copy-of-copy-of-virtual-intern

September 8, 2023

```
[2]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.preprocessing import LabelEncoder
     from sklearn import metrics
     from sklearn.model_selection import RandomizedSearchCV
     from sklearn.model_selection import train_test_split
     from sklearn.ensemble import RandomForestRegressor
     from sklearn.ensemble import ExtraTreesRegressor
     import pickle
```

1 loading data

```
[3]: train_data = pd.read_excel(r"Data_Train.xlsx")
```

2 training dataset

```
[4]: pd.set_option("display.max_columns",None)
```

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

```
[5]:        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
```

[6]:
```
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
```

[7]:
```
train_data.shape
```

[7]:
```
(10683, 11)
```

[8]:
```
train_data.describe()
```

[8]:
```
              Price
count  10683.000000
mean    9087.064121
std     4611.359167
min     1759.000000
25%     5277.000000
50%     8372.000000
75%    12373.000000
max    79512.000000
```

3 checking null values in training set

[9]:
```
train_data.isnull().sum()
```

[9]:
```
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
```

4 deleting null values column since it has 1 null value in two columns

[10]:
```
train_data.dropna(inplace=True)
```

5 again checking null values

[11]:
```
train_data.isnull().sum()
```

[11]:
```
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
```

#checking if there are any duplicate values

[12]:
```
train_data[train_data.duplicated()]
```

[12]:
```
           Airline Date_of_Journey    Source Destination  \
683    Jet Airways       1/06/2019     Delhi      Cochin
1061     Air India      21/05/2019     Delhi      Cochin
1348     Air India      18/05/2019     Delhi      Cochin
1418   Jet Airways       6/06/2019     Delhi      Cochin
1674        IndiGo      24/03/2019  Banglore   New Delhi
…              …               …        …         …
10594  Jet Airways      27/06/2019     Delhi      Cochin
10616  Jet Airways       1/06/2019     Delhi      Cochin
10634  Jet Airways       6/06/2019     Delhi      Cochin
10672  Jet Airways      27/06/2019     Delhi      Cochin
```

```
10673  Jet Airways      27/05/2019     Delhi      Cochin

                      Route Dep_Time  Arrival_Time Duration Total_Stops  \
683    DEL → NAG → BOM → COK    14:35  04:25 02 Jun  13h 50m     2 stops
1061   DEL → GOI → BOM → COK    22:00  19:15 22 May  21h 15m     2 stops
1348   DEL → HYD → BOM → COK    17:15  19:15 19 May      26h     2 stops
1418   DEL → JAI → BOM → COK    05:30  04:25 07 Jun  22h 55m     2 stops
1674               BLR → DEL    18:25         21:20   2h 55m    non-stop
…                         …      …             …       …         …
10594  DEL → AMD → BOM → COK    23:05  12:35 28 Jun  13h 30m     2 stops
10616  DEL → JAI → BOM → COK    09:40  12:35 02 Jun  26h 55m     2 stops
10634  DEL → JAI → BOM → COK    09:40  12:35 07 Jun  26h 55m     2 stops
10672  DEL → AMD → BOM → COK    23:05  19:00 28 Jun  19h 55m     2 stops
10673  DEL → AMD → BOM → COK    13:25  04:25 28 May      15h     2 stops

                   Additional_Info  Price
683                        No info  13376
1061                       No info  10231
1348                       No info  12392
1418   In-flight meal not included  10368
1674                       No info   7303
…                              …     …
10594                      No info  12819
10616                      No info  13014
10634  In-flight meal not included  11733
10672  In-flight meal not included  11150
10673                      No info  16704

[220 rows x 11 columns]
```

#drop duplicate values

[13]:
```
train_data.drop_duplicates(keep='first',inplace=True)
```

#exploratory data analysis

#handling numerical values 1)Date_of_Journey 2)Dep_Time 3)Arrival_Time 4)Duration

#extracting day from date and journey

[14]:
```
train_data['Journey_day'] = pd.to_datetime(train_data.Date_of_Journey,␣
 ↪format="%d/%m/%Y").dt.day
```

#extracting month from date and journey

[15]:
```
train_data['Journey_month'] = pd.to_datetime(train_data.Date_of_Journey,␣
 ↪format="%d/%m/%Y").dt.month
```

[16]:
```
train_data.head(2)
```

[16]:
```
     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

   Journey_day  Journey_month
0           24              3
1            1              5
```

6 now Date_of_Journey is not important so delete it from training data as i extracted useful information from it

[17]:
```python
train_data.drop(["Date_of_Journey"],axis=1, inplace = True)
```

7 Extracting Minute and Hour from Dep_Time, after that appending in train_data

8 And Deleting Dep_Time

[18]:
```python
train_data["Dep_hour"] = pd.to_datetime(train_data['Dep_Time']).dt.hour
train_data["Dep_min"] = pd.to_datetime(train_data['Dep_Time']).dt.minute
train_data.drop(["Dep_Time"],axis=1, inplace = True)
```

[19]:
```python
train_data.head(2)
```

[19]:
```
     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

  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

   Dep_hour  Dep_min
0        22       20
1         5       50
```

9 Extracting Minute and Hour from Arrival_Time, after that appending in train_data

10 And Deleting Arrival_Time

[20]:
```
train_data["Arrival_hour"] = pd.to_datetime(train_data['Arrival_Time']).dt.hour
train_data["Arrival_min"] = pd.to_datetime(train_data['Arrival_Time']).dt.minute
train_data.drop(["Arrival_Time"],axis=1, inplace = True)
```

[21]:
```
train_data.head()
```

[21]:
```
       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
```

11 Formating the Duration to correct format (ex- 2h 10m, 0h 15m, 5h 0m)

[22]:
```
duration = list(train_data['Duration']) # convert to list
for i in range(len(duration)):
  if len(duration[i].split())!=2:
    if "h" in duration[i]:
      duration[i] = duration[i].strip()+' 0m'
    else:
      duration[i]= "0h "+duration[i]

duration_hours = []
duration_mins = []
```

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

[23]:
```
train_data['Duration_hours']= duration_hours
train_data['Duration_mins']= duration_mins
```

12 Now Duration is not important so delete it from training data as I extracted useful information from it.

[24]:
```
train_data.drop(["Duration"],axis=1, inplace = True)
```

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

[25]:
```
       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

  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

   Arrival_hour  Arrival_min  Duration_hours  Duration_mins
0             1           10               2             50
1            13           15               7             25
2             4           25              19              0
```

#Handling Categorical value

#Nominal Categorical data 1.Airline 2.Source 3.Destination

#Ordinal Categorical data 1.Total_stops

#Airline Column

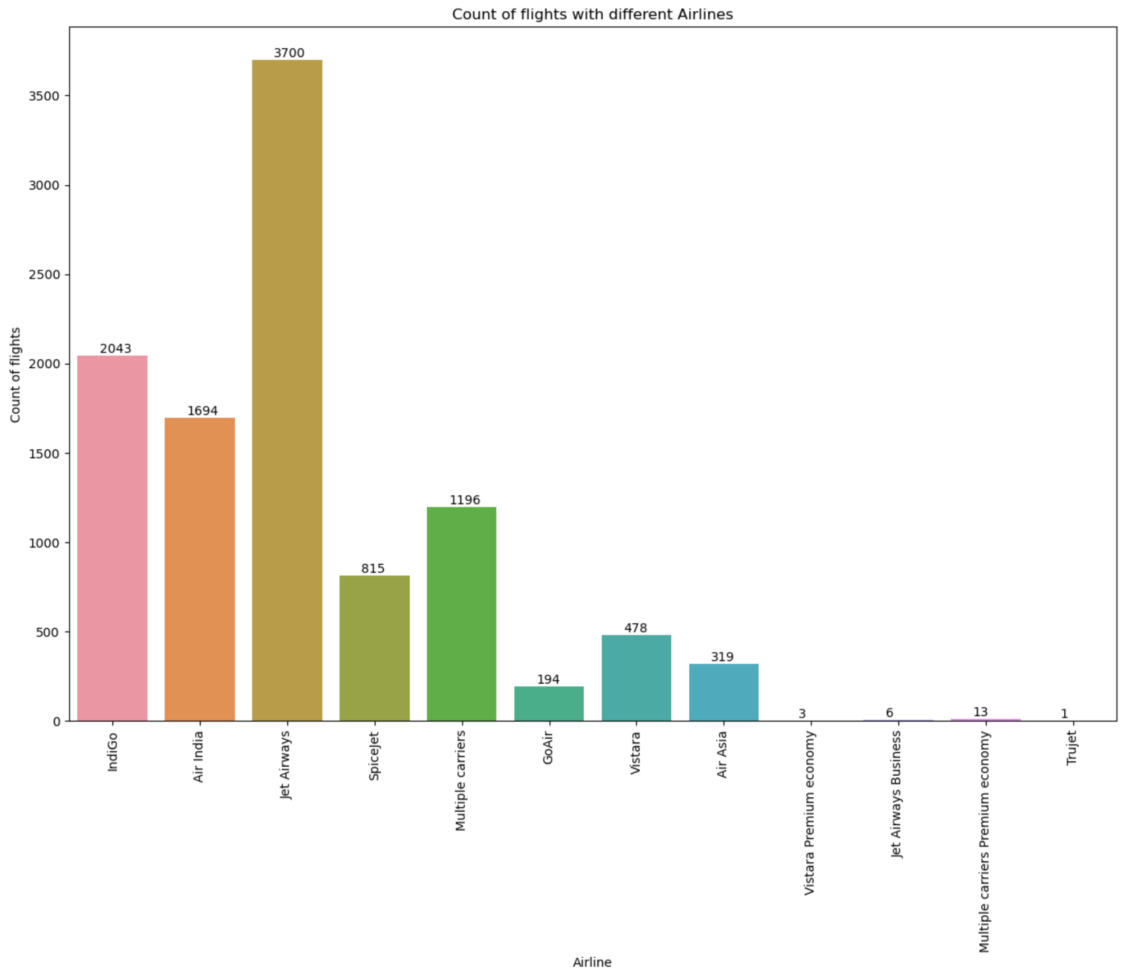
13 Checking value count of Airline column

[26]:
```
train_data['Airline'].value_counts()
```

[26]:
```
Jet Airways                          3700
IndiGo                               2043
Air India                            1694
Multiple carriers                    1196
SpiceJet                              815
Vistara                               478
```

```
Air Asia                                 319
GoAir                                    194
Multiple carriers Premium economy         13
Jet Airways Business                       6
Vistara Premium economy                    3
Trujet                                     1
Name: Airline, dtype: int64
```

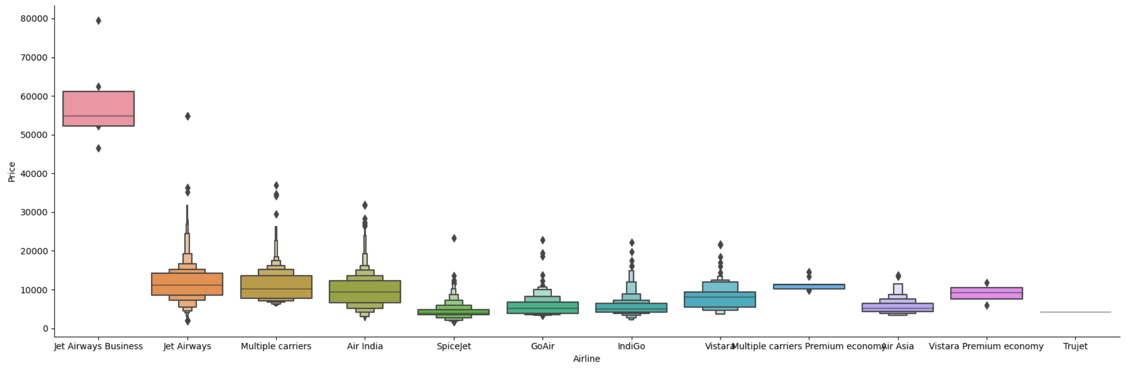
[27]:
```
plt.figure(figsize = (15, 10))
plt.title('Count of flights with different Airlines')
ax=sns.countplot(x = 'Airline', data =train_data)
plt.xlabel('Airline')
plt.ylabel('Count of flights')
plt.xticks(rotation = 90)
for p in ax.patches:
  ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1),␣
 ↪va='bottom',color= 'black')
plt.show()
```



Jet Airways Business, Vistara Premium economy, Trujet have actually almost negligible flights.

14 Plotting Price vs Airline to see individual airline company prices

[28]:
```
sns.catplot(y="Price",x = "Airline", data = train_data.
 ↪sort_values('Price',ascending=False),kind="boxen", height=6,aspect=3)
plt.show()
```



Clearly Jet Airways Business has the highest Price among all airlines

15 Replacing Multiple carriers Premium economy,Jet Airways Business, Vistara Premium economy, Trujet to Others

[29]:
```
train_data["Airline"].replace({'Multiple carriers Premium economy':'Other',␣
 ↪'Jet Airways Business':'Other','Vistara Premium economy':'Other','Trujet':
 ↪'Other'}, inplace=True)
```

16 As airline is nominal categorical data

17 Using One Hot Encoding for it and making dummy variables for Airline

[30]:
```
Airline = train_data[['Airline']]
Airline = pd.get_dummies(Airline,drop_first=True)
Airline.head()
```

[30]:
```
   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_Multiple carriers  Airline_Other  Airline_SpiceJet  Airline_Vistara
0                          0              0                 0                0
1                          0              0                 0                0
2                          0              0                 0                0
3                          0              0                 0                0
4                          0              0                 0                0
```

#Source

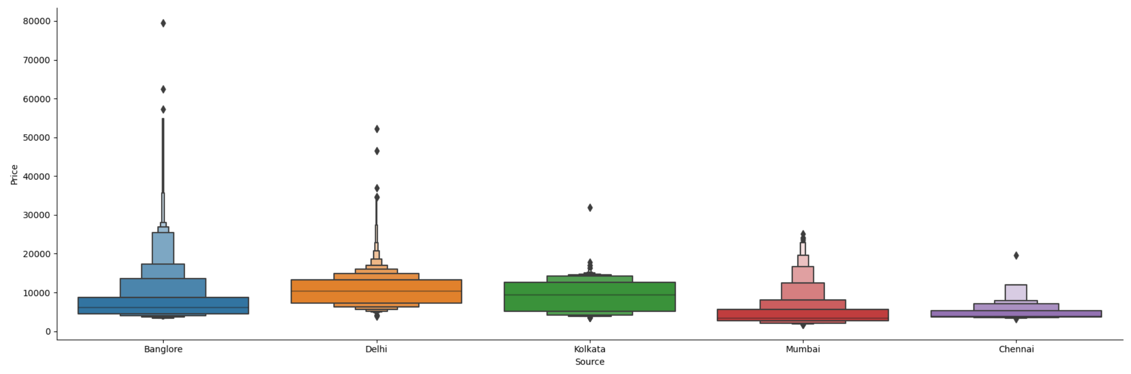
18 Checking value count for Source column

[31]:
```python
train_data['Source'].value_counts()
```

[31]:
```
Delhi       4345
Kolkata     2860
Banglore    2179
Mumbai       697
Chennai      381
Name: Source, dtype: int64
```

19 Plotting Price vs Source to see individual Sources prices

[32]:
```python
sns.catplot(y="Price",x = "Source", data = train_data.
 ↪sort_values('Price',ascending=False),kind="boxen", height=6,aspect=3)
plt.show()
```



Every Cities has almost similar price but there are some outlier also.

20 As Source is nominal categorical data

21 Using One Hot Encoding for it and making dummy variables for Source

[33]:
```
Source = train_data[['Source']]
Source = pd.get_dummies(Source,drop_first=True)
Source.head()
```

[33]:
```
   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

22 Checking value count for Destination

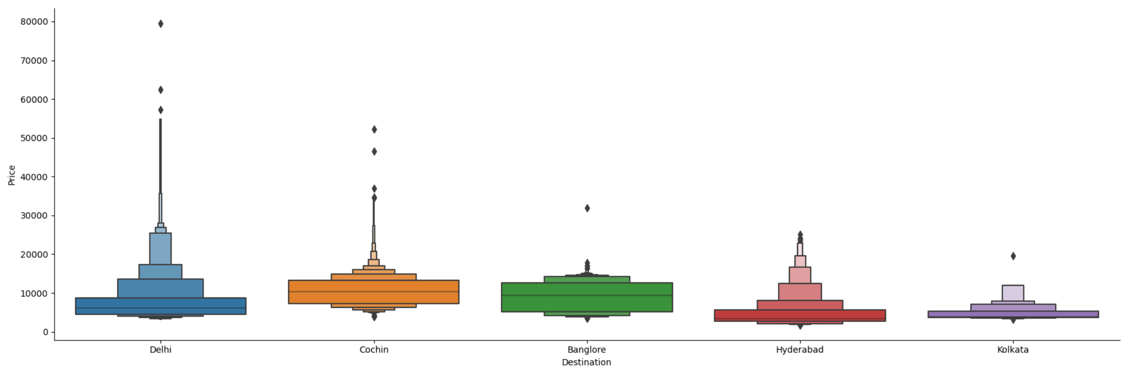
[34]:
```
train_data['Destination'].value_counts()
```

[34]:
```
Cochin       4345
Banglore     2860
Delhi        1265
New Delhi     914
Hyderabad     697
Kolkata       381
Name: Destination, dtype: int64
```

[35]:
```
train_data['Destination'].replace({'New Delhi':'Delhi'},inplace=True)
```

23 Plotting Price vs Destination to see individual Destination prices

[36]:
```
sns.catplot(y="Price",x = "Destination", data = train_data.
  ↪sort_values('Price',ascending=False),kind="boxen", height=6,aspect=3)
plt.show()
```



24 As Destination is nominal categorical data

25 Using One Hot Encoding for it and making dummy variables for Destination

[37]:
```
Destination = train_data[['Destination']]
Destination = pd.get_dummies(Destination,drop_first=True)
Destination.head()
```

[37]:
```
   Destination_Cochin  Destination_Delhi  Destination_Hyderabad  \
0                   0                  1                      0
1                   0                  0                      0
2                   1                  0                      0
3                   0                  0                      0
4                   0                  1                      0

   Destination_Kolkata
0                    0
1                    0
2                    0
3                    0
4                    0
```

Route, Additional_Info, Total_Stops

[38]:
```
train_data["Route"]
```

[38]:
```
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: 10462, dtype: object
```

26 Checking value count for Additional_Info

[39]:
```
train_data['Additional_Info'].value_counts()
```

[39]:
```
No info                        8182
In-flight meal not included    1926
No check-in baggage included    318
1 Long layover                   19
Change airports                   7
Business class                    4
No Info                           3
1 Short layover                   1
Red-eye flight                    1
2 Long layover                    1
Name: Additional_Info, dtype: int64
```

[40]:
```
train_data["Additional_Info"] = train_data["Additional_Info"].replace({'No␣
↪Info': 'No info'})
```

#Additional_Info

[41]:
```
train_data["Additional_Info"].replace({'Change airports':'Other', 'Business␣
↪class':'Other','1 Short layover':'Other','Red-eye flight':'Other','2 Long␣
↪layover':'Other',   }, inplace=True)
```

#Label encode and hot encode categorical columns

[42]:
```
label_encoder = LabelEncoder()
train_data["Additional_Info"]= label_encoder.
↪fit_transform(train_data["Additional_Info"])
```

[43]:
```
train_data["Additional_Info"]
```

[43]:
```
0        3
1        3
2        3
3        3
4        3
        ..
10678    3
```

```
10679    3
10680    3
10681    3
10682    3
Name: Additional_Info, Length: 10462, dtype: int32
```

27 Checking value count for Total_Stops

```
[44]: train_data['Total_Stops'].value_counts()
```

```
[44]: 1 stop      5625
non-stop    3475
2 stops     1318
3 stops       43
4 stops        1
Name: Total_Stops, dtype: int64
```

28 Now since Additional_Info contains almost 80% of No info

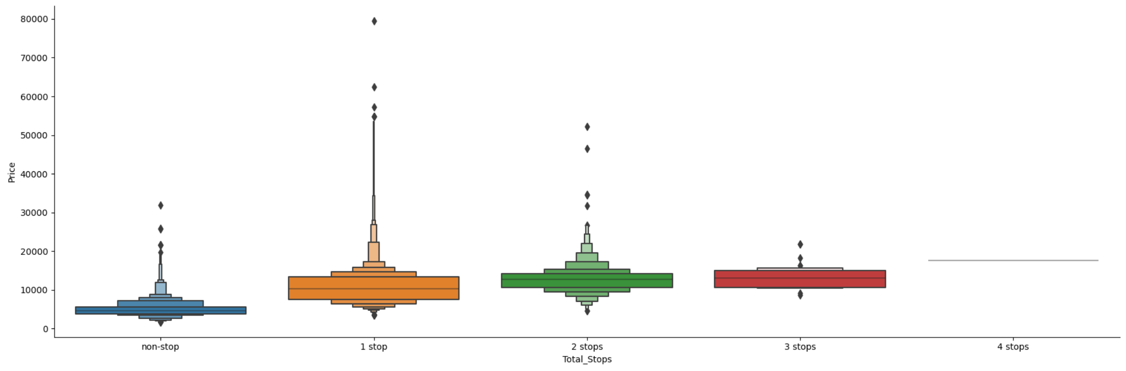
29 Route and Total_Stops are related to each other

30 Deleting Route

```
[45]: train_data.drop(['Route'],axis =1 , inplace = True)
```

31 Plotting Price vs Total_Stops to see individual Total_Stops prices

```
[46]: sns.catplot(y="Price",x = "Total_Stops", data = train_data.
       ↪sort_values('Price',ascending=True),kind="boxen", height=6,aspect=3)
plt.show()
```



As Total_Stops is increasing, Price is also increasing.

32 Total_Stops is ordinal categorical type

```
[47]: train_data.replace({'non-stop':0, '1 stop':1, '2 stops':2, '3 stops':3, '4␣
        ↪stops':4}, inplace = True)
      #Concatinating the Airline, Source and Destination to train_data
      data_train = pd.concat([train_data, Airline, Source, Destination], axis=1)
      data_train.head()
```

```
[47]:       Airline    Source Destination  Total_Stops  Additional_Info  Price  \
      0      IndiGo  Banglore       Delhi            0                3   3897
      1   Air India   Kolkata    Banglore            2                3   7662
      2 Jet Airways     Delhi      Cochin            2                3  13882
      3      IndiGo   Kolkata    Banglore            1                3   6218
      4      IndiGo  Banglore       Delhi            1                3  13302

         Journey_day  Journey_month  Dep_hour  Dep_min  Arrival_hour  Arrival_min  \
      0           24              3        22       20             1           10
      1            1              5         5       50            13           15
      2            9              6         9       25             4           25
      3           12              5        18        5            23           30
      4            1              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_Multiple carriers  \
      0               1                    0                          0
      1               0                    0                          0
      2               0                    1                          0
      3               1                    0                          0
      4               1                    0                          0

         Airline_Other  Airline_SpiceJet  Airline_Vistara  Source_Chennai  \
      0              0                 0                0               0
      1              0                 0                0               0
      2              0                 0                0               0
      3              0                 0                0               0
      4              0                 0                0               0
```

```
   Source_Delhi  Source_Kolkata  Source_Mumbai  Destination_Cochin  \
0             0               0              0                   0
1             0               1              0                   0
2             1               0              0                   1
3             0               1              0                   0
4             0               0              0                   0

   Destination_Delhi  Destination_Hyderabad  Destination_Kolkata
0                  1                      0                    0
1                  0                      0                    0
2                  0                      0                    0
3                  0                      0                    0
4                  1                      0                    0
```

[48]:
```
# Deleting Airline, Source and Destination from data_train as I extracted␣
 ↪useful information from it.
data_train.drop(["Airline"],axis=1, inplace = True)
data_train.drop(["Source"],axis=1, inplace = True)
data_train.drop(["Destination"],axis=1, inplace = True)
```

[49]:
```
data_train.head()
```

[49]:
```
   Total_Stops  Additional_Info  Price  Journey_day  Journey_month  Dep_hour  \
0            0                3   3897           24              3        22
1            2                3   7662            1              5         5
2            2                3  13882            9              6         9
3            1                3   6218           12              5        18
4            1                3  13302            1              3        16

   Dep_min  Arrival_hour  Arrival_min  Duration_hours  Duration_mins  \
0       20             1           10               2             50
1       50            13           15               7             25
2       25             4           25              19              0
3        5            23           30               5             25
4       50            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_Multiple carriers  Airline_Other  Airline_SpiceJet  \
0                          0              0                 0
1                          0              0                 0
```

```
2                        0           0              0
3                        0           0              0
4                        0           0              0

   Airline_Vistara  Source_Chennai  Source_Delhi  Source_Kolkata  \
0                0               0             0               0
1                0               0             0               1
2                0               0             1               0
3                0               0             0               1
4                0               0             0               0

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

   Destination_Hyderabad  Destination_Kolkata
0                      0                    0
1                      0                    0
2                      0                    0
3                      0                    0
4                      0                    0
```

[50]:
```
data_train.shape
```

[50]:
```
(10462, 27)
```

#Test Dataset

[51]:
```
test_data = pd.read_excel(r"Test_set.xlsx")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[51], line 1
----> 1 test_data = pd.read_excel(r"Test_set.xlsx")

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\util\_decorators.py:211,
 ↪in deprecate_kwarg.<locals>._deprecate_kwarg.<locals>.wrapper(*args, **kwargs
    209     else:
    210         kwargs[new_arg_name] = new_arg_value
--> 211 return func(*args, **kwargs)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\util\_decorators.py:331,
 ↪in deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args,␣
 ↪**kwargs)
    325 if len(args) > num_allow_args:
```

```
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\excel\_base.py:482, i␣
 ↪read_excel(io, sheet_name, header, names, index_col, usecols, squeeze, dtype,␣
 ↪engine, converters, true_values, false_values, skiprows, nrows, na_values,␣
 ↪keep_default_na, na_filter, verbose, parse_dates, date_parser, thousands,␣
 ↪decimal, comment, skipfooter, convert_float, mangle_dupe_cols, storage_options)
    480 if not isinstance(io, ExcelFile):
    481     should_close = True
--> 482     io = ExcelFile(io, storage_options=storage_options, engine=engine)
    483 elif engine and engine != io.engine:
    484     raise ValueError(
    485         "Engine should not be specified when passing "
    486         "an ExcelFile - ExcelFile already has the engine set"
    487     )

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\excel\_base.py:1652,␣
 ↪in ExcelFile.__init__(self, path_or_buffer, engine, storage_options)
   1650     ext = "xls"
   1651 else:
-> 1652     ext = inspect_excel_format(
   1653         content_or_path=path_or_buffer, storage_options=storage_options
   1654     )
   1655     if ext is None:
   1656         raise ValueError(
   1657             "Excel file format cannot be determined, you must specify "
   1658             "an engine manually."
   1659         )

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\excel\_base.py:1525,␣
 ↪in inspect_excel_format(content_or_path, storage_options)
   1522 if isinstance(content_or_path, bytes):
   1523     content_or_path = BytesIO(content_or_path)
-> 1525 with get_handle(
   1526     content_or_path, "rb", storage_options=storage_options, is_text=False
   1527 ) as handle:
   1528     stream = handle.handle
   1529     stream.seek(0)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\common.py:865, in␣
 ↪get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text,␣
 ↪errors, storage_options)
    856         handle = open(
    857             handle,
    858             ioargs.mode,
   (…)
    861             newline="",
    862         )
    863     else:
    864         # Binary mode
--> 865         handle = open(handle, ioargs.mode)
    866     handles.append(handle)
    868 # Convert BytesIO or file objects passed with an encoding

FileNotFoundError: [Errno 2] No such file or directory: 'Test_set.xlsx'
```

[ ]:
```
test_data.head()
```

[ ]:
```
test_data.shape
```

Perfoming same operation to Test dataset also.

[ ]:
```
print("Test data info")
print("\n\n")
print("--"*40)
print(test_data.info())
print("\n\n")

print("--"*40)
test_data.dropna(inplace=True)
print("Null value")
print("\n")
print(test_data.isnull().sum())
print("\n\n")

print("Exploratory Data Analysis")
print("\n")
print("--"*40)

# 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)

# Depature 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:
    if "h" in duration[i]:
      duration[i] = duration[i].strip()+' 0m'
    else:
      duration[i]= "0h "+duration[i]

duration_hours = []
duration_mins = []
for i in range(len(duration)):
  duration_hours.append(int(duration[i].split(sep="h")[0]))
  duration_mins.append(int(duration[i].split(sep="m")[0].split()[-1]))
test_data['Duration_hours']= duration_hours
test_data['Duration_mins']= duration_mins


# Airline
Airline = test_data[['Airline']]
Airline = pd.get_dummies(Airline,drop_first=True)
test_data["Airline"].replace({'Multiple carriers Premium economy':'Other', 'Jet␣
 ↪Airways Business':'Other','Vistara Premium economy':'Other','Trujet':
 ↪'Other'}, inplace=True)
# Source

Source = test_data[['Source']]
Source = pd.get_dummies(Source,drop_first=True)

# Destination
test_data['Destination'].replace({'New Delhi':'Delhi'},inplace=True)
Destination = test_data[['Destination']]
Destination = pd.get_dummies(Destination,drop_first=True)

# Additional Info
```

```
test_data["Additional_Info"].replace({'Change airports':'Other', 'Business␣
↪class':'Other','1 Short layover':'Other','Red-eye flight':'Other','2 Long␣
↪layover':'Other',   }, inplace=True)
test_data["Additional_Info"]= label_encoder.
↪fit_transform(test_data["Additional_Info"])
# now since Additional Info is contains almost 80% no info
# and Route and Total stops are related to each other
# droping Route

test_data.drop(['Route'],axis =1 , inplace = True)

# Total Stops
test_data.replace({'non-stop':0, '1 stop':1, '2 stops':2, '3 stops':3, '4␣
↪stops':4}, inplace = True)

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

data_test.drop(["Airline","Source","Destination","Duration"],axis=1, inplace =␣
↪True)
```

[ ]:
```
data_test.head()
```

#Train and Test Dataset

[ ]:
```
data_test.head()
```

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

[ ]:
```
print(data_train.shape)
print()
print(data_test.shape)
```

33 Feature Selection

[ ]:
```
data_train.columns
```

[ ]:
```
X = data_train.loc[:,['Total_Stops', 'Additional_Info', '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_Multiple carriers',
       'Airline_Other', 'Airline_SpiceJet', 'Airline_Vistara',
       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
       'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
       'Destination_Kolkata']]
X.head()
```

```
[ ]: y = data_train.iloc[:,2]
     y.head()
```

```
[ ]: # Checking Correlation

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

```
[ ]: # Checking all important feature using ExtraTreesRegressor

     selection = ExtraTreesRegressor()
     selection.fit(X,y)
```

```
[ ]: print(selection.feature_importances_)
```

```
[ ]: # Plotting important feature

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

```
[ ]: # Spiltting the Train data

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
      ↪random_state=42)
```

```
[ ]: # Using Random Forest Regressor

     reg_rf = RandomForestRegressor()
     reg_rf.fit(X_train, y_train)
```

```
[ ]: # Predicting the X_test

     y_pred = reg_rf.predict(X_test)
```

```
[ ]: reg_rf.score(X_train, y_train)
```

```
[ ]: reg_rf.score(X_test, y_test)
```

```
[ ]: sns.displot(y_test-y_pred)
     plt.show()
```

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

[ ]:
```
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)))
```

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

[1]:
```
# 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)
Cell In[1], line 5
      2 file = open('flight_rf.pkl', 'wb')
      4 # dump information to that file
----> 5 pickle.dump(reg_rf, file)

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

[2]:
```
import pickle

filename='flightpred'
pickle.dump(model,open(filename,'wb'))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 4
      1 import pickle
      3 filename='flightpred'
----> 4 pickle.dump(model,open(filename,'wb'))

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

[ ]:

[ ]: