

for (int i = 0; i < arr.length; i++)

arr[i] = arr[i] * 2; // arr[i] is the current element, arr[i] * 2 is the new value

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

for (int i = 0; i < arr.length; i++) arr[i] = arr[i] * 2; // arr[i] is the current element, arr[i] * 2 is the new value

for (int i = 0; i < arr.length; i++)

arr[i] = arr[i] * 2; // arr[i] is the current element, arr[i] * 2 is the new value

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

```
from sklearn.preprocessing import LabelEncoder  
le=LabelEncoder()
```

```
data.Airline=le.fit_transform(data.Airline)  
data.Source=le.fit_transform(data.Source)  
data.Destination=le.fit_transform(data.Destination)  
data.Total_Stops=le.fit_transform(data.Total_Stops)  
data.City1=le.fit_transform(data.City1)  
data.City2=le.fit_transform(data.City2)  
data.City3=le.fit_transform(data.City3)  
data.Additional_Info=le.fit_transform(data.Additional_Info)  
data.head()
```

	Airline	Source	Destination	Total_Stops	Additional_Info	Price	City1	City2	City3
0	3	0	5	4	7	3897	0	13	29
1	1	3	0	1	7	7662	2	25	1
2	4	2	1	1	7	13882	3	32	4
3	3	3	0	0	7	6218	2	34	3
4	3	0	5	0	7	13302	0	34	8

```
from sklearn.preprocessing import LabelEncoder  
le=LabelEncoder()
```

```
data.Airline=le.fit_transform(data.Airline)  
data.Source=le.fit_transform(data.Source)  
data.Destination=le.fit_transform(data.Destination)  
data.Total_Stops=le.fit_transform(data.Total_Stops)  
data.City1=le.fit_transform(data.City1)  
data.City2=le.fit_transform(data.City2)  
data.City3=le.fit_transform(data.City3)  
data.Additional_Info=le.fit_transform(data.Additional_Info)  
data.head()
```

Airline	Source	Destination	Total_Stops	Additional_Info	Price	City1	City2	City3	Dat
---------	--------	-------------	-------------	-----------------	-------	-------	-------	-------	-----

```
data[data['Travel_Hours'] == '5m']
```

Price	City1	City2	City3	Date	Month	Year	Dep_Time_Hour	Dep_Time_Mins	Arrival
17327	BOM	GOI	PNQ	6	3	2019	16	50	

INFO4110EX: 10082 flights, 0 to 10082

Data columns (total 19 columns):

#	Column	Non-Null Count	Dtype
0	Airline	10682 non-null	object
1	Source	10682 non-null	object
2	Destination	10682 non-null	object
3	Total_Stops	10682 non-null	object
4	Additional_Info	10682 non-null	object
5	Price	10682 non-null	int64
6	City1	10682 non-null	object
7	City2	10682 non-null	object
8	City3	10682 non-null	object
9	Date	10682 non-null	object
10	Month	10682 non-null	object
11	Year	10682 non-null	object
12	Dep_Time_Hour	10682 non-null	object
13	Dep_Time_Mins	10682 non-null	object
14	Arrival_date	10682 non-null	object
15	Arrival_Time_Hour	10682 non-null	object
16	Arrival_Time_Mins	10682 non-null	object
17	Travel_Hours	10682 non-null	object
18	Travel_Mins	10682 non-null	object

dtypes: int64(1), object(18)

memory usage: 1.6+ MB

Hence, we try to change the datatype of the required columns

```
#changing the numerical columns from object to int
#data.Total_Stops=data.Total_Stops.astype('int64')
data.Date=data.Date.astype('int64')
data.Month=data.Month.astype('int64')
data.Year=data.Year.astype('int64')
data.Dep_Time_Hour=data.Dep_Time_Hour.astype('int64')
data.Dep_Time_Hour=data.Dep_Time_Hour.astype('int64')
data.Dep_Time_Mins=data.Dep_Time_Mins.astype('int64')
data.Arrival_date=data.Arrival_date.astype('int64')
data.Arrival_Time_Hour=data.Arrival_Time_Hour.astype('int64')
data.Arrival_Time_Mins=data.Arrival_Time_Mins.astype('int64')
#data.Travel_Hours=data.Travel_Hours.astype('int64')
data.Travel_Mins=data.Travel_Mins.astype('int64')
```

```
#filling City3 as None, the missing values are less  
data['City3'].fillna('None',inplace=True)
```

```
#filling Arrival_Date as Departure_Date  
data['Arrival_date'].fillna(data['Date'],inplace=True)
```

Replacing Missing Values

We further replace 'NaN' values in 'City3' with 'None', since rows where 'City3' is missing did not have any stop, just the source and the destination.

We also replace missing values in 'Arrival_date' column with values in 'Date' column, since the missing values are those values where the flight took off and landed on

```
#Checking Null Values
```

```
data.isnull().sum()
```

```
Airline      0
```

```
Source       0
```

```
Destination  0
```

```
Total_Stops 0
```

```
Additional Info 0
```



```
data.isnull().sum()
```

```
Airline      0
Date_of_Journey  0
Source       0
Destination  0
Route        0
Dep_Time     0
Arrival_Time 0
Duration     0
Total_Stops  0
Additional_Info 0
Price        0
City1        0
City2        0
City3        3491
City4        9116
City5       10636
City6       10681
Date         0
Month        0
Year         0
Dep_Time_Hour 0
Dep_Time_Mins 0
Arrival_date 6348
Time_of_Arrival 0
Arrival_Time_Hour 0
Arrival_Time_Mins 0
Travel_Hours 0
Travel_Mins  1032
dtype: int64
```

#We also drop some columns like 'city6' and 'city5', since majority of the
`data.drop(['City4','City5','City6'],axis=1,inplace=True)`

```
data.drop(['Date_of_Journey','Route','Dep_Time','Arrival_Time','Duration'])
data.drop(['Time_of_Arrival'],axis=1,inplace=True)
```

```
data.isnull().sum()
```

Airline	0
Date_of_Journey	0
Source	0
Destination	0
Route	0
Non Time	0

```
data.Additional_Info.unique()
```

```
array(['No info', 'In-flight meal not included',  
      'No check-in baggage included', '1 Short layover', 'No Info',  
      '1 Long layover', 'Change airports', 'Business class',  
      'Red eye flight', '2 Long layovers'], dtype=object)
```

#Next, we divide the 'Duration' column to 'Travel_hours' and 'Travel_mi

```
data.Duration=data.Duration.str.split(' ')
```

```
data['Travel_Hours']=data.Duration.str[0]  
data['Travel_Hours']=data['Travel_Hours'].str.split('h')  
data['Travel_Hours']=data['Travel_Hours'].str[0]  
data.Travel_Hours=data.Travel_Hours  
data['Travel_Mins']=data.Duration.str[1]  
  
data.Travel_Mins=data.Travel_Mins.str.split('m')  
data.Travel_Mins=data.Travel_Mins.str[0]
```

#Next, we divide the 'Duration' column to 'Travel_hours' and 'Travel_m

```
data.Duration=data.Duration.str.split(' ')
```

```
data['Travel_Hours']=data.Duration.str[0]  
data['Travel_Hours']=data['Travel_Hours'].str.split('h')  
data['Travel_Hours']=data['Travel_Hours'].str[0]  
data.Travel_Hours=data.Travel_Hours  
data['Travel_Mins']=data.Duration.str[1]  
  
data.Travel_Mins=data.Travel_Mins.str.split('m')  
data.Travel_Mins=data.Travel_Mins.str[0]
```

#In the similar manner, we split the Dep_time column, and create separate

```
data.Dep_Time=data.Dep_Time.str.split(':')
```

```
data['Dep_Time_Hour']=data.Dep_Time.str[0]  
data['Dep_Time_Mins']=data.Dep_Time.str[1]
```

```
#We now split the Date column to extract the 'Date', 'Month' and 'Year'
```

```
data.Date_of_Journey=data.Date_of_Journey.str.split('/')  

```

```
data.Date_of_Journey
```

```
0      [24, 03, 2019]
```

```
1      [1, 05, 2019]
```

```
2      [9, 06, 2019]
```

```
3     [12, 05, 2019]
```

```
4     [01, 03, 2019]
```

```
...
```

```
10678    [9, 04, 2019]
```

```
10679   [27, 04, 2019]
```

```
10680   [27, 04, 2019]
```

```
10681   [01, 03, 2019]
```

```
10682    [9, 05, 2019]
```

```
Name: Date_of_Journey, Length: 10682, dtype: object
```

```
#Treating the data_column
```

```
data['Date']=data.Date_of_Journey.str[0]
```

```
data['Month']=data.Date_of_Journey.str[1]
```

```
data['Year']=data.Date_of_Journey.str[2]
```

```
for i in category:  
    print(i, data[i].unique())
```

Airline ['IndiGo' 'Air India' 'Jet Airways' 'Sp
'Vistara' 'Air Asia' 'Vistara Premium economy'
'Multiple carriers Premium economy' 'Trujet']
Source ['Bangalore' 'Kolkata' 'Delhi' 'Chennai'
Destination ['New Delhi' 'Bangalore' 'Cochin' 'M
Additional_Info ['No info' 'In-flight meal not
'1 Short layover' 'No Info' '1 Long layover'
'Business class' 'Red-eye flight' '2 Long layo

Data Preparation

As we have understood how the data is let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling outliers
- Scaling Techniques
- Splitting dataset into training and test set


```
data=pd.read_csv("Data_Train.csv")
```

```
data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Tim
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:2
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:5
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:2
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:0
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:5

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
import warnings
import pickle
from scipy import stats
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')
```

Collect The Dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link:

<https://www.kaggle.com/code/a/price-prediction/data>

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a number