Problem Statement

Which model is suitable for Dataset

Importing All The Required Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

STEP-1: Data Collection

In [2]:

```
train_df=pd.read_csv(r"C:\Users\thara\Downloads\Data_Train-Flight.csv")
train_df
```

Out[2]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Dura
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h
...	...	...	...	...	...	...	...	
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU ? BLR	19:55	22:25	2h
10679	Air India	27/04/2019	Kolkata	Banglore	CCU ? BLR	20:45	23:20	2h
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR ? DEL	08:20	11:20	
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR ? DEL	11:30	14:10	2h
10682	Air India	9/05/2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h

10683 rows × 11 columns

In [3]:

```
test_df=pd.read_csv(r"C:\Users\thara\Downloads\Test_set-Flight.csv")
test_df
```

Out[3]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL ? BOM ? COK	17:30	04:25 07 Jun	10h 5
1	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? MAA ? BLR	06:20	10:20	
2	Jet Airways	21/05/2019	Delhi	Cochin	DEL ? BOM ? COK	19:15	19:00 22 May	23h 4
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL ? BOM ? COK	08:00	21:00	
4	Air Asia	24/06/2019	Banglore	Delhi	BLR ? DEL	23:55	02:45 25 Jun	2h 5
...	...	...	...	...	...	...	...	
2666	Air India	6/06/2019	Kolkata	Banglore	CCU ? DEL ? BLR	20:30	20:25 07 Jun	23h 5
2667	IndiGo	27/03/2019	Kolkata	Banglore	CCU ? BLR	14:20	16:55	2h 3
2668	Jet Airways	6/03/2019	Delhi	Cochin	DEL ? BOM ? COK	21:50	04:25 07 Mar	6h 3
2669	Air India	6/03/2019	Delhi	Cochin	DEL ? BOM ? COK	04:00	19:15	15h 1
2670	Multiple carriers	15/06/2019	Delhi	Cochin	DEL ? BOM ? COK	04:55	19:15	14h 2

2671 rows × 10 columns

STEP-2: Data Preprocessing And Cleaning

In [4]:

```
# The shape of a DataFrame is a tuple of array dimensions that tells the number of rows

train_df.shape
```

Out[4]:

```
(10683, 11)
```

In [5]:

```
test_df.shape
```

Out[5]:

```
(2671, 10)
```

In [6]:

```
# The head() returns the first n rows for the object based on position.

train_df.head()
```

Out[6]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m

In [7]:

```
test_df.head()
```

Out[7]:

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

In [8]:

```
# The tail function in Python displays the last five rows of the dataframe by default.

train_df.tail()
```

Out[8]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Dura
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU ? BLR	19:55	22:25	2h
10679	Air India	27/04/2019	Kolkata	Banglore	CCU ? BLR	20:45	23:20	2h
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR ? DEL	08:20	11:20	
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR ? DEL	11:30	14:10	2h
10682	Air India	9/05/2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h

In [9]:

```
test_df.tail()
```

Out[9]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duratic
2666	Air India	6/06/2019	Kolkata	Banglore	CCU ? DEL ? BLR	20:30	20:25 07 Jun	23h 55
2667	IndiGo	27/03/2019	Kolkata	Banglore	CCU ? BLR	14:20	16:55	2h 35
2668	Jet Airways	6/03/2019	Delhi	Cochin	DEL ? BOM ? COK	21:50	04:25 07 Mar	6h 35
2669	Air India	6/03/2019	Delhi	Cochin	DEL ? BOM ? COK	04:00	19:15	15h 15
2670	Multiple carriers	15/06/2019	Delhi	Cochin	DEL ? BOM ? COK	04:55	19:15	14h 20

In [10]:

```
train_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 [11]:

```
test_df.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
```

In [12]:

```
# Explore numerical columns stats.
# The describe function only displays the numerical columns present in the Dataset.

train_df.describe()
```

Out[12]:

	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 [13]:

```
train_df.Price.describe()
```

Out[13]:

```
count    10683.000000
mean      9087.064121
std       4611.359167
min       1759.000000
25%       5277.000000
50%       8372.000000
75%      12373.000000
max      79512.000000
Name: Price, dtype: float64
```

In [14]:

```
test_df.describe()
```

Out[14]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Dura
count	2671	2671	2671	2671	2671	2671	2671	2
unique	11	44	5	6	100	199	704	
top	Jet Airways	9/05/2019	Delhi	Cochin	DEL ? BOM ? COK	10:00	19:00	2h
freq	897	144	1145	1145	624	62	113	

In [15]:

```
test_df.Dep_Time.describe()
```

Out[15]:

```
count      2671
unique      199
top       10:00
freq         62
Name: Dep_Time, dtype: object
```

In [16]:

```
# If an integer or string or any items in a list are repeated more than one time, they a

train_df.duplicated().sum()
```

Out[16]:

```
220
```

In [17]:

```
# To remove the duplicates values.

train_df=train_df.drop_duplicates()
train_df
```

Out[17]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Dura
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h
...	...	...	...	...	...	...	...	
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU ? BLR	19:55	22:25	2h
10679	Air India	27/04/2019	Kolkata	Banglore	CCU ? BLR	20:45	23:20	2h
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR ? DEL	08:20	11:20	
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR ? DEL	11:30	14:10	2h
10682	Air India	9/05/2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h

10463 rows × 11 columns

In [18]:

```
test_df.duplicated().sum()
```

Out[18]:

26

In [19]:

```
test_df=test_df.drop_duplicates()
test_df
```

Out[19]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL ? BOM ? COK	17:30	04:25 07 Jun	10h 5
1	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? MAA ? BLR	06:20	10:20	
2	Jet Airways	21/05/2019	Delhi	Cochin	DEL ? BOM ? COK	19:15	19:00 22 May	23h 4
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL ? BOM ? COK	08:00	21:00	
4	Air Asia	24/06/2019	Banglore	Delhi	BLR ? DEL	23:55	02:45 25 Jun	2h 5
...	...	...	...	...	...	...	...	
2666	Air India	6/06/2019	Kolkata	Banglore	CCU ? DEL ? BLR	20:30	20:25 07 Jun	23h 5
2667	IndiGo	27/03/2019	Kolkata	Banglore	CCU ? BLR	14:20	16:55	2h 3
2668	Jet Airways	6/03/2019	Delhi	Cochin	DEL ? BOM ? COK	21:50	04:25 07 Mar	6h 3
2669	Air India	6/03/2019	Delhi	Cochin	DEL ? BOM ? COK	04:00	19:15	15h 1
2670	Multiple carriers	15/06/2019	Delhi	Cochin	DEL ? BOM ? COK	04:55	19:15	14h 2

2645 rows × 10 columns

In [20]:

```
# To check the null values.

train_df.isnull().sum()
```

Out[20]:

```
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
```

Checking whether there are any null values in the dataset

In [21]:

```
train_df.isnull().sum()
```

Out[21]:

```
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 [22]:

```
test_df.isnull().sum()
```

Out[22]:

```
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
```

Removing Null Values from the dataset

In [23]:

```
train_df.dropna(inplace=True)
```

```
C:\Users\thara\AppData\Local\Temp\ipykernel_28160\519058362.py:1: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  train_df.dropna(inplace=True)
```

In [24]:

```
train_df.isnull().sum()
```

Out[24]:

```
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
```

Conversion of datatype of values from String to Numerical Values

In [25]:

```
train_df['Airline'].value_counts()
```

Out[25]:

```
Airline
Jet Airways                          3700
IndiGo                               2043
Air India                            1694
Multiple carriers                    1196
SpiceJet                              815
Vistara                               478
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: count, dtype: int64
```

In [26]:

```
train_df['Source'].value_counts()
```

Out[26]:

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

In [27]:

```
train_df['Destination'].value_counts()
```

Out[27]:

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

In [28]:

```
train_df['Total_Stops'].value_counts()
```

Out[28]:

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

In [29]:

```
airline={"Airline":{"Jet Airways":0,"IndiGo":1,"Air India":2,"Multiple carriers":3,"Spic
train_df=train_df.replace(airline)
train_df
```

Out[29]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	1	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 5
1	2	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 2
2	0	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	
3	1	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 2
4	1	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 4
...	...	...	...	...	...	...	...	
10678	6	9/04/2019	Kolkata	Banglore	CCU ? BLR	19:55	22:25	2h 3
10679	2	27/04/2019	Kolkata	Banglore	CCU ? BLR	20:45	23:20	2h 3
10680	0	27/04/2019	Banglore	Delhi	BLR ? DEL	08:20	11:20	
10681	5	01/03/2019	Banglore	New Delhi	BLR ? DEL	11:30	14:10	2h 4
10682	2	9/05/2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 2

10462 rows × 11 columns

In [30]:

```
city={"Source":{"Delhi":0,"Kolkata":1,"Banglore":2,"Mumbai":3,"Chennai":4}}
train_df=train_df.replace(city)
train_df
```

Out[30]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duratio
0	1	24/03/2019	2	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50
1	2	1/05/2019	1	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25
2	0	9/06/2019	0	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19
3	1	12/05/2019	1	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25
4	1	01/03/2019	2	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45
...	...	...	...	...	...	...	...	
10678	6	9/04/2019	1	Banglore	CCU ? BLR	19:55	22:25	2h 30
10679	2	27/04/2019	1	Banglore	CCU ? BLR	20:45	23:20	2h 35
10680	0	27/04/2019	2	Delhi	BLR ? DEL	08:20	11:20	
10681	5	01/03/2019	2	New Delhi	BLR ? DEL	11:30	14:10	2h 40
10682	2	9/05/2019	0	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20

10462 rows × 11 columns

In [31]:

```
destination={"Destination":{"Cochin":0,"Banglore":1,"Delhi":2,"New Delhi":3,"Hyderabad":
train_df=train_df.replace(destination)
train_df
```

Out[31]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duratio
0	1	24/03/2019	2	3	BLR ? DEL	22:20	01:10 22 Mar	2h 50
1	2	1/05/2019	1	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25
2	0	9/06/2019	0	0	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19
3	1	12/05/2019	1	1	CCU ? NAG ? BLR	18:05	23:30	5h 25
4	1	01/03/2019	2	3	BLR ? NAG ? DEL	16:50	21:35	4h 45
...	...	...	...	...	...	...	...	
10678	6	9/04/2019	1	1	CCU ? BLR	19:55	22:25	2h 30
10679	2	27/04/2019	1	1	CCU ? BLR	20:45	23:20	2h 35
10680	0	27/04/2019	2	2	BLR ? DEL	08:20	11:20	
10681	5	01/03/2019	2	3	BLR ? DEL	11:30	14:10	2h 40
10682	2	9/05/2019	0	0	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20

10462 rows × 11 columns

In [32]:

```
total={"Total_Stops":{"non-stop":0,"1 stop":1,"2 stops":2,"3 stops":3,"4 stops":4}}
train_df=train_df.replace(total)
train_df
```

Out[32]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duratio
0	1	24/03/2019	2	3	BLR ? DEL	22:20	01:10 22 Mar	2h 50
1	2	1/05/2019	1	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25
2	0	9/06/2019	0	0	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19
3	1	12/05/2019	1	1	CCU ? NAG ? BLR	18:05	23:30	5h 25
4	1	01/03/2019	2	3	BLR ? NAG ? DEL	16:50	21:35	4h 45
...	...	...	...	...	...	...	...	
10678	6	9/04/2019	1	1	CCU ? BLR	19:55	22:25	2h 30
10679	2	27/04/2019	1	1	CCU ? BLR	20:45	23:20	2h 35
10680	0	27/04/2019	2	2	BLR ? DEL	08:20	11:20	
10681	5	01/03/2019	2	3	BLR ? DEL	11:30	14:10	2h 40
10682	2	9/05/2019	0	0	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20

10462 rows × 11 columns

In [33]:

```
train_df
```

Out[33]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duratio
0	1	24/03/2019	2	3	BLR ? DEL	22:20	01:10 22 Mar	2h 50
1	2	1/05/2019	1	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25
2	0	9/06/2019	0	0	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19
3	1	12/05/2019	1	1	CCU ? NAG ? BLR	18:05	23:30	5h 25
4	1	01/03/2019	2	3	BLR ? NAG ? DEL	16:50	21:35	4h 45
...	...	...	...	...	...	...	...	
10678	6	9/04/2019	1	1	CCU ? BLR	19:55	22:25	2h 30
10679	2	27/04/2019	1	1	CCU ? BLR	20:45	23:20	2h 35
10680	0	27/04/2019	2	2	BLR ? DEL	08:20	11:20	
10681	5	01/03/2019	2	3	BLR ? DEL	11:30	14:10	2h 40
10682	2	9/05/2019	0	0	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20

10462 rows × 11 columns

Feature Scaling :

To Split the data into training data and test data

In [34]:

```
x=train_df[['Airline','Source','Destination','Total_Stops']]
y=train_df['Price']
```

In [35]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=100)
```

STEP-3: Data Visualization

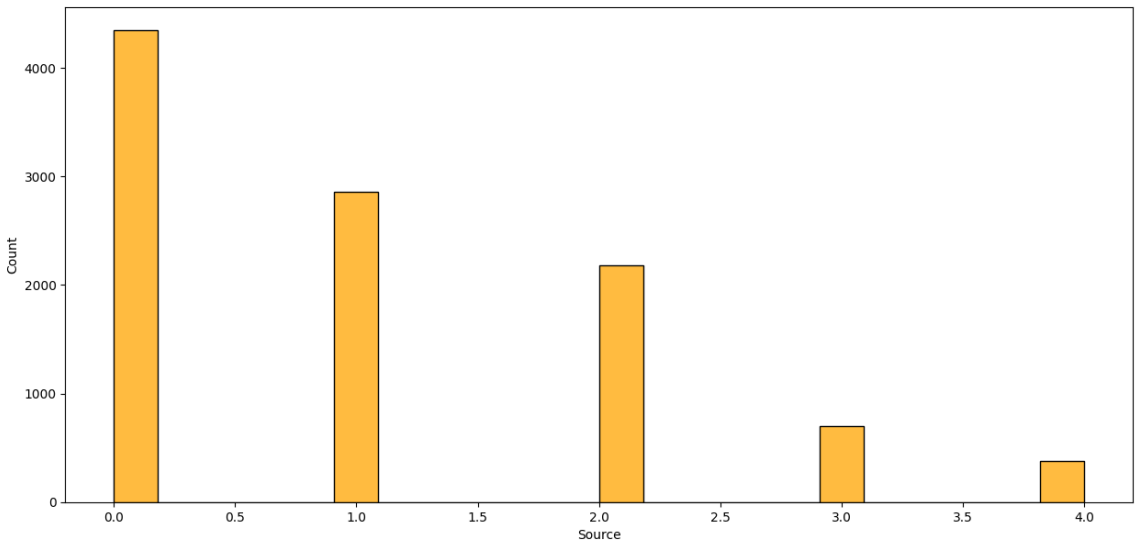
In [65]:

```
# Visualize our BMI column.
# A histogram is a graph showing frequency distributions.
# It is a graph showing the number of observations within each given interval.

plt.figure(figsize=(15,7))
sns.histplot(data=train_df, x='Source',color='orange')
```

Out[65]:

```
<Axes: xlabel='Source', ylabel='Count'>
```

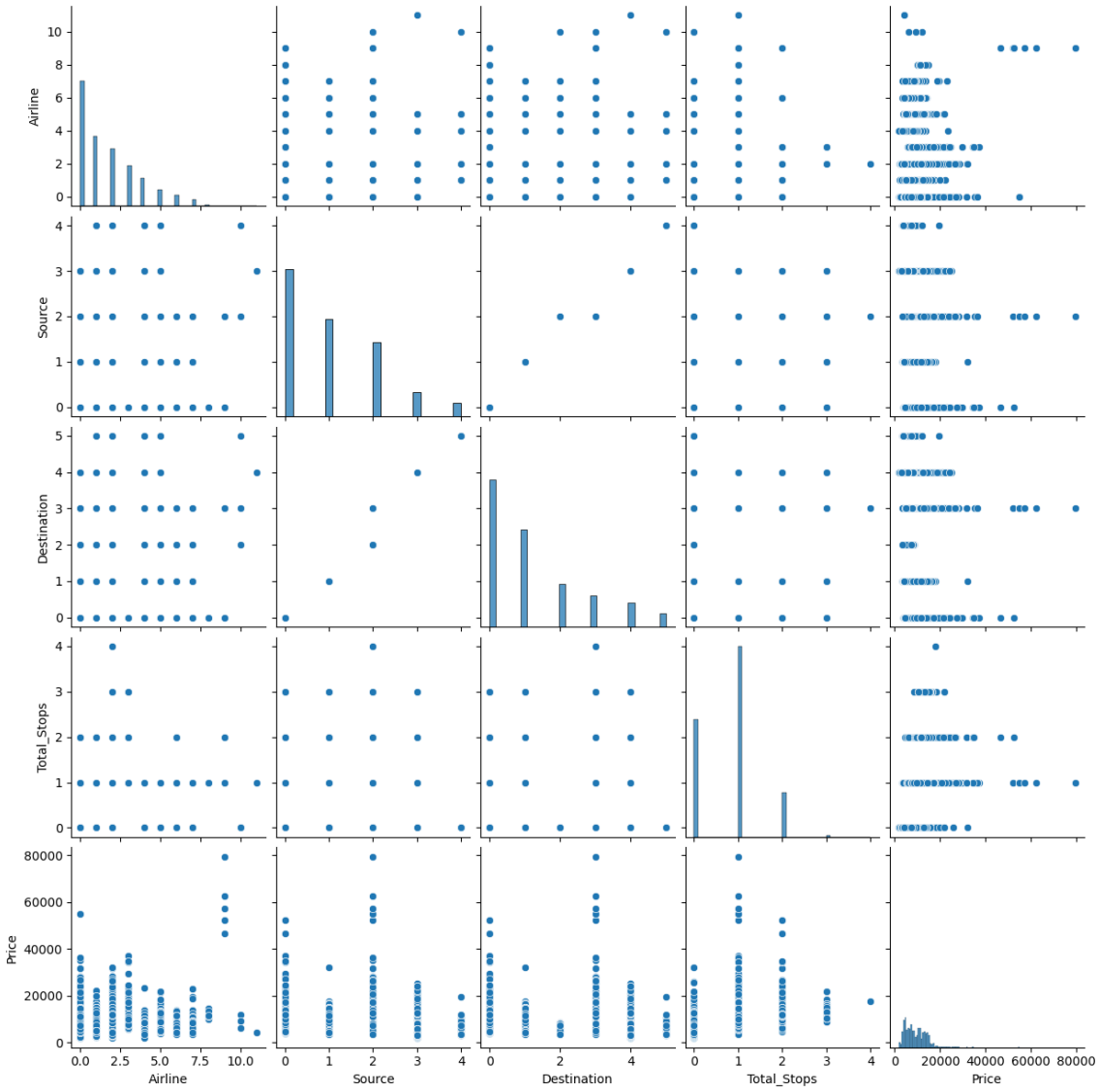


In [58]:

```
sns.pairplot(train_df)
```

Out[58]:

```
<seaborn.axisgrid.PairGrid at 0x2db3ddebc10>
```



In [59]:

```
df=train_df[['Airline','Source','Destination','Total_Stops','Price']]
sns.heatmap(df.corr(),annot=True)
```

Out[59]:

<Axes: >

Step-4:- Data Modelling

Applying LinearRegression

In [70]:

```
# Fit Model to the training set

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

Out[70]:

```
▾ LinearRegression
LinearRegression()
```

In [71]:

```
# Predict the test set result

y_pred = regressor.predict(x_test)
```

In [72]:

```
from sklearn.metrics import r2_score
score = r2_score(y_test, y_pred)
```

In [73]:

```
score
```

Out[73]:

```
0.3841668403360148
```

Applying LogisticRegression

In [64]:

```
x=np.array(df['Price']).reshape(-1,1)
y=np.array(df['Total_Stops']).reshape(-1,1)
df.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(max_iter=10000)
import warnings
warnings.simplefilter(action='ignore')
```

In [50]:

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

Out[50]:

```
LogisticRegression
LogisticRegression(max_iter=10000)
```

In [51]:

```
score=lr.score(x_test,y_test)
print(score)
```

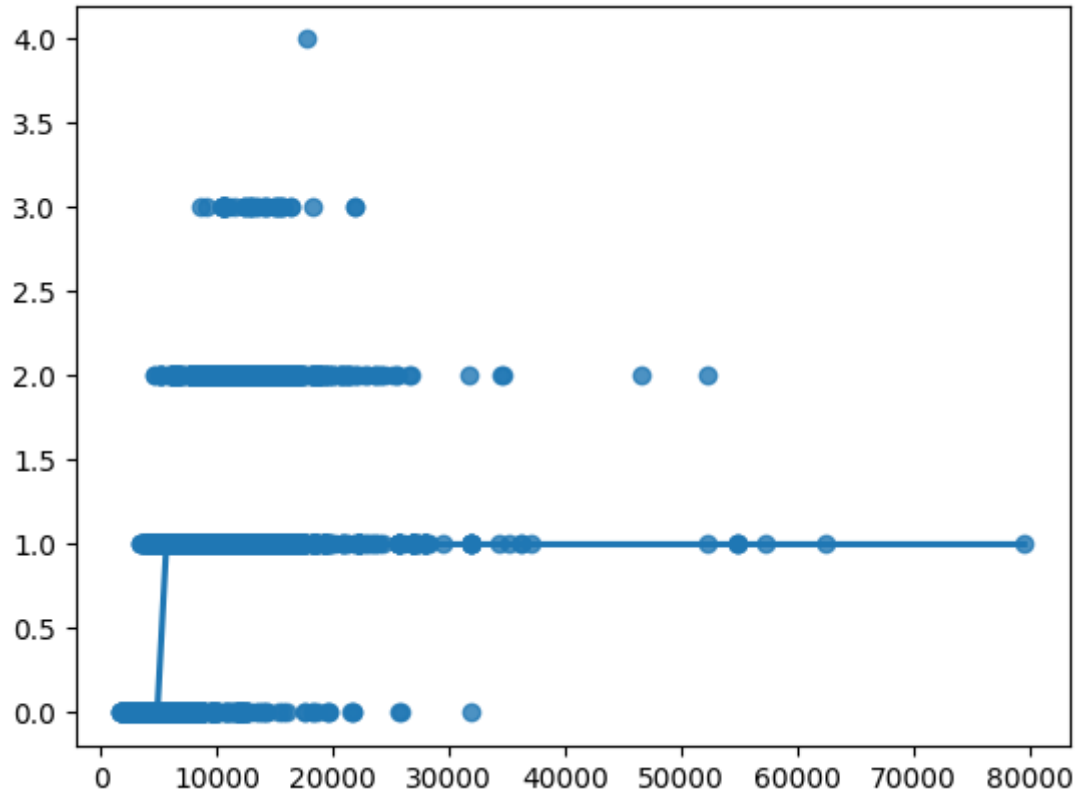
```
0.7311245619624084
```

In [52]:

```
sns.regplot(x=x,y=y,data=df,logistic=True,ci=None)
```

Out[52]:

```
<Axes: >
```



Since we did not get the accuracy for Logistic Regression we are going to implement Decision Tree and Random Forest and make a comparative study for finding the best model for the dataset

Decision tree

In [53]:

```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier(random_state=0)
clf.fit(x_train,y_train)
```

Out[53]:

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [54]:

```
score=clf.score(x_test,y_test)
print(score)
```

```
0.9327811404906021
```

Random Forest

In [55]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(X_train,y_train)
```

Out[55]:

```
RandomForestClassifier
RandomForestClassifier()
```

In [56]:

```
params={'max_depth':[2,3,5,10,20],
'min_samples_leaf':[5,10,20,50,100,200],
'n_estimators':[10,25,30,50,100,200]}
```

In [74]:

```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
grid_search.fit(X_train,y_train)
```

Out[74]:

```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [75]:

```
grid_search.best_score_
```

Out[75]:

```
0.5381674464080405
```

In [76]:

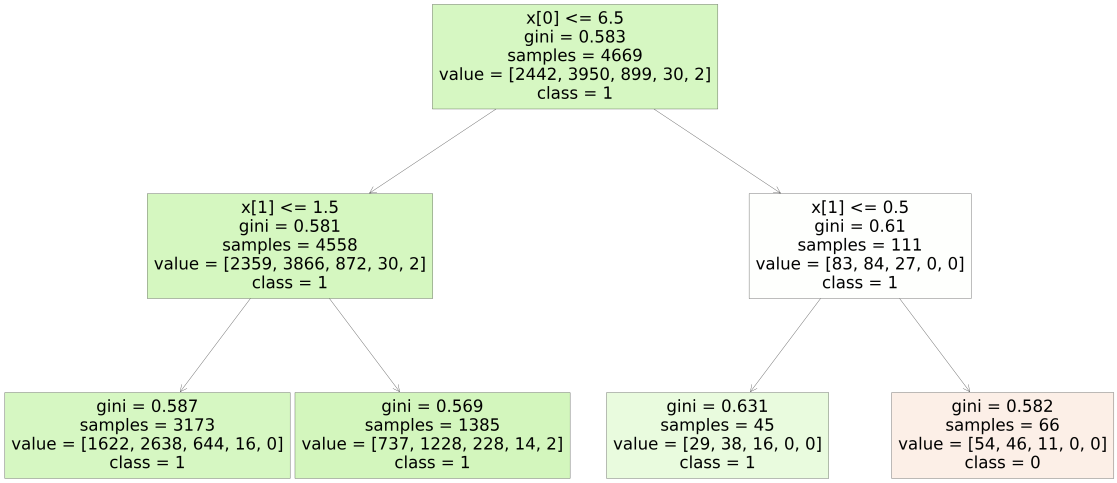
```
rf_best=grid_search.best_estimator_
rf_best
```

Out[76]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=2, min_samples_leaf=5, n_estimators=10)
```

In [77]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],class_names=['0','1','2','3','4'],filled=True);
```



In [79]:

```
score=rfc.score(X_test,y_test)
print(score)
```

```
0.5275565466709143
```

Conlusion:

Based on accuracy scores of all models that were implemented we can conclude that "Decision Tree" is the best model for the given dataset