Article

Flight Fare Prediction

Problem Statement

Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, it will be a different story. We might have often heard travelers saying that flight ticket prices are so unpredictable. Here you will be provided with prices of flight tickets for various airlines between the months of March and June of 2019 and between various cities.

Problem Definition

In this article, we are going to analyze and predict flight fare prediction using Machine Learning, and exploratory data analysis techniques. with the help of features, we are going to predict the flight price. So basically there are two datasets available

Train dataset
Test dataset

About the Dataset

The dataset contains several features based on that features we are going to predict the price.

FEATURES:

Airline: The name of the airline.

Date_of_Journey: The date of the journey

Source: The source from which the service begins.

Destination: The destination where the service ends.

Route: The route was taken by the flight to reach the destination.

Dep_Time: The time when the journey starts from the source.

Arrival_Time: Time of arrival at the destination.

Duration: Total duration of the flight.

Total_Stops: Total stops between the source and destination.

Additional_Info: Additional information about the flight

Price: The price of the ticket

Importing libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pickle
from pandas.plotting import scatter_matrix
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
from sklearn.metrics import r2_score
from sklearn import metrics
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.ensemble import RandomForestClassifier
```

DATA

Train Data

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302
5	SpiceJet	24/06/2019	Kolkata	Banglore	CCU → BLR	09:00	11:25	2h 25m	non-stop	No info	3873
6	Jet Airways	12/03/2019	Banglore	New Delhi	BLR → BOM → DEL	18:55	10:25 13 Mar	15h 30m	1 stop	In-flight meal not included	11087
7	Jet Airways	01/03/2019	Banglore	New Delhi	BLR → BOM → DEL	08:00	05:05 02 Mar	21h 5m	1 stop	No info	22270
8	Jet Airways	12/03/2019	Banglore	New Delhi	BLR → BOM → DEL	08:55	10:25 13 Mar	25h 30m	1 stop	In-flight meal not included	11087
9	Multiple carriers	27/05/2019	Delhi	Cochin	DEL → BOM → COK	11:25	19:15	7h 50m	1 stop	No info	8625

Test Data

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL → BOM → COK	17:30	04:25 07 Jun	10h 55m	1 stop	No info
1	IndiGo	12/05/2019	Kolkata	Banglore	CCU → MAA → BLR	06:20	10:20	4h	1 stop	No info
2	Jet Airways	21/05/2019	Delhi	Cochin	DEL → BOM → COK	19:15	19:00 22 May	23h 45m	1 stop	In-flight meal not included
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL → BOM → COK	08:00	21:00	13h	1 stop	No info
4	Air Asia	24/06/2019	Banglore	Delhi	BLR → DEL	23:55	02:45 25 Jun	2h 50m	non-stop	No info
5	Jet Airways	12/06/2019	Delhi	Cochin	DEL → BOM → COK	18:15	12:35 13 Jun	18h 20m	1 stop	In-flight meal not included
6	Air India	12/03/2019	Banglore	New Delhi	BLR → TRV → DEL	07:30	22:35	15h 5m	1 stop	No info
7	IndiGo	1/05/2019	Kolkata	Banglore	CCU → HYD → BLR	15:15	20:30	5h 15m	1 stop	No info
8	IndiGo	15/03/2019	Kolkata	Banglore	CCU → BLR	10:10	12:55	2h 45m	non-stop	No info
9	Jet Airways	18/05/2019	Kolkata	Banglore	CCU → BOM → BLR	16:30	22:35	6h 5m	1 stop	No info

First I am train data

Checking null value in train data

Here we can see that in Total_stops column there is a Null value

data_train.isnull().sum()

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

knowing some information regarding dataset

data_train.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
```

Dropping null values

data_train =data_train.dropna(axis =0)
data_train.isnull().sum()

```
Airline            0
```

```
Date_of_Journey    0
Source             0
Destination        0
Route              0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Additional_Info    0
Price              0
dtype: int64
```

Using value_counts function in some of the columns

data_train["Additional_Info"].value_counts()

```
No info                         8344
In-flight meal not included     1982
No check-in baggage included     320
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
```

data_train['Duration'].value_counts()

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

```
data_train['Airline'].value_counts()
Jet Airways                          3849
IndiGo                               2053
Air India                            1751
Multiple carriers                    1196
```

```
SpiceJet                                  818
Vistara                                   479
Air Asia                                  319
GoAir                                     194
Multiple carriers Premium economy          13
Jet Airways Business                        6
Vistara Premium economy                     3
Trujet                                      1
Name: Airline, dtype: int64
```

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

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

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

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

```
data_train["Route"].value_counts()
```

```
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
```

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

```
1 stop      5625
non-stop    3491
```

```
2 stops     1520
3 stops       45
4 stops        1
Name: Total_Stops, dtype: int64
```

Dropping some columns

data_train =data_train.drop(['Additional_Info'],axis =1)
data_train =data_train.drop(['Route'],axis =1)

data_train.head()

	Airline	Date_of_Jouey	Source	Destination	Dep_Time	Arrival_Time	Duration	Total_Stops	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	22:20	01:10 22 Mar	2h 50m	non-stop	3897
1	Air India	1/05/2019	Kolkata	Banglore	05:50	13:15	7h 25m	2 stops	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	09:25	04:25 10 Jun	19h	2 stops	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	18:05	23:30	5h 25m	1 stop	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	16:50	21:35	4h 45m	1 stop	13302

Importing Date time function

Further we are going to clean the data

import datetime

data_train['Day_of_journey'] =pd.to_datetime(data_train['Date_of_Journey'], format ="%d/%m/%Y").dt.day

data_train['Month_of_journey'] =pd.to_datetime(data_train['Date_of_Journey'], format ="%d/%m/%Y").dt.month

data_train.head()

	Airline	Date_of_Journey	Source	Destination	Dep_Time	Arrival_Time	Duration	Total_Stops	Price	Day_of_journey	Month_of_journey
0	IndiGo	24/03/2019	Banglore	New Delhi	22:20	01:10 22 Mar	2h 50m	non-stop	3897	24	3
1	Air India	1/05/2019	Kolkata	Banglore	05:50	13:15	7h 25m	2 stops	7662	1	5
2	Jet Airways	9/06/2019	Delhi	Cochin	09:25	04:25 10 Jun	19h	2 stops	13882	9	6
3	IndiGo	12/05/2019	Kolkata	Banglore	18:05	23:30	5h 25m	1 stop	6218	12	5
4	IndiGo	01/03/2019	Banglore	New Delhi	16:50	21:35	4h 45m	1 stop	13302	1	3

So with the help of the DateTime function here I have separated the day and month from the Date of Journey column and made a new column for both Day of journey, Month of Journey

As I have extracted the data from the Date of journey column I am going to drop that column with the help of drop function

```
data_train =data_train.drop(['Date_of_Journey'],axis =1)
data_train.head()
```

Same extracted both minute and hour from the Dep_time column and made two new column containg
Dep_min and Dep_hour

```
data_train['Dep_min'] =pd.to_datetime(data_train['Dep_Time']).dt.minute
data_train['Dep_hour'] =pd.to_datetime(data_train['Dep_Time']).dt.hour
```

Drop the column Dep_time

```
data_train =data_train.drop(['Dep_Time'],axis =1)
data_train.head()
```

	Airline	Source	Destination	Arrival_Time	Duration	Total_Stops	Price	Day_of_journey	Month_of_journey	Dep_min	Dep_hour
0	IndiGo	Banglore	New Delhi	01:10 22 Mar	2h 50m	non-stop	3897	24	3	20	22
1	Air India	Kolkata	Banglore	13:15	7h 25m	2 stops	7662	1	5	50	5
2	Jet Airways	Delhi	Cochin	04:25 10 Jun	19h	2 stops	13882	9	6	25	9
3	IndiGo	Kolkata	Banglore	23:30	5h 25m	1 stop	6218	12	5	5	18
4	IndiGo	Banglore	New Delhi	21:35	4h 45m	1 stop	13302	1	3	50	16

Separate both arrival min and arrival hour from the Arrival-time column

```
data_train['Arrival_min'] =pd.to_datetime(data_train['Arrival_Time']).dt.minute
data_train['Arrival_hour'] =pd.to_datetime(data_train['Arrival_Time']).dt.hour
```

```
data_train =data_train.drop(['Arrival_Time'],axis =1)
data_train.head()
```

	Airline	Source	Destination	Duration	Total_Stops	Price	Day_of_journey	Month_of_journey	Dep_min	Dep_hour	Arrival_min	Arrival_hour
0	IndiGo	Banglore	New Delhi	2h 50m	non-stop	3897	24	3	20	22	10	1
1	Air India	Kolkata	Banglore	7h 25m	2 stops	7662	1	5	50	5	15	13
2	Jet Airways	Delhi	Cochin	19h	2 stops	13882	9	6	25	9	25	4

	Airline	Source	Destination	Duration	Total_Stops	Price	Day_of_journey	Month_of_journey	Dep_min	Dep_hour	Arrival_min	Arrival_hour
3	IndiGo	Kolkata	Banglore	5h 25m	1 stop	6218	12	5	5	18	30	23
4	IndiGo	Banglore	New Delhi	4h 45m	1 stop	13302	1	3	50	16	35	21

For the duration I have defined a function

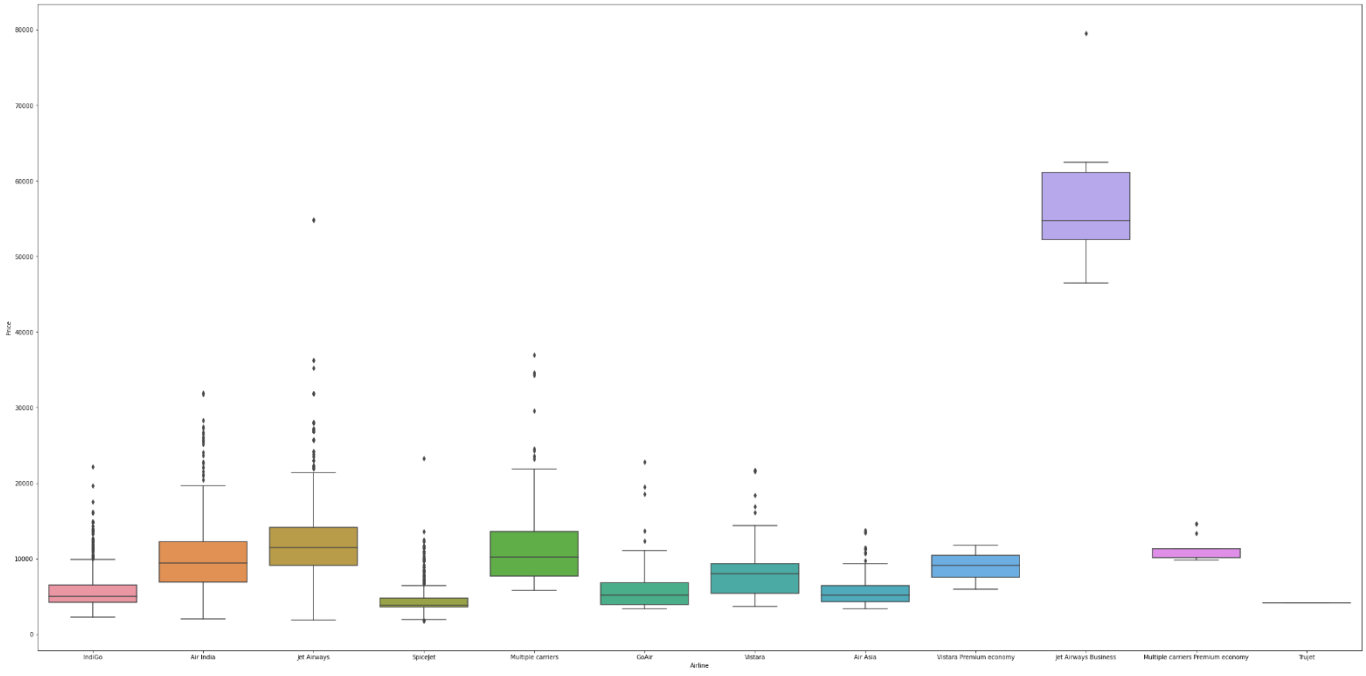
```
duration = list(data_train["Duration"])

for i in range(len(duration)):
   if len(duration[i].split()) != 2:
      if "h" in duration[i]:
         duration[i] = duration[i].strip() + " 0m"   # 0m = 0 minute
      else:
         duration[i] = "0h " + duration[i]          # 0h = 0 hour

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

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

data_train =data_train.drop(['Duration'],axis =1)
data_train.head()
```

	Airline	Source	Destination	Total_Stops	Price	Day_of_journey	Month_of_journey	Dep_min	Dep_hour	Arrival_min	Arrival_hour	Duration_hours	Duration_mins
0	IndiGo	Banglore	New Delhi	non-stop	3897	24	3	20	22	10	1	2	50
1	Air India	Kolkata	Banglore	2 stops	7662	1	5	50	5	15	13	7	25
2	Jet Air	Delhi	Cochin	2 stops	13882	9	6	25	9	25	4	19	0

	Airline	Source	Destination	Total_Stops	Price	Day_of_journey	Month_of_journey	Dep_min	Dep_hour	Arrival_min	Arrival_hour	Duration_hours	Duration_mins
	ways												
3	IndiGo	Kolkata	Bangalore	1 stop	6218	12	5	5	18	30	23	5	25
4	IndiGo	Banglore	New Delhi	1 stop	13302	1	3	50	16	35	21	4	45

Using boxplot method for better visualization

```
plt.figure(figsize=(40,20))
sns.boxplot( x ="Airline" , y="Price" ,data =data_train)
plt.show()
```

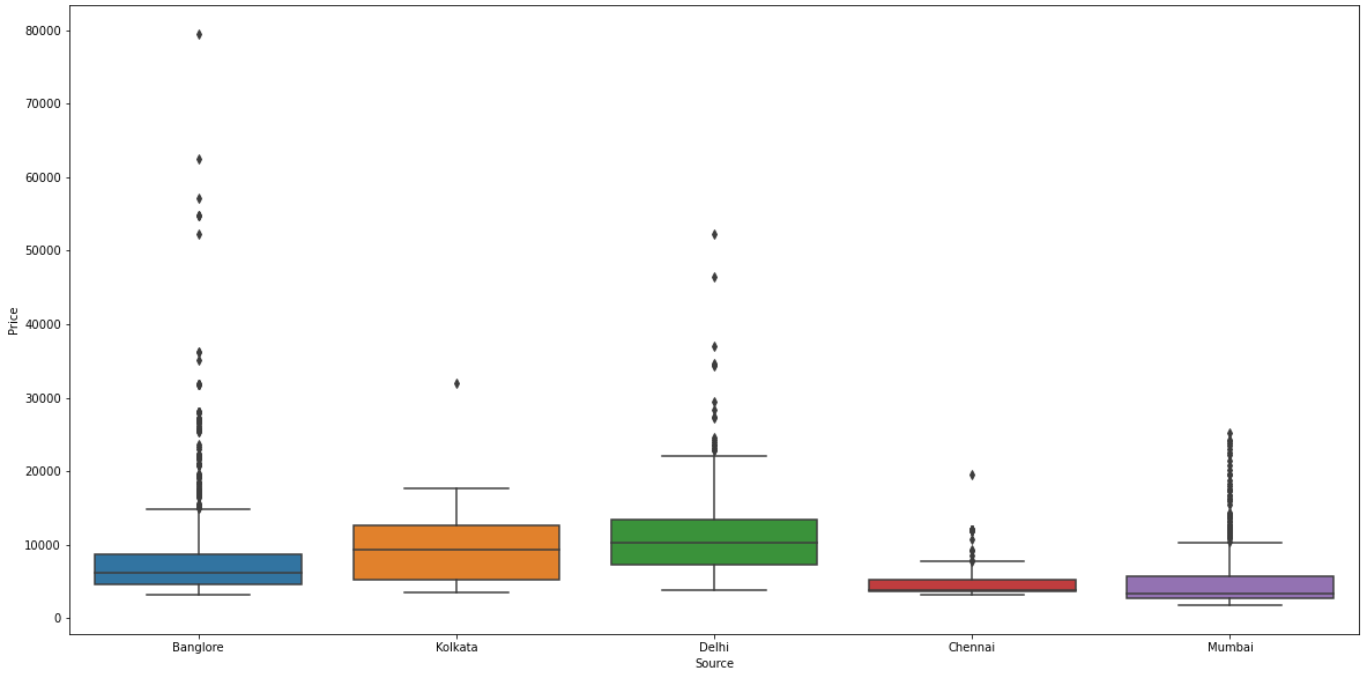


Making dummies for further model making

```
Airline = data_train[["Airline"]]
```

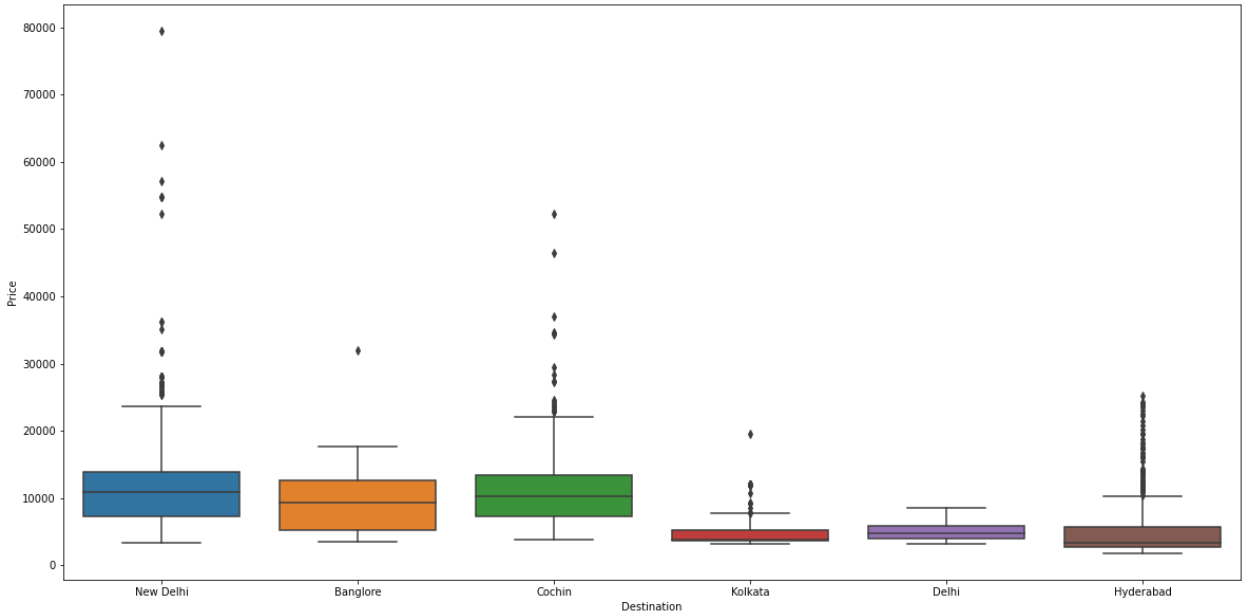
```
Airline = pd.get_dummies(Airline, drop_first= True)
```

```
plt.figure(figsize=(20,10))
sns.boxplot( x ="Source" , y="Price" ,data =data_train)
plt.show()
```



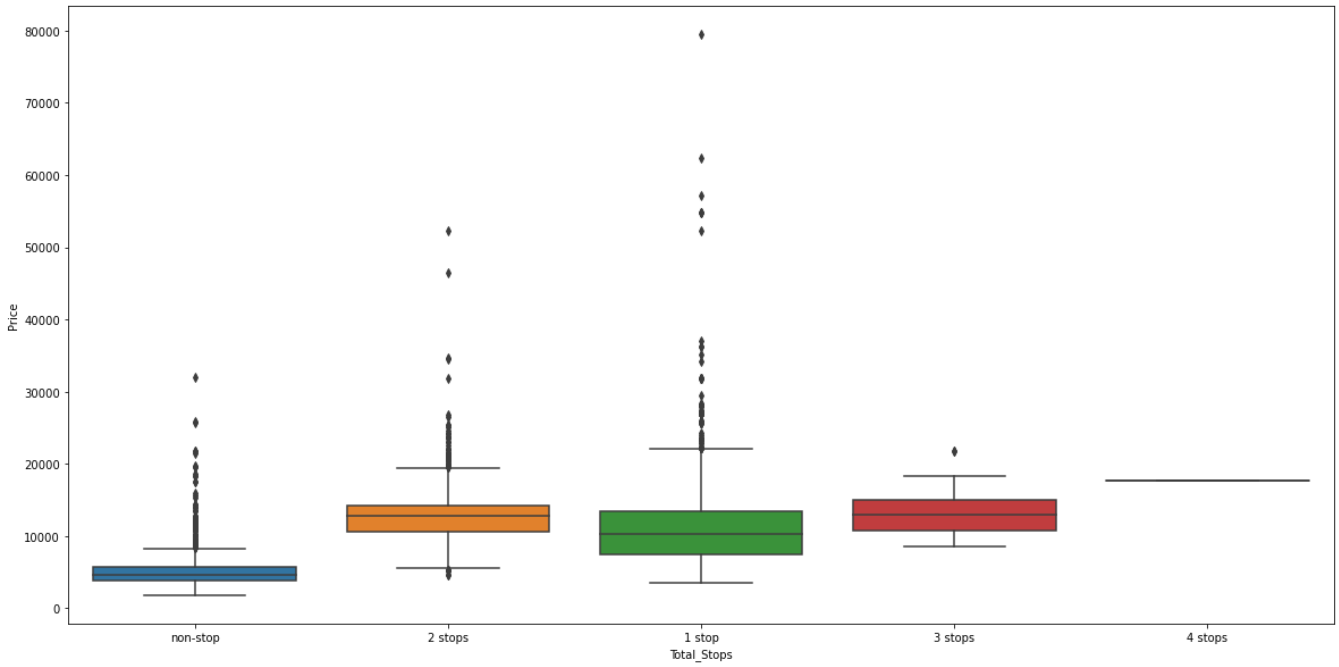

```
Source = data_train[["Source"]]
Source = pd.get_dummies(Source, drop_first= True)
```

```
plt.figure(figsize=(20,10))
sns.boxplot( x ="Destination" , y="Price" ,data =data_train)
plt.show()
```




```
Destination = data_train[["Destination"]]
Destination = pd.get_dummies(Destination, drop_first= True)
```

```
plt.figure(figsize=(20,10))
sns.boxplot( x ="Total_Stops" , y="Price" ,data =data_train)
plt.show()
```



Using replace method for model building

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

using Concat function to add up all the column

```
data_train1 =pd.concat([data_train,Airline,Source,Destination], axis=1)
```

we have these columns' data already so drooping the columns

```
data_train1.drop(["Airline","Source","Destination"] ,axis =1 ,inplace =True)
```

Using Histogram as we have cleaned all the data and for better visualization

As we have clean and visualize train data

Now further we are working on Test data I have followed same functions, methods for the cleaning of data as I have used above in train data

Dataset

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL → BOM → COK	17:30	04:25 07 Jun	10h 55m	1 stop	No info
1	IndiGo	12/05/2019	Kolkata	Banglore	CCU → MAA → BLR	06:20	10:20	4h	1 stop	No info
2	Jet Airways	21/05/2019	Delhi	Cochin	DEL → BOM → COK	19:15	19:00 22 May	23h 45m	1 stop	In-flight meal not included
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL → BOM → COK	08:00	21:00	13h	1 stop	No info
4	Air Asia	24/06/2019	Banglore	Delhi	BLR → DEL	23:55	02:45 25 Jun	2h 50m	non-stop	No info
5	Jet Airways	12/06/2019	Delhi	Cochin	DEL → BOM → COK	18:15	12:35 13 Jun	18h 20m	1 stop	In-flight meal not included
6	Air India	12/03/2019	Banglore	New Delhi	BLR → TRV → DEL	07:30	22:35	15h 5m	1 stop	No info
7	IndiGo	1/05/2019	Kolkata	Banglore	CCU → HYD → BLR	15:15	20:30	5h 15m	1 stop	No info
8	IndiGo	15/03/2019	Kolkata	Banglore	CCU → BLR	10:10	12:55	2h 45m	non-stop	No info
9	Jet Airways	18/05/2019	Kolkata	Banglore	CCU → BOM → BLR	16:30	22:35	6h 5m	1 stop	No info

Checking the shape of dataset using .shape function

data_test.shape

```
(2671, 10)
```

This dataset contains 2671 rows and 10 columns

Checking information in the dataset

data_test.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2671 entries, 0 to 2670
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          2671 non-null   object
 1   Date_of_Journey  2671 non-null   object
 2   Source           2671 non-null   object
 3   Destination      2671 non-null   object
 4   Route            2671 non-null   object
 5   Dep_Time         2671 non-null   object
 6   Arrival_Time     2671 non-null   object
 7   Duration         2671 non-null   object
 8   Total_Stops      2671 non-null   object
 9   Additional_Info  2671 non-null   object
dtypes: object(10)
memory usage: 208.8+ KB
```

Dropping Null values

```
data_test =data_test.dropna(axis =0)
data_test.isnull().sum()
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Additional_Info    0
dtype: int64
```

Drooped irrelevant columns from the dataset

```
data_test =data_test.drop(['Additional_Info'],axis =1)
data_test =data_test.drop(['Route'],axis =1)
data_test.head()
```

	Airline	Date_of_Journey	Source	Destination	Dep_Time	Arrival_Time	Duration	Total_Stops
0	Jet Airways	6/06/2019	Delhi	Cochin	17:30	04:25 07 Jun	10h 55m	1 stop
1	IndiGo	12/05/2019	Kolkata	Banglore	06:20	10:20	4h	1 stop
2	Jet Airways	21/05/2019	Delhi	Cochin	19:15	19:00 22 May	23h 45m	1 stop
3	Multiple carriers	21/05/2019	Delhi	Cochin	08:00	21:00	13h	1 stop
4	Air Asia	24/06/2019	Banglore	Delhi	23:55	02:45 25 Jun	2h 50m	non-stop

With the help of date time separating minutes hours

```
data_test['Day_of_journey'] =pd.to_datetime(data_test['Date_of_Journey'], format
="%d/%m/%Y").dt.day
data_test['Month_of_journey'] =pd.to_datetime(data_test['Date_of_Journey'], format
="%d/%m/%Y").dt.month
data_test =data_test.drop(['Date_of_Journey'],axis =1)
data_test.head()
```

	Airline	Source	Destination	Dep_Time	Arrival_Time	Duration	Total_Stops	Day_of_journey	Month_of_journey
0	Jet Airways	Delhi	Cochin	17:30	04:25 07 Jun	10h 55m	1 stop	6	6
1	IndiGo	Kolkata	Banglore	06:20	10:20	4h	1 stop	12	5
2	Jet Airways	Delhi	Cochin	19:15	19:00 22 May	23h 45m	1 stop	21	5
3	Multiple carriers	Delhi	Cochin	08:00	21:00	13h	1 stop	21	5
4	Air Asia	Banglore	Delhi	23:55	02:45 25 Jun	2h 50m	non-stop	24	6

Same for other columns

```
data_test['Dep_min'] =pd.to_datetime(data_test['Dep_Time']).dt.minute
data_test['Dep_hour'] =pd.to_datetime(data_test['Dep_Time']).dt.hour
data_test =data_test.drop(['Dep_Time'],axis =1)
```

```
data_test['Arrival_min'] =pd.to_datetime(data_test['Arrival_Time']).dt.minute
data_test['Arrival_hour'] =pd.to_datetime(data_test['Arrival_Time']).dt.hour
data_test =data_test.drop(['Arrival_Time'],axis =1)
```

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

for i in range(len(duration)):
    if len(duration[i].split()) != 2:
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"   # 0m = 0 minute
        else:
            duration[i] = "0h " + duration[i]          # 0h = 0 hour

duration_hours = []
duration_mins = []
for i in range(len(duration)):
```

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


data_test["Duration_hours"] = duration_hours
data_test["Duration_mins"] = duration_mins
data_test =data_test.drop(['Duration'],axis =1)
```

Creating dummies

```
Airline = data_test[["Airline"]]
Airline = pd.get_dummies(Airline, drop_first= True)

Source = data_test[["Source"]]
Source = pd.get_dummies(Source, drop_first= True)

Destination = data_test[["Destination"]]
Destination = pd.get_dummies(Destination, drop_first= True)
```

Replacing the values

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

	Airline	Source	Destination	Total_Stops	Day_of_journey	Month_of_journey	Dep_min	Dep_hour	Arrival_min	Arrival_hour	Duration_hours	Duration_mins
0	Jet Airways	Delhi	Cochin	1	6	6	30	17	25	4	10	55
1	IndiGo	Kolkata	Banglore	1	12	5	20	6	20	10	4	0
2	Jet Airways	Delhi	Cochin	1	21	5	15	19	0	19	23	45
3	Multiple carriers	Delhi	Cochin	1	21	5	0	8	0	21	13	0

	Airline	Source	Destination	Total_Stops	Day_of_journey	Month_of_journey	Dep_min	Dep_hour	Arrival_min	Arrival_hour	Duration_hours	Duration_mins
4	Air Asia	Banglore	Delhi	0	24	6	55	23	45	2	2	50

Adding the columns using concatenation function

```
data_test1 =pd.concat([data_test,Airline,Source,Destination], axis=1)
```

we have these columns data already so drooping the columns

```
data_test1.drop(["Airline","Source","Destination"] ,axis =1 ,inplace =True)
```

Checking the size of the dataset

```
data_test1.shape
```

```
(2671, 28)
```

Plotting histogram

```
data_test1.hist(color ="pink" , figsize =(20,20))
plt.show()
```

Building model

Separating features and label

x =data_train1.drop(['Price'], axis =1)
y =data_train1['Price']

Using Boxplot for better visualization

plt.figure(figsize=(50,50), facecolor='pink')

```
graph=1

for column in x:
    if graph<=30:
        ax=plt.subplot(10,3,graph)
        sns.boxplot(x[column])
        plt.xlabel(column,fontsize=20)


    graph+=1
plt.show()
```

Using Standard Scaler for scaling
And Fitting and transform the scaled data

```
sc = StandardScaler()
sc.fit_transform(x)
```

Using train test split for further model building

```
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.2 ,random_state=5)
```

using linear regression for predictions

```
lr = LinearRegression()
lr.fit(x_train,y_train)
```

```
lr = LinearRegression()
lr.fit(x_train,y_train)
array([11231.49084379,  4547.90828167, 11436.03552346, ...,
        5338.02095875,  3888.54931379,  4108.6131531 ])
```

```
print(r2_score(y_pred,y_test))
0.35664214783409176
```

Used different model for better prediction

Using Random forest classifier

```
# Random Forest Classifier
rf =RandomForestClassifier()
rf.fit(x_train,y_train)
```

```
y_predict =rf.predict(x_test)
y_predict
array([14781,  4483,  6795, ...,  5224,  3100,  3841], dtype=int64)
```

```
rf.score(x_train,y_train)
0.8821533060269163
```

```
rf.score(x_test,y_test)
0.33879270004679457
```

Calculating Metrics

```
MAE: 1485.3093121197942
MSE  7093620.016846046
RMSE: 2663.3850673242964


metrics.r2_score(y_test,y_predict)
0.6884453949370779


Using Decision Tree Classifier

# Decision Tree Classifier
dtc =DecisionTreeClassifier()
dtc.fit(x_train,y_train)

y_prediction =dtc.predict(x_test)
y_prediction
array([14781,  3383,  6795, ...,  5224,  3100,  3841], dtype=int64)


metrics.r2_score(y_test,y_prediction)
0.6535169589522694
```

Conclusion

In this article, we saw how to apply Different libraries to choose the best machine learning algorithm for the task at hand.

We analyzed the dataset and then find the null values, information regarding the dataset removed all the Null values and then used some methods to clean the data. and build a machine learning model further.

They used a standard scaler for the scaling and tried different machine learning models for better prediction or result. One can choose either based on the situation at hand.

thanku