MINI PROJECT

PROBLEM STATEMENT : Which model is suitable for Flight Price Prediction

Importing Packages

In [1]:

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

Read the Data

In [2]:

```
traindf=pd.read_csv(r"C:\Users\DHEEPAK\Desktop\Copy of Data_Train.csv")
traindf
```

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]:

```
testdf=pd.read_csv(r"C:\Users\DHEEPAK\Desktop\Copy of Test_set.csv")
testdf
```

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

Data Collection and Preprocessing

In [4]:

```
traindf.head()
```

Out[4]:

	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 [5]:

```
testdf.head()
```

Out[5]:

	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 [6]:

```
traindf.tail()
```

Out[6]:

	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 [7]:

```
testdf.tail()
```

Out[7]:

	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 [8]:

```
traindf.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]:

```
testdf.describe()
```

Out[9]:

	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 [10]:

```
traindf.shape
```

Out[10]:

```
(10683, 11)
```

In [11]:

```
testdf.shape
```

Out[11]:

```
(2671, 10)
```

In [12]:

```
traindf.columns
```

Out[12]:

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

In [13]:

```
testdf.columns
```

Out[13]:

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

In [14]:

```
traindf.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 [15]:

```
testdf.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 whether there are any null values in the dataset

In [16]:

```
traindf.isnull().sum()
```

Out[16]:

```
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 [17]:

```
testdf.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
```

Removing Null Values from the dataset

In [18]:

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

In [19]:

```
traindf.isnull().sum()
```

Out[19]:

```
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 [20]:

```
traindf.shape
```

Out[20]:

```
(10682, 11)
```

Conversion of datatype of values from String to Numerical Values

In [21]:

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

Out[21]:

```
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 [22]:

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

Out[22]:

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

In [23]:

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

Out[23]:

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

In [24]:

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

Out[24]:

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

In [25]:

```
airline={"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}}
traindf=traindf.replace(airline)
traindf
```

Out[25]:

	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

10682 rows × 11 columns

In [26]:

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

Out[26]:

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

10682 rows × 11 columns

In [27]:

```
destination={"Destination":{"Cochin":0,"Banglore":1,"Delhi":2,
 "New Delhi":3,"Hyderabad":4,"Kolkata":5}}
traindf=traindf.replace(destination)
traindf
```

Out[27]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durati
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	1
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

10682 rows × 11 columns

In [28]:

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

Out[28]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durati
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	1
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

10682 rows × 11 columns

In [29]:

```
traindf
```

Out[29]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durati
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	1
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

10682 rows × 11 columns

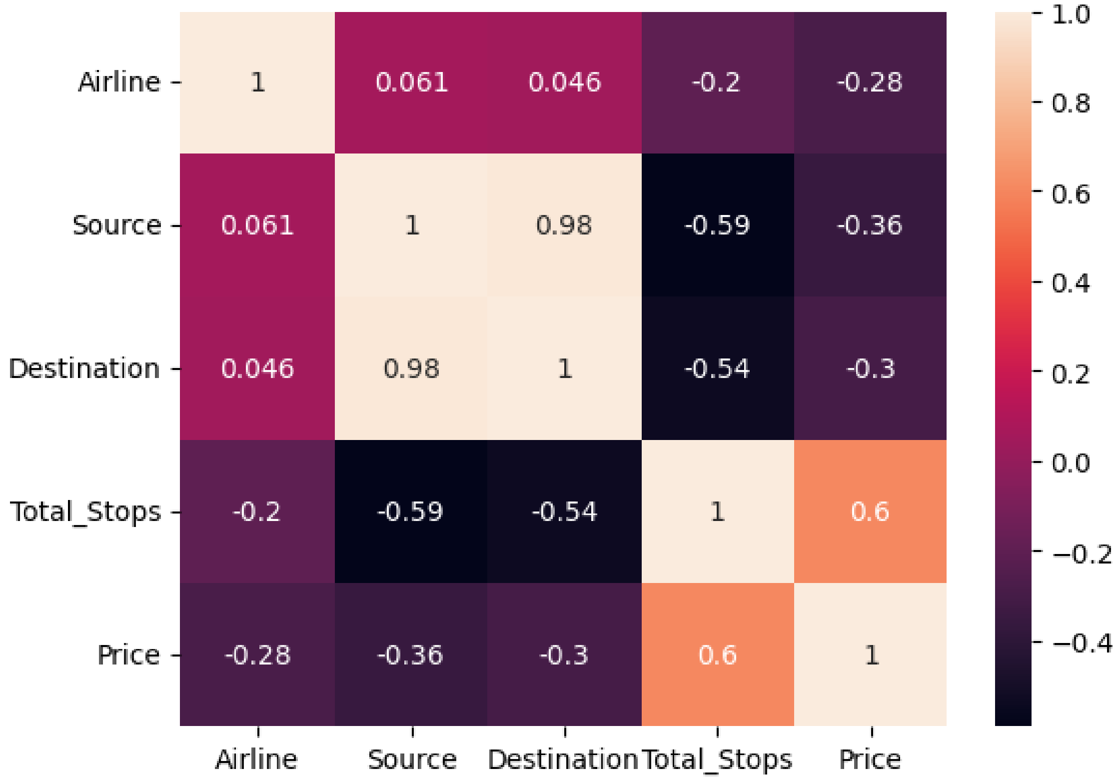
Data Visualization

In [30]:

```
#EDA
fdf=traindf[['Airline','Source','Destination','Total_Stops','Price']]
sns.heatmap(fdf.corr(),annot=True)
```

Out[30]:

```
<Axes: >
```



Feature Scaling : To Split the data into training data and test data

In [31]:

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

In [32]:

```
#Linear Regression
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)
```

Linear Regression

In [33]:

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

7211.098088897471

Out[33]:

	coefficient
Airline	-418.483922
Source	-3275.073380
Destination	2505.480291
Total_Stops	3541.798053

In [34]:

```
#Linear Rgeression
score=regr.score(X_test,y_test)
print(score)
```

0.41083048909283415

In [35]:

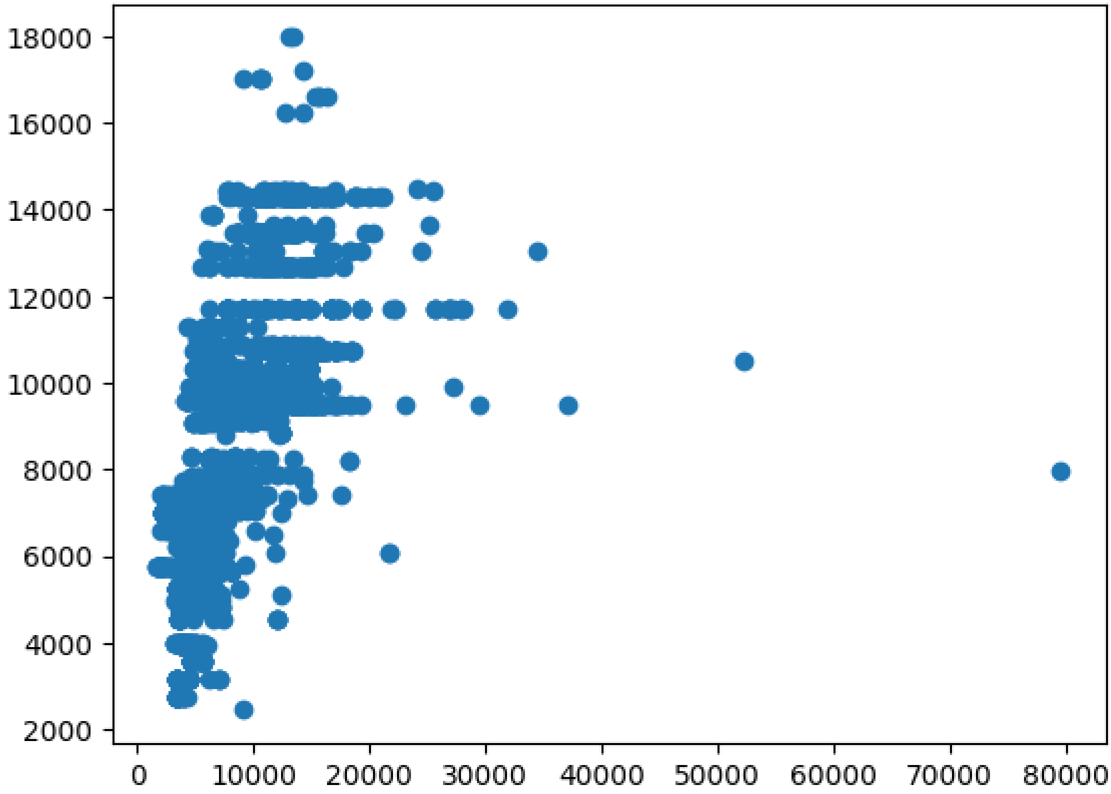
```
predictions=regr.predict(X_test)
```

In [36]:

```
plt.scatter(y_test,predictions)
```

Out[36]:

```
<matplotlib.collections.PathCollection at 0x2178830df90>
```



In [37]:

```
x=np.array(fdf['Price']).reshape(-1,1)
y=np.array(fdf['Total_Stops']).reshape(-1,1)
fdf.dropna(inplace=True)
```

```
C:\Users\DHEEPAK\AppData\Local\Temp\ipykernel_33112\3026288769.py:3: Setti
ngWithCopyWarning:
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)
  fdf.dropna(inplace=True)
```

In [38]:

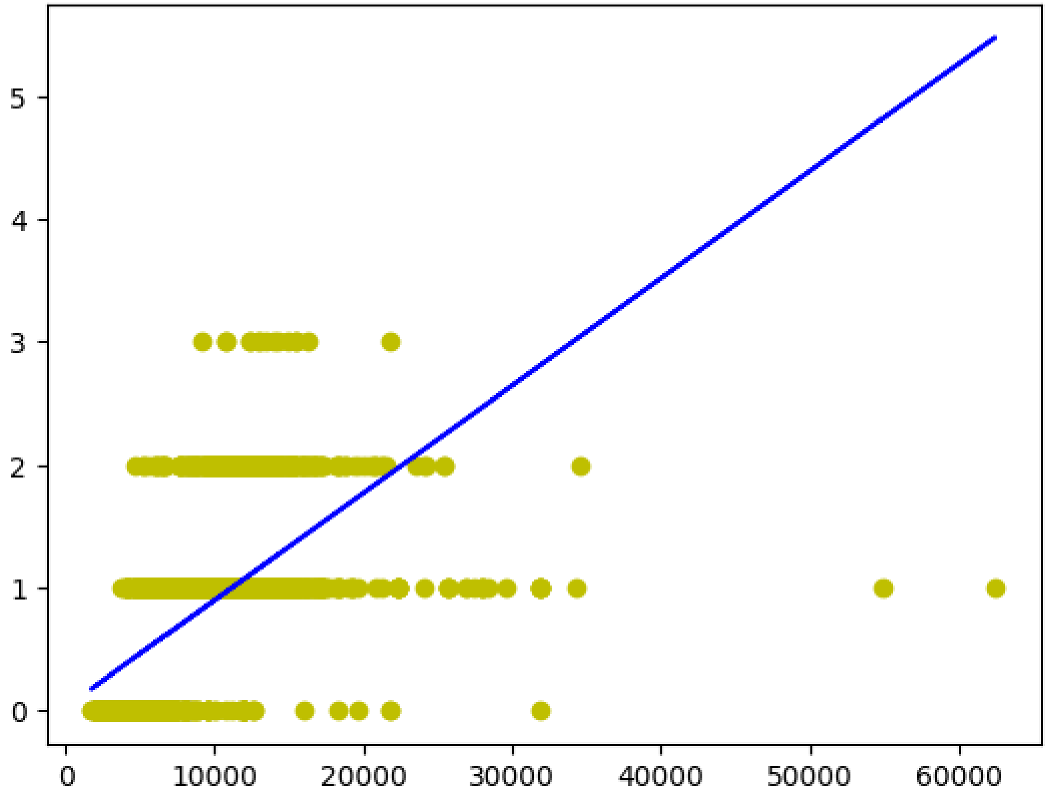
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

Out[38]:

```
▾ LinearRegression
LinearRegression()
```

In [39]:

```
y_pred=regr.predict(X_test)
plt.scatter(X_test,y_test,color='y')
plt.plot(X_test,y_pred,color='b')
plt.show()
```



Since we did not get the accuracy for LinearRegression we are going to implement Logistic Regression

Logistic Regression

In [40]:

```
#Logistic Regression
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\DHEEPAK\AppData\Local\Temp\ipykernel_33112\325765256.py:4: Settin
gWithCopyWarning:
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)
  fdf.dropna(inplace=True)
```

In [41]:

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

```
C:\Users\DHEEPAK\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[41]:

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

In [42]:

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

```
0.7160686427457098
```

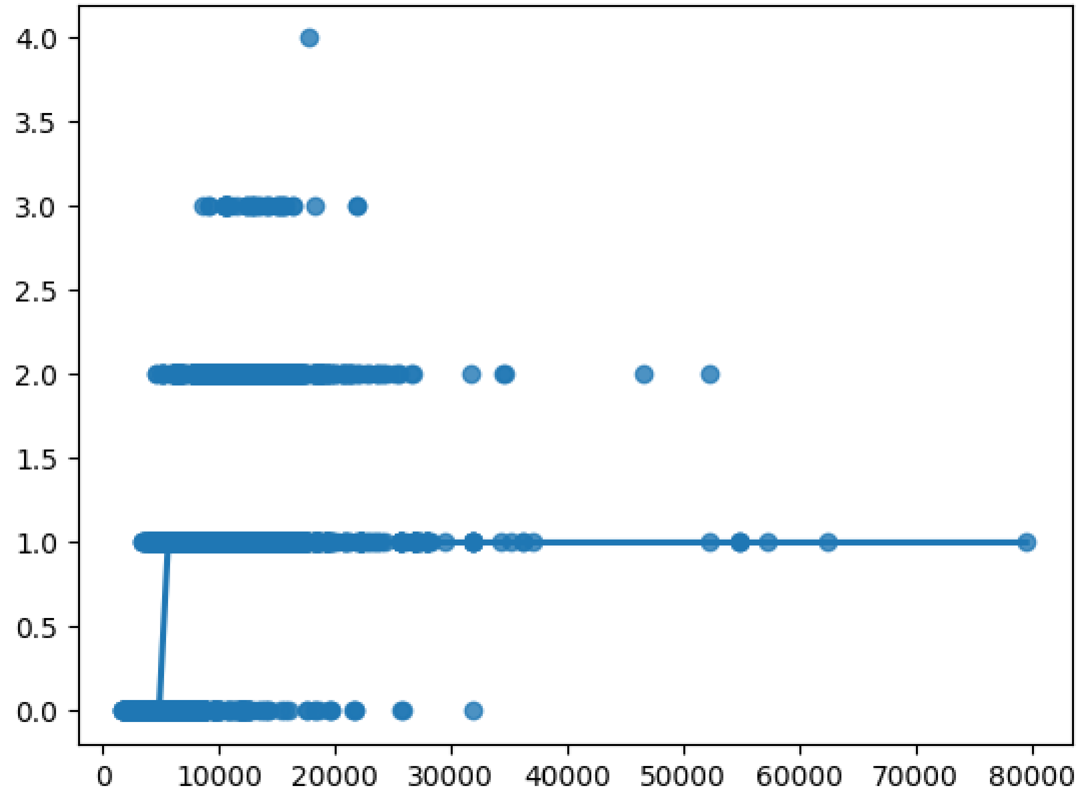
In [43]:

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

```
C:\Users\DHEEPAK\AppData\Local\Programs\Python\Python311\Lib\site-packages
\statsmodels\genmod\families\links.py:198: RuntimeWarning: overflow encoun
tered in exp
  t = np.exp(-z)
```

Out[43]:

```
<Axes: >
```



Since we did not get the accuracy for LogisticRegression 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 [44]:

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

Out[44]:

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

In [45]:

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

```
0.9369734789391576
```

Random Forest

In [46]:

```
#Random forest classifier
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(X_train,y_train)
```

```
C:\Users\DHEEPAK\AppData\Local\Temp\ipykernel_33112\1232785509.py:4: DataC
onversionWarning: A column-vector y was passed when a 1d array was expecte
d. 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]:

```
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 [48]:

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

In [49]:

```
grid_search.fit(X_train,y_train)
```

```
ges\sklearn\model_selection\_validation.py:686: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please chang
e the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\DHEEPAK\AppData\Local\Programs\Python\Python311\Lib\site-packa
ges\sklearn\model_selection\_validation.py:686: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please chang
e the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\DHEEPAK\AppData\Local\Programs\Python\Python311\Lib\site-packa
ges\sklearn\model_selection\_validation.py:686: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please chang
e the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\DHEEPAK\AppData\Local\Programs\Python\Python311\Lib\site-packa
ges\sklearn\model_selection\_validation.py:686: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please chang
e the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\DHEEPAK\AppData\Local\Programs\Python\Python311\Lib\site-packa
```

In [50]:

```
grid_search.best_score_
```

Out[50]:

```
0.5237394412946068
```

In [51]:

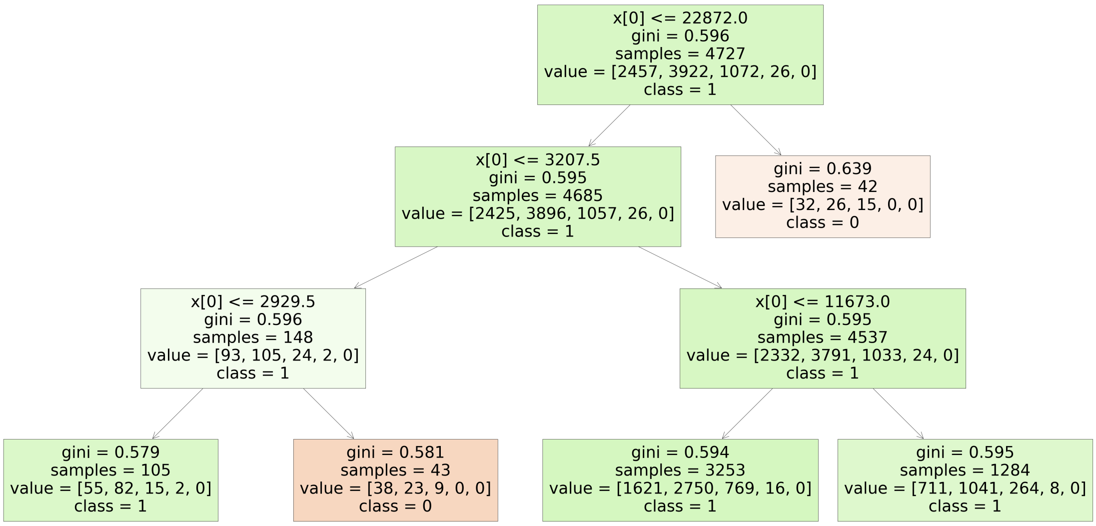
```
rf_best=grid_search.best_estimator_
rf_best
```

Out[51]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=3, min_samples_leaf=20, n_estimators=25)
```

In [52]:

```
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 [53]:

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

0.47051482059282373

Here when we compare between Decision Tree and Random Forest, we can confirm that Decision Treehas more accuracy than Random Forest which makes it the best model for this dataset. It makes DecisionTree to perform better than Random Forest. But it may vary for the other datasets where in most cases Random Forest performs better as it has reduced overfitting and robust to outliers. ¶

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

In [ ]:

```

```