Final Project Submission

Please fill out:

- Student name: Marife Ramoran
- Student pace: Self paced
- Scheduled project review date/time: 17 December 2023
- Instructor name: Hardik Idnani
- Blog post URL:

Flight Price Prediction

Overview

In this project, we aim to predict domestic flight prices in India for the period between March 2019 and July 2019 using linear regression and data analysis techniques. The goal is to provide travelers with accurate predictions to assist in planning and budgeting for their flights.

Business Problem

The business seeks to develop a robust solution utilizing linear regression and data analysis techniques. The absence of a comprehensive predictive model not only hinders the efficient allocation of travel budgets but also diminishes the overall satisfaction and confidence of travelers in the planning process. Therefore, our business problem revolves around the imperative need to fill this void and deliver a sophisticated predictive tool that empowers travelers with accurate flight price forecasts, ultimately transforming the travel planning landscape.

Exploratory Data Analysis

Utilizing visualization techniques, we will analyze the distribution of flight prices, discern trends, and investigate potential correlations among various features. This phase is instrumental in guiding informed decisions related to feature selection and gaining insights into the inherent patterns within the dataset. The dataset comprises 10,682 entries encompassing comprehensive details about flights, including information about the airline, date of journey, destination, and duration.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.metrics import accuracy_score,confusion_matrix

import warnings
warnings.filterwarnings('ignore')

!pip install missingno
```

```
Requirement already satisfied: missingno in /Users/mariferamoran/a
naconda3/lib/python3.11/site-packages (0.5.2)
Requirement already satisfied: numpy in /Users/mariferamoran/anaco
nda3/lib/python3.11/site-packages (from missingno) (1.24.3)
Requirement already satisfied: matplotlib in /Users/mariferamoran/
anaconda3/lib/python3.11/site-packages (from missingno) (3.7.1)
Requirement already satisfied: scipy in /Users/mariferamoran/anaco
nda3/lib/python3.11/site-packages (from missingno) (1.10.1)
Requirement already satisfied: seaborn in /Users/mariferamoran/ana
conda3/lib/python3.11/site-packages (from missingno) (0.12.2)
Requirement already satisfied: contourpy>=1.0.1 in /Users/marifera
moran/anaconda3/lib/python3.11/site-packages (from matplotlib->mis
```

```
moran/anaconda3/lib/python3.11/site-packages (from matplotlib->mis
singno) (1.0.5)
Requirement already satisfied: cycler>=0.10 in /Users/mariferamora
n/anaconda3/lib/python3.11/site-packages (from matplotlib->missing
no) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Users/marifer
amoran/anaconda3/lib/python3.11/site-packages (from matplotlib->mi
ssingno) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/marifer
amoran/anaconda3/lib/python3.11/site-packages (from matplotlib->mi
ssingno) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /Users/mariferam
oran/anaconda3/lib/python3.11/site-packages (from matplotlib->miss
ingno) (23.0)
Requirement already satisfied: pillow>=6.2.0 in /Users/mariferamor
an/anaconda3/lib/python3.11/site-packages (from matplotlib->missin
gno) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/marifera
moran/anaconda3/lib/python3.11/site-packages (from matplotlib->mis
singno) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /Users/mari
feramoran/anaconda3/lib/python3.11/site-packages (from matplotlib-
>missingno) (2.8.2)
Requirement already satisfied: pandas>=0.25 in /Users/mariferamora
n/anaconda3/lib/python3.11/site-packages (from seaborn->missingno)
(1.5.3)
Requirement already satisfied: pytz>=2020.1 in /Users/mariferamora
n/anaconda3/lib/python3.11/site-packages (from pandas>=0.25->seabo
rn->missingno) (2022.7)
Requirement already satisfied: six>=1.5 in /Users/mariferamoran/an
aconda3/lib/python3.11/site-packages (from python-dateutil>=2.7->m
atplotlib->missingno) (1.16.0)
```

Dataset Importation:

1. To import the dataset, we utilize the pandas library and employ the `read_excel` method since the data is structured in an Excel file.
2. Following the import, a crucial step involves examining for null values within specific columns or rows.
3. If null values are identified, several strategies can be applied:
    - Filling NaN values with the mean, median, or mode using the `fillna()` method.
    - In cases where the number of missing values is minimal, we may opt to drop those entries altogether.

In [2]:
```
pip install openpyxl
```

```
Requirement already satisfied: openpyxl in /Users/mariferamoran/an
aconda3/lib/python3.11/site-packages (3.0.10)
Requirement already satisfied: et_xmlfile in /Users/mariferamoran/
anaconda3/lib/python3.11/site-packages (from openpyxl) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
df = pd.read_excel('Data_Train.xlsx')
```

In [4]:
```
df.head(10)
```

Out[4]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duratic
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45
5	SpiceJet	24/06/2019	Kolkata	Banglore	CCU → BLR	09:00	11:25	2h 25
6	Jet Airways	12/03/2019	Banglore	New Delhi	BLR → BOM → DEL	18:55	10:25 13 Mar	15h 30
7	Jet Airways	01/03/2019	Banglore	New Delhi	BLR → BOM → DEL	08:00	05:05 02 Mar	21h 5
8	Jet Airways	12/03/2019	Banglore	New Delhi	BLR → BOM → DEL	08:55	10:25 13 Mar	25h 30
9	Multiple carriers	27/05/2019	Delhi	Cochin	DEL → BOM → COK	11:25	19:15	7h 50

In [5]:
```
df.info()
```

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

In [6]:
```
df.describe()
```

Out[6]:

	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

In [7]:
```
df.shape
```

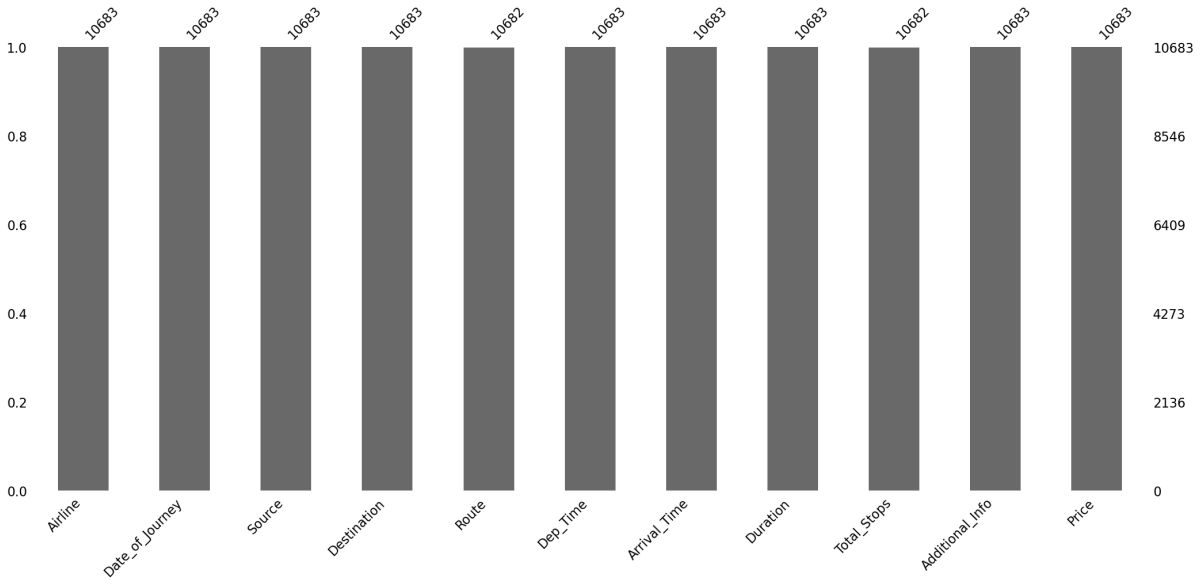
Out[7]: (10683, 11)

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

In [9]:
```
msno.bar(df)
plt.show
```

Out[9]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [10]:
```
# We identify two instances of missing values,
# and it is feasible to promptly eliminate these entries, given the
df.dropna(inplace= True)
```

In [11]:
```
df.isnull().sum()
```

Out[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
```

Data Cleaning

In [12]:
```
# Datatypes
df.dtypes
```

Out[12]:
```
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
```

In [13]:
```
# # The data types of `Date_of_journey`, `Arrival_Time`, and `Dep_T
# To facilitate accurate prediction, it is imperative to convert th

# Utilizing the `dt.day` method will extract the day from the given
# while the `dt.month` method will specifically extract the month i

def change_into_datetime(col):
    df[col]=pd.to_datetime(df[col])
```

In [14]:
```
df.columns
```

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

In [15]:
```
for i in ['Date_of_Journey', 'Dep_Time', 'Arrival_Time']:
    change_into_datetime(i)
```

In [16]:
```
df.dtypes
```

Out[16]:
```
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
```

In [17]:
```
# Subsequently, we extract the day and month from the 'Date_of_jour
# As a result, the 'Date_of_Journey' column becomes redundant, and 

df['journey_day'] = df['Date_of_Journey'].dt.day
df['journey_month'] = df['Date_of_Journey'].dt.month
```

In [18]:
```
df.head(10)
```

Out[18]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duratic
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2023-12-13 22:20:00	2023-03-22 01:10:00	2h 50
1	Air India	2019-01-05	Kolkata	Banglore	CCU → IXR → BBI → BLR	2023-12-13 05:50:00	2023-12-13 13:15:00	7h 25
2	Jet Airways	2019-09-06	Delhi	Cochin	DEL → LKO → BOM → COK	2023-12-13 09:25:00	2023-06-10 04:25:00	19
3	IndiGo	2019-12-05	Kolkata	Banglore	CCU → NAG → BLR	2023-12-13 18:05:00	2023-12-13 23:30:00	5h 25
4	IndiGo	2019-01-03	Banglore	New Delhi	BLR → NAG → DEL	2023-12-13 16:50:00	2023-12-13 21:35:00	4h 45
5	SpiceJet	2019-06-24	Kolkata	Banglore	CCU → BLR	2023-12-13 09:00:00	2023-12-13 11:25:00	2h 25
6	Jet Airways	2019-12-03	Banglore	New Delhi	BLR → BOM → DEL	2023-12-13 18:55:00	2023-03-13 10:25:00	15h 30
7	Jet Airways	2019-01-03	Banglore	New Delhi	BLR → BOM → DEL	2023-12-13 08:00:00	2023-03-02 05:05:00	21h 5
8	Jet Airways	2019-12-03	Banglore	New Delhi	BLR → BOM → DEL	2023-12-13 08:55:00	2023-03-13 10:25:00	25h 30
9	Multiple carriers	2019-05-27	Delhi	Cochin	DEL → BOM → COK	2023-12-13 11:25:00	2023-12-13 19:15:00	7h 50

In [19]:
```
df.drop('Date_of_Journey', axis=1, inplace= True)
```

In [20]:
```
# Extracting hours and minutes from the 'Arrival_time' and 'Dept_ti
# we store them in new columns before subsequently dropping the ori

# Function for extracting hour and minutes
def extract_hour(data,col):
    data[col+'_hour']=data[col].dt.hour

def extract_min(data,col):
    data[col+'_min']=data[col].dt.minute


def drop_col(data,col):
    data.drop(col, axis=1, inplace= True)
```

In [21]:
```
# Execute the function.
# Departure time denotes the moment a plane departs from the gate.
# Similar to the process applied to 'Date_of_Journey,' we can extra

extract_hour(df,'Dep_Time')

# Extracting minutes
extract_min(df,'Dep_Time')

# Drop the column
drop_col(df,'Dep_Time')
```

In [22]:
```
# Extracting hour
extract_hour(df,'Arrival_Time')

# Extracting min
extract_min(df,'Arrival_Time')


# Drop the column
drop_col(df,'Arrival_Time')
```

In [23]:
```
df.head(10)
```

Out[23]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	jou
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	
5	SpiceJet	Kolkata	Banglore	CCU → BLR	2h 25m	non-stop	No info	3873	
6	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	15h 30m	1 stop	In-flight meal not included	11087	
7	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	21h 5m	1 stop	No info	22270	
8	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	25h 30m	1 stop	In-flight meal not included	11087	
9	Multiple carriers	Delhi	Cochin	DEL → BOM → COK	7h 50m	1 stop	No info	8625	

In [24]:
```
# Let's carry out pre-processing on the 'duration' column
# by isolating the duration hours and minutes from the duration val

duration= list(df['Duration'])
for i in range(len(duration)):
    if len(duration[i].split(' '))==2:
        pass
    else:
        if 'h' in duration[i]: # Check if duration contains hour on
            duration[i]=duration[i] + ' 0m' # Adds zero minute
        else:
            duration[i]='0h '+ duration[i]
```

In [25]:
```
df['duration']= duration
# fix duplicate on this
```

In [26]:
```
df.head()
```

Out[26]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	jour
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	

In [27]:
```
def hour(x):
    return x.split(' ')[0][0:-1]

def minutes(x):
    return x.split(' ')[1][0:-1]
```

In [28]:
```
df['dur_hour']= df['Duration'].apply(hour)
```

In [33]:
```
df['dur_min']=df['Duration'].apply(minutes)
```

In [34]:
```
def minutes(x):
    split_values = x.split(' ')
    if len(split_values) == 2:
        return split_values[1][0:-1]
    else:
        return None

df['dur_min'] = df['Duration'].apply(minutes)
```

In [35]:
```
df.head(10)

# fix no.2 dur_min
```

Out[35]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	jou
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	

5	SpiceJet	Kolkata	Banglore	CCU → BLR	2h 25m	non-stop	No info	3873
6	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	15h 30m	1 stop	In-flight meal not included	11087
7	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	21h 5m	1 stop	No info	22270
8	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	25h 30m	1 stop	In-flight meal not included	11087
9	Multiple carriers	Delhi	Cochin	DEL → BOM → COK	7h 50m	1 stop	No info	8625

In [36]:
```
drop_col(df, 'Duration')
```

In [37]:
```
df.dtypes
```

Out[37]:
```
Airline              object
Source               object
Destination          object
Route                object
Total_Stops          object
Additional_Info      object
Price                 int64
journey_day           int64
journey_month         int64
Dep_Time_hour         int64
Dep_Time_min          int64
Arrival_Time_hour     int64
Arrival_Time_min      int64
duration             object
dur_hour             object
dur_min              object
dtype: object
```

In [38]:
```
# Finding the categorical values

column= [column for column in df.columns if df[column].dtype=='obje
```

In [39]:
```
# Finding the continuous values

continuous_col = [column for column in df.columns if df[column].dty
```

Handling Categorical Data

We are using two main Encoding Techniques to covert Categorical data into some numerical format

Nominal data -- Data that are not in any order > one hot encoding

Ordinal data -- Data are in order > labelEncoder

In [40]:
```
categorical= df[column]
```

In [41]:
```
categorical.head(10)
```

Out[41]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	duration	dur_hour
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	2h 50m	2
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	7h 25m	7
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info	19h 0m	19
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info	5h 25m	5
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info	4h 45m	4
5	SpiceJet	Kolkata	Banglore	CCU → BLR	non-stop	No info	2h 25m	2
6	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	1 stop	In-flight meal not included	15h 30m	15
7	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	1 stop	No info	21h 5m	21
8	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	1 stop	In-flight meal not included	25h 30m	25
9	Multiple carriers	Delhi	Cochin	DEL → BOM → COK	1 stop	No info	7h 50m	7

In [42]:
```
categorical['Airline'].value_counts()
```

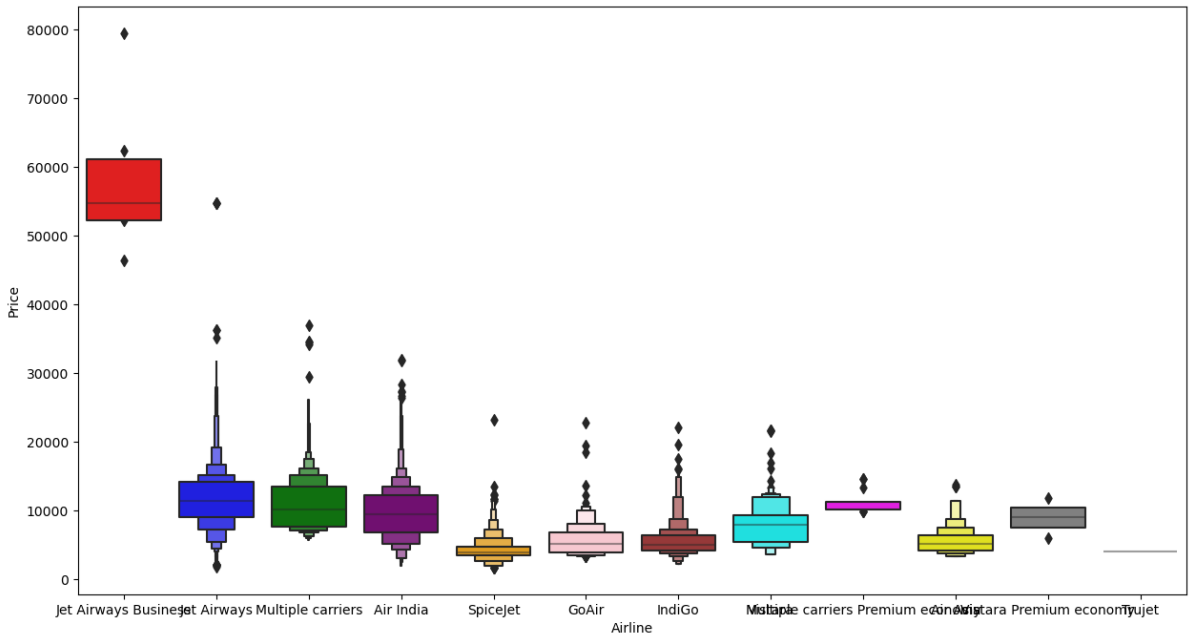
Out[42]:
```
Jet Airways                          3849
IndiGo                               2053
Air India                            1751
Multiple carriers                    1196
SpiceJet                              818
Vistara                               479
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
```

Airline vs Price Analysis

In [43]:
```
colors = ['red', 'blue', 'green', 'purple', 'orange', 'pink', 'brow

plt.figure(figsize=(15, 8))
sns.boxenplot(x='Airline', y='Price', data=df.sort_values('Price',
```

Out[43]:
```
<Axes: xlabel='Airline', ylabel='Price'>
```

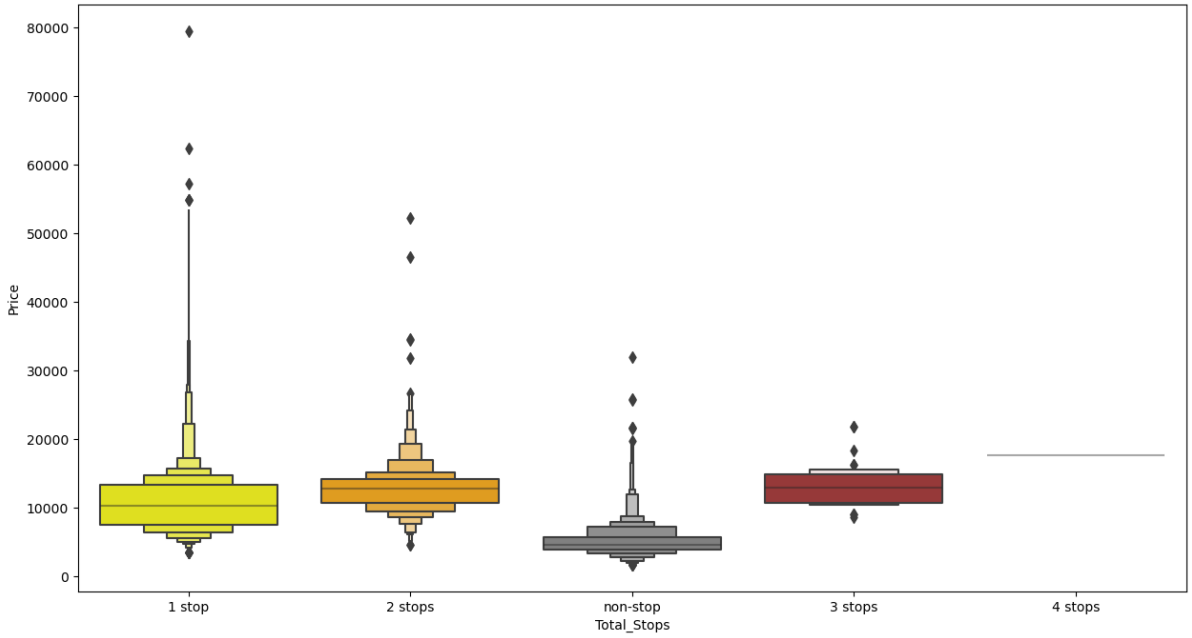


The graphical representation indicates that Jet Airways Business exhibits the highest pricing, while the remaining airlines, with the exception of the first one, display relatively similar median prices.

Perform Total_Stops vs Price Analysis

In [44]:
```
plt.figure(figsize=(15,8))
sns.boxenplot(x='Total_Stops',y='Price',data=df.sort_values('Price'
```

Out[44]: <Axes: xlabel='Total_Stops', ylabel='Price'>



In [45]:
```
# Since the airline data is nominal categorical, we will apply One-
Airline=pd.get_dummies(categorical['Airline'],drop_first=True)
```

In [46]:
```
Airline.head()
```

Out[46]:

	Air India	GoAir	IndiGo	Jet Airways	Jet Airways Business	Multiple carriers	Multiple carriers Premium economy	SpiceJet	Trujet	Vistara	F e
0	0	0	1	0	0	0	0	0	0	0	
1	1	0	0	0	0	0	0	0	0	0	
2	0	0	0	1	0	0	0	0	0	0	
3	0	0	1	0	0	0	0	0	0	0	
4	0	0	1	0	0	0	0	0	0	0	

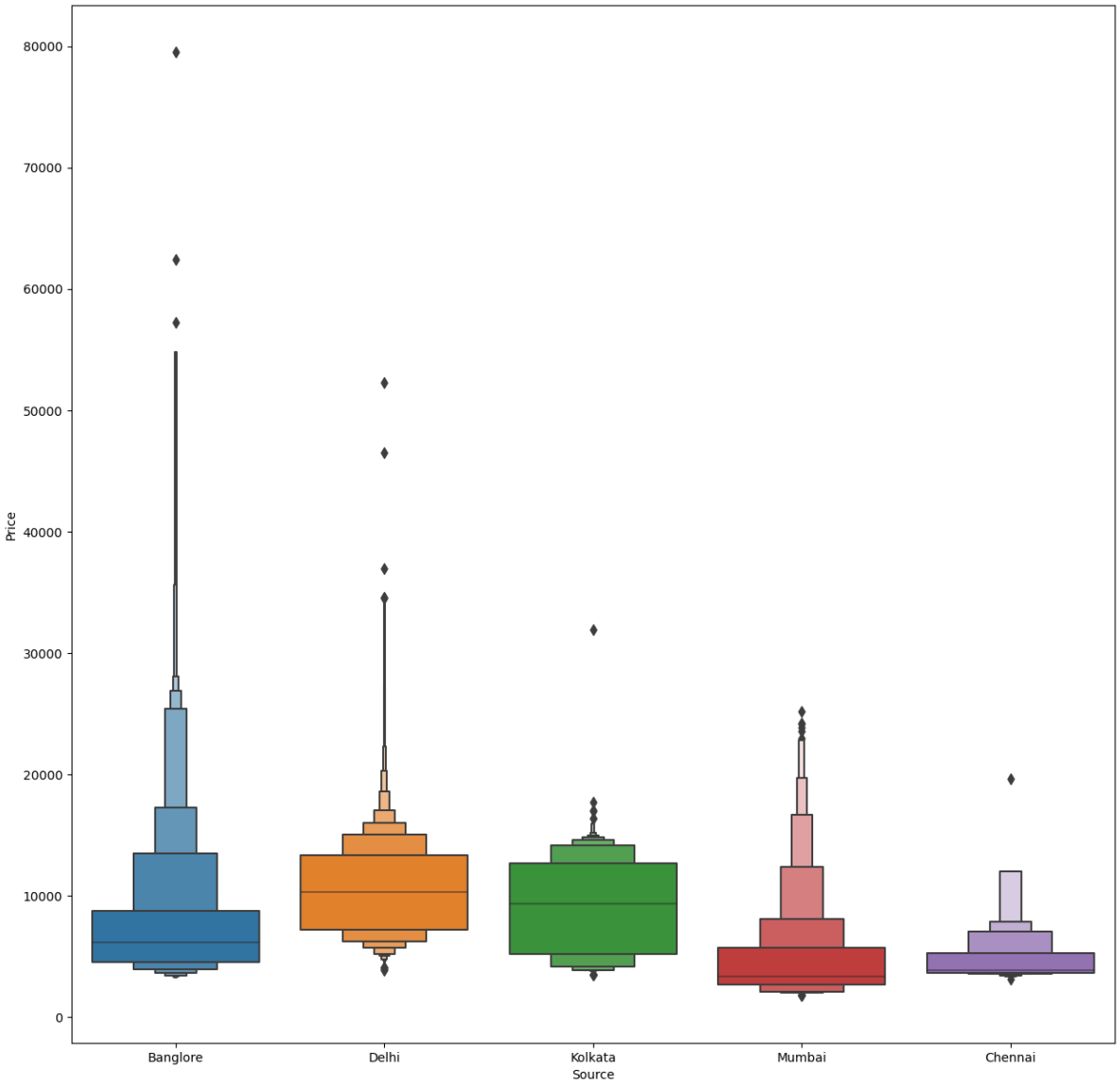
In [47]:
```
categorical['Source'].value_counts()
```

Out[47]:
```
Delhi       4536
Kolkata     2871
Banglore    2197
Mumbai       697
Chennai      381
Name: Source, dtype: int64
```

Source vs Price

In [48]:
```
plt.figure(figsize=(15,15))
sns.boxenplot(x='Source',y='Price',data=df.sort_values('Price',asce
```

Out[48]:
```
<Axes: xlabel='Source', ylabel='Price'>
```



In [49]:
```
#encoding of source column
source=pd.get_dummies(categorical['Source'],drop_first=True)
source.head()
```

Out[49]:

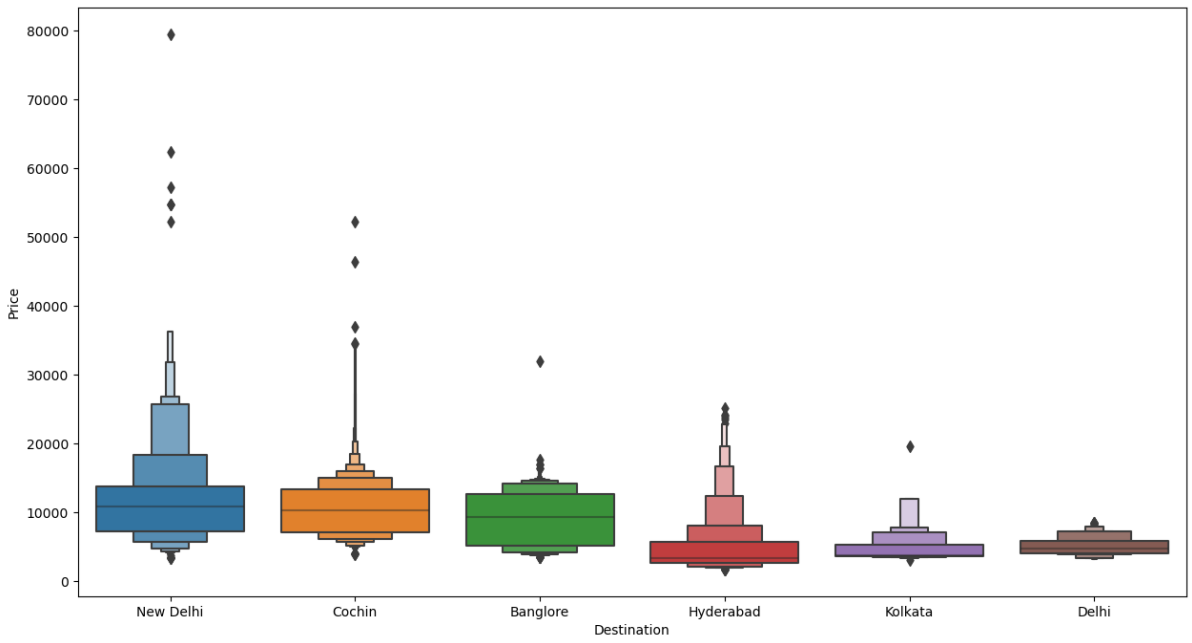
	Chennai	Delhi	Kolkata	Mumbai
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	0

In [50]:
```
categorical['Destination'].value_counts()
```

Out[50]:
```
Cochin       4536
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: Destination, dtype: int64
```

In [51]:
```
plt.figure(figsize=(15,8))
sns.boxenplot(x='Destination',y='Price',data=df.sort_values('Price'
```

Out[51]:
```
<Axes: xlabel='Destination', ylabel='Price'>
```



In [52]:
```
#encoding of destination column
destination=pd.get_dummies(categorical['Destination'],drop_first=Tr
destination.head()
```

Out[52]:

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

In [53]:
```
# now work on route column
categorical['Route'].value_counts()
```

Out[53]:
```
DEL → BOM → COK          2376
BLR → DEL                1552
CCU → BOM → BLR           979
CCU → BLR                 724
BOM → HYD                 621
                         ...
CCU → VTZ → BLR             1
CCU → IXZ → MAA → BLR       1
BOM → COK → MAA → HYD       1
BOM → CCU → HYD             1
BOM → BBI → HYD             1
Name: Route, Length: 128, dtype: int64
```

In [54]:
```
categorical['Route1']=categorical['Route'].str.split('→').str[0]
categorical['Route2']=categorical['Route'].str.split('→').str[1]
categorical['Route3']=categorical['Route'].str.split('→').str[2]
categorical['Route4']=categorical['Route'].str.split('→').str[3]
categorical['Route5']=categorical['Route'].str.split('→').str[4]
```

In [55]:
```
categorical.head()
```

Out[55]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	duration	dur_hour	d
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	2h 50m	2	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	7h 25m	7	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info	19h 0m	19	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info	5h 25m	5	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info	4h 45m	4	

In [56]:
```
drop_col(categorical, 'Route')
```

In [57]:
```
categorical.isnull().sum()
```

Out[57]:
```
Airline              0
Source               0
Destination          0
Total_Stops          0
Additional_Info      0
duration             0
dur_hour             0
dur_min           1032
Route1               0
Route2               0
Route3            3491
Route4            9116
Route5           10636
dtype: int64
```

In [58]:
```
categorical.columns
```

Out[58]:
```
Index(['Airline', 'Source', 'Destination', 'Total_Stops', 'Additio
nal_Info',
       'duration', 'dur_hour', 'dur_min', 'Route1', 'Route2', 'Rou
te3',
       'Route4', 'Route5'],
      dtype='object')
```

In [59]:
```
for i in ['Route3', 'Route4', 'Route5']:
    categorical[i].fillna('None',inplace=True)
```

In [60]:
```
categorical.isnull().sum()
```

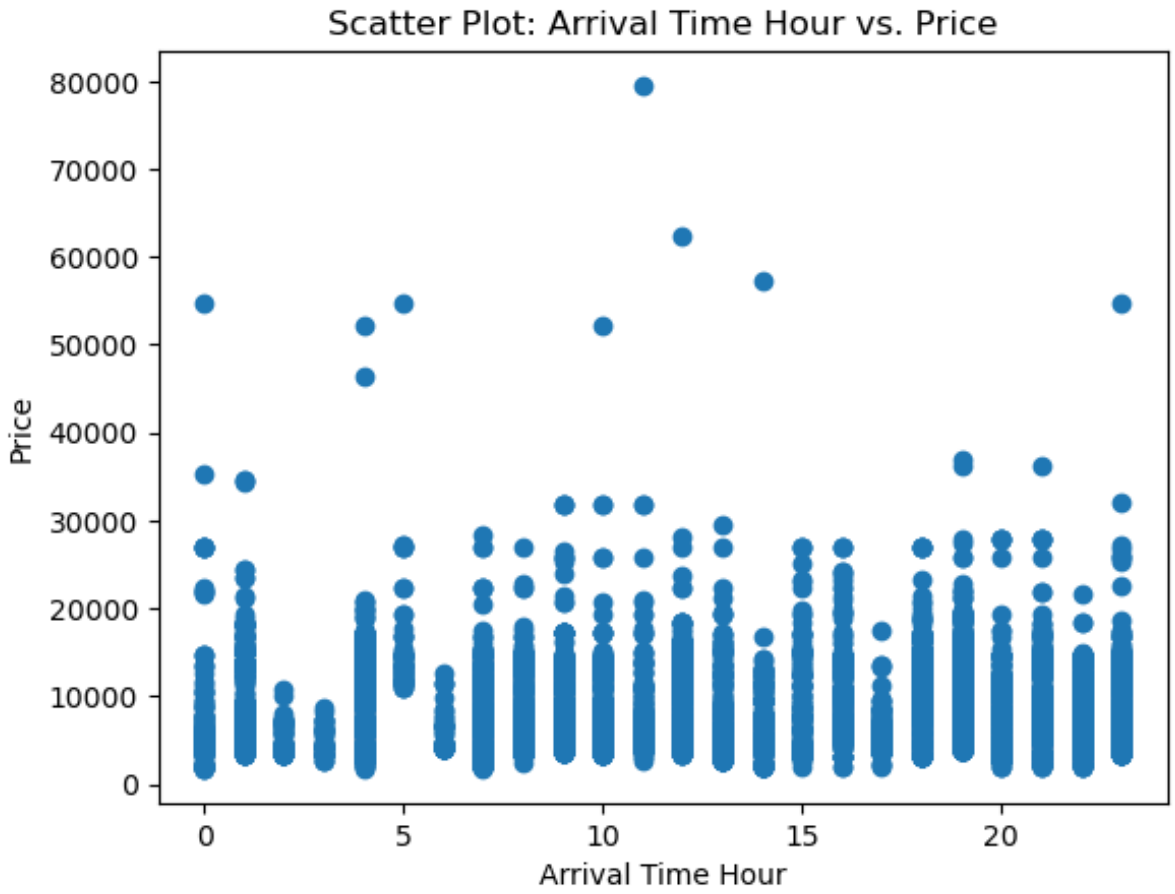
Out[60]:
```
Airline              0
Source               0
Destination          0
Total_Stops          0
Additional_Info      0
duration             0
dur_hour             0
dur_min           1032
Route1               0
Route2               0
Route3               0
Route4               0
Route5               0
dtype: int64
```

In [61]:
```
for i in categorical.columns:
    print('{} has total {} categories'.format(i,len(categorical[i].
```

```
Airline has total 12 categories
Source has total 5 categories
Destination has total 6 categories
Total_Stops has total 5 categories
Additional_Info has total 10 categories
duration has total 368 categories
dur_hour has total 43 categories
dur_min has total 11 categories
Route1 has total 5 categories
Route2 has total 45 categories
Route3 has total 30 categories
Route4 has total 14 categories
Route5 has total 6 categories
```

In [62]:
```
plt.scatter(df['Arrival_Time_hour'], df['Price'])
plt.xlabel('Arrival Time Hour')
plt.ylabel('Price')
plt.title('Scatter Plot: Arrival Time Hour vs. Price')

plt.show()
```



In [63]:
```
# Applying label encoder
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

In [64]:
```
for i in ['Route1', 'Route2', 'Route3', 'Route4', 'Route5']:
    categorical[i]=encoder.fit_transform(categorical[i])
```

In [65]:
```
categorical.head()
```

Out[65]:

	Airline	Source	Destination	Total_Stops	Additional_Info	duration	dur_hour	dur_min
0	IndiGo	Banglore	New Delhi	non-stop	No info	2h 50m	2	50
1	Air India	Kolkata	Banglore	2 stops	No info	7h 25m	7	25
2	Jet Airways	Delhi	Cochin	2 stops	No info	19h 0m	19	None
3	IndiGo	Kolkata	Banglore	1 stop	No info	5h 25m	5	25
4	IndiGo	Banglore	New Delhi	1 stop	No info	4h 45m	4	45

In [66]:
```
drop_col(categorical, 'Additional_Info')
```

In [67]:
```
categorical['Total_Stops'].unique()
```

Out[67]:
```
array(['non-stop', '2 stops', '1 stop', '3 stops', '4 stops'],
      dtype=object)
```

In [68]:
```
# encoding Total stops
dict={'non-stop':0, '2 stops':2, '1 stop':1, '3 stops':3, '4 stops'
categorical['Total_Stops']=categorical['Total_Stops'].map(dict)
```

In [69]:
```
categorical['Total_Stops']
```

Out[69]:
```
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
```

In [70]:
```
drop_col(categorical,'Source')
drop_col(categorical,'Destination')
drop_col(categorical,'Airline')
```

Model

In [95]:
```
final_df=pd.concat([categorical,Airline,source,destination,df[conti
```

In [96]:
```
final_df.head()
```

Out[96]:

	Total_Stops	duration	dur_hour	dur_min	Route1	Route2	Route3	Route4	Route5	A Ind
0	0	2h 50m	2	50	0	13	29	13	5	
1	2	7h 25m	7	25	2	25	1	3	5	
2	2	19h 0m	19	None	3	32	4	5	5	
3	1	5h 25m	5	25	2	34	3	13	5	
4	1	4h 45m	4	45	0	34	8	13	5	

5 rows × 36 columns

In [97]:
```
pd.set_option('display.max_columns',33)
final_df.head()
```

Out[97]:

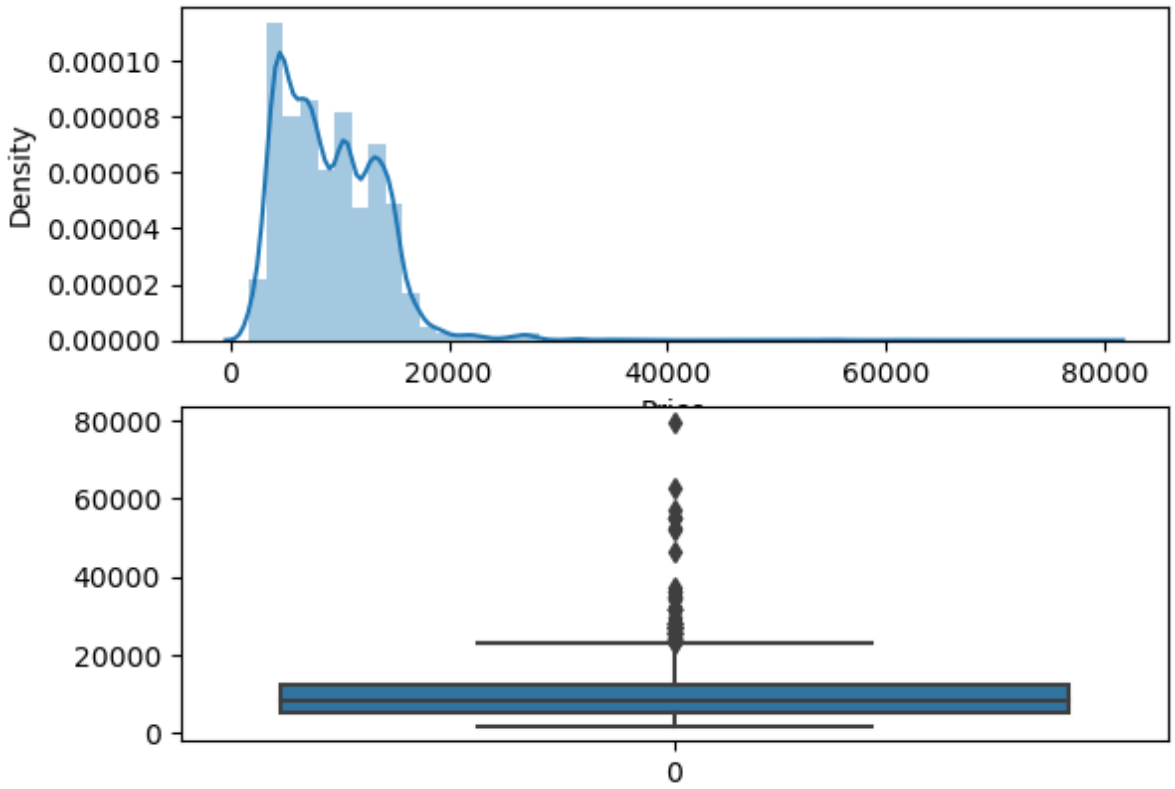
	Total_Stops	duration	dur_hour	dur_min	Route1	Route2	Route3	Route4	Route5	A Ind
0	0	2h 50m	2	50	0	13	29	13	5	
1	2	7h 25m	7	25	2	25	1	3	5	
2	2	19h 0m	19	None	3	32	4	5	5	
3	1	5h 25m	5	25	2	34	3	13	5	
4	1	4h 45m	4	45	0	34	8	13	5	

5 rows × 36 columns

Checking for outliers

In [98]:
```
def plot(data,col):
    fig,(ax1,ax2)=plt.subplots(2,1)
    sns.distplot(data[col],ax=ax1)
    sns.boxplot(data[col],ax=ax2)
```

In [99]:
```
plot(final_df, 'Price')
```

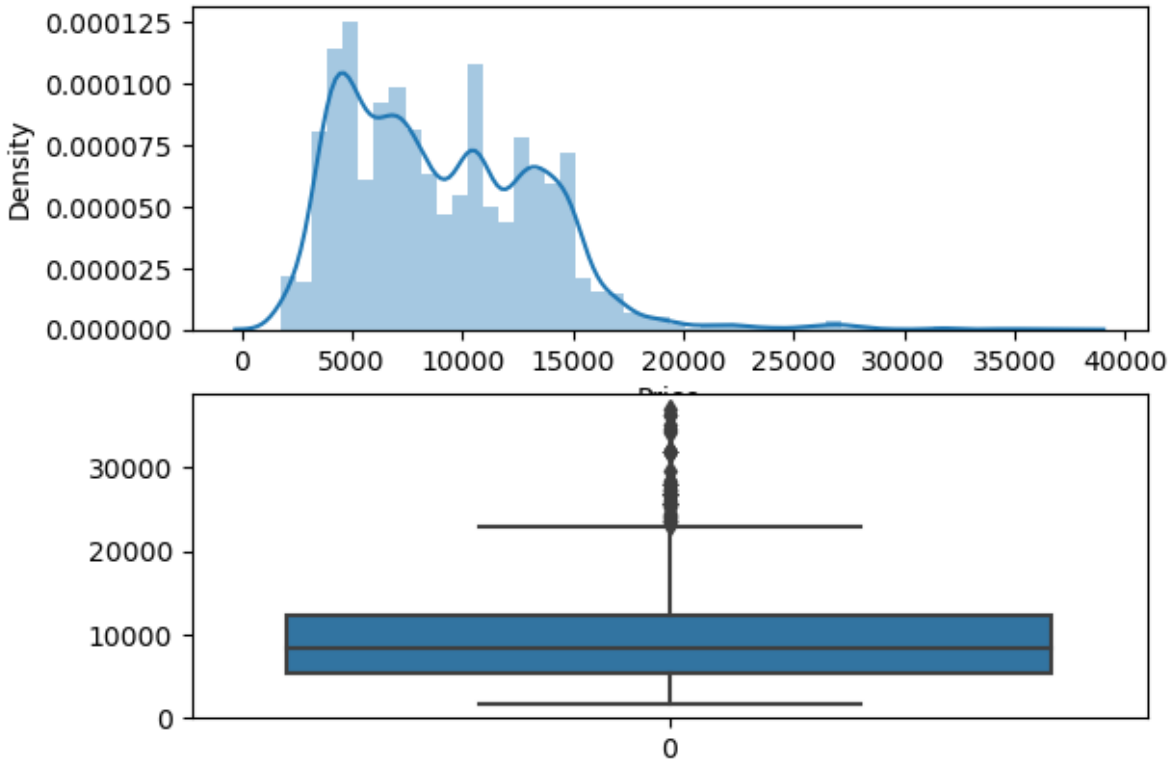


Handling outliers:

As there is some outliers in price feature,so we replace it with median.

In [100]:
```
final_df['Price']=np.where(final_df['Price']>=40000,final_df['Price
```

In [101]:
```
plot(final_df, 'Price')
```



Separate the dataset in X and Y columns

In [102]:
```
X= final_df.drop('Price',axis=1)
y= df['Price']
```

Feature Selection

This revolves around identifying the optimal feature that exhibits a strong relationship with the independent variable, consequently mitigating issues associated with dimensionality reduction. To address this, we employ the mutual_info_classif method.

In [103]:
```
from sklearn.feature_selection import mutual_info_classif
```

In [112]:
```
mutual_info_classif(X,y)
```

```
---------------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
Cell In[112], line 1
----> 1 mutual_info_classif(X,y)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/_param
_validation.py:211, in validate_params.<locals>.decorator.<locals>
.wrapper(*args, **kwargs)
    205 try:
    206     with config_context(
    207         skip_parameter_validation=(
    208             prefer_skip_nested_validation or
global_skip_validation
    209         )
    210     ):
--> 211         return func(*args, **kwargs)
    212 except InvalidParameterError as e:
```

In [113]:
```
imp= pd.DataFrame(mutual_info_classif(X,y),index=X.columns)
imp
```

```
---------------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
Cell In[113], line 1
----> 1 imp= pd.DataFrame(mutual_info_classif(X,y),index=X.columns
)
      2 imp

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/_param
_validation.py:211, in validate_params.<locals>.decorator.<locals>
.wrapper(*args, **kwargs)
    205 try:
    206     with config_context(
    207         skip_parameter_validation=(
    208             prefer_skip_nested_validation or
global_skip_validation
    209         )
    210     ):
--> 211         return func(*args, **kwargs)
    212 except InvalidParameterError as e:
    213     # When the function is just a wrapper around an estima
tor, we allow
    214     # the function to delegate validation to the estimator
, but we replace
    215     # the name of the estimator by the name of the functio
n in the error
    216     # message to avoid confusion.
```

```
    217     msg = re.sub(
    218         r"parameter of \w+ must be",
    219         f"parameter of {func.__qualname__} must be",
    220         str(e),
    221     )

File ~/anaconda3/lib/python3.11/site-packages/sklearn/feature_sele
ction/_mutual_info.py:493, in mutual_info_classif(X, y, discrete_f
eatures, n_neighbors, copy, random_state)
    419 """Estimate mutual information for a discrete target varia
ble.
    420
    421 Mutual information (MI) [1]_ between two random variables
is a non-negative
   (...)
    490        of a Random Vector:, Probl. Peredachi Inf., 23:2 (1
987), 9-16
    491 """
    492 check_classification_targets(y)
--> 493 return _estimate_mi(X, y, discrete_features, True, n_neigh
bors, copy, random_state)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/feature_sele
ction/_mutual_info.py:258, in _estimate_mi(X, y, discrete_features
, discrete_target, n_neighbors, copy, random_state)
    201 def _estimate_mi(
    202     X,
    203     y,
   (...)
    208     random_state=None,
    209 ):
    210     """Estimate mutual information between the features an
d the target.
    211
    212     Parameters
   (...)
    256           Data Sets". PLoS ONE 9(2), 2014.
    257     """
--> 258     X, y = check_X_y(X, y, accept_sparse="csc", y_numeric=
not discrete_target)
    259     n_samples, n_features = X.shape
    261     if isinstance(discrete_features, (str, bool)):

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/valida
tion.py:1147, in check_X_y(X, y, accept_sparse, accept_large_spars
e, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, mult
i_output, ensure_min_samples, ensure_min_features, y_numeric, esti
mator)
   1142         estimator_name = _check_estimator_name(estimator)
   1143     raise ValueError(
   1144         f"{estimator_name} requires y to be passed, but th
e target y is None"
   1145     )
-> 1147 X = check_array(
```

```
   1148     X,
   1149     accept_sparse=accept_sparse,
   1150     accept_large_sparse=accept_large_sparse,
   1151     dtype=dtype,
   1152     order=order,
   1153     copy=copy,
   1154     force_all_finite=force_all_finite,
   1155     ensure_2d=ensure_2d,
   1156     allow_nd=allow_nd,
   1157     ensure_min_samples=ensure_min_samples,
   1158     ensure_min_features=ensure_min_features,
   1159     estimator=estimator,
   1160     input_name="X",
   1161 )
   1163 y = _check_y(y, multi_output=multi_output, y_numeric=y_num
eric, estimator=estimator)
   1165 check_consistent_length(X, y)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/valida
tion.py:917, in check_array(array, accept_sparse, accept_large_spa
rse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, en
sure_min_samples, ensure_min_features, estimator, input_name)
    915         array = xp.astype(array, dtype, copy=False)
    916     else:
--> 917         array = _asarray_with_order(array, order=order, dt
ype=dtype, xp=xp)
    918 except ComplexWarning as complex_warning:
    919     raise ValueError(
    920         "Complex data not supported\n{}\n".format(array)
    921     ) from complex_warning

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/_array
_api.py:380, in _asarray_with_order(array, dtype, order, copy, xp)
    378     array = numpy.array(array, order=order, dtype=dtype)
    379 else:
--> 380     array = numpy.asarray(array, order=order, dtype=dtype)
    382 # At this point array is a NumPy ndarray. We convert it to
an array
    383 # container that is consistent with the input's namespace.
    384 return xp.asarray(array)

File ~/anaconda3/lib/python3.11/site-packages/pandas/core/generic.
py:2070, in NDFrame.__array__(self, dtype)
   2069 def __array__(self, dtype: npt.DTypeLike | None = None) ->
np.ndarray:
-> 2070     return np.asarray(self._values, dtype=dtype)

ValueError: could not convert string to float: '2h 50m'
```

In [114]:
```
imp.columns=['importance']
imp.sort_values(by='importance',ascending=False)
```

```
---------------------------------------------------------------------
---------
NameError                                 Traceback (most recent c
all last)
Cell In[114], line 1
----> 1 imp.columns=['importance']
      2 imp.sort_values(by='importance',ascending=False)

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

As we can see from the table, there are several features that have approx 0 value,so we should remove it after authorizing it.But for now,we are not removing it.

Models

In [115]:
```
# spiliting the dataset
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20
```

In [116]:
```
from sklearn.metrics import r2_score,mean_absolute_error,mean_squar
def predict(ml_model):
    print('Model is: {}'.format(ml_model))
    model= ml_model.fit(X_train,y_train)
    print("Training score: {}".format(model.score(X_train,y_train))
    predictions = model.predict(X_test)
    print("Predictions are: {}".format(predictions))
    print('\n')
    r2score=r2_score(y_test,predictions)
    print("r2 score is: {}".format(r2score))

    print('MAE:{}'.format(mean_absolute_error(y_test,predictions)))
    print('MSE:{}'.format(mean_squared_error(y_test,predictions)))
    print('RMSE:{}'.format(np.sqrt(mean_squared_error(y_test,predic

    sns.distplot(y_test-predictions)
```

In [117]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor,RandomForestl
```

In [118]:
```
predict(RandomForestRegressor())
```

```
Model is: RandomForestRegressor()

---------------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
Cell In[118], line 1
----> 1 predict(RandomForestRegressor())

Cell In[116], line 4, in predict(ml_model)
      2 def predict(ml_model):
      3     print('Model is: {}'.format(ml_model))
----> 4     model= ml_model.fit(X_train,y_train)
      5     print("Training score: {}".format(model.score(X_train,
y_train)))
      6     predictions = model.predict(X_test)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/base.py:1151
, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *
args, **kwargs)
```

From the graph,it is clear that we predicted 84% correctly.

In [119]:
```
predict(LogisticRegression())
```

```
Model is: LogisticRegression()

---------------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
Cell In[119], line 1
----> 1 predict(LogisticRegression())

Cell In[116], line 4, in predict(ml_model)
      2 def predict(ml_model):
      3     print('Model is: {}'.format(ml_model))
----> 4     model= ml_model.fit(X_train,y_train)
      5     print("Training score: {}".format(model.score(X_train,
y_train)))
      6     predictions = model.predict(X_test)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/base.py:1151
, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *
args, **kwargs)
   1144     estimator._validate_params()
   1146 with config_context(
   1147     skip_parameter_validation=(
   1148         prefer_skip_nested_validation or
```

```
global_skip_validation
   1149     )
   1150 ):
-> 1151     return fit_method(estimator, *args, **kwargs)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/linear_model
/_logistic.py:1207, in LogisticRegression.fit(self, X, y, sample_w
eight)
   1204 else:
   1205     _dtype = [np.float64, np.float32]
-> 1207 X, y = self._validate_data(
   1208     X,
   1209     y,
   1210     accept_sparse="csr",
   1211     dtype=_dtype,
   1212     order="C",
   1213     accept_large_sparse=solver not in ["liblinear", "sag",
"saga"],
   1214 )
   1215 check_classification_targets(y)
   1216 self.classes_ = np.unique(y)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/base.py:621,
in BaseEstimator._validate_data(self, X, y, reset, validate_separa
tely, cast_to_ndarray, **check_params)
    619         y = check_array(y, input_name="y", **check_y_param
s)
    620     else:
--> 621         X, y = check_X_y(X, y, **check_params)
    622     out = X, y
    624 if not no_val_X and check_params.get("ensure_2d", True):

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/valida
tion.py:1147, in check_X_y(X, y, accept_sparse, accept_large_spars
e, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, mult
i_output, ensure_min_samples, ensure_min_features, y_numeric, esti
mator)
   1142         estimator_name = _check_estimator_name(estimator)
   1143     raise ValueError(
   1144         f"{estimator_name} requires y to be passed, but th
e target y is None"
   1145     )
-> 1147 X = check_array(
   1148     X,
   1149     accept_sparse=accept_sparse,
   1150     accept_large_sparse=accept_large_sparse,
   1151     dtype=dtype,
   1152     order=order,
   1153     copy=copy,
   1154     force_all_finite=force_all_finite,
   1155     ensure_2d=ensure_2d,
   1156     allow_nd=allow_nd,
   1157     ensure_min_samples=ensure_min_samples,
   1158     ensure_min_features=ensure_min_features,
```

```
   1159     estimator=estimator,
   1160     input_name="X",
   1161 )
   1163 y = _check_y(y, multi_output=multi_output, y_numeric=y_num
eric, estimator=estimator)
   1165 check_consistent_length(X, y)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/valida
tion.py:917, in check_array(array, accept_sparse, accept_large_spa
rse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, en
sure_min_samples, ensure_min_features, estimator, input_name)
    915         array = xp.astype(array, dtype, copy=False)
    916     else:
--> 917         array = _asarray_with_order(array, order=order, dt
ype=dtype, xp=xp)
    918 except ComplexWarning as complex_warning:
    919     raise ValueError(
    920         "Complex data not supported\n{}\n".format(array)
    921     ) from complex_warning

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/_array
_api.py:380, in _asarray_with_order(array, dtype, order, copy, xp)
    378     array = numpy.array(array, order=order, dtype=dtype)
    379 else:
--> 380     array = numpy.asarray(array, order=order, dtype=dtype)
    382 # At this point array is a NumPy ndarray. We convert it to
an array
    383 # container that is consistent with the input's namespace.
    384 return xp.asarray(array)

File ~/anaconda3/lib/python3.11/site-packages/pandas/core/generic.
py:2070, in NDFrame.__array__(self, dtype)
   2069 def __array__(self, dtype: npt.DTypeLike | None = None) ->
np.ndarray:
-> 2070     return np.asarray(self._values, dtype=dtype)

ValueError: could not convert string to float: '1h 30m'
```

In [120]:
```
predict(KNeighborsRegressor())
```

```
Model is: KNeighborsRegressor()

---------------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
Cell In[120], line 1
----> 1 predict(KNeighborsRegressor())

Cell In[116], line 4, in predict(ml_model)
      2 def predict(ml_model):
      3     print('Model is: {}'.format(ml_model))
----> 4     model= ml_model.fit(X_train,y_train)
      5     print("Training score: {}".format(model.score(X_train,
y_train)))
      6     predictions = model.predict(X_test)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/base.py:1151
, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *
args, **kwargs)
```

In [121]:
```
predict(DecisionTreeRegressor())
```

```
Model is: DecisionTreeRegressor()

---------------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
Cell In[121], line 1
----> 1 predict(DecisionTreeRegressor())

Cell In[116], line 4, in predict(ml_model)
      2 def predict(ml_model):
      3     print('Model is: {}'.format(ml_model))
----> 4     model= ml_model.fit(X_train,y_train)
      5     print("Training score: {}".format(model.score(X_train,
y_train)))
      6     predictions = model.predict(X_test)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/base.py:1151
, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *
args, **kwargs)
   1144     estimator._validate_params()
   1146 with config_context(
   1147     skip_parameter_validation=(
   1148         prefer_skip_nested_validation or
global_skip_validation
   1149     )
   1150 ):
-> 1151     return fit_method(estimator, *args, **kwargs)
```

```
File ~/anaconda3/lib/python3.11/site-packages/sklearn/tree/_classes.py:1320, in DecisionTreeRegressor.fit(self, X, y, sample_weight, check_input)
   1290 @_fit_context(prefer_skip_nested_validation=True)
   1291 def fit(self, X, y, sample_weight=None, check_input=True):
   1292     """Build a decision tree regressor from the training set (X, y).
   1293
   1294     Parameters
   (...)
   1317         Fitted estimator.
   1318     """
-> 1320     super()._fit(
   1321         X,
   1322         y,
   1323         sample_weight=sample_weight,
   1324         check_input=check_input,
   1325     )
   1326     return self

File ~/anaconda3/lib/python3.11/site-packages/sklearn/tree/_classes.py:242, in BaseDecisionTree._fit(self, X, y, sample_weight, check_input, missing_values_in_feature_mask)
    238 check_X_params = dict(
    239     dtype=DTYPE, accept_sparse="csc", force_all_finite=False
    240 )
    241 check_y_params = dict(ensure_2d=False, dtype=None)
--> 242 X, y = self._validate_data(
    243     X, y, validate_separately=(check_X_params, check_y_params)
    244 )
    246 missing_values_in_feature_mask = (
    247     self._compute_missing_values_in_feature_mask(X)
    248 )
    249 if issparse(X):

File ~/anaconda3/lib/python3.11/site-packages/sklearn/base.py:616, in BaseEstimator._validate_data(self, X, y, reset, validate_separately, cast_to_ndarray, **check_params)
    614 if "estimator" not in check_X_params:
    615     check_X_params = {**default_check_params, **check_X_params}
--> 616 X = check_array(X, input_name="X", **check_X_params)
    617 if "estimator" not in check_y_params:
    618     check_y_params = {**default_check_params, **check_y_params}

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/validation.py:917, in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator, input_name)
    915         array = xp.astype(array, dtype, copy=False)
```

```
    916     else:
--> 917         array = _asarray_with_order(array, order=order, dt
ype=dtype, xp=xp)
    918 except ComplexWarning as complex_warning:
    919     raise ValueError(
    920         "Complex data not supported\n{}\n".format(array)
    921     ) from complex_warning

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/_array
_api.py:380, in _asarray_with_order(array, dtype, order, copy, xp)
    378     array = numpy.array(array, order=order, dtype=dtype)
    379 else:
--> 380     array = numpy.asarray(array, order=order, dtype=dtype)
    382 # At this point array is a NumPy ndarray. We convert it to
an array
    383 # container that is consistent with the input's namespace.
    384 return xp.asarray(array)

File ~/anaconda3/lib/python3.11/site-packages/pandas/core/generic.
py:2070, in NDFrame.__array__(self, dtype)
   2069 def __array__(self, dtype: npt.DTypeLike | None = None) ->
np.ndarray:
-> 2070     return np.asarray(self._values, dtype=dtype)

ValueError: could not convert string to float: '1h 30m'
```

In [122]:
```
from sklearn.svm import SVR
predict(SVR())
```

```
Model is: SVR()

---------------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
Cell In[122], line 2
      1 from sklearn.svm import SVR
----> 2 predict(SVR())

Cell In[116], line 4, in predict(ml_model)
      2 def predict(ml_model):
      3     print('Model is: {}'.format(ml_model))
----> 4     model= ml_model.fit(X_train,y_train)
      5     print("Training score: {}".format(model.score(X_train,
y_train)))
      6     predictions = model.predict(X_test)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/base.py:1151
, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *
args, **kwargs)
   1144     estimator._validate_params()
   1146 with config_context(
```

```
   1147     skip_parameter_validation=(
   1148         prefer_skip_nested_validation or
global_skip_validation
   1149     )
   1150 ):
-> 1151     return fit_method(estimator, *args, **kwargs)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/svm/_base.py
:190, in BaseLibSVM.fit(self, X, y, sample_weight)
    188     check_consistent_length(X, y)
    189 else:
--> 190     X, y = self._validate_data(
    191         X,
    192         y,
    193         dtype=np.float64,
    194         order="C",
    195         accept_sparse="csr",
    196         accept_large_sparse=False,
    197     )
    199 y = self._validate_targets(y)
    201 sample_weight = np.asarray(
    202     [] if sample_weight is None else sample_weight, dtype=
np.float64
    203 )

File ~/anaconda3/lib/python3.11/site-packages/sklearn/base.py:621,
in BaseEstimator._validate_data(self, X, y, reset, validate_separa
tely, cast_to_ndarray, **check_params)
    619         y = check_array(y, input_name="y", **check_y_param
s)
    620     else:
--> 621         X, y = check_X_y(X, y, **check_params)
    622     out = X, y
    624 if not no_val_X and check_params.get("ensure_2d", True):

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/valida
tion.py:1147, in check_X_y(X, y, accept_sparse, accept_large_spars
e, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, mult
i_output, ensure_min_samples, ensure_min_features, y_numeric, esti
mator)
   1142         estimator_name = _check_estimator_name(estimator)
   1143     raise ValueError(
   1144         f"{estimator_name} requires y to be passed, but th
e target y is None"
   1145     )
-> 1147 X = check_array(
   1148     X,
   1149     accept_sparse=accept_sparse,
   1150     accept_large_sparse=accept_large_sparse,
   1151     dtype=dtype,
   1152     order=order,
   1153     copy=copy,
   1154     force_all_finite=force_all_finite,
   1155     ensure_2d=ensure_2d,
```

```
   1156     allow_nd=allow_nd,
   1157     ensure_min_samples=ensure_min_samples,
   1158     ensure_min_features=ensure_min_features,
   1159     estimator=estimator,
   1160     input_name="X",
   1161 )
   1163 y = _check_y(y, multi_output=multi_output, y_numeric=y_num
eric, estimator=estimator)
   1165 check_consistent_length(X, y)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/valida
tion.py:917, in check_array(array, accept_sparse, accept_large_spa
rse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, en
sure_min_samples, ensure_min_features, estimator, input_name)
    915         array = xp.astype(array, dtype, copy=False)
    916     else:
--> 917         array = _asarray_with_order(array, order=order, dt
ype=dtype, xp=xp)
    918 except ComplexWarning as complex_warning:
    919     raise ValueError(
    920         "Complex data not supported\n{}\n".format(array)
    921     ) from complex_warning

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/_array
_api.py:380, in _asarray_with_order(array, dtype, order, copy, xp)
    378     array = numpy.array(array, order=order, dtype=dtype)
    379 else:
--> 380     array = numpy.asarray(array, order=order, dtype=dtype)
    382 # At this point array is a NumPy ndarray. We convert it to
an array
    383 # container that is consistent with the input's namespace.
    384 return xp.asarray(array)

File ~/anaconda3/lib/python3.11/site-packages/pandas/core/generic.
py:2070, in NDFrame.__array__(self, dtype)
   2069 def __array__(self, dtype: npt.DTypeLike | None = None) ->
np.ndarray:
-> 2070     return np.asarray(self._values, dtype=dtype)

ValueError: could not convert string to float: '1h 30m'
```

In [123]:
```
predict(GradientBoostingRegressor())
```

```
Model is: GradientBoostingRegressor()

---------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
Cell In[123], line 1
----> 1 predict(GradientBoostingRegressor())
```

```
Cell In[116], line 4, in predict(ml_model)
      2 def predict(ml_model):
      3     print('Model is: {}'.format(ml_model))
----> 4     model= ml_model.fit(X_train,y_train)
      5     print("Training score: {}".format(model.score(X_train,
y_train)))
      6     predictions = model.predict(X_test)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/base.py:1151
, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *
args, **kwargs)
   1144     estimator._validate_params()
   1146 with config_context(
   1147     skip_parameter_validation=(
   1148         prefer_skip_nested_validation or
global_skip_validation
   1149     )
   1150 ):
-> 1151     return fit_method(estimator, *args, **kwargs)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/ensemble/_gb
.py:416, in BaseGradientBoosting.fit(self, X, y, sample_weight, mo
nitor)
    410     self._clear_state()
    412 # Check input
    413 # Since check_array converts both X and y to the same dtyp
e, but the
    414 # trees use different types for X and y, checking them sep
arately.
--> 416 X, y = self._validate_data(
    417     X, y, accept_sparse=["csr", "csc", "coo"], dtype=DTYPE
, multi_output=True
    418 )
    420 sample_weight_is_none = sample_weight is None
    422 sample_weight = _check_sample_weight(sample_weight, X)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/base.py:621,
in BaseEstimator._validate_data(self, X, y, reset, validate_separa
tely, cast_to_ndarray, **check_params)
    619         y = check_array(y, input_name="y", **check_y_param
s)
    620     else:
--> 621         X, y = check_X_y(X, y, **check_params)
    622     out = X, y
    624 if not no_val_X and check_params.get("ensure_2d", True):

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/valida
tion.py:1147, in check_X_y(X, y, accept_sparse, accept_large_spars
e, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, mult
i_output, ensure_min_samples, ensure_min_features, y_numeric, esti
mator)
   1142         estimator_name = _check_estimator_name(estimator)
   1143     raise ValueError(
   1144         f"{estimator_name} requires y to be passed, but th
```

```
e target y is None"
   1145     )
-> 1147 X = check_array(
   1148     X,
   1149     accept_sparse=accept_sparse,
   1150     accept_large_sparse=accept_large_sparse,
   1151     dtype=dtype,
   1152     order=order,
   1153     copy=copy,
   1154     force_all_finite=force_all_finite,
   1155     ensure_2d=ensure_2d,
   1156     allow_nd=allow_nd,
   1157     ensure_min_samples=ensure_min_samples,
   1158     ensure_min_features=ensure_min_features,
   1159     estimator=estimator,
   1160     input_name="X",
   1161 )
   1163 y = _check_y(y, multi_output=multi_output, y_numeric=y_num
eric, estimator=estimator)
   1165 check_consistent_length(X, y)

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/valida
tion.py:917, in check_array(array, accept_sparse, accept_large_spa
rse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, en
sure_min_samples, ensure_min_features, estimator, input_name)
    915         array = xp.astype(array, dtype, copy=False)
    916     else:
--> 917         array = _asarray_with_order(array, order=order, dt
ype=dtype, xp=xp)
    918 except ComplexWarning as complex_warning:
    919     raise ValueError(
    920         "Complex data not supported\n{}\n".format(array)
    921     ) from complex_warning

File ~/anaconda3/lib/python3.11/site-packages/sklearn/utils/_array
_api.py:380, in _asarray_with_order(array, dtype, order, copy, xp)
    378     array = numpy.array(array, order=order, dtype=dtype)
    379 else:
--> 380     array = numpy.asarray(array, order=order, dtype=dtype)
    382 # At this point array is a NumPy ndarray. We convert it to
an array
    383 # container that is consistent with the input's namespace.
    384 return xp.asarray(array)

File ~/anaconda3/lib/python3.11/site-packages/pandas/core/generic.
py:2070, in NDFrame.__array__(self, dtype)
   2069 def __array__(self, dtype: npt.DTypeLike | None = None) ->
np.ndarray:
-> 2070     return np.asarray(self._values, dtype=dtype)

ValueError: could not convert string to float: '1h 30m'
```

Hyper tuning the model

In [ ]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [ ]:
```
random_grid = {
    'n_estimators' : [100, 120, 150, 180, 200,220],
    'max_features':['auto','sqrt'],
    'max_depth':[5,10,15,20],
    }
```

In [ ]:
```
rf=RandomForestRegressor()
rf_random=RandomizedSearchCV(estimator=rf,param_distributions=rando

rf_random.fit(X_train,y_train)

# best parameter
rf_random.best_params_
```

In [ ]:
```
# best parameter
rf_random.best_params_
```

In [ ]:
```
#predicting the values
prediction = rf_random.predict(X_test)

#distribution plot between actual value and predicted value
sns.displot(y_test-prediction)
```

In [ ]:
```
r2_score(y_test,prediction)
```

After hypertuning,the accuracy increases .

In [ ]:
```

```