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

sns.set()
```

Importing dataset

1. Since data is in form of excel file we have to use pandas read_excel to load the data
2. After loading it is important to check the complete information of data as it can indication many of the hidden infomation such as null values in a column or a row
3. Check whether any null values are there or not. if it is present then following can be done, A. Imputing data using Imputation method in sklearn B. Filling NaN values with mean, median and mode using fillna() method
4. Describe data --> which can give statistical analysis

```
train_data = pd.read_excel(r"Data_Train.xlsx")

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

train_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
```

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
```

```
 0   Airline          10683 non-null  object
 1   Date_of_Journey  10683 non-null  object
 2   Source           10683 non-null  object
 3   Destination      10683 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10683 non-null  object
 6   Arrival_Time     10683 non-null  object
 7   Duration         10683 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10683 non-null  object
 10  Price            10683 non-null  int64
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

```
train_data["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
```

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

```
train_data.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
```

EDA

From description we can see that Date_of_Journey is a object data type, Therefore, we have to convert this datatype into timestamp so as to use this column properly for prediction

For this we require pandas to_datetime to convert object data type to datetime dtype.

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

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

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


train_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
```

```
# Since we have converted Date_of_Journey column into integers, Now we
can drop as it is of no use.
```

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

# Departure time is when a plane leaves the gate.
# Similar to Date_of_Journey we can extract values from Dep_Time

# Extracting Hours
train_data["Dep_hour"] =
pd.to_datetime(train_data["Dep_Time"]).dt.hour

# Extracting Minutes
train_data["Dep_min"] =
pd.to_datetime(train_data["Dep_Time"]).dt.minute

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

train_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
```

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

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

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

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

train_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_min
0       20             1           10
1       50            13           15
2       25             4           25
```

```
3        5            23           30
4       50            21           35

# Time taken by plane to reach destination is called Duration
# It is the differnce betwwen Departure Time and Arrival time


# Assigning and converting Duration column into list
duration = list(train_data["Duration"])

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
minute
        else:
            duration[i] = "0h " + duration[i]           # Adds 0 hour

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

train_data["Duration_hours"] = duration_hours
train_data["Duration_mins"] = duration_mins

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

train_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_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

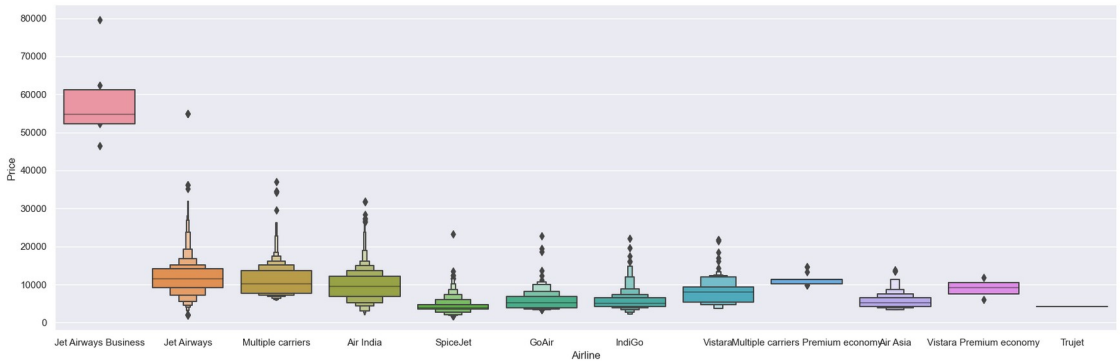
One can find many ways to handle categorical data. Some of them categorical data are,

1. Nominal data --> data are not in any order --> OneHotEncoder is used in this case
   Ordinal data --> data are in order --> 2. LabelEncoder is used in this case

```
train_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
```

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

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




```
# As Airline is Nominal Categorical data we will perform
OneHotEncoding

Airline = train_data[["Airline"]]

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

Airline.head()
```

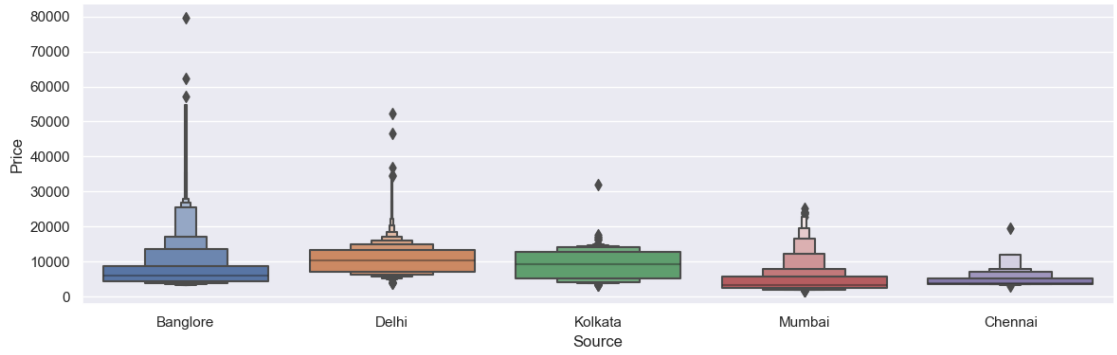
```
   Airline_Air India  Airline_GoAir  Airline_IndiGo  Airline_Jet
Airways  \
0                  0              0               1
0
1                  1              0               0
0
2                  0              0               0
1
3                  0              0               1
0
4   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

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

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

```
# Source vs Price

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




```
# As Source is Nominal Categorical data we will perform OneHotEncoding

Source = train_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
```

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

```
Cochin       4536
Banglore     2871
```

```
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: Destination, dtype: int64
```

```
# As Destination is Nominal Categorical data we will perform
OneHotEncoding

Destination = train_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
```

```
train_data["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
```

```
# Additional_Info contains almost 80% no_info
# Route and Total_Stops are related to each other

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

train_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 this is case of Ordinal Categorical type we perform LabelEncoder
# Here Values are assigned with corresponding keys
```

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

```
train_data.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 dataframe --> train_data + Airline + Source +
Destination
```

```
data_train = pd.concat([train_data, Airline, Source, Destination],
axis = 1)
```

```
data_train.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
```

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

data_train.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
```

```
data_train.shape
```

```
(10682, 30)
```

Test Set

```
test_data = pd.read_excel(r"Test_set.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
```

```
# Preprocessing

print("Test data Info")
print("-"*75)
print(test_data.info())

print()
print()

print("Null values :")
print("-"*75)
test_data.dropna(inplace = True)
print(test_data.isnull().sum())

# EDA

# 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)

# Dep_Time
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)

# Arrival_Time
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
duration = list(test_data["Duration"])

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
```

```
minute
        else:
            duration[i] = "0h " + duration[i]           # Adds 0 hour

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 column to test set
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)


# Additional_Info contains almost 80% no_info
# Route and Total_Stops are related to each other
test_data.drop(["Route", "Additional_Info"], axis = 1, inplace = True)

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

```
# Concatenate dataframe --> test_data + Airline + Source + Destination
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)
```

```
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
None


Null values :
-----------------------------------------------------------------------
-----
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
dtype: int64
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)

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  GoAir
IndiGo  \
0           25              10             55          0      0
0
1           20               4              0          0      0
1
2            0              23             45          0      0
0
3            0              13              0          0      0
0
4           45               2             50          0      0
0

   Jet Airways  Jet Airways Business  Multiple carriers  \
0            1                     0                  0
1            0                     0                  0
2            1                     0                  0
3            0                     0                  1
4            0                     0                  0

   Multiple carriers Premium economy  SpiceJet  Vistara  \
0                                  0         0        0
1                                  0         0        0
2                                  0         0        0
3                                  0         0        0
4                                  0         0        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
```

Feature Selection

Finding out the best feature which will contribute and have good relation with target variable. Following are some of the feature selection methods,

1. heatmap
2. feature_importance_
3. SelectKBest

```
data_train.shape
```

```
(10682, 30)
```

```
data_train.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')
```

```
X = data_train.loc[:, ['Total_Stops', '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']]
X.head()
```

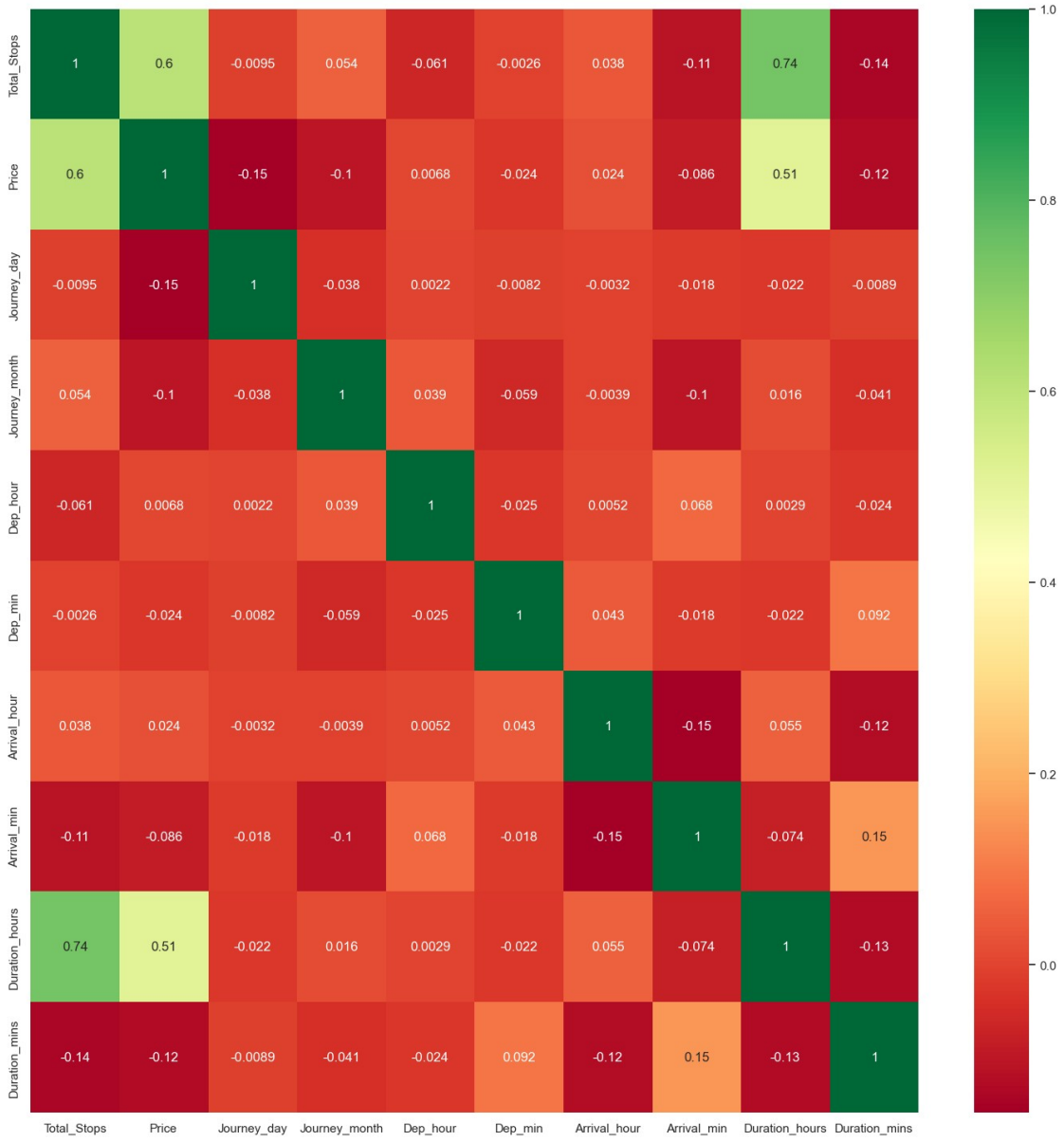
```
   Total_Stops  Journey_day  Journey_month  Dep_hour  Dep_min
Arrival_hour  \
0            0           24              3        22       20
1
1            2            1              5         5       50
13
2            2            9              6         9       25
4
3            1           12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

```python
y = data_train.iloc[:, 1]
y.head()
```

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

```python
# Finds correlation between Independent and dependent attributes

plt.figure(figsize = (18,18))
sns.heatmap(train_data.corr(), annot = True, cmap = "RdYlGn")

plt.show()
```




```
# Important feature using ExtraTreesRegressor
```

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

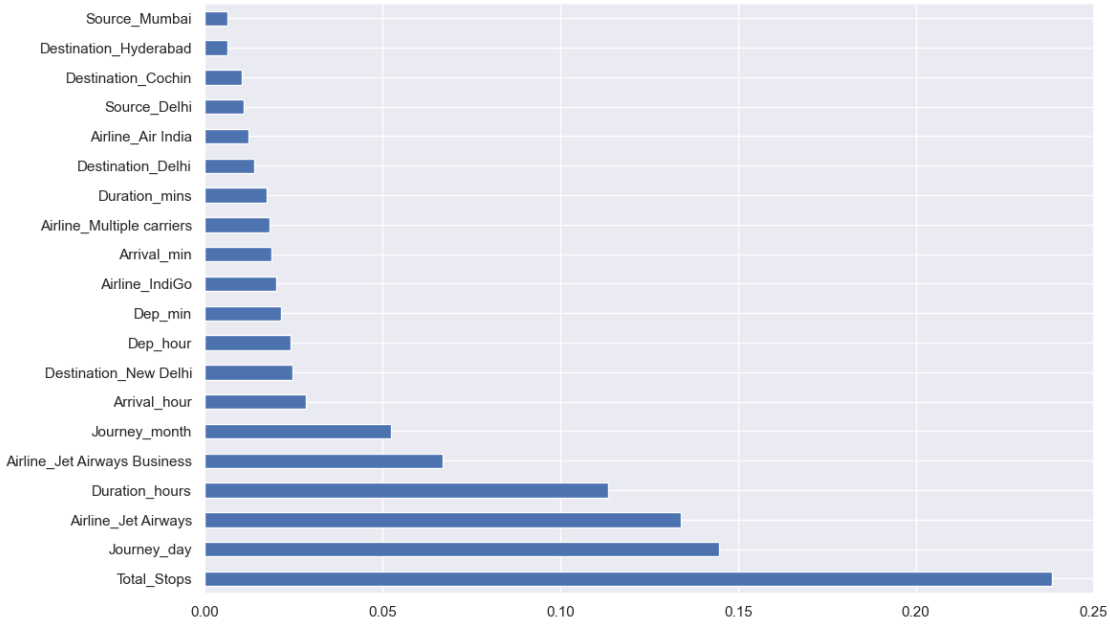
```
ExtraTreesRegressor()
```

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

```
[2.10019048e-01 1.43594012e-01 5.29988557e-02 2.43389435e-02
 2.14446918e-02 2.74590333e-02 1.96060177e-02 1.43784273e-01
 1.81903453e-02 9.77314977e-03 1.71805093e-03 1.76919749e-02
 1.35299402e-01 6.77707191e-02 1.86517978e-02 8.70553321e-04
 2.56094792e-03 1.11462535e-04 4.84264774e-03 8.29715015e-05
```

```
 4.23821068e-04 1.19889832e-02 3.35619979e-03 6.77522957e-03
 9.59449368e-03 1.44852639e-02 7.06792489e-03 4.80525293e-04
 2.50186610e-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()
```



Fitting model using Random Forest

1. Split dataset into train and test set in order to prediction w.r.t X_test
2. If needed do scaling of data .Scaling is not done in Random forest
3. Import model
4. Fit the data
5. Predict w.r.t X_test
6. In regression check RSME Score
7. Plot graph

```
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)
```

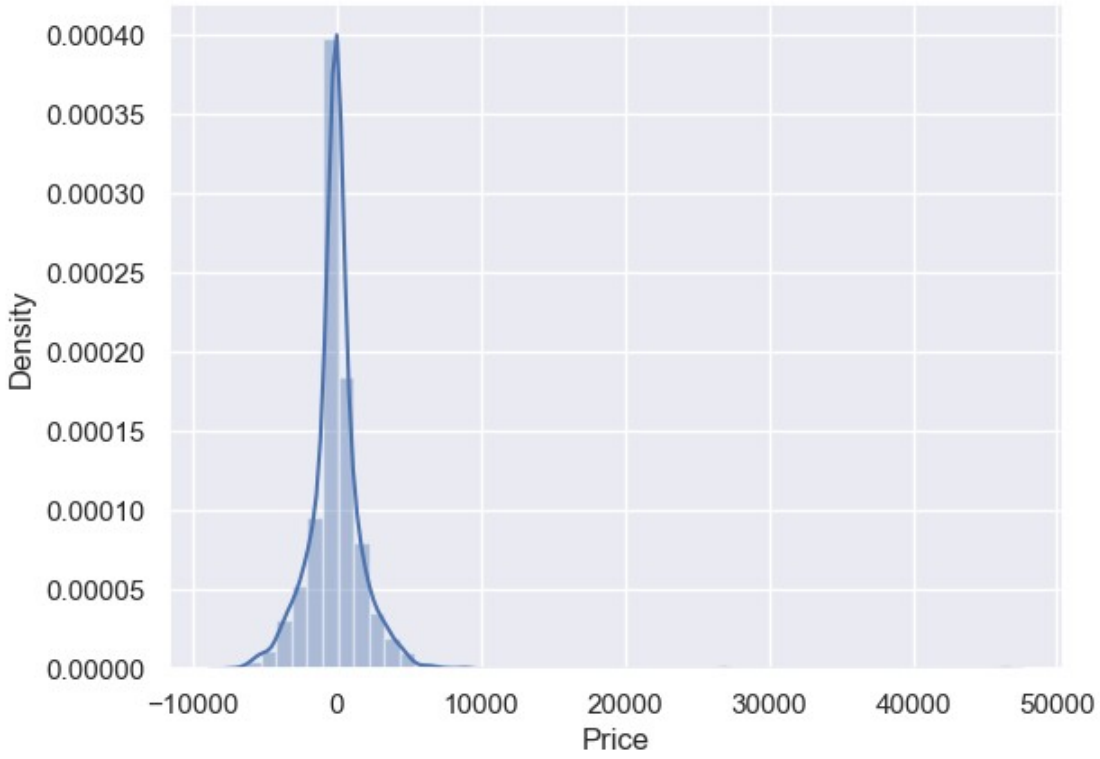
```
0.9524909728852449
```

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

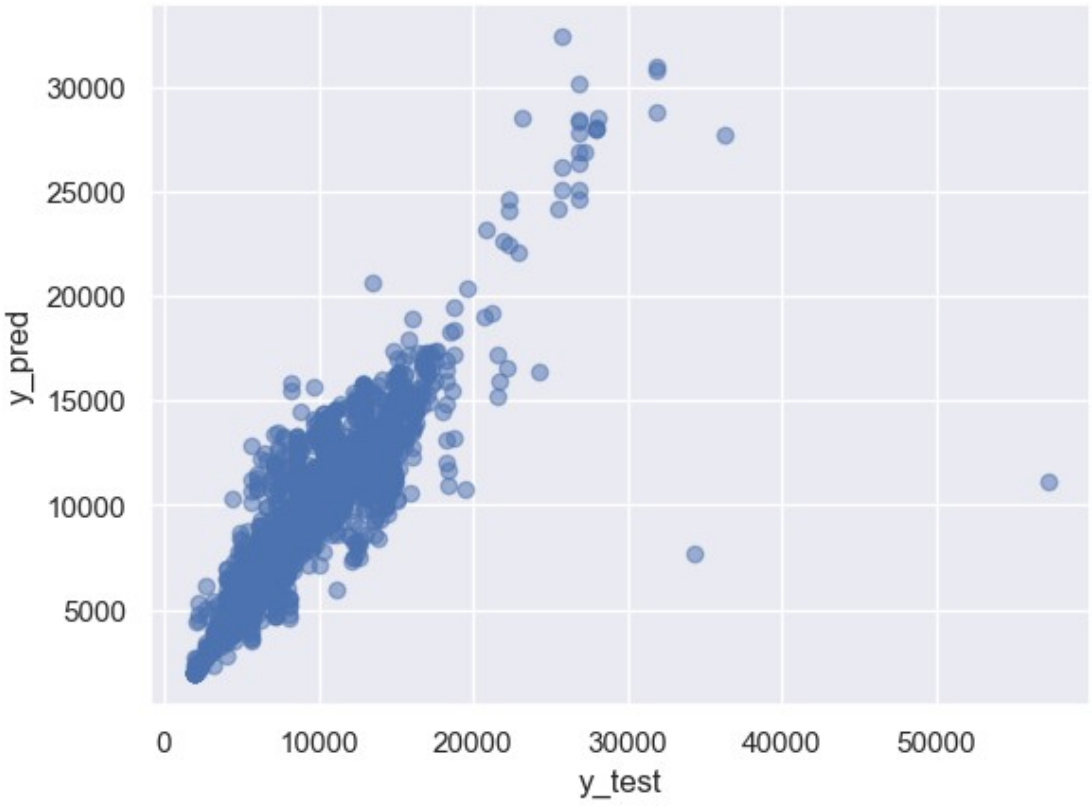
```
0.7971934493754069
```

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

```
C:\Users\nites\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()
```



Metrics

```
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: 1176.0260353790256
MSE: 4372926.267476443
RMSE: 2091.1542906912546
```

```
# RMSE/(max(DV)-min(DV))

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

```
0.026887077025966846
```

```
metrics.r2_score(y_test, y_pred)
```

```
0.7971934493754069
```

#Hyperparameter Tuning . Choose following method for hyperparameter tuning

1. RandomizedSearchCV --> Fast

2. GridSearchCV . Assign hyperparameters in form of dictionery . Fit the model . Check best paramters and best score#

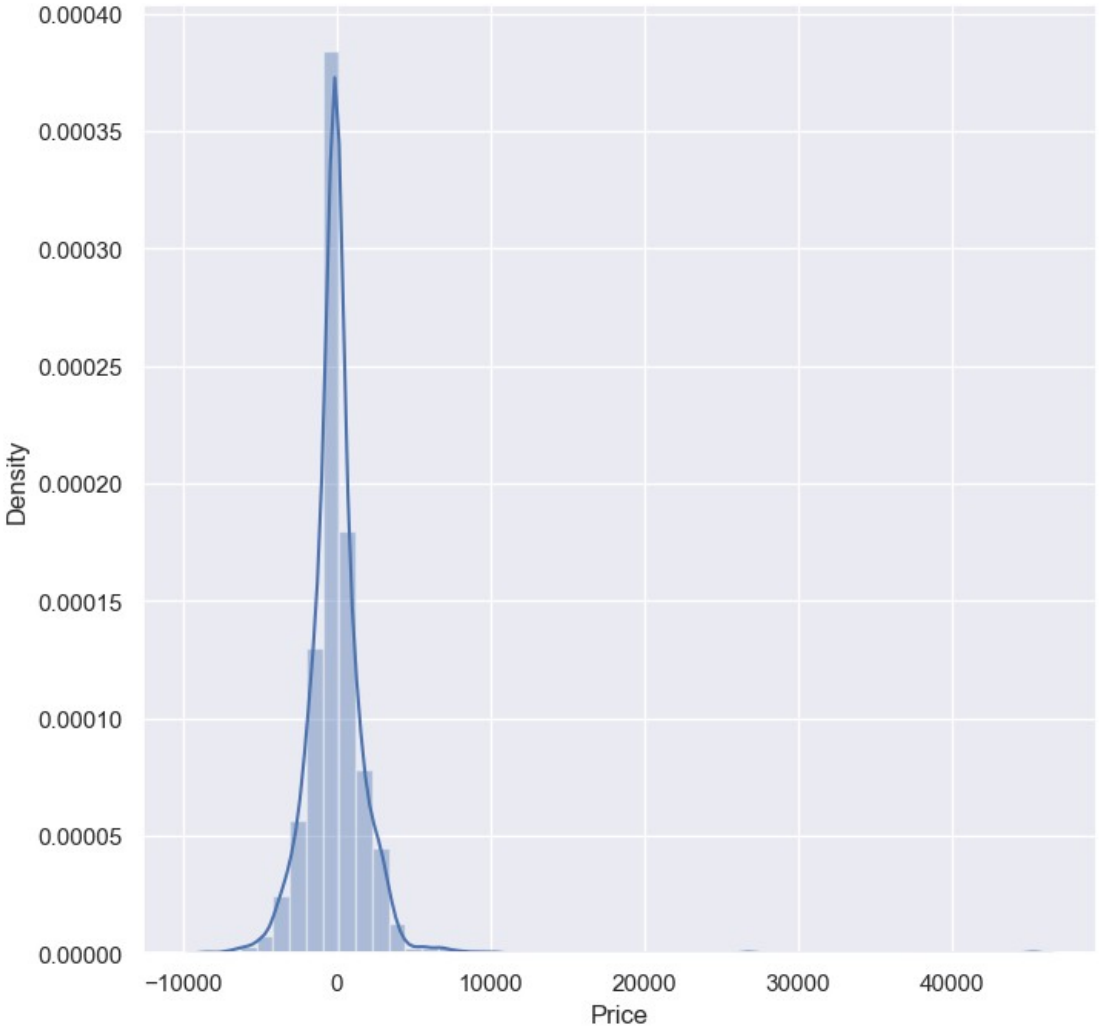
```
from sklearn.model_selection import RandomizedSearchCV

#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200,
num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]

# Create the random grid

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,
# 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] END max_depth=10, max_features=sqrt, min_samples_leaf=5,
min_samples_split=5, n_estimators=900; total time=   3.0s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5,
min_samples_split=5, n_estimators=900; total time=   2.7s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5,
min_samples_split=5, n_estimators=900; total time=   2.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5,
min_samples_split=5, n_estimators=900; total time=   2.7s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5,
min_samples_split=5, n_estimators=900; total time=   2.7s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=1100; total time=   4.4s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=1100; total time=   5.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=1100; total time=   5.4s
```

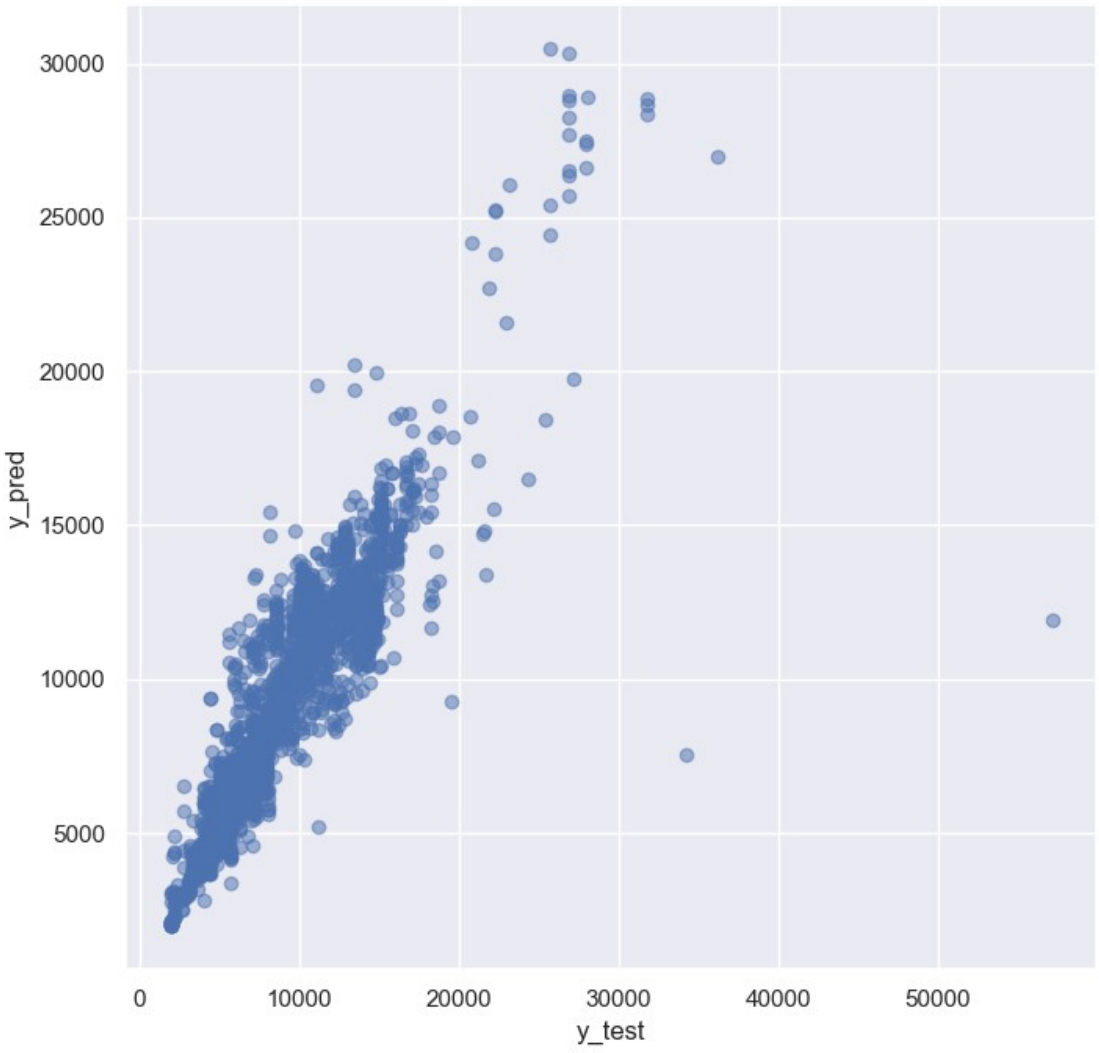
```
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=1100; total time=   5.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=1100; total time=   6.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=100, n_estimators=300; total time=   3.7s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=100, n_estimators=300; total time=   2.9s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=100, n_estimators=300; total time=   2.6s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=100, n_estimators=300; total time=   2.8s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=100, n_estimators=300; total time=   3.3s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=400; total time=   5.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=400; total time=   4.9s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=400; total time=   4.6s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=400; total time=   4.7s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=400; total time=   4.7s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10,
min_samples_split=5, n_estimators=700; total time=   7.3s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10,
min_samples_split=5, n_estimators=700; total time=   7.2s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10,
min_samples_split=5, n_estimators=700; total time=   7.3s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10,
min_samples_split=5, n_estimators=700; total time=   7.2s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10,
min_samples_split=5, n_estimators=700; total time=   9.2s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1,
min_samples_split=2, n_estimators=1000; total time=   7.6s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1,
min_samples_split=2, n_estimators=1000; total time=   6.6s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1,
min_samples_split=2, n_estimators=1000; total time=   6.6s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1,
min_samples_split=2, n_estimators=1000; total time=   6.6s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1,
min_samples_split=2, n_estimators=1000; total time=   6.6s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10,
min_samples_split=15, n_estimators=1100; total time=   2.5s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10,
min_samples_split=15, n_estimators=1100; total time=   2.6s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10,
min_samples_split=15, n_estimators=1100; total time=   2.1s
```

```
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10,
min_samples_split=15, n_estimators=1100; total time=   2.2s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10,
min_samples_split=15, n_estimators=1100; total time=   2.2s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=15, n_estimators=300; total time=   1.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=15, n_estimators=300; total time=   1.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=15, n_estimators=300; total time=   1.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=15, n_estimators=300; total time=   1.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=15, n_estimators=300; total time=   1.0s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=700; total time=   1.4s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=700; total time=   1.3s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=700; total time=   1.3s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=700; total time=   1.4s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=700; total time=   1.3s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=700; total time=   9.1s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=700; total time=   8.9s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=700; total time=   9.6s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=700; total time=   8.8s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=700; total time=   8.9s

RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), 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]},
                   random_state=42, 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}

prediction = rf_random.predict(X_test)

plt.figure(figsize = (8,8))
sns.distplot(y_test-prediction)
plt.show()

C:\Users\nites\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: 1165.1443099178487
MSE: 4054301.6119071487
RMSE: 2013.5296401858973
```

Save the model to reuse it again

```
import pickle
# open a file, where you ant to store the data
file = open('flight_rf.pkl', 'wb')

# dump information to that file
pickle.dump(reg_rf, file)
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_17964\3020833835.py in <module>
      4
      5 # dump information to that file
----> 6 pickle.dump(reg_rf, file)

NameError: name 'reg_rf' is not defined
```

```
model = open('flight_price_rf.pkl','rb')
forest = pickle.load(model)
```

```
---------------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_17964\4234317093.py in <module>
----> 1 model = open('flight_price_rf.pkl','rb')
      2 forest = pickle.load(model)

FileNotFoundError: [Errno 2] No such file or directory:
'flight_price_rf.pkl'
```