

PROBLEM STATEMENT: Which model is suitable for flight price prediction dataset.

Import Libraries

```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

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

Requirement already satisfied: openpyxl in c:\users\dell\appdata\local\programs\python\python311\lib\site-packages (3.1.2)

Requirement already satisfied: et-xmlfile in c:\users\dell\appdata\local\programs\python\python311\lib\site-packages (from openpyxl) (1.1.0)

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip available: 22.3.1 -> 23.1.2

[notice] To update, run: python.exe -m pip install --upgrade pip

```
In [3]: train_df=pd.read_excel(r"C:\Users\DELL\Desktop\Data_Train.xlsx")
train_df
```

Out[3]:

	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
...
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU → BLR	19:55	22:25	2h 30m
10679	Air India	27/04/2019	Kolkata	Banglore	CCU → BLR	20:45	23:20	2h 35m
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR → DEL	08:20	11:20	3h
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR → DEL	11:30	14:10	2h 40m
10682	Air India	9/05/2019	Delhi	Cochin	DEL → GOI → BOM → COK	10:55	19:15	8h 20m

10683 rows × 11 columns



Data Cleaning & Preprocessing

In [4]:

train_df.head()

Out[4]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Tot
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 [5]: `train_df.tail()`

Out[5]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU → BLR	19:55	22:25	2h 30m
10679	Air India	27/04/2019	Kolkata	Banglore	CCU → BLR	20:45	23:20	2h 35m
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR → DEL	08:20	11:20	3h
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR → DEL	11:30	14:10	2h 40m
10682	Air India	9/05/2019	Delhi	Cochin	DEL → GOI → BOM → COK	10:55	19:15	8h 20m

In [6]: `train_df.shape`

Out[6]: (10683, 11)

In [7]: `train_df.columns`

Out[7]: Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route', 'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops', 'Additional_Info', 'Price'], dtype='object')

In [8]: `train_df.describe()`

Out[8]:

	Price
count	10683.000000
mean	9087.064121
std	4611.359167
min	1759.000000
25%	5277.000000
50%	8372.000000
75%	12373.000000
max	79512.000000

```
In [9]: 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 [10]: test_df=pd.read_excel(r"C:\Users\DELL\Desktop\Test_set.xlsx")
test_df
```

Out[10]:

	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
...
2666	Air India	6/06/2019	Kolkata	Banglore	CCU → DEL → BLR	20:30	20:25 07 Jun	23h 55m
2667	IndiGo	27/03/2019	Kolkata	Banglore	CCU → BLR	14:20	16:55	2h 35m
2668	Jet Airways	6/03/2019	Delhi	Cochin	DEL → BOM → COK	21:50	04:25 07 Mar	6h 35m
2669	Air India	6/03/2019	Delhi	Cochin	DEL → BOM → COK	04:00	19:15	15h 15m
2670	Multiple carriers	15/06/2019	Delhi	Cochin	DEL → BOM → COK	04:55	19:15	14h 20m

2671 rows × 10 columns



In [11]: test_df.head()

Out[11]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Tot
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 [12]: test_df.tail()

Out[12]:

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

In [13]: `test_df.shape`

Out[13]: (2671, 10)

In [14]: `test_df.columns`

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

In [15]: `test_df.describe()`

Out[15]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration
count	2671	2671	2671	2671	2671	2671	2671	2671
unique	11	44	5	6	100	199	704	320
top	Jet Airways	9/05/2019	Delhi	Cochin	DEL → BOM → COK	10:00	19:00	2h 50m
freq	897	144	1145	1145	624	62	113	122

In [16]: `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
```

Checking any null values


```
In [17]: train_df.isnull().sum()
```

```
Out[17]: 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 [18]: test_df.isnull().sum()
```

```
Out[18]: 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
```

```
In [19]: train_df.dropna(inplace=True)
```

```
In [20]: train_df.isnull().sum()
```

```
Out[20]: 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
```

```
In [21]: train_df.shape
```

```
Out[21]: (10682, 11)
```

```
In [22]: train_df['Airline'].value_counts()
```

```
Out[22]: Airline
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: count, dtype: int64
```

```
In [23]: train_df['Source'].value_counts()
```

```
Out[23]: Source
Delhi      4536
Kolkata    2871
Bangalore  2197
Mumbai     697
Chennai    381
Name: count, dtype: int64
```

```
In [24]: train_df['Destination'].value_counts()
```

```
Out[24]: Destination
Cochin      4536
Bangalore   2871
Delhi       1265
New Delhi   932
Hyderabad   697
Kolkata     381
Name: count, dtype: int64
```

```
In [25]: train_df['Additional_Info'].value_counts
```

```
Out[25]: <bound method IndexOpsMixin.value_counts of 0      No info
1      No info
2      No info
3      No info
4      No info
...
10678  No info
10679  No info
10680  No info
10681  No info
10682  No info
Name: Additional_Info, Length: 10682, dtype: object>
```

Convert string to numerical values

```
In [26]: convert={"Airline":{"Jet Airways":0,"IndiGo":1,"Air India":2,"Multiple carriers":3,"SpiceJet":4,"Vistara":5,"Air Asia":6,"GoAir":7,"Multiple carriers Premium economy":8,"Jet Airways Business":9,"Vistara Premium economy":10,"Trujet":11}}
train_df=train_df.replace(convert)
train_df
```

Out[26]:

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

10682 rows × 11 columns



```
In [27]: convert={"Source":{"Delhi":0,"Kolkata":1,"Banglore":2,  
    "Mumbai":3,"Chennai":4}}  
train_df=train_df.replace(convert)  
train_df
```

Out[27]:

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

10682 rows × 11 columns



```
In [28]: convert={"Destination":{"Cochin":0,"Banglore":1,"Delhi":2,  
    "New Delhi":3,"Hyderabad":4,"Kolkata":5}}  
train_df=train_df.replace(convert)  
train_df
```


Out[28]:

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

10682 rows × 11 columns



```
In [29]: convert={"Additional_Info":{"No info":3}}
train_df=train_df.replace(convert)
train_df
```

Out[29]:

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

10682 rows × 11 columns



```
In [30]: train_df['Total_Stops'].value_counts()
```

```
Out[30]: Total_Stops
1 stop      5625
non-stop    3491
2 stops     1520
3 stops       45
4 stops        1
Name: count, dtype: int64
```

```
In [31]: convert={"Total_Stops":{"non-stop":0,"1 stop":1,"2 stops":2,  
    "3 stops":3,"4 stops":4}}  
train_df=train_df.replace(convert)  
train_df
```

Out[31]:

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

10682 rows × 11 columns



```
In [32]: train_df['Route'].value_counts()
```

```
Out[32]: Route
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: count, Length: 128, dtype: int64
```

```
In [33]: convert={"Route":{"DEL → BOM → COK ":0,"BLR → DEL":1,"CCU → BOM → BLR ":2,
"CCU → BLR ":3,"BOM → HYD ":4,"CCU → VTZ → BLR ":5,"CCU → IXZ → MAA → BLR":6,
"BOM → CCU → HYD":8,"BOM → BBI → HYD":9}}
train_df=train_df.replace(convert)
train_df
```

Out[33]:

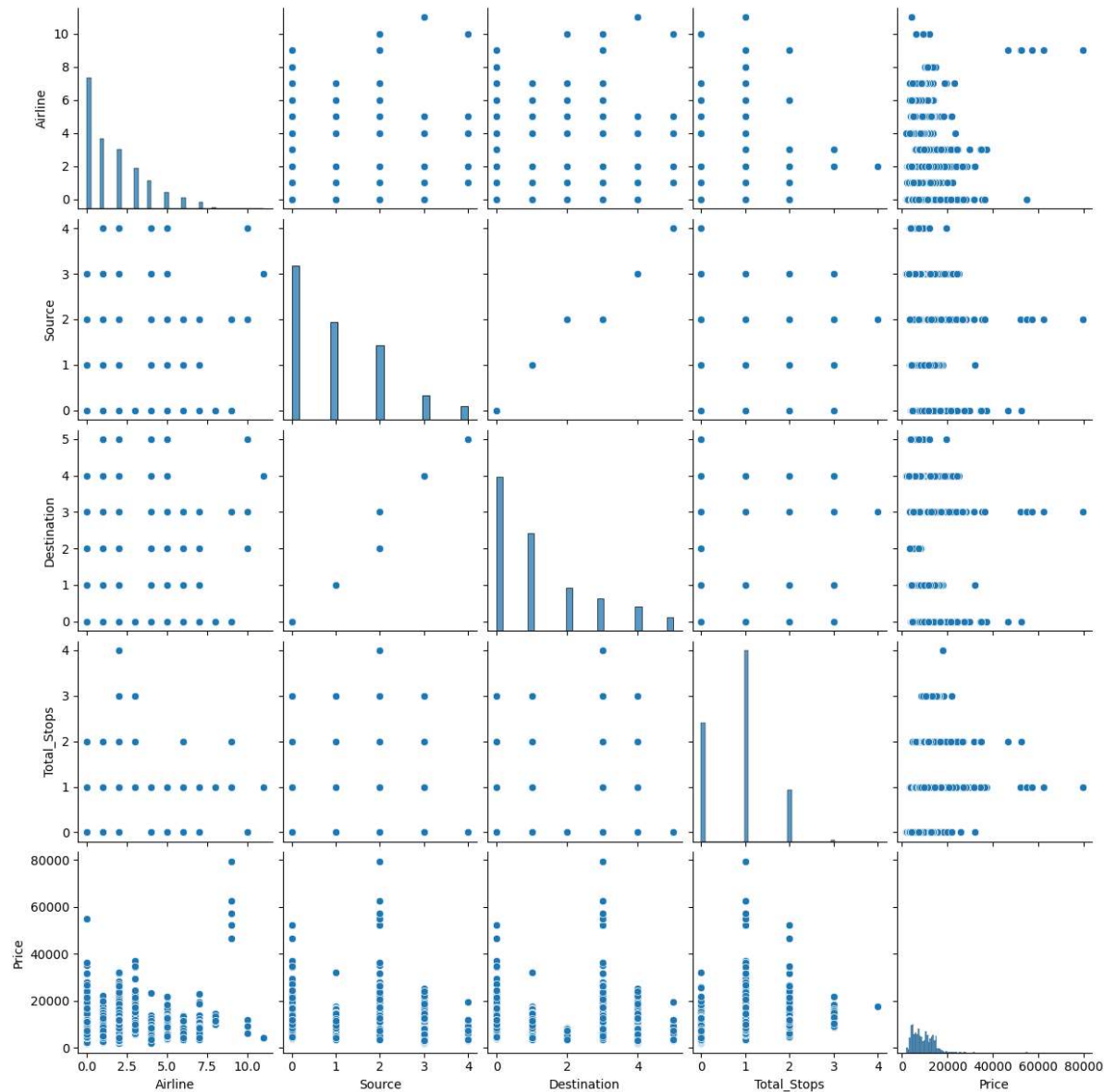
	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	1
0	1	24/03/2019	2	3	1	22:20	01:10 22 Mar	2h 50m	
					CCU → IXR → BBI → BLR				
1	2	1/05/2019	1	1		05:50	13:15	7h 25m	
					DEL → LKO → BOM → COK				
2	0	9/06/2019	0	0		09:25	04:25 10 Jun	19h	
					CCU → NAG → BLR				
3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

10682 rows × 11 columns

Data Visualisation

```
In [34]: sns.pairplot(train_df)
```

```
Out[34]: <seaborn.axisgrid.PairGrid at 0x1b6c87b9710>
```



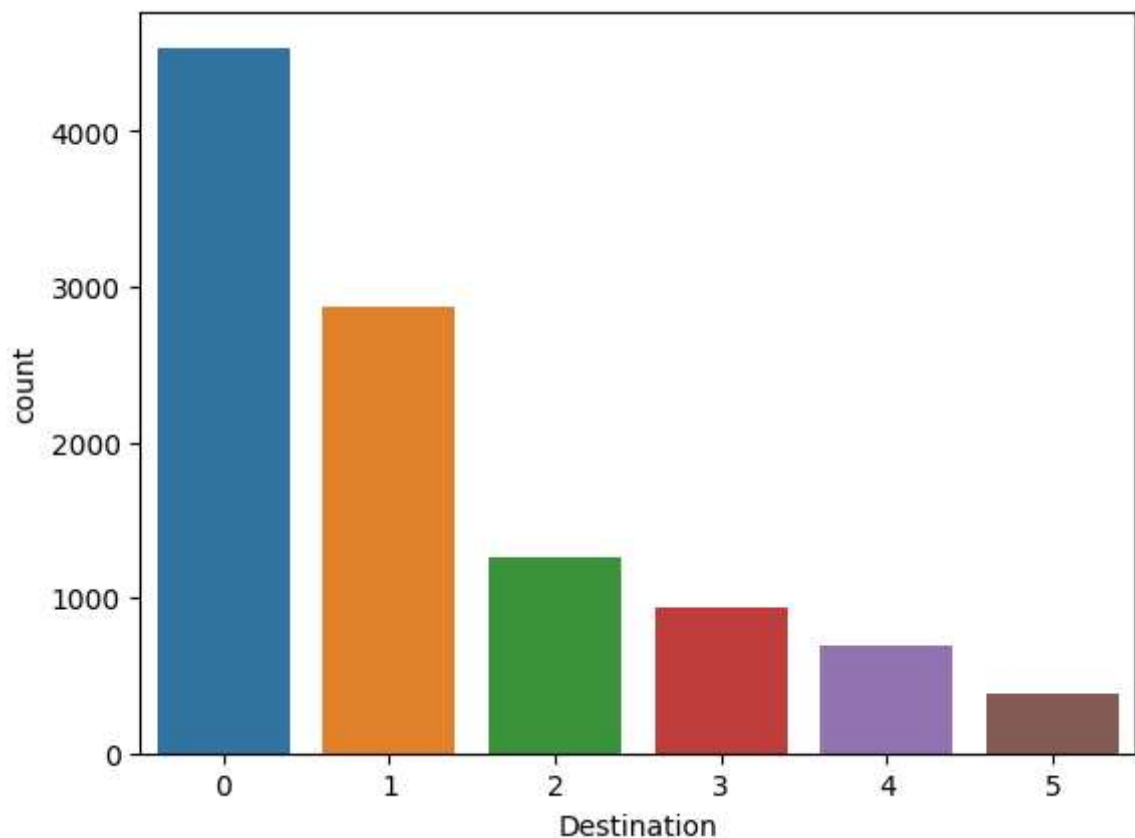

```
In [35]: fdf=train_df[['Airline','Source','Destination','Total_Stops','Price']]
sns.heatmap(fdf.corr(),annot=True)
```

Out[35]: <Axes: >



```
In [36]: sns.countplot(x="Destination", data=train_df)
```

```
Out[36]: <Axes: xlabel='Destination', ylabel='count'>
```



Feature Scaling: To split the data into train and test data.

```
In [37]: x=fdf[['Airline', 'Source', 'Destination', 'Total_Stops']]  
y=fdf['Price']
```

```
In [38]: 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=1)
```

Linear Regression

```
In [39]: from sklearn.linear_model import LinearRegression
```

```
In [40]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)  
regr=LinearRegression()  
regr.fit(x_train,y_train)  
print(regr.score(x_test,y_test))
```

0.3858214433842241

Logistic Regression

```
In [41]: x=np.array(fdf['Price']).reshape(-1,1)
y=np.array(fdf['Total_Stops']).reshape(-1,1)
fdf.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)
```

C:\Users\DELL\AppData\Local\Temp\ipykernel_8688\497261869.py:3: SettingWithCopyWarning:

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-docs/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)

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

```
In [42]: lr.fit(x_train,y_train)
```

C:\Users\DELL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

```
y = column_or_1d(y, warn=True)
```

```
Out[42]: LogisticRegression
LogisticRegression(max_iter=10000)
```

```
In [43]: score=lr.score(x_test,y_test)
print(score)
```

```
0.7160686427457098
```

Decision Tree

```
In [44]: from sklearn.tree import DecisionTreeClassifier
clt=DecisionTreeClassifier(random_state=0)
clt.fit(x_train,y_train)
```

```
Out[44]: DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

```
In [45]: score=clt.score(x_test,y_test)
print(score)
```

0.9369734789391576

Random Forest

```
In [46]: from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

C:\Users\DELL\AppData\Local\Temp\ipykernel_8688\2210184639.py:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

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

```
Out[46]: RandomForestClassifier
RandomForestClassifier()
```

```
In [47]: rf=RandomForestClassifier()
```

```
In [48]: 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 [49]: from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

C:\Users\DELL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection_split.py:700: UserWarning: The least populated class in y has only 1 members, which is less than n_splits=2.

```
warnings.warn(
```

C:\Users\DELL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

```
estimator.fit(X_train, y_train, **fit_params)
```

C:\Users\DELL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

```
estimator.fit(X_train, y_train, **fit_params)
```

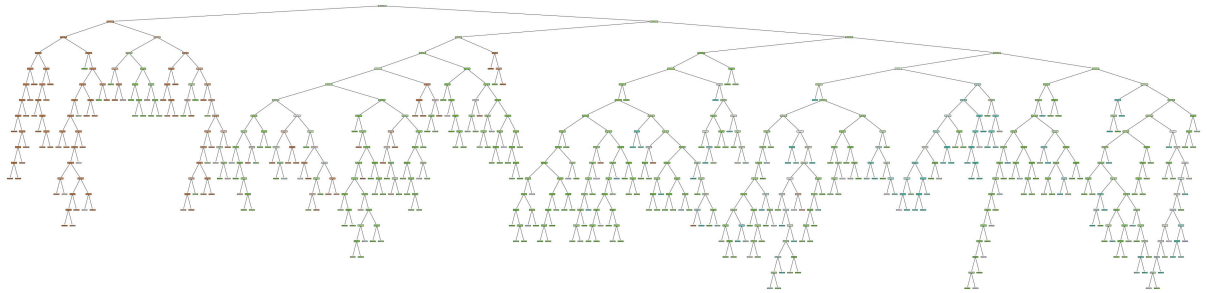
C:\Users\DELL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

```
estimator.fit(X_train, y_train, **fit_params)
```

```
In [50]: rf_best=grid_search.best_estimator_  
rf_best
```

```
Out[50]: RandomForestClassifier  
RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_estimators=30)
```

```
In [51]: from sklearn.tree import plot_tree  
plt.figure(figsize=(80,20))  
plot_tree(rf_best.estimators_[5],filled=True);
```



```
In [52]: score=rfc.score(x_test,y_test)  
print(score)
```

0.9375975039001561

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

```
In [ ]:
```