

Problem Statement: Which model is best fit for given dataset

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt, seaborn as sns
from sklearn.model_selection import train_test_split
```

Data Collection

```
In [2]: train_df=pd.read_csv(r"C:\Users\teppa\Downloads\Flight_train.csv")
train_df
```

```
Out[2]:
```

	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



```
In [3]: test_df=pd.read_csv(r"C:\Users\teppa\Downloads\flight_test.csv")
test_df
```

Out[3]:

	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



Data Cleaning & Preprocessing

In [4]: `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 [5]: `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 [6]: train_df.head()

Out[6]:

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

Out[8]:

	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 [9]: `test_df.tail()`

Out[9]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration
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 [10]: train_df.describe
```

```
Out[10]: <bound method NDFrame.describe of
e Destination
0      IndiGo      24/03/2019  Bangalore  New Delhi  \
1      Air India    1/05/2019   Kolkata    Bangalore
2      Jet Airways  9/06/2019    Delhi      Cochin
3      IndiGo      12/05/2019   Kolkata    Bangalore
4      IndiGo      01/03/2019   Bangalore  New Delhi
...      ...      ...      ...      ...
10678   Air Asia    9/04/2019   Kolkata    Bangalore
10679   Air India   27/04/2019   Kolkata    Bangalore
10680   Jet Airways 27/04/2019   Bangalore  Delhi
10681   Vistara     01/03/2019   Bangalore  New Delhi
10682   Air India    9/05/2019    Delhi      Cochin

      Route Dep_Time  Arrival_Time  Duration  Total_Stops  \
0      BLR ? DEL    22:20    01:10 22 Mar    2h 50m    non-stop  \
1  CCU ? IXR ? BBI ? BLR    05:50          13:15    7h 25m    2 stops
2  DEL ? LKO ? BOM ? COK    09:25    04:25 10 Jun    19h    2 stops
3      CCU ? NAG ? BLR    18:05          23:30    5h 25m    1 stop
4      BLR ? NAG ? DEL    16:50          21:35    4h 45m    1 stop
...      ...      ...      ...      ...      ...
10678      CCU ? BLR    19:55          22:25    2h 30m    non-stop
10679      CCU ? BLR    20:45          23:20    2h 35m    non-stop
10680      BLR ? DEL    08:20          11:20     3h    non-stop
10681      BLR ? DEL    11:30          14:10    2h 40m    non-stop
10682  DEL ? GOI ? BOM ? COK    10:55          19:15    8h 20m    2 stops

      Additional_Info  Price
0      No info    3897
1      No info    7662
2      No info   13882
3      No info    6218
4      No info   13302
...      ...      ...
10678   No info    4107
10679   No info    4145
10680   No info    7229
10681   No info   12648
10682   No info   11753
```

```
[10683 rows x 11 columns]>
```

```
In [11]: test_df.describe
```

```
Out[11]: <bound method NDFrame.describe of
Source Destination
0 Jet Airways 6/06/2019 Delhi Cochin \
1 IndiGo 12/05/2019 Kolkata Bangalore
2 Jet Airways 21/05/2019 Delhi Cochin
3 Multiple carriers 21/05/2019 Delhi Cochin
4 Air Asia 24/06/2019 Bangalore Delhi
...
2666 Air India 6/06/2019 Kolkata Bangalore
2667 IndiGo 27/03/2019 Kolkata Bangalore
2668 Jet Airways 6/03/2019 Delhi Cochin
2669 Air India 6/03/2019 Delhi Cochin
2670 Multiple carriers 15/06/2019 Delhi Cochin

Route Dep_Time Arrival_Time Duration Total_Stops
0 DEL ? BOM ? COK 17:30 04:25 07 Jun 10h 55m 1 stop \
1 CCU ? MAA ? BLR 06:20 10:20 4h 1 stop
2 DEL ? BOM ? COK 19:15 19:00 22 May 23h 45m 1 stop
3 DEL ? BOM ? COK 08:00 21:00 13h 1 stop
4 BLR ? DEL 23:55 02:45 25 Jun 2h 50m non-stop
...
2666 CCU ? DEL ? BLR 20:30 20:25 07 Jun 23h 55m 1 stop
2667 CCU ? BLR 14:20 16:55 2h 35m non-stop
2668 DEL ? BOM ? COK 21:50 04:25 07 Mar 6h 35m 1 stop
2669 DEL ? BOM ? COK 04:00 19:15 15h 15m 1 stop
2670 DEL ? BOM ? COK 04:55 19:15 14h 20m 1 stop

Additional_Info
0 No info
1 No info
2 In-flight meal not included
3 No info
4 No info
...
2666 No info
2667 No info
2668 No info
2669 No info
2670 No info
```

```
[2671 rows x 10 columns]>
```

```
In [12]: train_df.shape
```

```
Out[12]: (10683, 11)
```

```
In [13]: test_df.shape
```

```
Out[13]: (2671, 10)
```



```
In [14]: train_df.isnull().sum()
```

```
Out[14]: 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 [15]: train_df.fillna(method='ffill',inplace=True)
```

```
In [16]: train_df.isnull().sum()
```

```
Out[16]: 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 [17]: test_df.isnull().sum()
```

```
Out[17]: 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 [18]: train_df['Airline'].value_counts()
```

```
Out[18]: Airline
Jet Airways          3849
IndiGo               2053
Air India            1752
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 [19]: train_df['Source'].value_counts()
```

```
Out[19]: Source
Delhi      4537
Kolkata    2871
Banglore   2197
Mumbai     697
Chennai    381
Name: count, dtype: int64
```

```
In [20]: train_df['Additional_Info'].value_counts()
```

```
Out[20]: Additional_Info
No info          8345
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: count, dtype: int64
```

```
In [21]: train_df['Destination'].value_counts()
```

```
Out[21]: Destination
Cochin      4537
Banglore    2871
Delhi       1265
New Delhi   932
Hyderabad   697
Kolkata     381
Name: count, dtype: int64
```

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

```
Out[22]: Total_Stops
1 stop      5625
non-stop    3492
2 stops     1520
3 stops       45
4 stops        1
Name: count, dtype: int64
```

```
In [23]: a={"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(a)
train_df
```

Out[23]:

	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

10683 rows × 11 columns



```
In [24]: b={"Source":{"Delhi":1,"Kolkata":2,"Banglore":3,"Mumbai":4,"Chennai":5}}
train_df=train_df.replace(b)
train_df
```

Out[24]:

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

10683 rows × 11 columns

```
In [25]: c={"Destination":{"Cochin":1,"Banglore":3,"Delhi":4,"New Delhi":5,"Hyderabad":
train_df=train_df.replace(c)
train_df
```

Out[25]:

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

10683 rows × 11 columns

```
In [26]: d={"Total_Stops":{"non-stop":0,"1 stop":1,"2 stops":2,"3 stops":3,"4 stops":4}
train_df=train_df.replace(d)
train_df
```

Out[26]:

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

10683 rows × 11 columns

```
In [27]: train_df
```

Out[27]:

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

10683 rows × 11 columns



Data Visualization

```
In [28]: df=train_df[['Airline','Source','Destination','Total_Stops','Price']]
sns.heatmap(df.corr(),annot=True)
```

Out[28]: <Axes: >



```
In [29]: x=df[['Airline','Source','Destination','Total_Stops']]
y=df['Price']
```

Model Building

Linear Regression:

```
In [30]: 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=
```

```
In [31]: from sklearn.linear_model import LinearRegression
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.intercept_)
coeff_df=pd.DataFrame(regr.coef_,columns=['coefficient'])
coeff_df
```

8298.528534891828

Out[31]:

	coefficient
--	-------------

0	-349.864027
---	-------------

1	-4177.926793
---	--------------

2	2472.364335
---	-------------

3	3589.079598
---	-------------

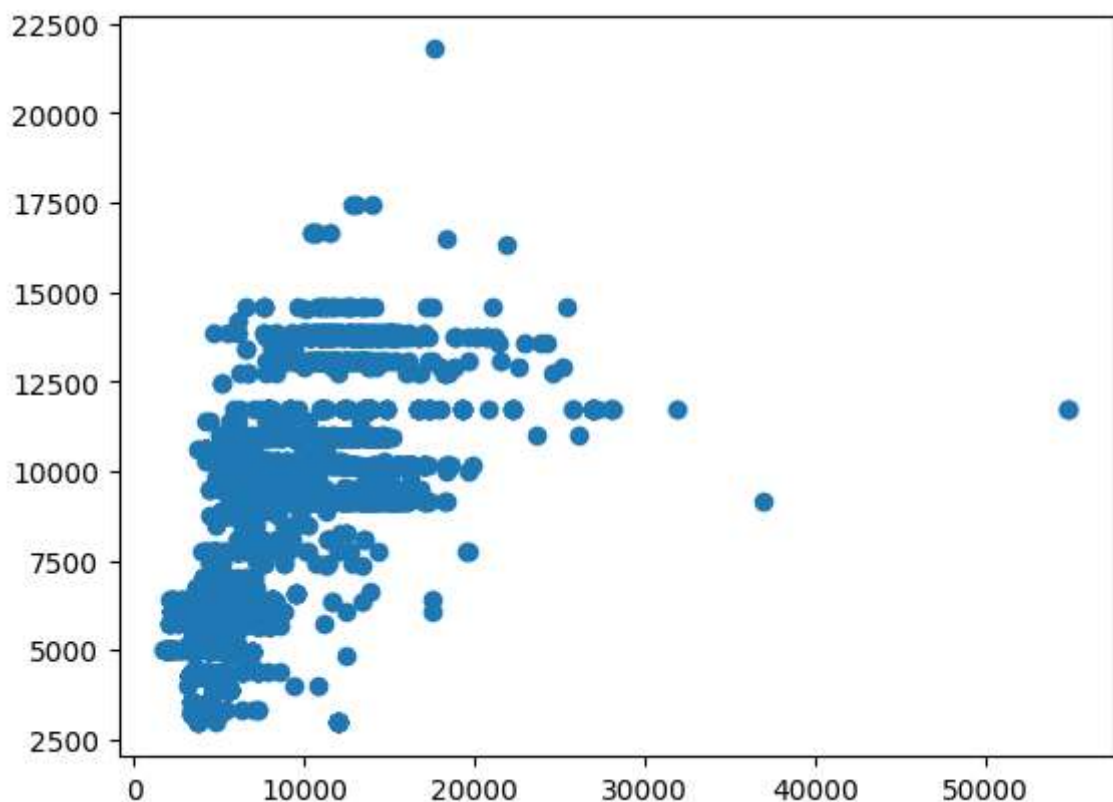
Evaluation

```
In [32]: score=regr.score(x_test,y_test)
print(score)
```

0.43702395281202056

```
In [33]: predictions=regr.predict(x_test)
plt.scatter(y_test,predictions)
```

Out[33]: <matplotlib.collections.PathCollection at 0x1ed35df30d0>



```
In [34]: x=np.array(df['Price']).reshape(-1,1)
y=np.array(df['Total_Stops']).reshape(-1,1)
df.dropna()
```

```
Out[34]:
```

	Airline	Source	Destination	Total_Stops	Price
0	1	3	5	0	3897
1	2	2	3	2	7662
2	0	1	1	2	13882
3	1	2	3	1	6218
4	1	3	5	1	13302
...
10678	6	2	3	0	4107
10679	2	2	3	0	4145
10680	0	3	4	0	7229
10681	5	3	5	0	12648
10682	2	1	1	2	11753

10683 rows × 5 columns

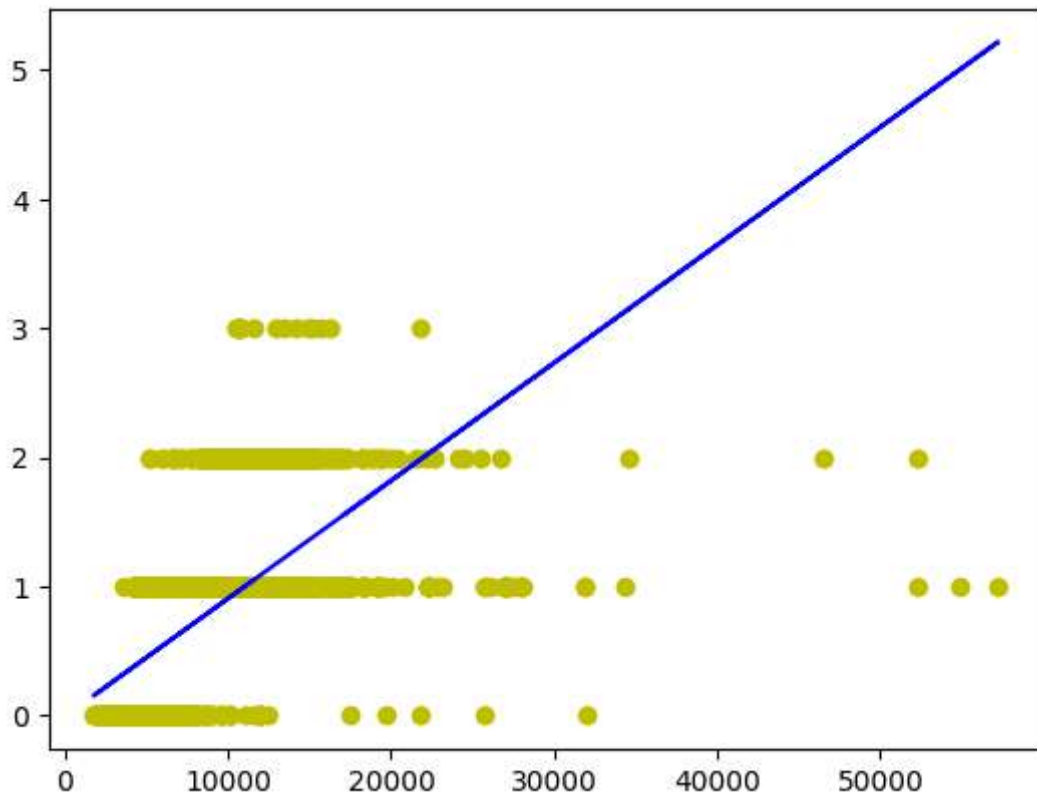
```
In [35]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
regr.fit(x_train,y_train)
regr.fit(x_train,y_train)
```

```
Out[35]: LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [36]: y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='y')
plt.plot(x_test,y_pred,color='b')
plt.show()
```



Logistic Regression:

```
In [37]: from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

```
In [38]: x=np.array(df['Price']).reshape(-1,1)
y=np.array(df['Total_Stops']).reshape(-1,1)
```

```
In [39]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=
```

```
In [40]: from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(max_iter=10000)
```

```
In [41]: lr.fit(x_train,y_train)
```

```
C:\Users\pappu\AppData\Local\Programs\Python\Python310\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[41]: LogisticRegression(max_iter=10000)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

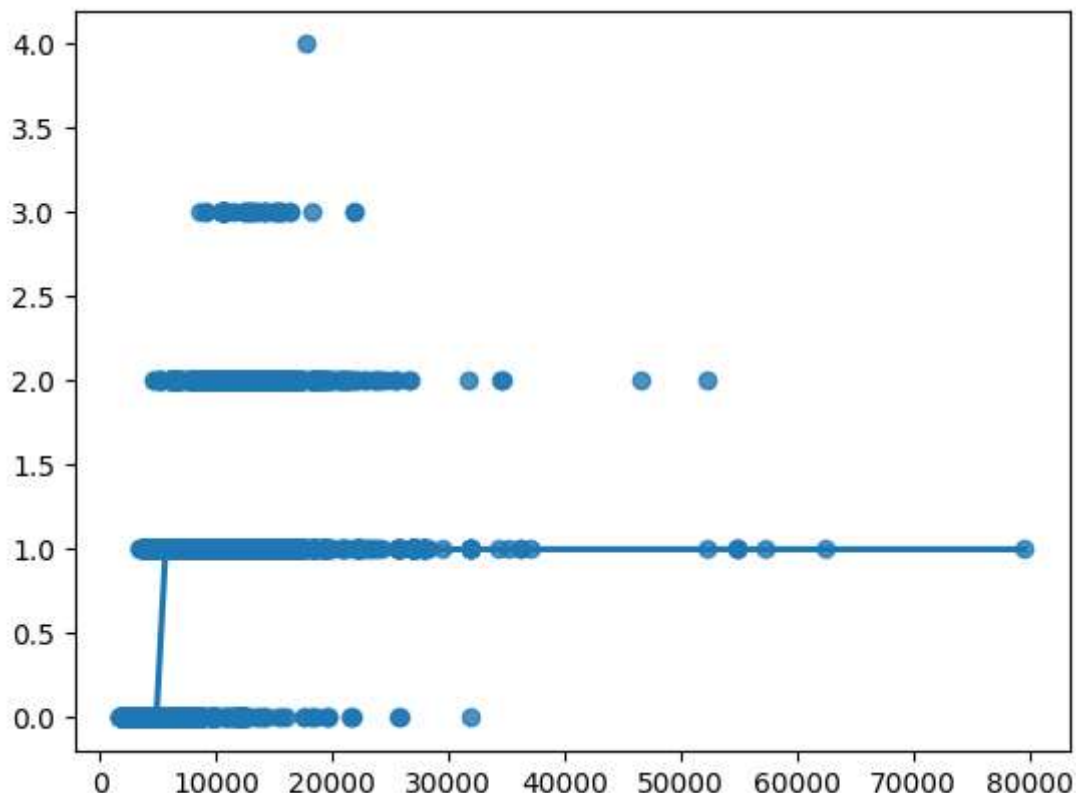
```
In [42]: score=lr.score(x_test,y_test)
print(score)
```

```
0.7101404056162246
```

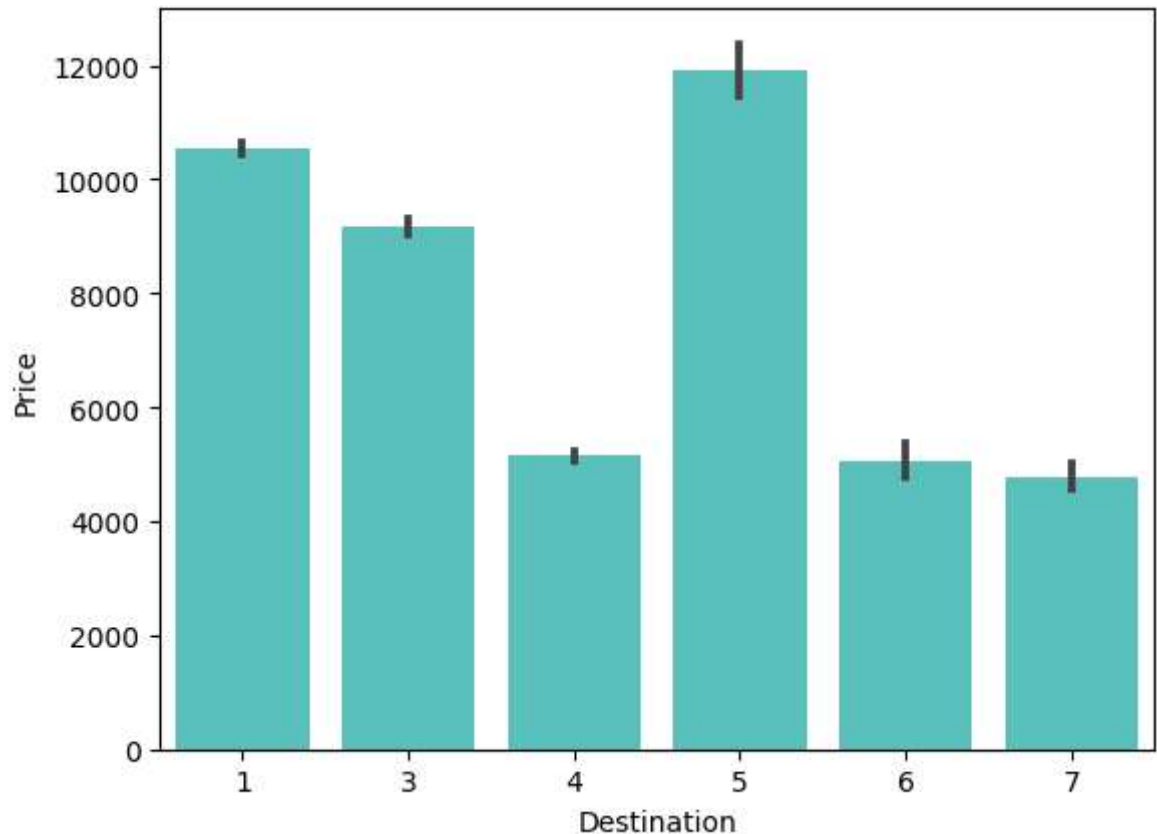
```
In [43]: sns.regplot(x=x,y=y,data=df,logistic=True,ci=None)
```

```
C:\Users\pappu\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels\genmod\family\links.py:198: RuntimeWarning: overflow encountered in exp
  t = np.exp(-z)
```

```
Out[43]: <Axes: >
```



```
In [44]: sns.barplot(x='Destination', y='Price', data=df, color="mediumturquoise")  
plt.show()
```



Decision Tree:

```
In [45]: from sklearn.tree import DecisionTreeClassifier
```

```
In [46]: clf=DecisionTreeClassifier(random_state=0)  
clf.fit(x_train,y_train)
```

Out[46]: DecisionTreeClassifier(random_state=0)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [47]: score=clf.score(x_test,y_test)  
print(score)
```

0.9351014040561623

Random Forest:

```
In [48]: from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

C:\Users\pappu\AppData\Local\Temp\ipykernel_25636\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[48]: RandomForestClassifier()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [49]: 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 [50]: from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
```

```
In [51]: grid_search.fit(x_train,y_train)
```

C:\Users\pappu\AppData\Local\Programs\Python\Python310\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\pappu\AppData\Local\Programs\Python\Python310\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\pappu\AppData\Local\Programs\Python\Python310\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\pappu\AppData\Local\Programs\Python\Python310\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 [53]: grid_search.best_score_
```

Out[53]: 0.8685477400374432

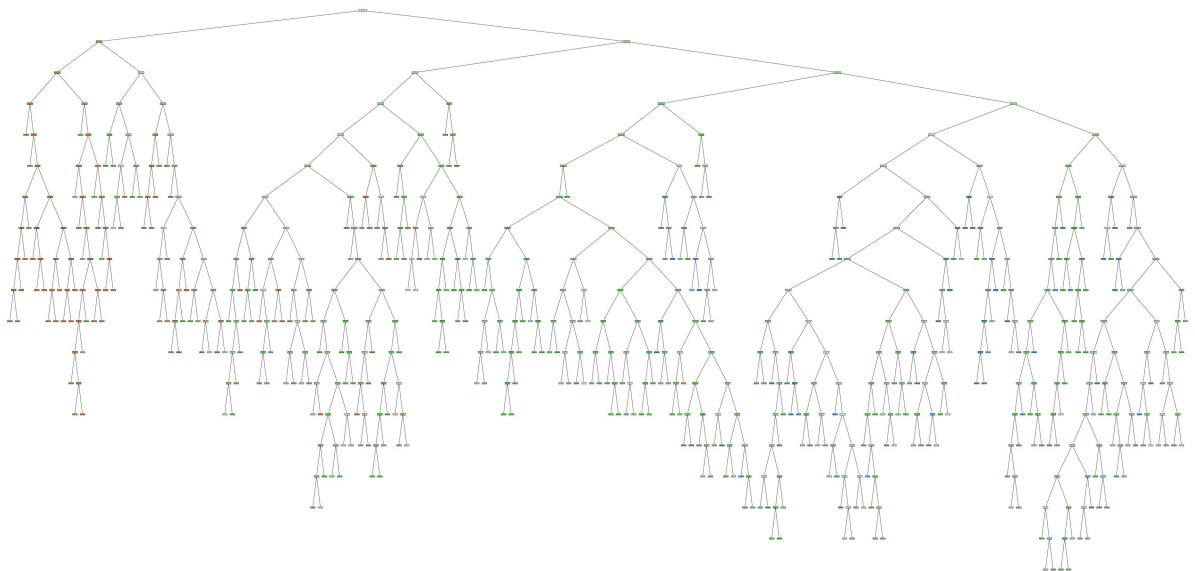
```
In [54]: rf_best=grid_search.best_estimator_  
rf_best
```

Out[54]: RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_estimators=50)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [55]: 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 [56]: score=rfc.score(x_test,y_test)  
print(score)
```

0.9354134165366614

Conclusion

*Based on the accuracy scores of the above implemented models, we can conclude that **Random Forest** is best model for the given dataset.*