → DEL', 'BLR → VGA → DEL',
       'CCU → JAI → DEL → BLR', 'CCU → AMD → BLR',
       'CCU → VNS → DEL → BLR', 'BLR → BOM → IDR → DEL',
       'BLR → BBI → DEL', 'BLR → GOI → DEL', 'BOM → AMD → ISK → HYD',
       'BOM → DED → DEL → HYD', 'DEL → IXC → BOM → COK',
       'CCU → PAT → BLR', 'BLR → CCU → BBI → DEL',
       'CCU → BBI → HYD → BLR', 'BLR → BOM → NAG → DEL',
       'BLR → CCU → BBI → HYD → DEL', 'BLR → GAU → DEL',
       'BOM → BHO → DEL → HYD', 'BOM → JLR → HYD',
       'BLR → HYD → VGA → DEL', 'CCU → KNU → BLR',
       'CCU → BOM → PNQ → BLR', 'DEL → BBI → COK',
       'BLR → VGA → HYD → DEL', 'BOM → JDH → JAI → DEL → HYD',
       'DEL → GWL → IDR → BOM → COK', 'CCU → RPR → HYD → BLR',
       'CCU → VTZ → BLR', 'CCU → DEL → VGA → BLR',
       'BLR → BOM → IDR → GWL → DEL', 'CCU → DEL → COK → TRV → BLR',
       'BOM → COK → MAA → HYD', 'BOM → NDC → HYD', 'BLR → BDQ → DEL',
       'CCU → BOM → TRV → BLR', 'CCU → BOM → HBX → BLR',
       'BOM → BDQ → DEL → HYD', 'BOM → CCU → HYD',
       'BLR → TRV → COK → DEL', 'BLR → IDR → DEL',
       'CCU → IXZ → MAA → BLR', 'CCU → GAU → IMF → DEL → BLR',
       'BOM → GOI → PNQ → HYD', 'BOM → BLR → CCU → BBI → HYD',
       'BOM → MAA → HYD', 'BLR → BOM → UDR → DEL',
       'BOM → UDR → DEL → HYD', 'BLR → VGA → VTZ → DEL',
       'BLR → HBX → BOM → BHO → DEL', 'CCU → IXA → BLR',
       'BOM → RPR → VTZ → HYD', 'BLR → HBX → BOM → AMD → DEL',
       'BOM → IDR → DEL → HYD', 'BOM → BLR → HYD', 'BLR → STV → DEL',
       'CCU → IXB → DEL → BLR', 'BOM → JAI → DEL → HYD',
       'BOM → VNS → DEL → HYD', 'BLR → HBX → BOM → NAG → DEL', nan,
       'BLR → BOM → IXC → DEL', 'BLR → CCU → BBI → HYD → VGA → DEL',
       'BOM → BBI → HYD'], dtype=object)
```

unique value in Total_Stops

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

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

There is only one value in Total_Stops & Route so we can drop null value from daaset

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

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

```
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
```

```
df.head(20)
```

```
              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
5            SpiceJet      24/06/2019   Kolkata    Banglore
6         Jet Airways      12/03/2019  Banglore   New Delhi
7         Jet Airways      01/03/2019  Banglore   New Delhi
8         Jet Airways      12/03/2019  Banglore   New Delhi
9   Multiple carriers      27/05/2019     Delhi      Cochin
10          Air India       1/06/2019     Delhi      Cochin
11             IndiGo      18/04/2019   Kolkata    Banglore
12          Air India      24/06/2019   Chennai     Kolkata
13        Jet Airways       9/05/2019   Kolkata    Banglore
14             IndiGo      24/04/2019   Kolkata    Banglore
15          Air India       3/03/2019     Delhi      Cochin
16           SpiceJet      15/04/2019     Delhi      Cochin
17        Jet Airways      12/06/2019     Delhi      Cochin
18          Air India      12/06/2019     Delhi      Cochin
19        Jet Airways      27/05/2019     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

5               CCU → BLR    09:00         11:25   2h 25m    non-stop

6         BLR → BOM → DEL    18:55  10:25 13 Mar  15h 30m      1 stop

7         BLR → BOM → DEL    08:00  05:05 02 Mar   21h 5m      1 stop

8         BLR → BOM → DEL    08:55  10:25 13 Mar  25h 30m      1 stop

9         DEL → BOM → COK    11:25         19:15   7h 50m      1 stop

10        DEL → BLR → COK    09:45         23:00  13h 15m      1 stop

11              CCU → BLR    20:20         22:55   2h 35m    non-stop

12              MAA → CCU    11:40         13:55   2h 15m    non-stop

13        CCU → BOM → BLR    21:10  09:20 10 May  12h 10m      1 stop
```

```
14              CCU → BLR    17:15         19:50   2h 35m    non-stop
15  DEL → AMD → BOM → COK    16:40  19:15 04 Mar  26h 35m     2 stops
16        DEL → PNQ → COK    08:45         13:15   4h 30m      1 stop
17        DEL → BOM → COK    14:00  12:35 13 Jun  22h 35m      1 stop
18  DEL → CCU → BOM → COK    20:15  19:15 13 Jun      23h     2 stops
19        DEL → BOM → COK    16:00  12:35 28 May  20h 35m      1 stop

                Additional_Info  Price
0                       No info   3897
1                       No info   7662
2                       No info  13882
3                       No info   6218
4                       No info  13302
5                       No info   3873
6   In-flight meal not included  11087
7                       No info  22270
8   In-flight meal not included  11087
9                       No info   8625
10                      No info   8907
11                      No info   4174
12                      No info   4667
13  In-flight meal not included   9663
14                      No info   4804
15                      No info  14011
16                      No info   5830
17  In-flight meal not included  10262
18                      No info  13381
19  In-flight meal not included  12898
```

Now we extract day values and month values from Date_of_Journey and create two new columns Journey_day & Journey_month

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

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

df.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_day
0           24           24
1            1            1
2            9            9
3           12           12
4            1            1
```

so after we create two new column from date_of_journey , now we drop Date_of_Journey column from dataset

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

```
----------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
d:\DATA SCIENCE\GIThub\project\CASE STUDY\Python_Case_Study_Fl.ipynb
Cell 29' in <module>
----> <a
href='vscode-notebook-cell:/d%3A/DATA%20SCIENCE/GIThub/project/CASE
%20STUDY/Python_Case_Study_Fl.ipynb#ch0000033?line=1'>2</a>
df.drop(["journey_day"], axis = 1, inplace = True)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\
pandas\util\_decorators.py:311, in
deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*arg
s, **kwargs)
    <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/util/_decorators.py?line=304'>305</a> if
len(args) > num_allow_args:
    <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/util/_decorators.py?line=305'>306</a>
warnings.warn(
    <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/util/_decorators.py?line=306'>307</a>
msg.format(arguments=arguments),
    <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/util/_decorators.py?line=307'>308</a>
FutureWarning,
    <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/util/_decorators.py?line=308'>309</a>
stacklevel=stacklevel,
    <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/util/_decorators.py?line=309'>310</
a>     )
--> <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/util/_decorators.py?line=310'>311</a>
return func(*args, **kwargs)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\
pandas\core\frame.py:4956, in DataFrame.drop(self, labels, axis,
index, columns, level, inplace, errors)
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4807'>4808</a>
@deprecate_nonkeyword_arguments(version=None, allowed_args=["self",
"labels"])
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4808'>4809</a> def
drop(
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4809'>4810</a>
self,
   (...)
   <a
```

```
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4816'>4817</a>
errors: str = "raise",
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4817'>4818</a> ):
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4818'>4819</a>     """
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4819'>4820</a>     Drop
specified labels from rows or columns.
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4820'>4821</a>
   (...)
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4953'>4954</a>
weight  1.0     0.8
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4954'>4955</a>     """
-> <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4955'>4956</a>
return super().drop(
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4956'>4957</a>
labels=labels,
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4957'>4958</a>
axis=axis,
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4958'>4959</a>
index=index,
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4959'>4960</a>
columns=columns,
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4960'>4961</a>
level=level,
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
```

```
10/lib/site-packages/pandas/core/frame.py?line=4961'>4962</a>
inplace=inplace,
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4962'>4963</a>
errors=errors,
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/frame.py?line=4963'>4964</a>     )

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\
pandas\core\generic.py:4279, in NDFrame.drop(self, labels, axis,
index, columns, level, inplace, errors)
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/generic.py?line=4276'>4277</a> for
axis, labels in axes.items():
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/generic.py?line=4277'>4278</a>     if
labels is not None:
-> <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/generic.py?line=4278'>4279</a>
obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/generic.py?line=4280'>4281</a> if
inplace:
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/generic.py?line=4281'>4282</a>
self._update_inplace(obj)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\
pandas\core\generic.py:4323, in NDFrame._drop_axis(self, labels, axis,
level, errors, consolidate, only_slice)
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/generic.py?line=4320'>4321</a>
new_axis = axis.drop(labels, level=level, errors=errors)
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/generic.py?line=4321'>4322</a>
else:
-> <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/generic.py?line=4322'>4323</a>
new_axis = axis.drop(labels, errors=errors)
   <a
```

```
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/generic.py?line=4323'>4324</a>
indexer = axis.get_indexer(new_axis)
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/generic.py?line=4325'>4326</a> # Case
for non-unique axis
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/generic.py?line=4326'>4327</a> else:

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\
pandas\core\indexes\base.py:6644, in Index.drop(self, labels, errors)
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/indexes/base.py?line=6641'>6642</a>
if mask.any():
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/indexes/base.py?line=6642'>6643</a>
if errors != "ignore":
-> <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/indexes/base.py?line=6643'>6644</a>
raise KeyError(f"{list(labels[mask])} not found in axis")
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/indexes/base.py?line=6644'>6645</a>
indexer = indexer[~mask]
   <a
href='file:///c%3A/Users/CREATOR/AppData/Local/Programs/Python/Python3
10/lib/site-packages/pandas/core/indexes/base.py?line=6645'>6646</a>
return self.delete(indexer)

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

same things we have do with Dep_time column , we create two new column Dep_hour and Dep_min from extract hour and min from Dep_Time

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

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

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

df.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
```

Similar to Date_of_Journey we can extract values from Arrival_Time

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

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

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

```
df.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_min
0       20             1           10
1       50            13           15
2       25             4           25
3        5            23           30
4       50            21           35
```

check the all the values in Duration

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

```
['2h 50m',
 '7h 25m',
 '19h',
 '5h 25m',
 '4h 45m',
 '2h 25m',
 '15h 30m',
 '21h 5m',
 '25h 30m',
 '7h 50m',
 '13h 15m',
 '2h 35m',
 '1h 20m',
 '22h 55m',
 '1h 30m',
 '2h 35m',
 '37h 20m',
 '2h 50m',
 '2h 50m',
 '36h 10m',
 '9h 15m',
 '1h 25m',
 '5h 5m',
 '2h 15m',
 '15h 10m',
 '25h 55m',
 '5h 40m',
 '7h 30m',
 '17h 45m',
 '2h 20m',
 '8h 15m',
 '3h 40m',
 '5h 55m',
 '2h 30m',
 '17h 30m',
 '13h 50m',
 '2h 15m',
 '21h 35m',
 '9h 30m',
 '8h 40m',
 '10h 25m',
 '35h 5m',
 '14h 15m',
 '3h',
 '25h 50m',
 '2h 30m',
 '8h 40m',
 '10h',
 '11h 30m',
 '7h 30m',
 ...]
```

here i create loop for check duration contains only hour min and if yes add min or hour in it

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

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

```
duration
```

```
['2h 50m',
 '7h 25m',
 '19h 0m',
 '5h 25m',
 '4h 45m',
 '2h 25m',
 '15h 30m',
 '21h 5m',
 '25h 30m',
 '7h 50m',
 '13h 15m',
 '2h 35m',
 '2h 15m',
 '12h 10m',
 '2h 35m',
 '26h 35m',
 '4h 30m',
 '22h 35m',
 '23h 0m',
 '20h 35m',
 '5h 10m',
 '15h 20m',
 '2h 50m',
 '2h 55m',
 '13h 20m',
 '15h 10m',
 '5h 45m',
 '5h 55m',
 '2h 50m',
 '2h 15m',
 '2h 15m',
 '13h 25m',
 '2h 50m',
 '22h 0m',
 '5h 30m',
 '10h 25m',
 '2h 15m',
 '21h 35m',
 '9h 30m',
 '8h 40m',
 '10h 25m',
 '35h 5m',
 '14h 15m',
 '3h 0m',
 '25h 50m',
 '2h 30m',
 '8h 40m',
 '10h 0m',
 '11h 30m',
 '7h 30m',
 ...]
```

now Extract hour and min from duration column and create two new column Duration_hours & Duration_mins

```
duration[i].split(sep = "h")[0]
```

```
'8'
```

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

# Adding duration_hours and duration_mins list to train_data dataframe

df["Duration_hours"] = duration_hours
df["Duration_mins"] = duration_mins
```

Now drop Duration column

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

```
df.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_min  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
```

Handling Categorical Data

check the value counts in Airline

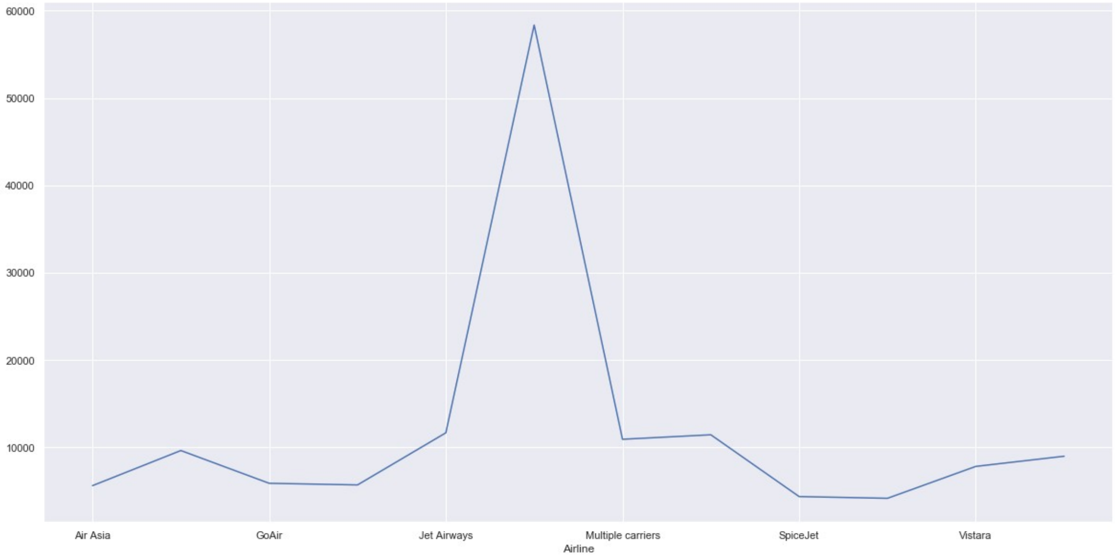
```
df["Airline"].value_counts()
```

```
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
```

Display price according to Airline

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

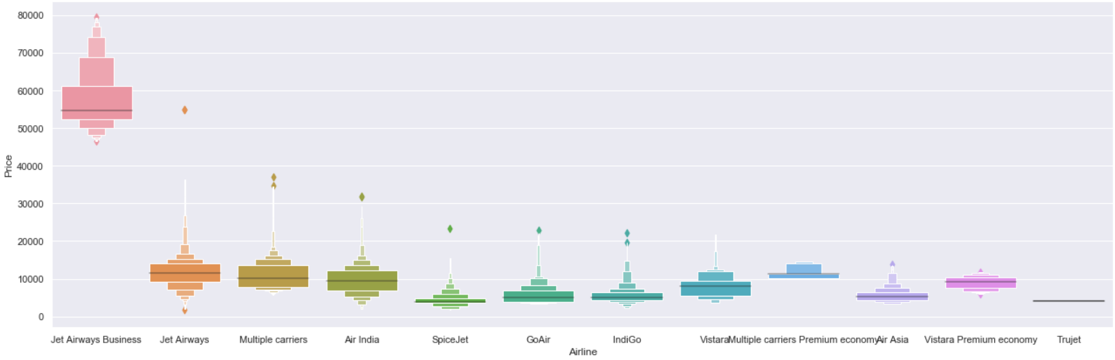
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c73acc7548>
```



here we can say that jet airways price higher than any onther airline

chech the average price according to Airline

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



now we hace to create dummy data for Categorical

Airline is Nominal Categorical data we will perform OneHotEncoding

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

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

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
```

check the Source values counts

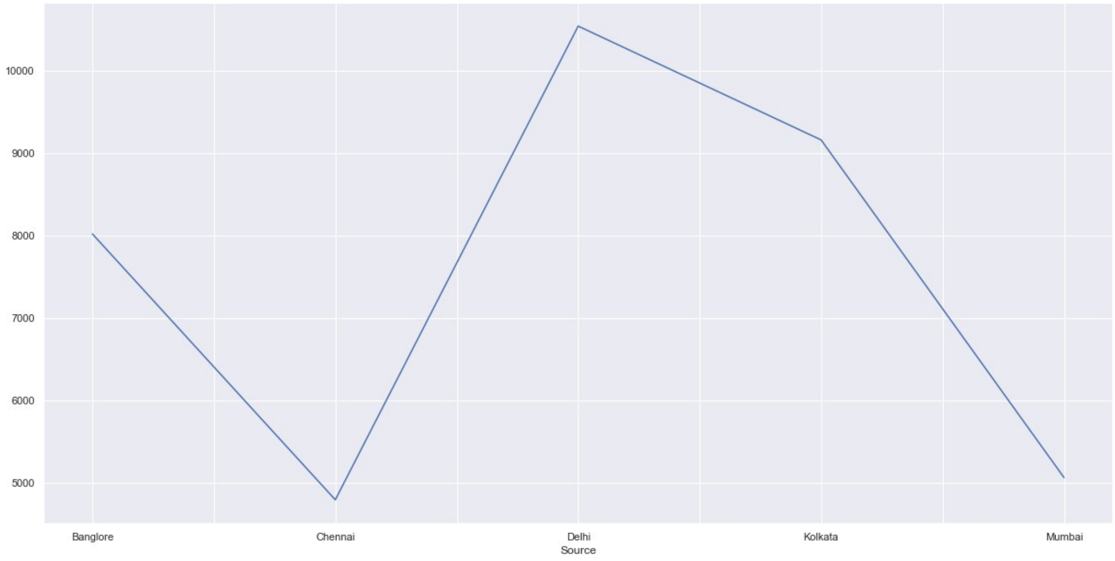
```
df["Source"].value_counts()
```

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

display average price according to source

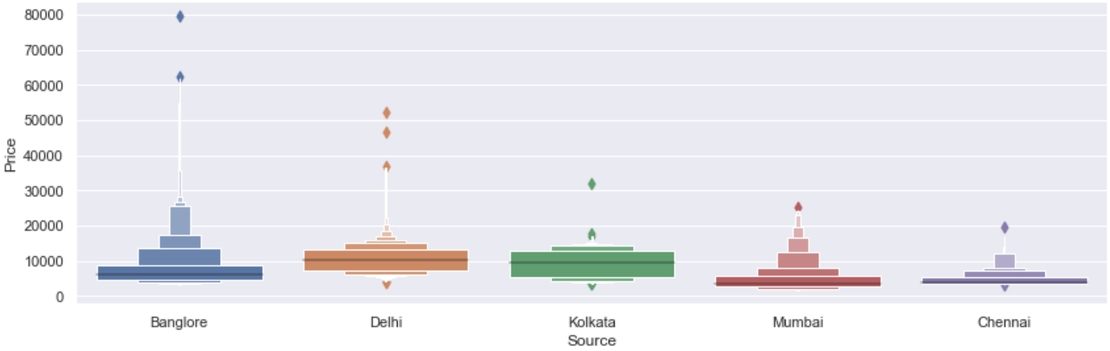
```
df.groupby('Source')['Price'].mean().plot(figsize=(20,10))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c73bb3ae48>
```



Source vs Price

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



Source is Nominal Categorical data we will perform OneHotEncoding

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

check Destination value counts

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

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

Destination is Nominal Categorical data we will perform OneHotEncoding

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

check Route column

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

```
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
```

Drop Route and Additional_Info columns because its unrelevant

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

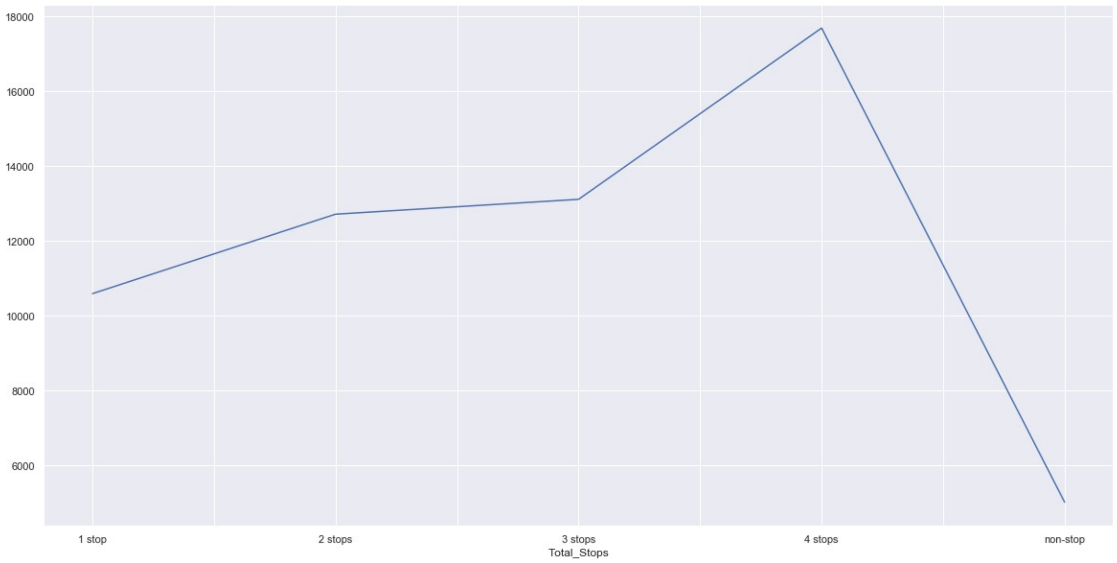
```
df["Total_Stops"].value_counts()
```

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

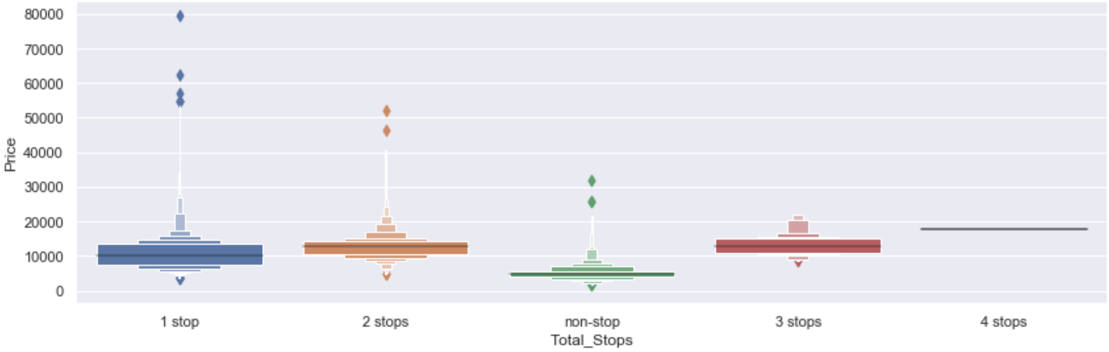
plot Average price of all Total_Stops

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c73bb40d88>
```



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



Now replace categorical value in Total_stop with numeric value by manually

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

```
df.head()
```

```
       Airline    Source Destination  Total_Stops  Price  Journey_day
\
0       IndiGo  Banglore   New Delhi            0   3897           24

1    Air India   Kolkata    Banglore            2   7662            1

2  Jet Airways     Delhi      Cochin            2  13882            9
```

```
3       IndiGo   Kolkata    Banglore            1   6218          12

4       IndiGo  Banglore   New Delhi            1  13302           1


   Journey_month  Dep_hour  Dep_min  Arrival_hour  Arrival_min  \
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
```

concatenate all dummy data which we created with our orignal dataset

```
# Concatenate dataframe --> train_data + Airline + Source +
Destination

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

df.head()
```

```
       Airline    Source Destination  Total_Stops  Price  Journey_day
\
0       IndiGo  Banglore   New Delhi            0   3897          24

1    Air India   Kolkata    Banglore            2   7662           1

2  Jet Airways     Delhi      Cochin            2  13882           9

3       IndiGo   Kolkata    Banglore            1   6218          12

4       IndiGo  Banglore   New Delhi            1  13302           1


   Journey_month  Dep_hour  Dep_min  Arrival_hour  Arrival_min  \
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  Airline_Air India  Airline_GoAir  \
0               2             50                  0              0
1               7             25                  1              0
2              19              0                  0              0
3               5             25                  0              0
4               4             45                  0              0

   Airline_IndiGo  Airline_Jet Airways  Airline_Jet Airways
Business  \
0               1                    0                             0

1               0                    0                             0

2               0                    1                             0

3               1                    0                             0

4               1                    0                             0


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

   Airline_SpiceJet  Airline_Trujet  Airline_Vistara  \
0                 0               0                0
1                 0               0                0
2                 0               0                0
3                 0               0                0
4                 0               0                0

   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  
```

Drop Categorical columns from dataset

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

```
df.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_min  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_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

   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

   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
```

```
df.shape
```

```
(10682, 30)
```

Feature Selection

```
df.shape
```

```
(10682, 30)
```

Check all columns from dataset

```
df.columns
```

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

Create target and features set

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

```
X.head()
```

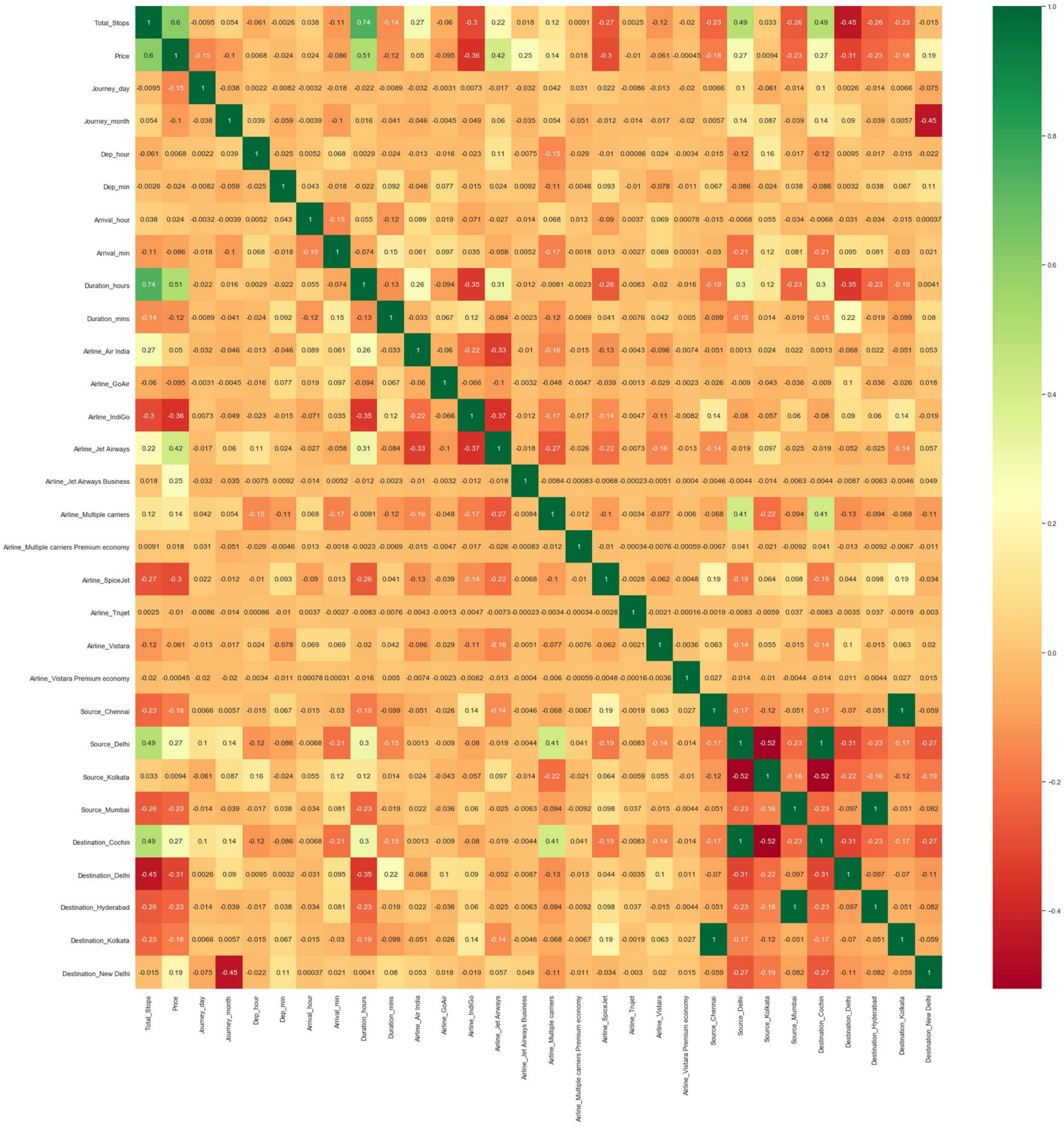
```
   Total_Stops  Journey_day  Journey_month  Dep_hour  Dep_min
Arrival_hour  \
0            0           24              3        22       20
1
1            2            1              5         5       50
13
2            2            9              6         9       25
4
3            1           12              5        18        5
23
```

```
4            1            1              3        16       50
21

   Arrival_min  Duration_hours  Duration_mins  Airline_Air India  \
0           10               2             50                  0
1           15               7             25                  1
2           25              19              0                  0
3           30               5             25                  0
4           35               4             45                  0

   Airline_GoAir  Airline_IndiGo  Airline_Jet Airways  \
0              0               1                    0
1              0               0                    0
2              0               0                    1
3              0               1                    0
4              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

   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

   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
```

Finds correlation between Independent and dependent attributes

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

plt.show()
```



Find the important Featuresn using ExtraTreesRegressor

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

```
ExtraTreesRegressor(bootstrap=False, ccp_alpha=0.0, criterion='mse',
                    max_depth=None, max_features='auto',
max_leaf_nodes=None,
                    max_samples=None, min_impurity_decrease=0.0,
                    min_impurity_split=None, min_samples_leaf=1,
                    min_samples_split=2, min_weight_fraction_leaf=0.0,
                    n_estimators=100, n_jobs=None, oob_score=False,
                    random_state=None, verbose=0, warm_start=False)
```

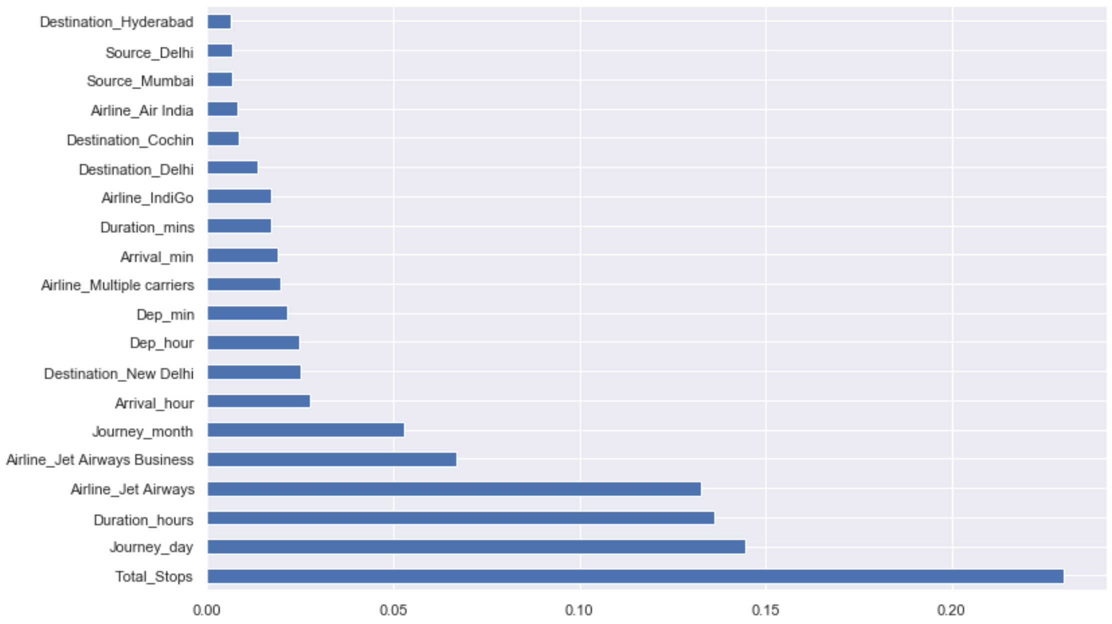
print all features importances

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

```
[2.30143659e-01 1.44672427e-01 5.28838029e-02 2.46389644e-02
 2.13257481e-02 2.77958625e-02 1.88825546e-02 1.36250308e-01
 1.73244828e-02 8.18570154e-03 1.63049606e-03 1.69865732e-02
 1.32573871e-01 6.70634625e-02 1.97137544e-02 8.85255822e-04
 2.71307546e-03 1.01912889e-04 4.85068129e-03 7.36838181e-05
 4.93473642e-04 6.56190557e-03 3.34568960e-03 6.71729993e-03
 8.62360191e-03 1.37031867e-02 6.22982636e-03 5.14917665e-04
 2.51138223e-02]
```

plot graph of feature importances for better visualization

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



create training and testing data

```
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)
```

Apply Linear regression on training dataset

```
from sklearn.linear_model import LinearRegression

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

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

print trainging and testing score

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

```
0.6240840020468167
```

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

```
0.61959437290701
```

How try all diffrent regression algorith and find the testing score

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

model =
[DecisionTreeRegressor,SVR,RandomForestRegressor,KNeighborsRegressor,A
daBoostRegressor,XGBRFRegressor]

for mod in model:
    reg = mod()
    reg = reg.fit(X_train,y_train)
    print(mod , 'accuracy',reg.score(X_test,y_test))
```

```
<class 'sklearn.tree._classes.DecisionTreeRegressor'> accuracy
0.7277833526169082
<class 'sklearn.svm._classes.SVR'> accuracy -0.00041646312498344606
<class 'sklearn.ensemble._forest.RandomForestRegressor'> accuracy
0.7969644029493801
<class 'sklearn.neighbors._regression.KNeighborsRegressor'> accuracy
0.570669349010061
<class 'sklearn.ensemble._weight_boosting.AdaBoostRegressor'> accuracy
0.4386200418675608
<class 'xgboost.sklearn.XGBRFRegressor'> accuracy 0.7161708929527828
```

Now apply Kflod and cross validation technique

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

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


results = []
names = []
for name,model in models:
    kfold = KFold(n_splits=10)
    cv_result =cross_val_score(model,X_train,y_train,cv=kfold)
    names.append(name)
    results.append(cv_result)
for i in range(len(names)):
    print(names[i],results[i].mean())
```

```
KNN 0.5639980977635727
CART 0.6909200089636072
RF 0.8035952735736792
SVM -0.00016124772787321496
AdaBoost 0.41986782275841605
XGB 0.7221475814068828
```

Here we see RandomForestRegressor gives us best score so we can use RandomForest Regressor algorithm

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

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto',
max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2,
min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

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

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

```
0.9528701383520191
```

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

```
0.798284510731937
```

Perform Hyper-prameter tuning using RandomizedSearchCV

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

create list for all possible parameter

```
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}
```

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

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

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

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5,
max_features=sqrt, max_depth=10

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1
concurrent workers.

[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5,
max_features=sqrt, max_depth=10, total=   5.6s
```

```
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5,
max_features=sqrt, max_depth=10

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    5.5s
remaining:    0.0s

[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5,
max_features=sqrt, max_depth=10, total=   5.2s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5,
max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5,
max_features=sqrt, max_depth=10, total=   4.9s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5,
max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5,
max_features=sqrt, max_depth=10, total=   4.0s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5,
max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5,
max_features=sqrt, max_depth=10, total=   6.8s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=15, total=   8.1s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=15, total=   7.1s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=15, total=   8.7s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=15, total=   8.4s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=15, total=   6.0s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5,
max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5,
max_features=auto, max_depth=15, total=   3.2s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5,
max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5,
max_features=auto, max_depth=15, total=   3.3s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5,
max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5,
max_features=auto, max_depth=15, total=   3.2s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5,
max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5,
max_features=auto, max_depth=15, total=   3.1s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5,
max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5,
max_features=auto, max_depth=15, total=   3.3s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5,
max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5,
max_features=auto, max_depth=15, total=   6.8s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5,
max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5,
max_features=auto, max_depth=15, total=   8.2s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5,
max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5,
max_features=auto, max_depth=15, total=   7.3s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5,
max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5,
max_features=auto, max_depth=15, total=   6.2s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5,
max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5,
max_features=auto, max_depth=15, total=   5.9s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10,
max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10,
max_features=auto, max_depth=20, total=   9.3s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10,
max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10,
max_features=auto, max_depth=20, total=   9.1s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10,
max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10,
max_features=auto, max_depth=20, total=   8.9s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10,
max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10,
max_features=auto, max_depth=20, total=   9.0s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10,
max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10,
max_features=auto, max_depth=20, total=   9.1s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1,
max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1,
max_features=sqrt, max_depth=25, total=  10.0s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1,
max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1,
max_features=sqrt, max_depth=25, total=   9.7s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1,
max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1,
max_features=sqrt, max_depth=25, total=   9.3s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1,
max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1,
max_features=sqrt, max_depth=25, total=   9.3s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1,
max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1,
max_features=sqrt, max_depth=25, total=   9.2s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10,
max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10,
max_features=sqrt, max_depth=5, total=   2.9s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10,
max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10,
max_features=sqrt, max_depth=5, total=   3.2s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10,
max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10,
max_features=sqrt, max_depth=5, total=   3.0s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10,
max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10,
max_features=sqrt, max_depth=5, total=   3.1s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10,
max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10,
max_features=sqrt, max_depth=5, total=   3.3s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1,
max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1,
max_features=sqrt, max_depth=15, total=   1.5s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1,
max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1,
max_features=sqrt, max_depth=15, total=   1.4s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1,
max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1,
max_features=sqrt, max_depth=15, total=   1.4s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1,
max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1,
max_features=sqrt, max_depth=15, total=   1.6s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1,
max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1,
max_features=sqrt, max_depth=15, total=   1.5s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=5

[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=5, total=   2.0s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=5, total=   2.1s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=5, total=   2.0s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=5, total=   2.1s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2,
max_features=sqrt, max_depth=5, total=   1.9s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1,
max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1,
max_features=auto, max_depth=20, total=  10.9s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1,
max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1,
max_features=auto, max_depth=20, total=  10.9s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1,
max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1,
max_features=auto, max_depth=20, total=  11.1s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1,
max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1,
max_features=auto, max_depth=20, total=  11.0s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1,
max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1,
max_features=auto, max_depth=20, total=  11.1s
```

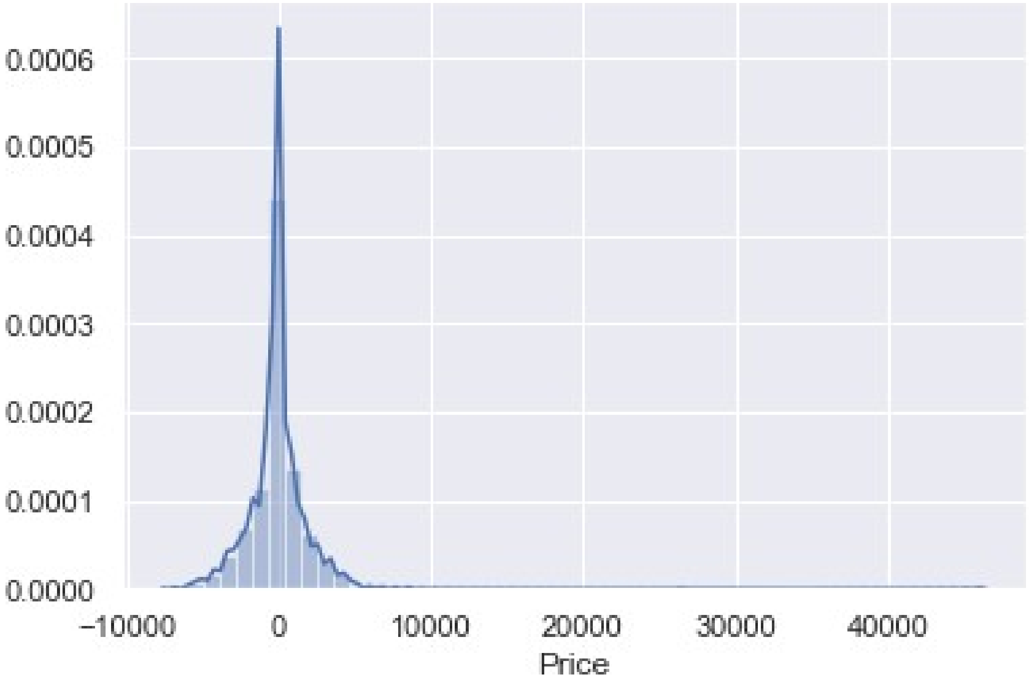
```
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:  4.9min finished

RandomizedSearchCV(cv=5, error_score=nan,
                   estimator=RandomForestRegressor(bootstrap=True,
                                                   ccp_alpha=0.0,
                                                   criterion='mse',
                                                   max_depth=None,

max_features='auto',

max_leaf_nodes=None,
                                                   max_samples=None,

min_impurity_decrease=0.0,

min_impurity_split=None,
                                                   min_samples_leaf=1,

min_samples_split=2,

min_weight_fraction_leaf=0.0,
                                                   n_estimators=100,
                                                   n_jobs=None,
oob_score=Fals...
                   iid='deprecated', n_iter=10, n_jobs=1,
                   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]},
                   pre_dispatch='2*n_jobs', random_state=42,
refit=True,
                   return_train_score=False,
scoring='neg_mean_squared_error',
                   verbose=2)
```

```
rf_random.best_params_
```

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

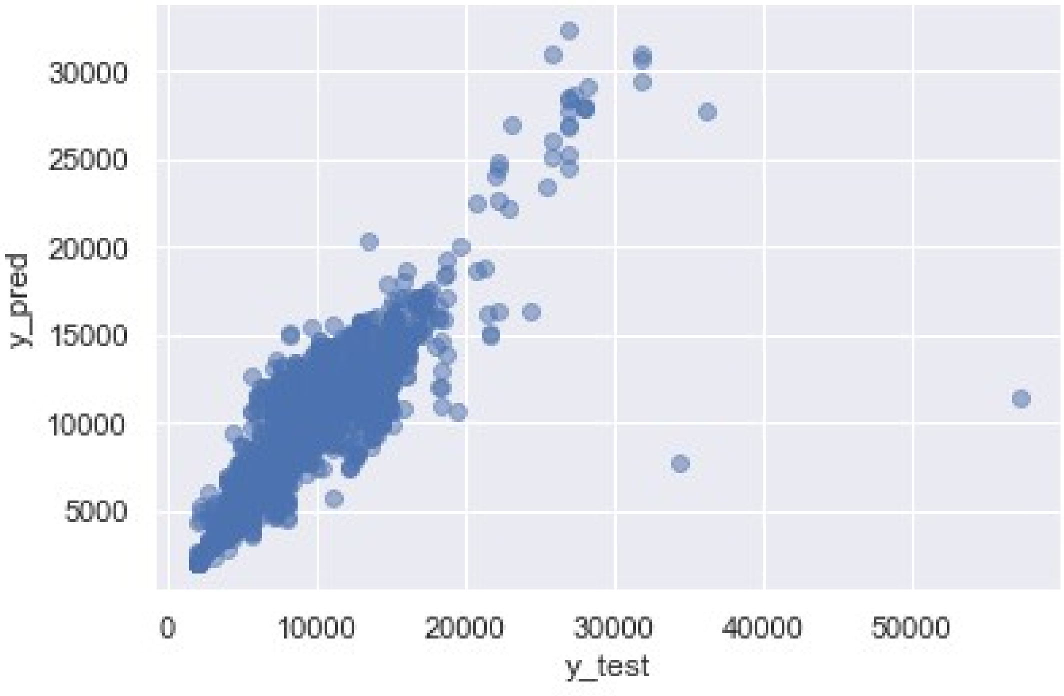
compare y_test and y_pred value using distplot

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



And scatter plot

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



Model Evalution

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

check mean_absolute_error

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

```
1179.9788104872175
```

check mean_squared_error

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

```
4349400.741053828
```

check r2_score

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

```
0.798284510731937
```