Airline Fare Prediction

February 6, 2025

1 Airline Fare Predicition 2025

1.1 Importing Libraries

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

[2]:
```
train_data = pd.read_excel(r"E:\DS Projects\1. Predict Fare of Airlines Tickets␣
↪using Machine Learning/Data_Train.xlsx")
```

[3]:
```
train_data.head(4)
```

[3]:
```
       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

  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]:
```
train_data.tail(4)
```

[4]:
```
           Airline Date_of_Journey    Source Destination  \
10679    Air India      27/04/2019   Kolkata    Banglore
10680  Jet Airways      27/04/2019  Banglore       Delhi
10681      Vistara      01/03/2019  Banglore   New Delhi
10682    Air India       9/05/2019     Delhi      Cochin

                       Route Dep_Time Arrival_Time Duration Total_Stops  \
10679              CCU → BLR    20:45        23:20   2h 35m    non-stop
10680              BLR → DEL    08:20        11:20       3h    non-stop
10681              BLR → DEL    11:30        14:10   2h 40m    non-stop
```

```
 10682  DEL → GOI → BOM → COK    10:55        19:15   8h 20m      2 stops

      Additional_Info  Price
 10679        No info   4145
 10680        No info   7229
 10681        No info  12648
 10682        No info  11753
```

1.2 Data Cleaning: Missing Values

[5]:
```
train_data.info()
```

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

[6]:
```
train_data.isnull().sum()
```

[6]:
```
 Airline            0
 Date_of_Journey    0
 Source             0
 Destination        0
 Route              1
 Dep_Time           0
 Arrival_Time       0
 Duration           0
 Total_Stops        1
 Additional_Info    0
 Price              0
 dtype: int64
```

[7]:
```
train_data['Total_Stops'].isnull()
```

[7]:
```
0        False
1        False
2        False
3        False
4        False
         …
10678    False
10679    False
10680    False
10681    False
10682    False
Name: Total_Stops, Length: 10683, dtype: bool
```

[8]:
```
train_data[train_data['Total_Stops'].isnull()]
```

[8]:
```
        Airline Date_of_Journey Source Destination Route Dep_Time  \
9039  Air India       6/05/2019  Delhi      Cochin   NaN    09:45

      Arrival_Time Duration Total_Stops Additional_Info  Price
9039  09:25 07 May  23h 40m         NaN         No info   7480
```

[9]:
```
train_data.dropna(inplace=True)
```

[10]:
```
train_data.isnull().sum()
```

[10]:
```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Additional_Info    0
Price              0
dtype: int64
```

[11]:
```
train_data.dtypes
```

[11]:
```
Airline            object
Date_of_Journey    object
Source             object
Destination        object
Route              object
Dep_Time           object
Arrival_Time       object
Duration           object
```

```
Total_Stops        object
Additional_Info    object
Price               int64
dtype: object
```

[12]:
```
train_data.info(memory_usage="deep")
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10682 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          10682 non-null  object
 1   Date_of_Journey  10682 non-null  object
 2   Source           10682 non-null  object
 3   Destination      10682 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10682 non-null  object
 6   Arrival_Time     10682 non-null  object
 7   Duration         10682 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10682 non-null  object
 10  Price            10682 non-null  int64 
dtypes: int64(1), object(10)
memory usage: 6.3 MB
```

1.3 Pre-Processing & Extraction of Derived Attributes

[13]:
```
data = train_data.copy()
```

[14]:
```
data.columns
```

[14]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price'],
      dtype='object')
```

[15]:
```
data.head(2)
```

[15]:
```
     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   

  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  
```

[16]:
```
data.dtypes
```

[16]:
```
Airline            object
Date_of_Journey    object
Source             object
Destination        object
Route              object
Dep_Time           object
Arrival_Time       object
Duration           object
Total_Stops        object
Additional_Info    object
Price               int64
dtype: object
```

[17]:
```
import warnings
from warnings import filterwarnings
filterwarnings("ignore")
```

1.4 Changing into Date Data type

[18]:
```
def change_into_Datetime(col):
    data[col] = pd.to_datetime(data[col])
```

[19]:
```
data.columns
```

[19]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price'],
      dtype='object')
```

[20]:
```
for feature in ['Dep_Time', 'Arrival_Time','Date_of_Journey']:
    change_into_Datetime(feature)
```

[21]:
```
data.dtypes
```

[21]:
```
Airline                    object
Date_of_Journey    datetime64[ns]
Source                     object
Destination                object
Route                      object
Dep_Time           datetime64[ns]
Arrival_Time       datetime64[ns]
Duration                   object
Total_Stops                object
Additional_Info            object
Price                       int64
dtype: object
```

1.5 Separating Day, Month and Year

[22]:
```
data["Journey_day"] = data['Date_of_Journey'].dt.day
```

[23]:
```
data["Journey_month"] = data['Date_of_Journey'].dt.month
```

[24]:
```
data["Journey_year"] = data['Date_of_Journey'].dt.year
```

[25]:
```
data.head(3)
```

[25]:
```
       Airline Date_of_Journey    Source Destination                  Route  \
0       IndiGo      2019-03-24  Banglore   New Delhi              BLR → DEL
1    Air India      2019-05-01   Kolkata    Banglore  CCU → IXR → BBI → BLR
2  Jet Airways      2019-06-09     Delhi      Cochin  DEL → LKO → BOM → COK

             Dep_Time        Arrival_Time Duration Total_Stops  \
0 2025-02-06 22:20:00 2025-03-22 01:10:00   2h 50m    non-stop
1 2025-02-06 05:50:00 2025-02-06 13:15:00   7h 25m     2 stops
2 2025-02-06 09:25:00 2025-06-10 04:25:00      19h     2 stops

  Additional_Info  Price  Journey_day  Journey_month  Journey_year
0         No info   3897           24              3          2019
1         No info   7662            1              5          2019
2         No info  13882            9              6          2019
```

1.6 Cleaning Dep_Time and Arrival_Time & Extracting Derived Attributes (Hour and Mins)

[26]:
```
def extract_hour_min(df , col):
   df[col+"_hour"] = df[col].dt.hour
   df[col+"_minute"] = df[col].dt.minute
   return df.head(3)
```

[27]:
```
data.columns
```

[27]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price', 'Journey_day', 'Journey_month',
       'Journey_year'],
      dtype='object')
```

[28]:
```
extract_hour_min(data, "Dep_Time")
```

[28]:
```
       Airline Date_of_Journey    Source Destination                  Route  \
0       IndiGo      2019-03-24  Banglore   New Delhi              BLR → DEL
1    Air India      2019-05-01   Kolkata    Banglore  CCU → IXR → BBI → BLR
2  Jet Airways      2019-06-09     Delhi      Cochin  DEL → LKO → BOM → COK
```

```
              Dep_Time        Arrival_Time Duration Total_Stops  \
0 2025-02-06 22:20:00 2025-03-22 01:10:00   2h 50m    non-stop
1 2025-02-06 05:50:00 2025-02-06 13:15:00   7h 25m     2 stops
2 2025-02-06 09:25:00 2025-06-10 04:25:00      19h     2 stops

  Additional_Info  Price  Journey_day  Journey_month  Journey_year  \
0         No info   3897           24              3          2019
1         No info   7662            1              5          2019
2         No info  13882            9              6          2019

   Dep_Time_hour  Dep_Time_minute
0             22               20
1              5               50
2              9               25
```

[29]:
```
extract_hour_min(data, "Arrival_Time")
```

[29]:
```
       Airline Date_of_Journey    Source Destination                  Route  \
0       IndiGo      2019-03-24  Banglore   New Delhi              BLR → DEL
1    Air India      2019-05-01   Kolkata    Banglore  CCU → IXR → BBI → BLR
2  Jet Airways      2019-06-09     Delhi      Cochin  DEL → LKO → BOM → COK

              Dep_Time        Arrival_Time Duration Total_Stops  \
0 2025-02-06 22:20:00 2025-03-22 01:10:00   2h 50m    non-stop
1 2025-02-06 05:50:00 2025-02-06 13:15:00   7h 25m     2 stops
2 2025-02-06 09:25:00 2025-06-10 04:25:00      19h     2 stops

  Additional_Info  Price  Journey_day  Journey_month  Journey_year  \
0         No info   3897           24              3          2019
1         No info   7662            1              5          2019
2         No info  13882            9              6          2019

   Dep_Time_hour  Dep_Time_minute  Arrival_Time_hour  Arrival_Time_minute
0             22               20                  1                   10
1              5               50                 13                   15
2              9               25                  4                   25
```

[30]:
```
cols_to_drop = ['Arrival_Time', 'Dep_Time']
data.drop(cols_to_drop , axis=1 , inplace=True)
```

[31]:
```
data.head(3)
```

[31]:
```
       Airline Date_of_Journey    Source Destination                  Route  \
0       IndiGo      2019-03-24  Banglore   New Delhi              BLR → DEL
1    Air India      2019-05-01   Kolkata    Banglore  CCU → IXR → BBI → BLR
2  Jet Airways      2019-06-09     Delhi      Cochin  DEL → LKO → BOM → COK
```

```
  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

   Journey_year  Dep_Time_hour  Dep_Time_minute  Arrival_Time_hour  \
0          2019             22               20                  1
1          2019              5               50                 13
2          2019              9               25                  4

   Arrival_Time_minute
0                   10
1                   15
2                   25
```

[32]:
```
data.shape
```

[32]:
```
(10682, 16)
```

1.7 Analysis: Most of the flights take-off

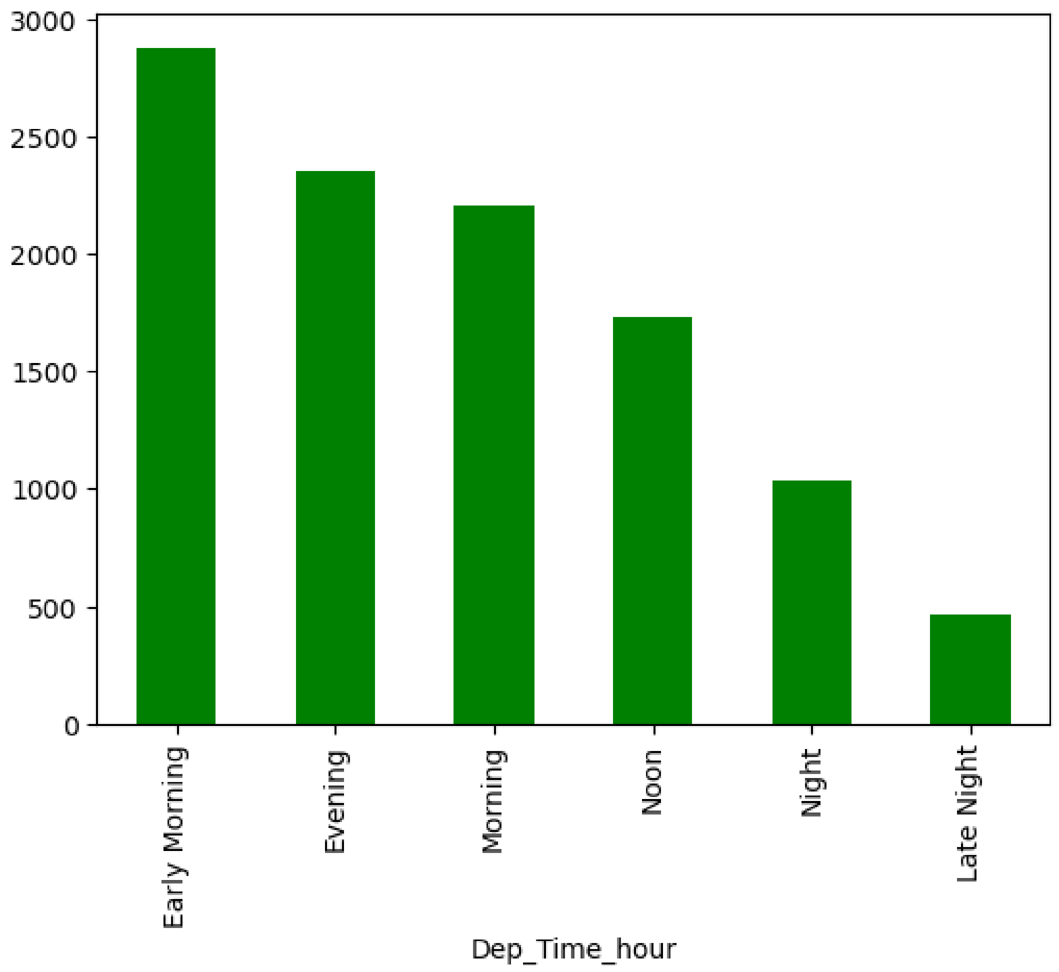
[33]:
```
data.columns
```

[33]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Duration', 'Total_Stops', 'Additional_Info', 'Price', 'Journey_day',
       'Journey_month', 'Journey_year', 'Dep_Time_hour', 'Dep_Time_minute',
       'Arrival_Time_hour', 'Arrival_Time_minute'],
      dtype='object')
```

[34]:
```
def flight_dep_time(x):
    if (x>4) and (x<=8):
        return "Early Morning"

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

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

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

    elif (x>20) and (x<=24):
        return "Night"
    else:
        return "Late Night"
```

[35]:
```
data['Dep_Time_hour'].apply(flight_dep_time).value_counts().plot(kind="bar" ,␣
↪color="g")
```

[35]:
```
<Axes: xlabel='Dep_Time_hour'>
```



[36]:
```
import plotly
import cufflinks as cf
from cufflinks.offline import go_offline
from plotly.offline import plot, iplot, init_notebook_mode , download_plotlyjs
init_notebook_mode(connected=True)
cf.go_offline()
```

[37]:
```
data['Dep_Time_hour'].apply(flight_dep_time).value_counts().iplot(kind="bar")
```

[38]:
```
data['Dep_Time_hour'].apply(flight_dep_time).value_counts()
```

[38]:
```
Dep_Time_hour
Early Morning    2880
Evening          2357
Morning          2209
Noon             1731
Night            1040
Late Night        465
Name: count, dtype: int64
```

1.8 Pre-Processing on Duration Feature

[39]:
```
def preprocess_duration(x):
    if 'h' not in x:
        x = '0h' + ' ' + x
    elif 'm' not in x:
        x = x + ' ' + '0m'

    return x
```

[40]:
```
data['Duration'] = data['Duration'].apply(preprocess_duration)
```

[41]:
```
data['Duration']
```

[41]:
```
0         2h 50m
1         7h 25m
2         19h 0m
3         5h 25m
4         4h 45m
           …
10678     2h 30m
10679     2h 35m
10680      3h 0m
10681     2h 40m
10682     8h 20m
Name: Duration, Length: 10682, dtype: object
```

[42]:
```
data['Duration'][0]
```

[42]:
```
'2h 50m'
```

[43]:
```
'2h 50m'.split(' ')
```

[43]:
```
['2h', '50m']
```

[44]:
```
'2h 50m'.split(' ')[0]
```

[44]:
```
'2h'
```

[45]:
```
'2h 50m'.split(' ')[0][0:-1]
```

[45]:
```
'2'
```

[46]:
```
type('2h 50m'.split(' ')[0][0:-1])
```

[46]:
```
str
```

[47]:
```
int('2h 50m'.split(' ')[0][0:-1])
```

[47]:
```
2
```

[48]:
```
int('2h 50m'.split(' ')[1][0:-1])
```

[48]:
```
50
```

1.9 Using Lambda as Annonymous function

[49]:
```
data['Duration_hours'] = data['Duration'].apply(lambda x: int(x.split(' ')[0][0:
 ↪-1]))
```

[50]:
```
data['Duration_mins'] = data['Duration'].apply(lambda x: int(x.split(' ')[1][0:
 ↪-1]))
```

[51]:
```
data.head(2)
```

[51]:
```
     Airline Date_of_Journey    Source Destination                  Route  \
0     IndiGo      2019-03-24  Banglore   New Delhi              BLR → DEL
1  Air India      2019-05-01   Kolkata    Banglore  CCU → IXR → BBI → BLR

  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

   Journey_year  Dep_Time_hour  Dep_Time_minute  Arrival_Time_hour  \
0          2019             22               20                  1
1          2019              5               50                 13

   Arrival_Time_minute  Duration_hours  Duration_mins
0                   10               2             50
1                   15               7             25
```

1.10 Analysis: Wheteher Duration Impacts On Price Or Not?

[52]:
```
data['Duration']
```

[52]:
```
0        2h 50m
1        7h 25m
```

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

```
[53]: 2*60
```

```
[53]: 120
```

```
[54]: '2*60'
```

```
[54]: '2*60'
```

```
[55]: eval('2*60')
```

```
[55]: 120
```

```
[56]: data['Duration'].str.replace('h' , "*60").str.replace(' ' , '+').str.
       ↪replace('m' , "*1")
```

```
[56]: 0         2*60+50*1
      1         7*60+25*1
      2         19*60+0*1
      3         5*60+25*1
      4         4*60+45*1
                   …
      10678     2*60+30*1
      10679     2*60+35*1
      10680      3*60+0*1
      10681     2*60+40*1
      10682     8*60+20*1
      Name: Duration, Length: 10682, dtype: object
```

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

```
[58]: data['Duration_total_mins']
```

```
[58]: 0          170
      1          445
      2         1140
      3          325
```

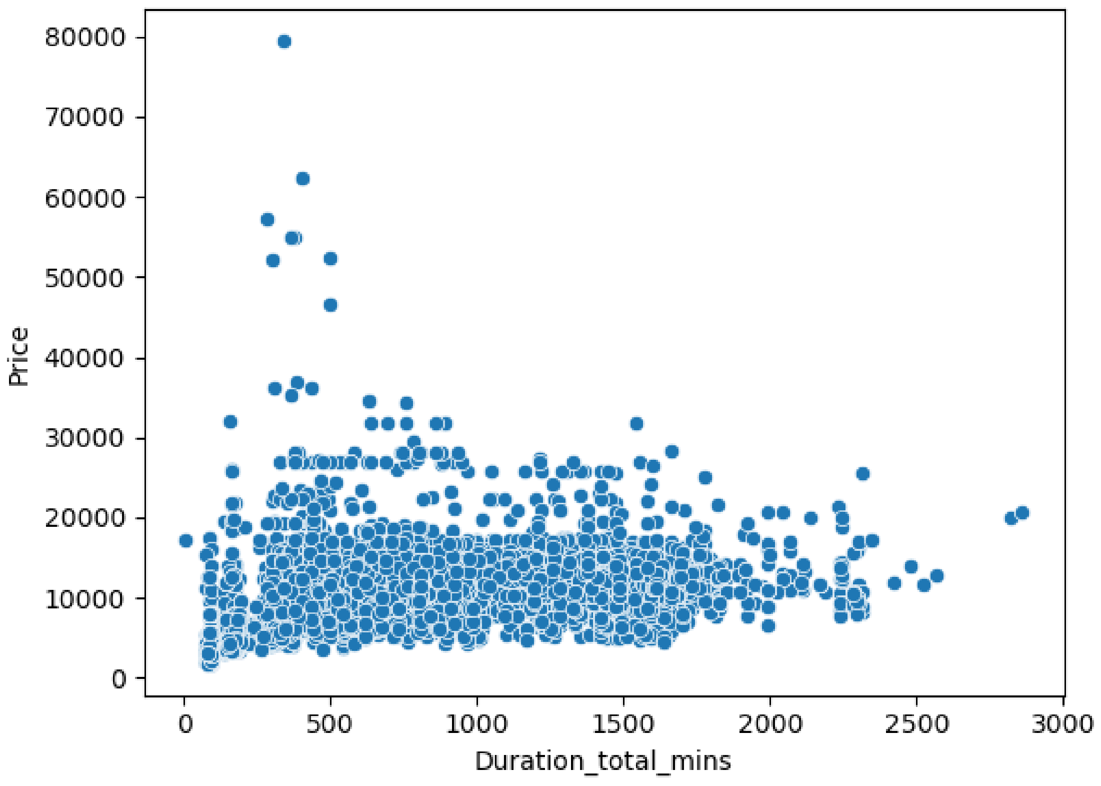
```
4          285
          …
10678      150
10679      155
10680      180
10681      160
10682      500
Name: Duration_total_mins, Length: 10682, dtype: int64
```

[59]:
```
data.columns
```

[59]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Duration', 'Total_Stops', 'Additional_Info', 'Price', 'Journey_day',
       'Journey_month', 'Journey_year', 'Dep_Time_hour', 'Dep_Time_minute',
       'Arrival_Time_hour', 'Arrival_Time_minute', 'Duration_hours',
       'Duration_mins', 'Duration_total_mins'],
      dtype='object')
```

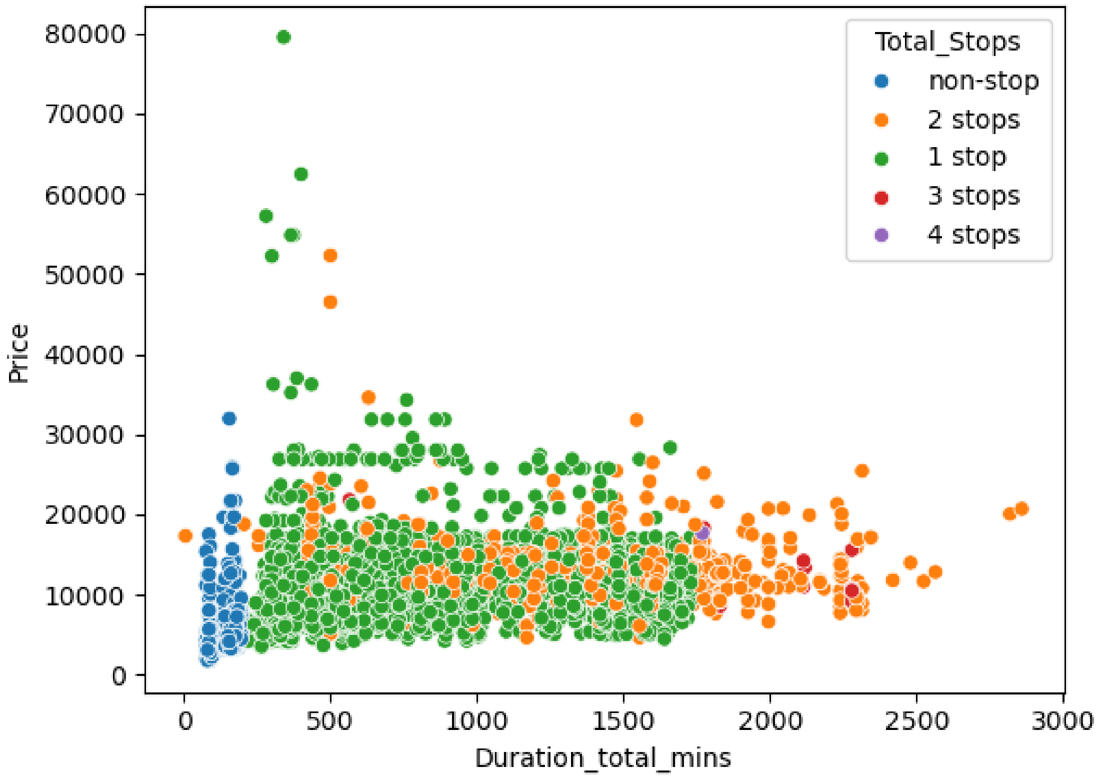
[60]:
```
sns.scatterplot(x="Duration_total_mins" , y="Price" , data=data)
```

[60]:
```
<Axes: xlabel='Duration_total_mins', ylabel='Price'>
```



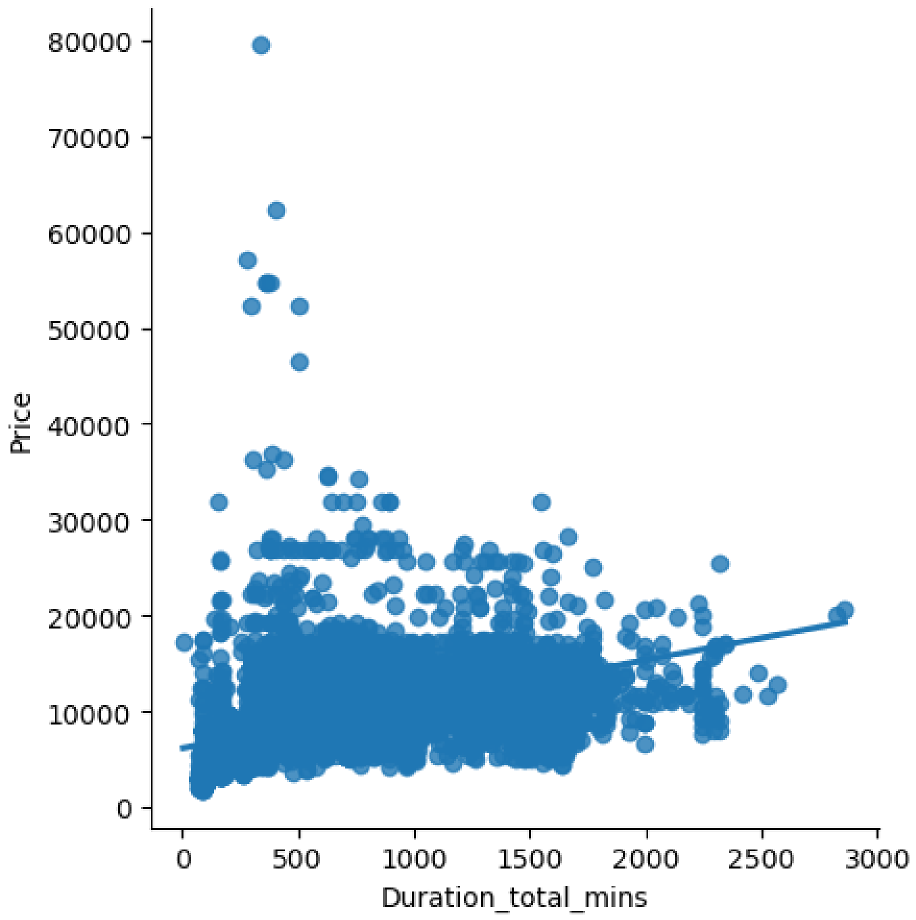
[61]:
```
sns.scatterplot(x="Duration_total_mins" , y="Price" , hue="Total_Stops",␣
    ↪data=data)
```

[61]:
```
<Axes: xlabel='Duration_total_mins', ylabel='Price'>
```



[62]:
```
sns.lmplot(x="Duration_total_mins" , y="Price" , data=data)
```

[62]:
```
<seaborn.axisgrid.FacetGrid at 0x19383ac5430>
```



1.11 Analysis:

1.12 1) On which route Jet Airways is extremely used?

1.13 2) Airline vs Price Analysis

[63]:
```
data[data['Airline']=='Jet Airways'].groupby('Route').size().
↪sort_values(ascending=False)
```

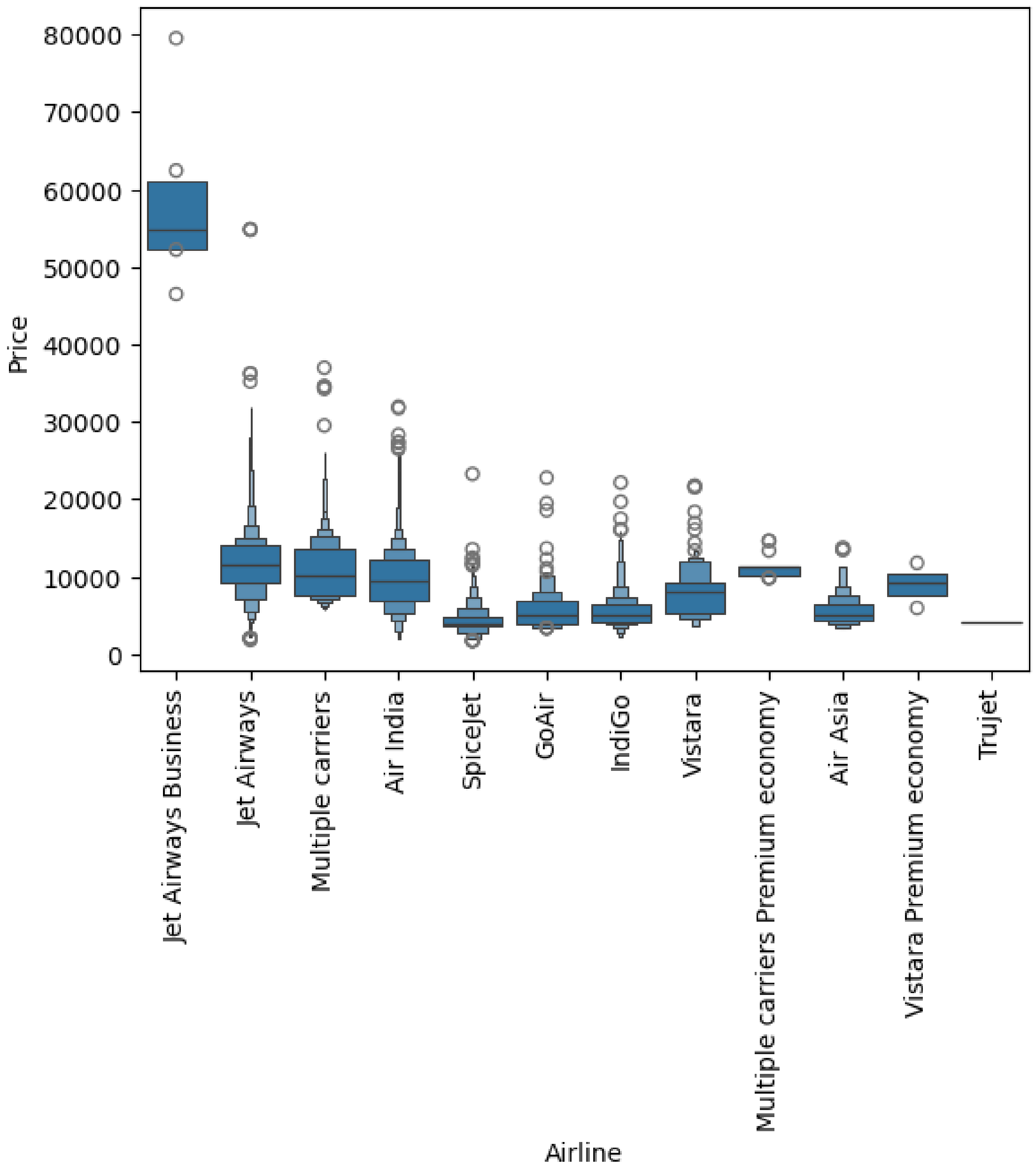
[63]:
```
Route
CCU → BOM → BLR          930
DEL → BOM → COK          875
BLR → BOM → DEL          385
BLR → DEL                382
CCU → DEL → BLR          300
BOM → HYD                207
DEL → JAI → BOM → COK    207
DEL → AMD → BOM → COK    141
DEL → IDR → BOM → COK      86
DEL → NAG → BOM → COK      61
DEL → ATQ → BOM → COK      38
DEL → COK                  34
DEL → BHO → BOM → COK      29
DEL → BDQ → BOM → COK      28
DEL → LKO → BOM → COK      25
DEL → JDH → BOM → COK      23
CCU → GAU → BLR            22
DEL → MAA → BOM → COK      16
DEL → IXC → BOM → COK      13
BLR → MAA → DEL            10
BLR → BDQ → DEL             8
DEL → UDR → BOM → COK       7
BOM → DEL → HYD             5
CCU → BOM → PNQ → BLR       4
BLR → BOM → JDH → DEL       3
DEL → DED → BOM → COK       2
BOM → BDQ → DEL → HYD       2
DEL → CCU → BOM → COK       1
BOM → VNS → DEL → HYD       1
BOM → UDR → DEL → HYD       1
BOM → JDH → DEL → HYD       1
BOM → IDR → DEL → HYD       1
BOM → DED → DEL → HYD       1
dtype: int64
```

[64]:
```
data.columns
```

[64]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Duration', 'Total_Stops', 'Additional_Info', 'Price', 'Journey_day',
       'Journey_month', 'Journey_year', 'Dep_Time_hour', 'Dep_Time_minute',
       'Arrival_Time_hour', 'Arrival_Time_minute', 'Duration_hours',
       'Duration_mins', 'Duration_total_mins'],
      dtype='object')
```

[65]:
```
sns.boxenplot(y='Price' , x='Airline' , data=data.sort_values('Price' ,␣
 ↪ascending=False))
plt.xticks(rotation="vertical")
plt.show
```

[65]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



1.14 Feature Engineering: One Hot Encoding

[66]:
```
data.head(2)
```

[66]:
```
      Airline Date_of_Journey    Source Destination                  Route  \
0      IndiGo      2019-03-24  Banglore   New Delhi              BLR → DEL
1   Air India      2019-05-01   Kolkata   Banglore  CCU → IXR → BBI → BLR

  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

   Journey_year  Dep_Time_hour  Dep_Time_minute  Arrival_Time_hour  \
0          2019             22               20                  1
1          2019              5               50                 13

   Arrival_Time_minute  Duration_hours  Duration_mins  Duration_total_mins
0                   10               2             50                  170
1                   15               7             25                  445
```

[67]:
```
cat_col = [col for col in data.columns if data[col].dtype=="object"]
```

[68]:
```
num_col = [col for col in data.columns if data[col].dtype!="object"]
```

[69]:
```
cat_col
```

[69]:
```
['Airline',
 'Source',
 'Destination',
 'Route',
 'Duration',
 'Total_Stops',
 'Additional_Info']
```

[70]:
```
data['Source'].unique()
```

[70]:
```
array(['Banglore', 'Kolkata', 'Delhi', 'Chennai', 'Mumbai'], dtype=object)
```

[71]:
```
data['Source'].apply(lambda x : 1 if x =='Banglore' else 0)
```

[71]:
```
0        1
1        0
2        0
3        0
4        1
        ..
10678    0
10679    0
10680    1
10681    1
10682    0
Name: Source, Length: 10682, dtype: int64
```

[72]:
```
for sub_category in data['Source'].unique():
    data['Source_'+sub_category] = data['Source'].apply(lambda x : 1 if x␣
 ↪==sub_category else 0)
```

```
[73]: data.head(3)
```

```
[73]:        Airline Date_of_Journey    Source Destination                  Route  \
      0       IndiGo      2019-03-24  Banglore   New Delhi              BLR → DEL
      1    Air India      2019-05-01   Kolkata    Banglore  CCU → IXR → BBI → BLR
      2  Jet Airways      2019-06-09     Delhi      Cochin  DEL → LKO → BOM → COK

        Duration Total_Stops Additional_Info  Price  Journey_day  …  \
      0   2h 50m    non-stop         No info   3897           24  …
      1   7h 25m     2 stops         No info   7662            1  …
      2   19h 0m     2 stops         No info  13882            9  …

         Arrival_Time_hour  Arrival_Time_minute  Duration_hours  Duration_mins  \
      0                  1                   10               2             50
      1                 13                   15               7             25
      2                  4                   25              19              0

         Duration_total_mins  Source_Banglore  Source_Kolkata  Source_Delhi  \
      0                  170                1               0             0
      1                  445                0               1             0
      2                 1140                0               0             1

         Source_Chennai  Source_Mumbai
      0               0              0
      1               0              0
      2               0              0

      [3 rows x 24 columns]
```

```
[74]: cat_col
```

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

```
[75]: data.head(2)
```

```
[75]:      Airline Date_of_Journey    Source Destination                  Route  \
      0     IndiGo      2019-03-24  Banglore   New Delhi              BLR → DEL
      1  Air India      2019-05-01   Kolkata    Banglore  CCU → IXR → BBI → BLR

        Duration Total_Stops Additional_Info  Price  Journey_day  …  \
      0   2h 50m    non-stop         No info   3897           24  …
```

```
1   7h 25m     2 stops         No info   7662            1  …

   Arrival_Time_hour  Arrival_Time_minute  Duration_hours  Duration_mins  \
0                  1                   10               2             50
1                 13                   15               7             25

   Duration_total_mins  Source_Banglore  Source_Kolkata  Source_Delhi  \
0                  170                1               0             0
1                  445                0               1             0

   Source_Chennai  Source_Mumbai
0               0              0
1               0              0

[2 rows x 24 columns]
```

[76]:
```
data['Airline'].nunique()
```

[76]:
```
12
```

[77]:
```
data['Airline'].unique()
```

[77]:
```
array(['IndiGo', 'Air India', 'Jet Airways', 'SpiceJet',
       'Multiple carriers', 'GoAir', 'Vistara', 'Air Asia',
       'Vistara Premium economy', 'Jet Airways Business',
       'Multiple carriers Premium economy', 'Trujet'], dtype=object)
```

1.15 Optimized Encoding: Target Guided Encoding

[78]:
```
data.groupby(['Airline'])['Price'].mean().sort_values()
```

[78]:
```
Airline
Trujet                                4140.000000
SpiceJet                              4338.284841
Air Asia                              5590.260188
IndiGo                                5673.682903
GoAir                                 5861.056701
Vistara                               7796.348643
Vistara Premium economy               8962.333333
Air India                             9612.427756
Multiple carriers                    10902.678094
Multiple carriers Premium economy    11418.846154
Jet Airways                          11643.923357
Jet Airways Business                 58358.666667
Name: Price, dtype: float64
```

[79]:
```
airlines = data.groupby(['Airline'])['Price'].mean().sort_values().index
```

```
[80]: airlines
```

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

1.16 Enumration

```
[81]: dict_airlines = {key:index for index , key in enumerate(airlines , 0)}
```

```
[82]: dict_airlines
```

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

```
[83]: data['Airline'] = data['Airline'].map(dict_airlines)
```

```
[84]: data['Airline']
```

```
[84]: 0         3
      1         7
      2        10
      3         3
      4         3
               ..
      10678     2
      10679     7
      10680    10
      10681     5
      10682     7
      Name: Airline, Length: 10682, dtype: int64
```

```
[85]: data.head(3)
```

```
[85]:    Airline Date_of_Journey    Source Destination                  Route  \
      0        3      2019-03-24  Banglore   New Delhi              BLR → DEL
```

```
1        7      2019-05-01   Kolkata    Banglore  CCU → IXR → BBI → BLR
2       10      2019-06-09     Delhi      Cochin  DEL → LKO → BOM → COK

  Duration Total_Stops Additional_Info  Price  Journey_day  …  \
0   2h 50m    non-stop         No info   3897           24  …
1   7h 25m     2 stops         No info   7662            1  …
2   19h 0m     2 stops         No info  13882            9  …

   Arrival_Time_hour  Arrival_Time_minute  Duration_hours  Duration_mins  \
0                  1                   10               2             50
1                 13                   15               7             25
2                  4                   25              19              0

   Duration_total_mins  Source_Banglore  Source_Kolkata  Source_Delhi  \
0                  170                1               0             0
1                  445                0               1             0
2                 1140                0               0             1

   Source_Chennai  Source_Mumbai
0               0              0
1               0              0
2               0              0

[3 rows x 24 columns]
```

[86]:
```python
data['Destination'].unique()
```

[86]:
```
array(['New Delhi', 'Banglore', 'Cochin', 'Kolkata', 'Delhi', 'Hyderabad'],
      dtype=object)
```

[87]:
```python
data['Destination'].replace('New Delhi' , 'Delhi' , inplace=True)
```

[88]:
```python
data['Destination'].unique()
```

[88]:
```
array(['Delhi', 'Banglore', 'Cochin', 'Kolkata', 'Hyderabad'],
      dtype=object)
```

[89]:
```python
dest = data.groupby(['Destination'])['Price'].mean().sort_values().index
```

[90]:
```python
dest
```

[90]:
```
Index(['Kolkata', 'Hyderabad', 'Delhi', 'Banglore', 'Cochin'], dtype='object',
name='Destination')
```

[91]:
```python
dict_dest = {key:index for index , key in enumerate(dest , 0)}
```

[92]:
```python
dict_dest
```

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

```
[93]: data['Destination'] = data['Destination'].map(dict_dest)
```

```
[94]: data['Destination']
```

```
[94]: 0        2
      1        3
      2        4
      3        3
      4        2
              ..
      10678    3
      10679    3
      10680    2
      10681    2
      10682    4
      Name: Destination, Length: 10682, dtype: int64
```

```
[95]: data.head(3)
```

```
[95]:    Airline Date_of_Journey    Source  Destination                  Route  \
      0        3      2019-03-24  Banglore            2              BLR → DEL
      1        7      2019-05-01   Kolkata            3  CCU → IXR → BBI → BLR
      2       10      2019-06-09     Delhi            4  DEL → LKO → BOM → COK

        Duration Total_Stops Additional_Info  Price  Journey_day  …  \
      0   2h 50m    non-stop         No info   3897           24  …
      1   7h 25m     2 stops         No info   7662            1  …
      2   19h 0m     2 stops         No info  13882            9  …

         Arrival_Time_hour  Arrival_Time_minute  Duration_hours  Duration_mins  \
      0                  1                   10               2             50
      1                 13                   15               7             25
      2                  4                   25              19              0

         Duration_total_mins  Source_Banglore  Source_Kolkata  Source_Delhi  \
      0                  170                1               0             0
      1                  445                0               1             0
      2                 1140                0               0             1

         Source_Chennai  Source_Mumbai
      0               0              0
      1               0              0
      2               0              0

      [3 rows x 24 columns]
```

1.17 Manual Label Encoding

[96]:
```
data['Total_Stops']
```

[96]:
```
0         non-stop
1          2 stops
2          2 stops
3           1 stop
4           1 stop
            …
10678     non-stop
10679     non-stop
10680     non-stop
10681     non-stop
10682      2 stops
Name: Total_Stops, Length: 10682, dtype: object
```

[97]:
```
data['Total_Stops'].unique()
```

[97]:
```
array(['non-stop', '2 stops', '1 stop', '3 stops', '4 stops'],
      dtype=object)
```

[98]:
```
stop = {'non-stop':0, '2 stops':2, '1 stop':1, '3 stops':3, '4 stops':4}
```

[99]:
```
data['Total_Stops'] = data['Total_Stops'].map(stop)
```

[100]:
```
data['Total_Stops']
```

[100]:
```
0        0
1        2
2        2
3        1
4        1
        ..
10678    0
10679    0
10680    0
10681    0
10682    2
Name: Total_Stops, Length: 10682, dtype: int64
```

[101]:
```
data.columns
```

[101]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Duration', 'Total_Stops', 'Additional_Info', 'Price', 'Journey_day',
       'Journey_month', 'Journey_year', 'Dep_Time_hour', 'Dep_Time_minute',
       'Arrival_Time_hour', 'Arrival_Time_minute', 'Duration_hours',
       'Duration_mins', 'Duration_total_mins', 'Source_Banglore',
       'Source_Kolkata', 'Source_Delhi', 'Source_Chennai', 'Source_Mumbai'],
      dtype='object')
```

[102]:
```
data['Additional_Info'].value_counts()/len(data)*100
```

[102]:
```
Additional_Info
No info                         78.112713
In-flight meal not included     18.554578
No check-in baggage included     2.995694
1 Long layover                   0.177869
Change airports                  0.065531
Business class                   0.037446
No Info                          0.028085
1 Short layover                  0.009362
Red-eye flight                   0.009362
2 Long layover                   0.009362
Name: count, dtype: float64
```

[103]:
```
data.head(4)
```

[103]:
```
   Airline Date_of_Journey    Source  Destination                  Route  \
0        3      2019-03-24  Banglore            2              BLR → DEL
1        7      2019-05-01   Kolkata            3  CCU → IXR → BBI → BLR
2       10      2019-06-09     Delhi            4  DEL → LKO → BOM → COK
3        3      2019-05-12   Kolkata            3        CCU → NAG → BLR

  Duration  Total_Stops Additional_Info  Price  Journey_day  …  \
0   2h 50m            0         No info   3897           24  …
1   7h 25m            2         No info   7662            1  …
2   19h 0m            2         No info  13882            9  …
3   5h 25m            1         No info   6218           12  …

   Arrival_Time_hour  Arrival_Time_minute  Duration_hours  Duration_mins  \
0                  1                   10               2             50
1                 13                   15               7             25
2                  4                   25              19              0
3                 23                   30               5             25

   Duration_total_mins  Source_Banglore  Source_Kolkata  Source_Delhi  \
0                  170                1               0             0
1                  445                0               1             0
2                 1140                0               0             1
3                  325                0               1             0

   Source_Chennai  Source_Mumbai
0               0              0
1               0              0
```

```
2                0              0
3                0              0

[4 rows x 24 columns]
```

[104]:
```
data.columns
```

[104]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Duration', 'Total_Stops', 'Additional_Info', 'Price', 'Journey_day',
       'Journey_month', 'Journey_year', 'Dep_Time_hour', 'Dep_Time_minute',
       'Arrival_Time_hour', 'Arrival_Time_minute', 'Duration_hours',
       'Duration_mins', 'Duration_total_mins', 'Source_Banglore',
       'Source_Kolkata', 'Source_Delhi', 'Source_Chennai', 'Source_Mumbai'],
      dtype='object')
```

[105]:
```
data.drop(columns=['Date_of_Journey' , 'Additional_Info' , 'Source' ,␣
 ↪'Journey_year'] , axis=1 , inplace=True)
```

[106]:
```
data.drop(columns=['Route'] , axis=1 , inplace=True)
```

[107]:
```
data.columns
```

[107]:
```
Index(['Airline', 'Destination', 'Duration', 'Total_Stops', 'Price',
       'Journey_day', 'Journey_month', 'Dep_Time_hour', 'Dep_Time_minute',
       'Arrival_Time_hour', 'Arrival_Time_minute', 'Duration_hours',
       'Duration_mins', 'Duration_total_mins', 'Source_Banglore',
       'Source_Kolkata', 'Source_Delhi', 'Source_Chennai', 'Source_Mumbai'],
      dtype='object')
```

[108]:
```
data.columns
```

[108]:
```
Index(['Airline', 'Destination', 'Duration', 'Total_Stops', 'Price',
       'Journey_day', 'Journey_month', 'Dep_Time_hour', 'Dep_Time_minute',
       'Arrival_Time_hour', 'Arrival_Time_minute', 'Duration_hours',
       'Duration_mins', 'Duration_total_mins', 'Source_Banglore',
       'Source_Kolkata', 'Source_Delhi', 'Source_Chennai', 'Source_Mumbai'],
      dtype='object')
```

[109]:
```
data.head(3)
```

[109]:
```
   Airline  Destination Duration  Total_Stops  Price  Journey_day  \
0        3            2   2h 50m            0   3897           24
1        7            3   7h 25m            2   7662            1
2       10            4   19h 0m            2  13882            9

   Journey_month  Dep_Time_hour  Dep_Time_minute  Arrival_Time_hour  \
0              3             22               20                  1
1              5              5               50                 13
```

```
2             6             9            25                  4

   Arrival_Time_minute  Duration_hours  Duration_mins  Duration_total_mins  \
0                   10               2             50                  170
1                   15               7             25                  445
2                   25              19              0                 1140

   Source_Banglore  Source_Kolkata  Source_Delhi  Source_Chennai  \
0                1               0             0               0
1                0               1             0               0
2                0               0             1               0

   Source_Mumbai
0              0
1              0
2              0
```

[110]:
```
data.drop(columns=['Duration_total_mins'] , axis=1 , inplace=True)
```

[111]:
```
data.drop(columns=['Duration'] , axis=1 , inplace=True)
```

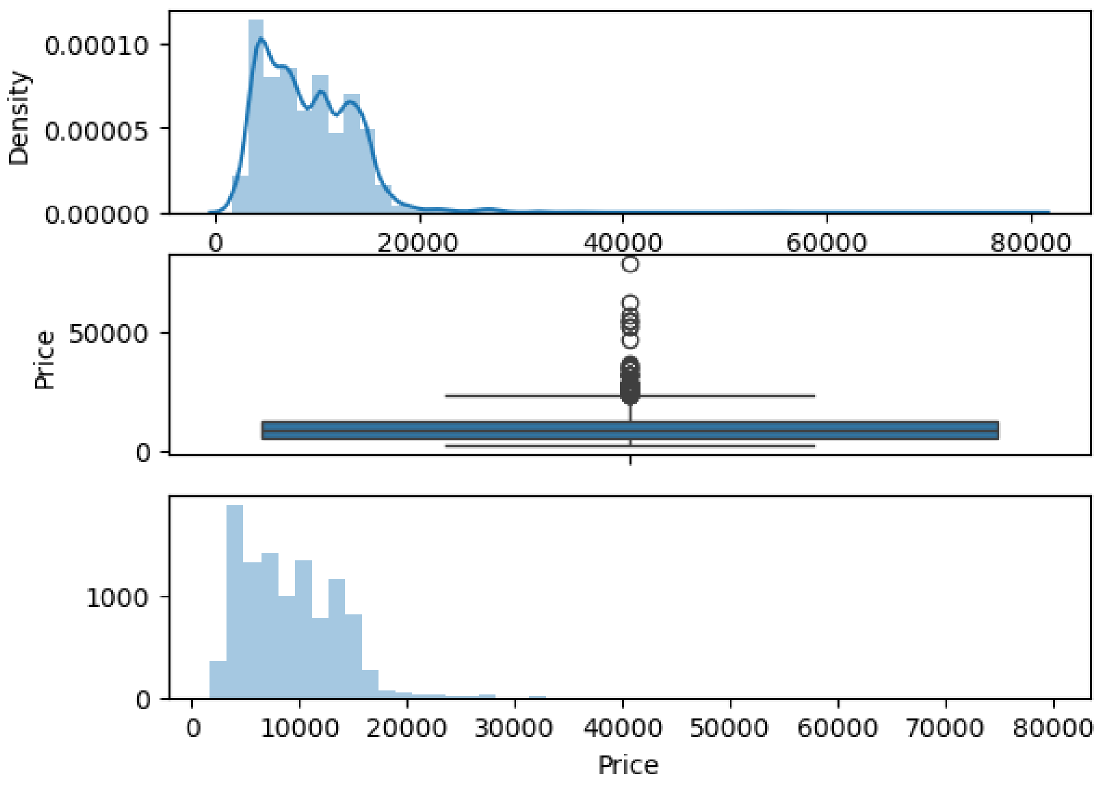
[112]:
```
data.head(3)
```

[112]:
```
   Airline  Destination  Total_Stops  Price  Journey_day  Journey_month  \
0        3            2            0   3897           24              3
1        7            3            2   7662            1              5
2       10            4            2  13882            9              6

   Dep_Time_hour  Dep_Time_minute  Arrival_Time_hour  Arrival_Time_minute  \
0             22               20                  1                   10
1              5               50                 13                   15
2              9               25                  4                   25

   Duration_hours  Duration_mins  Source_Banglore  Source_Kolkata  \
0               2             50                1               0
1               7             25                0               1
2              19              0                0               0

   Source_Delhi  Source_Chennai  Source_Mumbai
0             0               0              0
1             0               0              0
2             1               0              0
```

1.18 Outlier Detection

[113]:
```
def plot(df , col):
    fig , (ax1 , ax2 , ax3) = plt.subplots(3,1)

    sns.distplot(df[col] , ax=ax1)
    sns.boxplot(df[col] , ax=ax2)
    sns.distplot(df[col] , ax=ax3 , kde=False)
```

[114]:
```
plot(data , 'Price')
```



[115]:
```
q1 = data['Price'].quantile(0.25)
q3 = data['Price'].quantile(0.75)

iqr = q3 - q1

maximum = q3 + 1.5*iqr
minimum = q1 - 1.5*iqr
```

[116]:
```
print(maximum)
```

```
23017.0
```

```
[117]: print(minimum)
```

```
-5367.0
```

```
[118]: print([price for price in data['Price'] if price>maximum or price<minimum])
```

```
[27430, 36983, 26890, 26890, 25139, 27210, 52229, 26743, 26890, 25735, 27992,
26890, 26890, 23583, 26890, 23533, 24115, 25735, 54826, 31783, 27992, 26890,
26890, 25430, 36235, 27210, 26890, 25735, 54826, 26890, 35185, 79512, 28097,
27992, 26890, 25735, 26092, 31825, 25913, 25735, 27992, 31825, 23267, 62427,
54826, 31825, 25430, 26890, 36235, 23843, 26890, 25735, 28322, 25735, 25735,
31825, 26890, 27992, 34273, 46490, 29528, 26890, 26890, 26890, 34503, 26890,
27992, 26890, 26890, 23170, 24528, 26890, 27992, 25735, 34608, 25703, 26890,
23528, 31825, 27282, 25735, 27992, 52285, 24017, 31945, 26890, 24318, 23677,
27992, 24210, 57209, 26890, 31825, 26480]
```

```
[119]: len([price for price in data['Price'] if price>maximum or price<minimum])
```

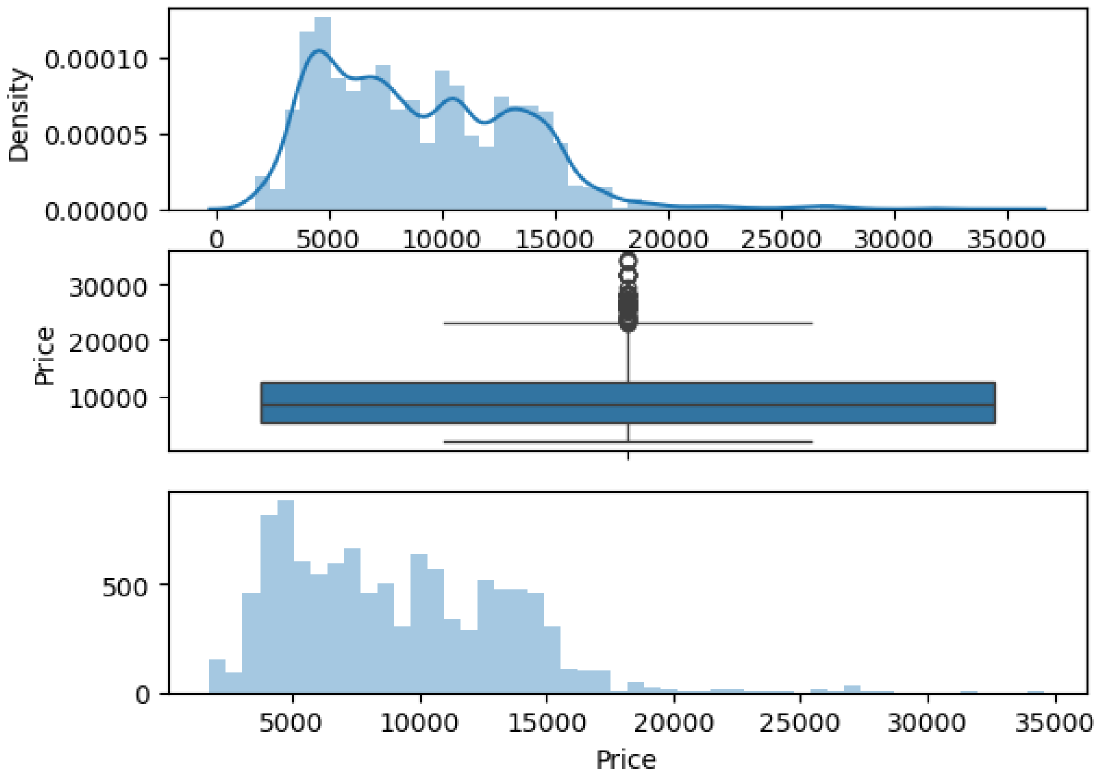
```
[119]: 94
```

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

```
[121]: data['Price']
```

```
[121]: 0         3897.0
       1         7662.0
       2        13882.0
       3         6218.0
       4        13302.0
                 …
       10678     4107.0
       10679     4145.0
       10680     7229.0
       10681    12648.0
       10682    11753.0
       Name: Price, Length: 10682, dtype: float64
```

```
[122]: plot(data , 'Price')
```



1.19 Feature Selection

[123]:
```
X = data.drop(['Price'] , axis=1)
```

[124]:
```
y = data['Price']
```

[125]:
```
from sklearn.feature_selection import mutual_info_regression
```

[126]:
```
imp = mutual_info_regression(X , y)
```

[127]:
```
imp
```

[127]:
```
array([1.32242588, 1.06873428, 0.78609461, 0.38237418, 0.61839087,
       0.93453414, 0.75535656, 1.13792216, 0.89756616, 1.12525252,
       0.67404378, 0.39688153, 0.45552395, 0.51221784, 0.13437008,
       0.19534493])
```

[128]:
```
imp_df = pd.DataFrame(imp , index=X.columns)
```

[129]:
```
imp_df.columns = ['importance']
```

[130]:
```
imp_df
```

[130]:
```
                     importance
Airline                1.322426
Destination            1.068734
Total_Stops            0.786095
Journey_day            0.382374
Journey_month          0.618391
Dep_Time_hour          0.934534
Dep_Time_minute        0.755357
Arrival_Time_hour      1.137922
Arrival_Time_minute    0.897566
Duration_hours         1.125253
Duration_mins          0.674044
Source_Banglore        0.396882
Source_Kolkata         0.455524
Source_Delhi           0.512218
Source_Chennai         0.134370
Source_Mumbai          0.195345
```

[131]:
```python
imp_df.sort_values(by='importance' , ascending=False)
```

[131]:
```
                     importance
Airline                1.322426
Arrival_Time_hour      1.137922
Duration_hours         1.125253
Destination            1.068734
Dep_Time_hour          0.934534
Arrival_Time_minute    0.897566
Total_Stops            0.786095
Dep_Time_minute        0.755357
Duration_mins          0.674044
Journey_month          0.618391
Source_Delhi           0.512218
Source_Kolkata         0.455524
Source_Banglore        0.396882
Journey_day            0.382374
Source_Mumbai          0.195345
Source_Chennai         0.134370
```

1.20 Machine Learning Model Building and Saving (Regression, Classificaion, Clustring)

[132]:
```python
from sklearn.model_selection import train_test_split
```

[133]:
```python
X_train, X_test, y_train, y_test = train_test_split(
  X, y, test_size=0.33, random_state=42)
```

[134]:
```python
from sklearn.ensemble import RandomForestRegressor
```

```
[135]: ml_model = RandomForestRegressor()
```

```
[136]: ml_model.fit(X_train , y_train)
```

```
[136]: RandomForestRegressor()
```

```
[137]: y_pred = ml_model.predict(X_test)
```

```
[138]: y_pred
```

```
[138]: array([16723.03 ,  5339.6  ,  8998.62 , …,  8166.02 ,  9107.56 ,
              11679.555])
```

```
[139]: from sklearn import metrics
```

```
[140]: metrics.r2_score(y_test , y_pred)
```

```
[140]: 0.8172808853427979
```

1.21 Saving Model

```
[141]: import pickle
```

```
[142]: file = open(r'E:\DS Projects\1. Predict Fare of Airlines Tickets using Machine␣
        ↪Learning/rf_random.pkl' , 'wb')
```

```
[143]: pickle.dump(ml_model , file)
```

```
[144]: model = open(r'E:\DS Projects\1. Predict Fare of Airlines Tickets using Machine␣
        ↪Learning/rf_random.pkl' , 'rb')
```

```
[145]: forest = pickle.load(model)
```

```
[146]: y_pred2 = forest.predict(X_test)
```

```
[147]: y_pred2
```

```
[147]: array([16723.03 ,  5339.6  ,  8998.62 , …,  8166.02 ,  9107.56 ,
              11679.555])
```

```
[148]: metrics.r2_score(y_test , y_pred2)
```

```
[148]: 0.8172808853427979
```

1.22 Define Evalution Metric and Automate Machine Learning Pipeline

[149]:
```
def mape(y_true , y_pred):
    y_true , y_pred = np.array(y_true) , np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

[150]:
```
mape(y_test , y_pred)
```

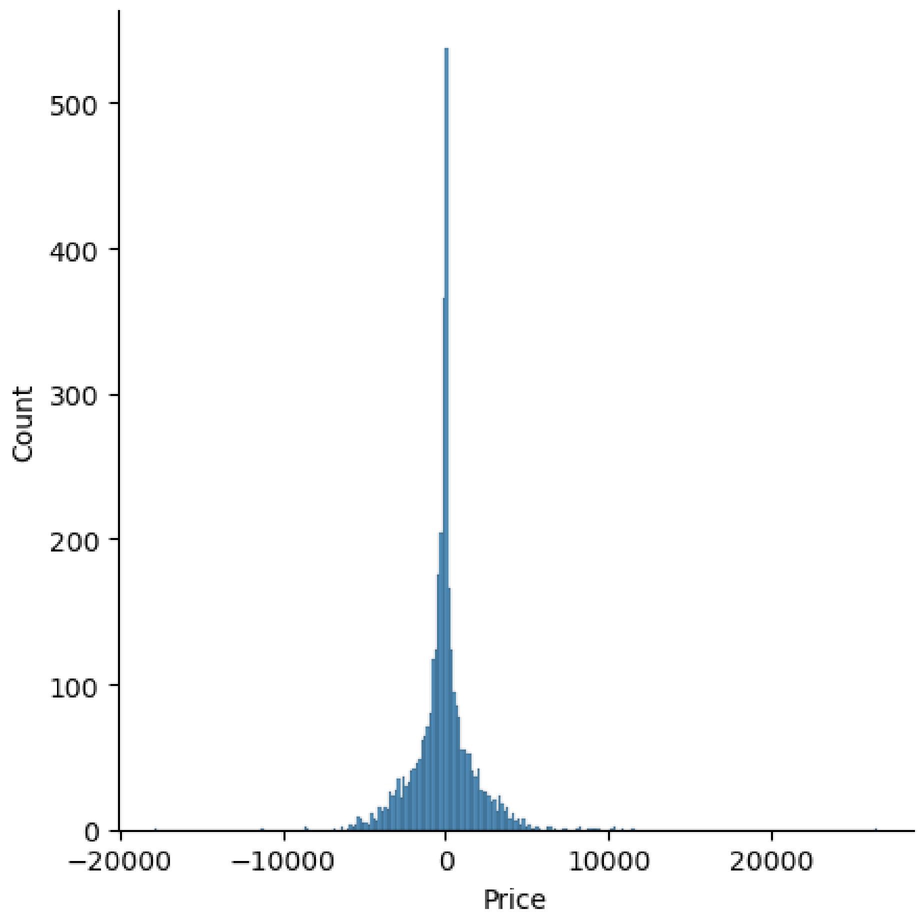
[150]:
```
13.124320812886314
```

[152]:
```
from sklearn import metrics
```

[156]:
```
def predict(ml_model):
    model = ml_model.fit(X_train, y_train)  # Fix indentation here
    print('Training score : {}'.format(model.score(X_train, y_train)))
    y_prediction = model.predict(X_test)
    print('Predictions are: {}'.format(y_prediction))
    print('\n')
    r2_score = metrics.r2_score(y_test, y_prediction)  # Fix indentation and␣
 ↪parentheses here
    print('R2 score : {}'.format(r2_score))
    print('MAE : {}'.format(metrics.mean_absolute_error(y_test, y_prediction)))
    print('MSE : {}'.format(metrics.mean_squared_error(y_test, y_prediction)))
    print('RMSE : {}'.format(np.sqrt(metrics.mean_squared_error(y_test,␣
 ↪y_prediction))))
    print('MAPE : {}'.format(mape(y_test, y_prediction)))
    sns.displot(y_test - y_prediction)
```

[157]:
```
predict(RandomForestRegressor())
```

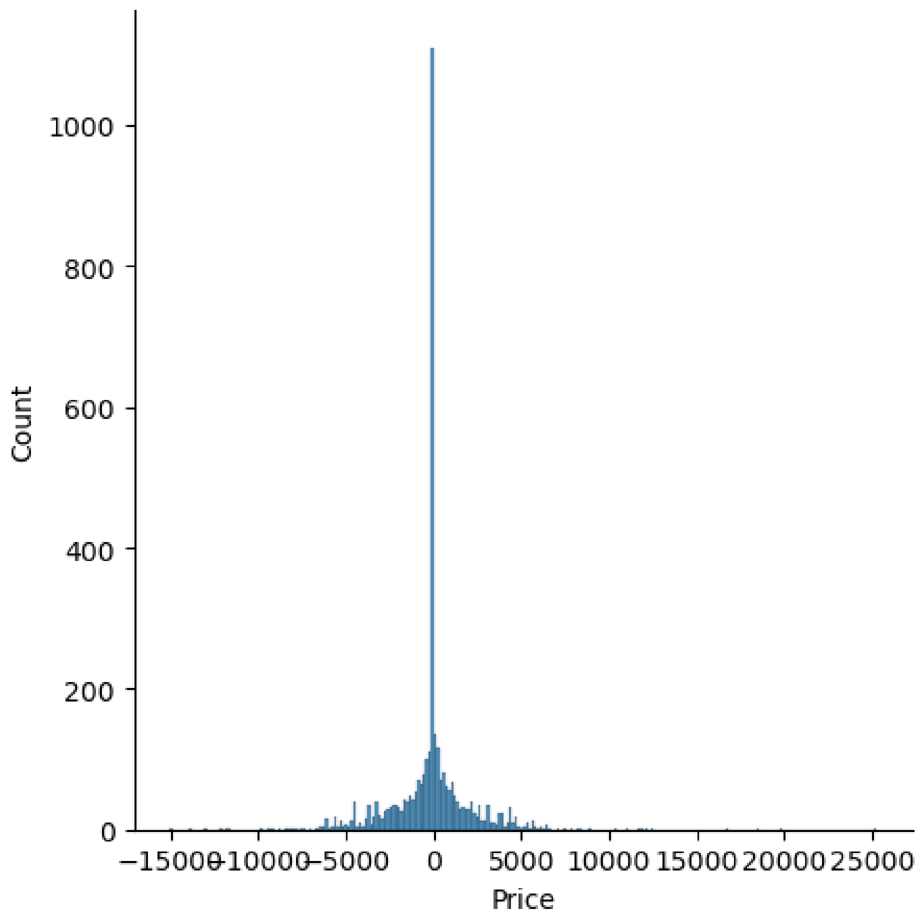
```
Training score : 0.9543029543825003
Predictions are: [16671.58        5337.8          8830.04        …  8037.09
  9220.47       12307.16866667]


R2 score : 0.8172578862267197
MAE : 1164.9011671313656
MSE : 3509328.433285369
RMSE : 1873.3201630488497
MAPE : 13.114686931362417
```



```
[158]: from sklearn.tree import DecisionTreeRegressor
```

```
[159]: predict(DecisionTreeRegressor())
```

```
Training score : 0.9696998040730191
Predictions are: [16840.   4959.   8085.  …  6442.  10141.  11652.5]


R2 score : 0.6904468907030217
MAE : 1397.1093590470787
MSE : 5944571.317674103
RMSE : 2438.1491582087638
MAPE : 15.538578369246133
```



1.23 Hypertune ML Model (Hyperparameter Optimization)

[194]:
```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV
import numpy as np

# Create an instance of RandomForestRegressor
reg_rf = RandomForestRegressor()

# Define parameter grid
n_estimators = [int(x) for x in np.linspace(start=100, stop=1200, num=6)]
max_features = ["auto", "sqrt"]
max_depth = [int(x) for x in np.linspace(start=5, stop=30, num=4)]
min_samples_split = [5, 10, 15, 100]

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

# Create the RandomizedSearchCV instance with the reg_rf instance
rf_random = RandomizedSearchCV(estimator=reg_rf,␣
 ↪param_distributions=random_grid, cv=3, n_jobs=-1, verbose=2)

# Fit the model
rf_random.fit(X_train, y_train)
```

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

[194]:
```
RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(), n_jobs=-1,
                   param_distributions={'max_depth': [5, 13, 21, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_split': [5, 10, 15, 100],
                                        'n_estimators': [100, 320, 540, 760,
                                                         980, 1200]},
                   verbose=2)
```

[195]:
```
rf_random.best_params_
```

[195]:
```
{'n_estimators': 980,
 'min_samples_split': 5,
 'max_features': 'sqrt',
 'max_depth': 13}
```

[197]:
```
rf_random.best_estimator_
```

[197]:
```
RandomForestRegressor(max_depth=13, max_features='sqrt', min_samples_split=5,
                      n_estimators=980)
```

[198]:
```
rf_random.best_score_
```

[198]:
```
0.7978165042231741
```