Problem Statement:To predict which is best fit for Flight Price Model Data Frame

In [1]:

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

In [2]:

```python
traindf=pd.read_csv(r"C:\Users\HP\Downloads\Data_Train1.csv")
traindf
```

Out[2]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 5
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 2
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 2
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 4
...	...	...	...	...	...	...	...	
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU ? BLR	19:55	22:25	2h 3
10679	Air India	27/04/2019	Kolkata	Banglore	CCU ? BLR	20:45	23:20	2h 3
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 4
10682	Air India	9/05/2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 2

10683 rows × 11 columns

In [4]:

```
1  testdf=pd.read_csv(r"C:\Users\HP\Downloads\Test_set1.csv")
2  testdf
```

Out[4]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durati
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL ? BOM ? COK	17:30	04:25 07 Jun	10h 55
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 45
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL ? BOM ? COK	08:00	21:00	1:
4	Air Asia	24/06/2019	Banglore	Delhi	BLR ? DEL	23:55	02:45 25 Jun	2h 50
...	...	...	...	...	...	...	...	
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

2671 rows × 10 columns

In [5]:

```
1  traindf.head()
```

Out[5]:

	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 [6]:

```
1  testdf.head()
```

Out[6]:

	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 [7]:

```
1  traindf.tail()
```

Out[7]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU ? BLR	19:55	22:25	2h 3
10679	Air India	27/04/2019	Kolkata	Banglore	CCU ? BLR	20:45	23:20	2h 3
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 4
10682	Air India	9/05/2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 2

In [8]:

```
1  testdf.tail()
```

Out[8]:

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

In [9]:

```
1  traindf.describe()
```

Out[9]:

	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 [10]:

```
1  testdf.describe()
```

Out[10]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
count	2671	2671	2671	2671	2671	2671	2671	26
unique	11	44	5	6	100	199	704	3
top	Jet Airways	9/05/2019	Delhi	Cochin	DEL ? BOM ? COK	10:00	19:00	2h 5
freq	897	144	1145	1145	624	62	113	1

In [11]:

```
1  traindf.shape
```

Out[11]:

(10683, 11)

In [12]:

```
1  testdf.shape
```

Out[12]:

(2671, 10)

In [13]:

```
1  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 [14]:

```
1  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
```

In [15]:

```
1  traindf.duplicated().sum()
```

Out[15]:

220

In [16]:

```
1  testdf.duplicated().sum()
```

Out[16]:

26

In [17]:

```
1  traindf.columns
```

Out[17]:

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

In [18]:

```
1  traindf.columns
```

Out[18]:

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

In [19]:

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

Out[19]:

```
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 [20]:

```
1  testdf.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
dtype: int64
```

In [21]:

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

In [22]:

```
1  traindf.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
Price              0
dtype: int64
```

In [23]:

```
1  traindf.shape
```

Out[23]:

```
(10682, 11)
```

In [24]:

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

Out[24]:

```
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 [25]:

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

Out[25]:

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

In [26]:

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

Out[26]:

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

In [27]:

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

Out[27]:

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

In [28]:

```
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[28]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durati
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	1
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 [29]:

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

Out[29]:

	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 50r
1	2	1/05/2019	1	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25r
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 25r
4	1	01/03/2019	2	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45r
...	...	...	...	...	...	...	...	.
10678	6	9/04/2019	1	Banglore	CCU ? BLR	19:55	22:25	2h 30r
10679	2	27/04/2019	1	Banglore	CCU ? BLR	20:45	23:20	2h 35r
10680	0	27/04/2019	2	Delhi	BLR ? DEL	08:20	11:20	3
10681	5	01/03/2019	2	New Delhi	BLR ? DEL	11:30	14:10	2h 40r
10682	2	9/05/2019	0	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20r

10682 rows × 11 columns

In [30]:

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

Out[30]:

	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 50r
1	2	1/05/2019	1	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25r
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 25r
4	1	01/03/2019	2	3	BLR ? NAG ? DEL	16:50	21:35	4h 45r
...	...	...	...	...	...	...	...	.
10678	6	9/04/2019	1	1	CCU ? BLR	19:55	22:25	2h 30r
10679	2	27/04/2019	1	1	CCU ? BLR	20:45	23:20	2h 35r
10680	0	27/04/2019	2	2	BLR ? DEL	08:20	11:20	3
10681	5	01/03/2019	2	3	BLR ? DEL	11:30	14:10	2h 40r
10682	2	9/05/2019	0	0	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20r

10682 rows × 11 columns

In [31]:

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

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 50r
1	2	1/05/2019	1	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25r
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 25r
4	1	01/03/2019	2	3	BLR ? NAG ? DEL	16:50	21:35	4h 45r
...	...	...	...	...	...	...	...	.
10678	6	9/04/2019	1	1	CCU ? BLR	19:55	22:25	2h 30r
10679	2	27/04/2019	1	1	CCU ? BLR	20:45	23:20	2h 35r
10680	0	27/04/2019	2	2	BLR ? DEL	08:20	11:20	3
10681	5	01/03/2019	2	3	BLR ? DEL	11:30	14:10	2h 40r
10682	2	9/05/2019	0	0	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20r

10682 rows × 11 columns

In [32]:

```
1  traindf
```

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 50r
1	2	1/05/2019	1	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25r
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 25r
4	1	01/03/2019	2	3	BLR ? NAG ? DEL	16:50	21:35	4h 45r
...	...	...	...	...	...	...	...	.
10678	6	9/04/2019	1	1	CCU ? BLR	19:55	22:25	2h 30r
10679	2	27/04/2019	1	1	CCU ? BLR	20:45	23:20	2h 35r
10680	0	27/04/2019	2	2	BLR ? DEL	08:20	11:20	3
10681	5	01/03/2019	2	3	BLR ? DEL	11:30	14:10	2h 40r
10682	2	9/05/2019	0	0	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20r

10682 rows × 11 columns

In [33]:

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

Out[33]:

```
<Axes: >
```


In [34]:

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

Linear Regression

In [35]:

```python
#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)
```

In [36]:

```python
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
```

7211.098088897488

Out[36]:

	coefficient
Airline	-418.483922
Source	-3275.073380
Destination	2505.480291
Total_Stops	3541.798053

In [37]:

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

0.4108304890928348

In [38]:

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

In [39]:

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

Out[39]:

`<matplotlib.collections.PathCollection at 0x1d6136e65d0>`
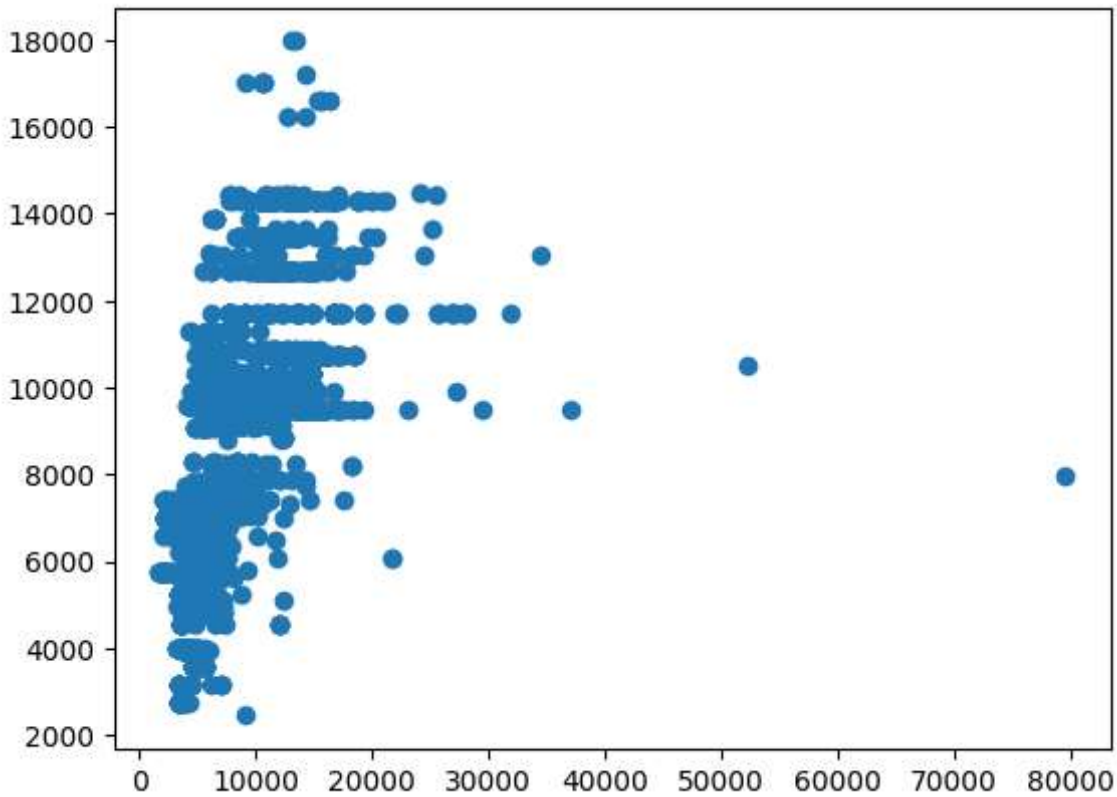
In [40]:

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

```
C:\Users\HP\AppData\Local\Temp\ipykernel_7140\521034954.py:3: SettingWithC
opyWarning:
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]:

```python
1  X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
2  regr.fit(X_train,y_train)
3  regr.fit(X_train,y_train)
```
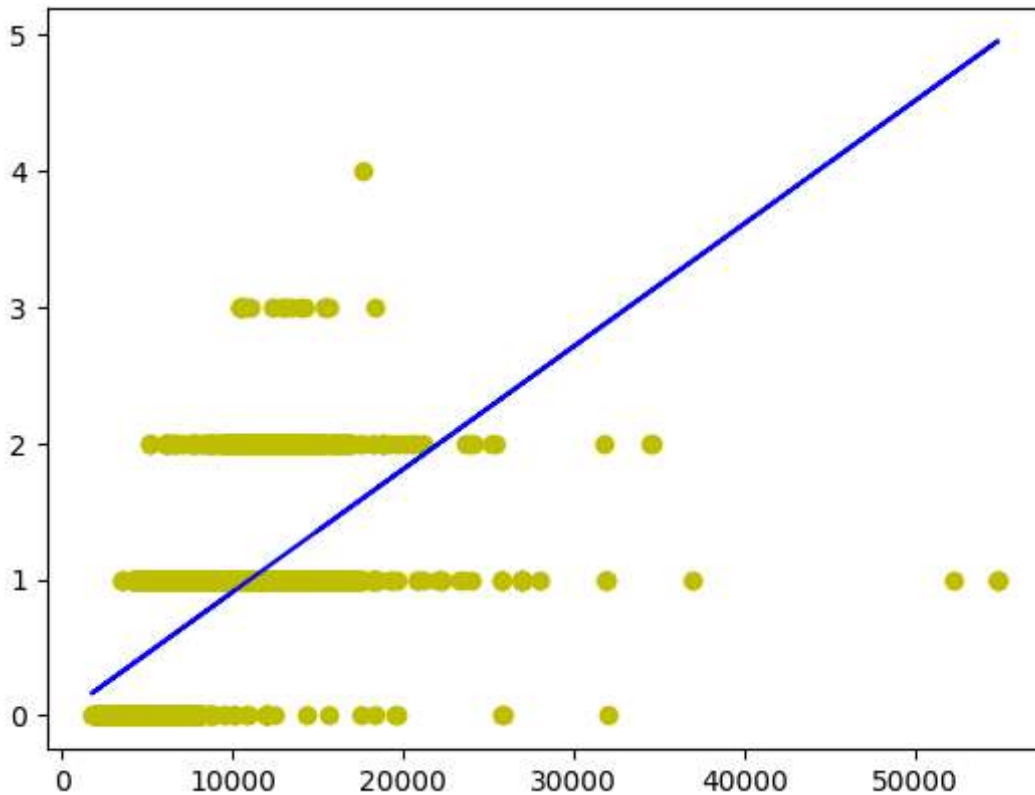
Out[41]:

```
▼ LinearRegression

LinearRegression()
```

In [42]:

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


Logistic Regression

In [43]:

```
1  #Logistic Regression
2  x=np.array(fdf['Price']).reshape(-1,1)
3  y=np.array(fdf['Total_Stops']).reshape(-1,1)
4  fdf.dropna(inplace=True)
5  x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
6  from sklearn.linear_model import LogisticRegression
7  lr=LogisticRegression(max_iter=10000)
```

C:\Users\HP\AppData\Local\Temp\ipykernel_7140\3604832714.py:4: SettingWith
CopyWarning:
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 [44]:

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

C:\Users\HP\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\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_sa
mples,), for example using ravel().
  y = column_or_1d(y, warn=True)

Out[44]:

```
▼           LogisticRegression

LogisticRegression(max_iter=10000)
```

In [45]:

```
1  score=lr.score(x_test,y_test)
2  print(score)
```
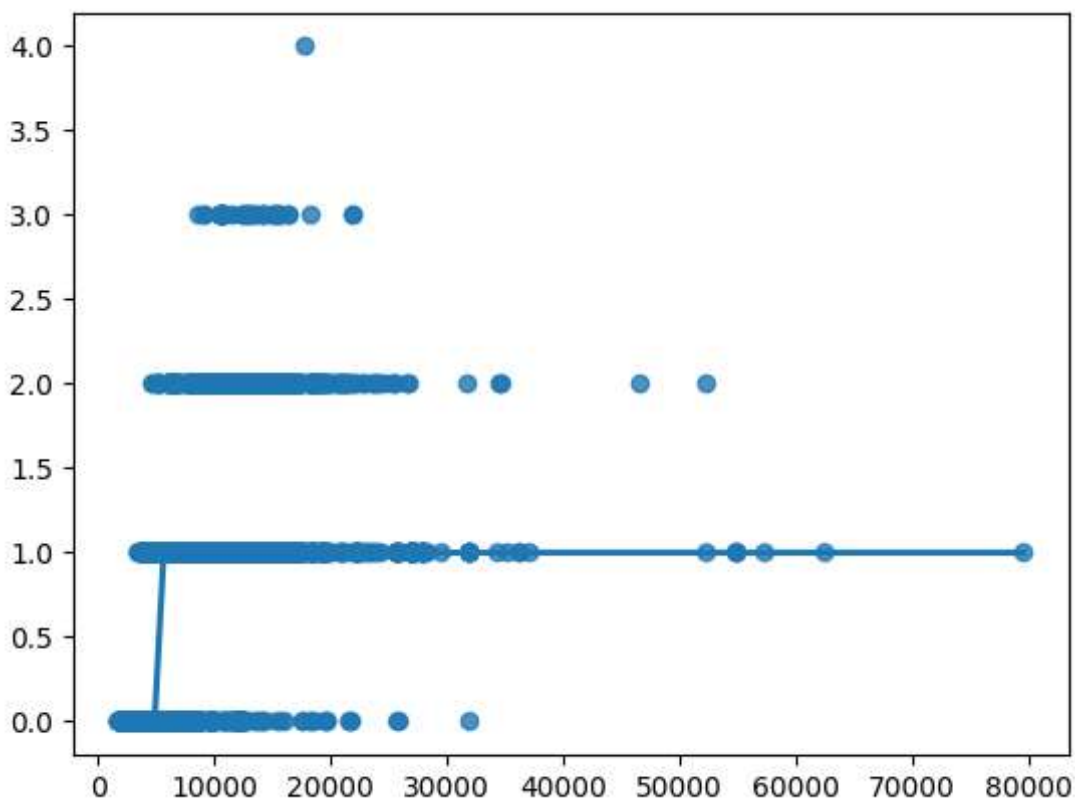
0.7160686427457098

In [46]:

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

C:\Users\HP\AppData\Local\Programs\Python\Python311\Lib\site-packages\stat
smodels\genmod\families\links.py:198: RuntimeWarning: overflow encountered
in exp
  t = np.exp(-z)

Out[46]:

<Axes: >

Decision Tree

In [47]:

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

Out[47]:

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

In [48]:

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

0.9369734789391576

Random Classifier

In [49]:

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

C:\Users\HP\AppData\Local\Temp\ipykernel_7140\1232785509.py:4: DataConvers
ionWarning: A column-vector y was passed when a 1d array was expected. Ple
ase change the shape of y to (n_samples,), for example using ravel().
  rfc.fit(X_train,y_train)

Out[49]:

```
▼ RandomForestClassifier
RandomForestClassifier()
```

In [50]:

```python
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 [51]:

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

In [52]:

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

```
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\HP\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\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\HP\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\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\HP\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\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\HP\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\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 [54]:

```
1  grid_search.best_score_
```

Out[54]:

0.5237394770692444

In [55]:

```
1  rf_best=grid_search.best_estimator_
2  rf_best
```
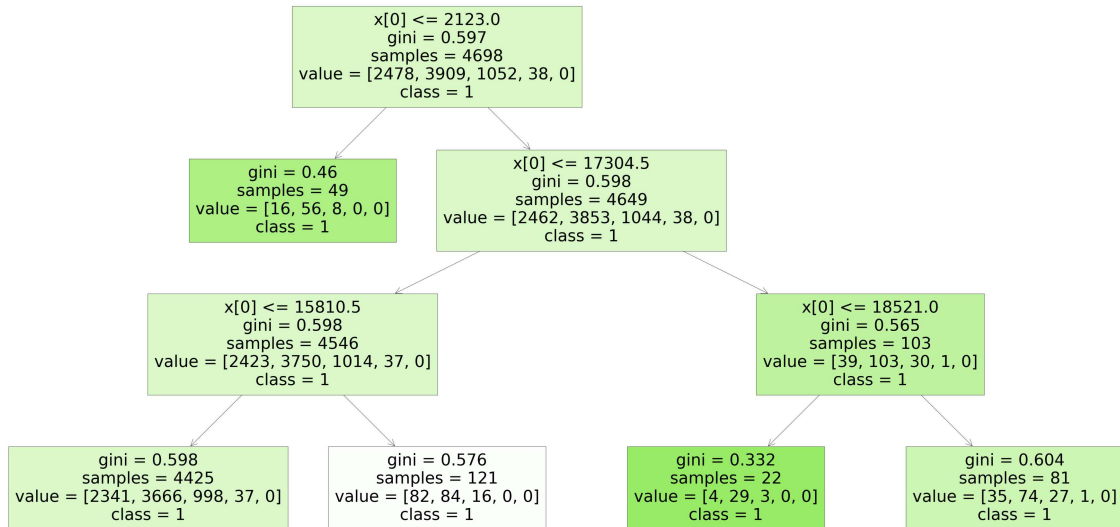
Out[55]:

```
▼                    RandomForestClassifier
RandomForestClassifier(max_depth=3, min_samples_leaf=20, n_estimators=50)
```

In [56]:

```python
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);
```

```
                          x[0] <= 2123.0
                          gini = 0.597
                          samples = 4698
                          value = [2478, 3909, 1052, 38, 0]
                          class = 1
              ┌───────────────────┴───────────────────┐
        gini = 0.46                           x[0] <= 17304.5
        samples = 49                          gini = 0.598
        value = [16, 56, 8, 0, 0]             samples = 4649
        class = 1                             value = [2462, 3853, 1044, 38, 0]
                                              class = 1
                          ┌───────────────────┴───────────────────┐
                   x[0] <= 15810.5                          x[0] <= 18521.0
                   gini = 0.598                             gini = 0.565
                   samples = 4546                           samples = 103
                   value = [2423, 3750, 1014, 37, 0]        value = [39, 103, 30, 1, 0]
                   class = 1                                class = 1
            ┌──────────┴──────────┐                  ┌──────────┴──────────┐
      gini = 0.598         gini = 0.576         gini = 0.332         gini = 0.604
      samples = 4425       samples = 121        samples = 22         samples = 81
      value = [2341,       value = [82, 84,     value = [4, 29,      value = [35, 74,
      3666, 998, 37, 0]    16, 0, 0]            3, 0, 0]             27, 1, 0]
      class = 1            class = 1            class = 1            class = 1
```

In [57]:

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

0.48642745709828394

```
1  # Conclusion:
2  For the given insurance data set have performed linear,logistic,random forest and
   decision tree models of regression and classifications.
3  #conclude that the most accuracy is occured in logistic regression,i.e 71percent
4  #conclude that the Logic Regression model is best fit for given data
```