1)problem Statement:which model is best for Flight Price Prediction Dataset

In [113]:

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

In [114]:

```
m=pd.read_csv(r"C:\Users\chinta pavani\Downloads\Data_Train.csv")
m
```

Out[114]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Dur
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 [115]:

```
s=pd.read_csv(r"C:\Users\chinta pavani\Downloads\Test_set.csv")
s
```

Out[115]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Dura
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL → BOM → COK	17:30	04:25 07 Jun	10h
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
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
...	...	...	...	...	...	...	...	
2666	Air India	6/06/2019	Kolkata	Banglore	CCU → DEL → BLR	20:30	20:25 07 Jun	23h
2667	IndiGo	27/03/2019	Kolkata	Banglore	CCU → BLR	14:20	16:55	2h
2668	Jet Airways	6/03/2019	Delhi	Cochin	DEL → BOM → COK	21:50	04:25 07 Mar	6h
2669	Air India	6/03/2019	Delhi	Cochin	DEL → BOM → COK	04:00	19:15	15h
2670	Multiple carriers	15/06/2019	Delhi	Cochin	DEL → BOM → COK	04:55	19:15	14h

2671 rows × 10 columns

In [116]:

```
m.head()
```

Out[116]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duratior
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 [117]:
```
s.head()
```

Out[117]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duratior
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 [118]:
```
m.tail()
```

Out[118]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Dur
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 [119]:

```
s.tail()
```

Out[119]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durati
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

In [120]:

```
m.describe()
```

Out[120]:

	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 [121]:
```
s.describe()
```

Out[121]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Dur
count	2671	2671	2671	2671	2671	2671	2671	
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 [122]:
```
m.shape
```

Out[122]:
```
(10683, 11)
```

In [123]:
```
s.shape
```

Out[123]:
```
(2671, 10)
```

In [124]:
```
m.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 [125]:

```
s.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 [126]:

```
m.duplicated().sum()
```

Out[126]:

```
220
```

In [127]:

```
s.duplicated().sum()
```

Out[127]:

```
26
```

In [128]:

```
m.columns
```

Out[128]:

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

In [129]:

```
s.columns
```

Out[129]:

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

```
In [130]: m.isnull().sum()

Out[130]: 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 [131]: s.isnull().sum()

Out[131]: 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 [132]: m.dropna(inplace=True)
```

```
In [133]: m.isnull().sum()

Out[133]: 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 [134]: m.shape

Out[134]: (10682, 11)
```

In [157]:

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

Out[157]:

```
Airline
0     3849
1     2053
2     1751
3     1196
4      818
5      479
6      319
7      194
8       13
9        6
10       3
11       1
Name: count, dtype: int64
```

In [158]:

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

Out[158]:

```
Source
0    4536
1    2871
2    2197
3     697
4     381
Name: count, dtype: int64
```

In [160]:

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

Out[160]:

```
Destination
0    4536
1    2871
2    1265
3     932
4     697
5     381
Name: count, dtype: int64
```

In [161]:

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

Out[161]:

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

In [162]:

```
flight={"Airline":{"Jet Airways":0,"IndiGo":1,"Air India":2,"Multiple carr
"SpiceJet":4,"Vistara":5,"Air Asia":6,"GoAir":7,
"Multiple carriers Premium economy":8,
"Jet Airways Business":9,"Vistara Premium economy":10,"Trujet":11}}
m=m.replace(flight)
m
```

Out[162]:

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

10682 rows × 11 columns

In [164]:

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

Out[164]:

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

10682 rows × 11 columns

In [165]:

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

Out[165]:

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

10682 rows × 11 columns

In [166]:

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

Out[166]:

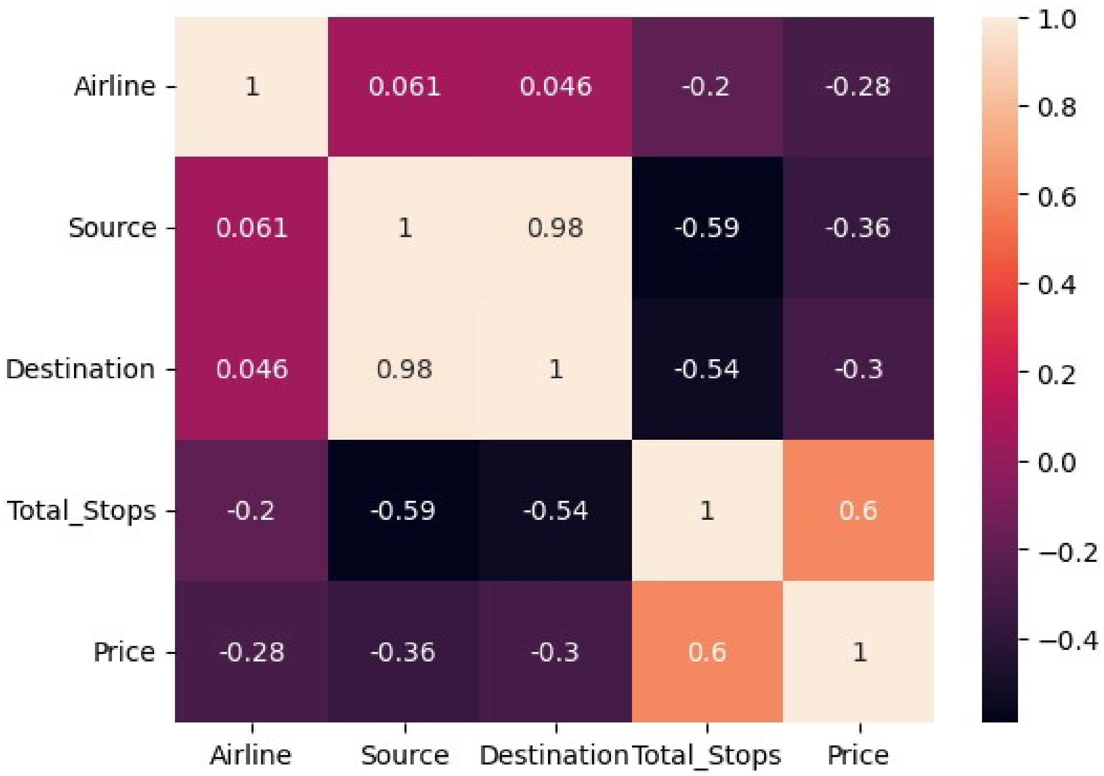
	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	1	24/03/2019	2	3	BLR → DEL	22:20	01:10 22 Mar	2h 5
1	2	1/05/2019	1	1	CCU → IXR → BBI → BLR	05:50	13:15	7h 2
2	0	9/06/2019	0	0	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	
3	1	12/05/2019	1	1	CCU → NAG → BLR	18:05	23:30	5h 2
4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

10682 rows × 11 columns

Data Visualization

In [168]:
```
fdf=m[['Airline','Source','Destination','Total_Stops','Price']]
sns.heatmap(fdf.corr(),annot=True)
```

Out[168]:
```
<Axes: >
```



In [169]:
```
x=fdf[['Airline','Source','Destination','Total_Stops']]
y=fdf['Price']
```

In [170]:
```
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_st
```

Linear Regression

In [171]:

```
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[171]:

	coefficient
Airline	-418.483922
Source	-3275.073380
Destination	2505.480291
Total_Stops	3541.798053

In [172]:

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

```
0.4108304890928348
```

In [173]:

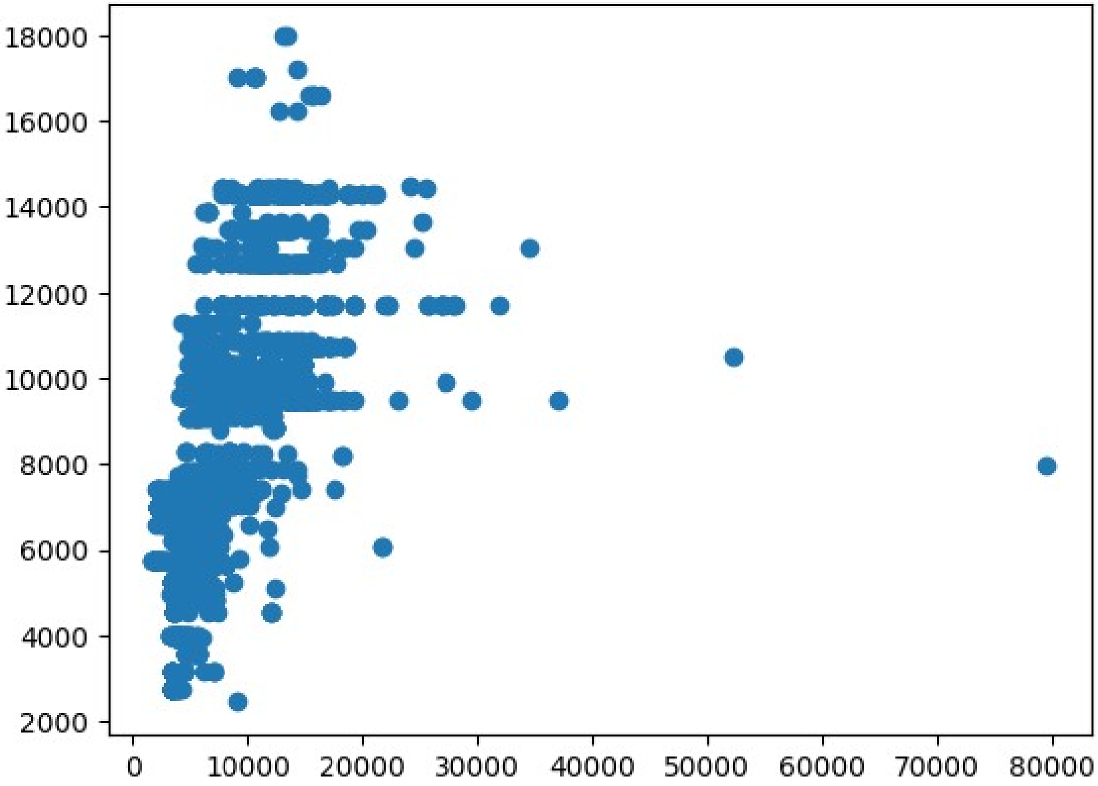
```
predictions=regr.predict(X_test)
```

In [174]:

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

Out[174]:

```
<matplotlib.collections.PathCollection at 0x19b2c0bd2d0>
```



In [175]:

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

```
C:\Users\chinta pavani\AppData\Local\Temp\ipykernel_5260\521034954.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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  fdf.dropna(inplace=True)
```

In [176]:

```
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)
```

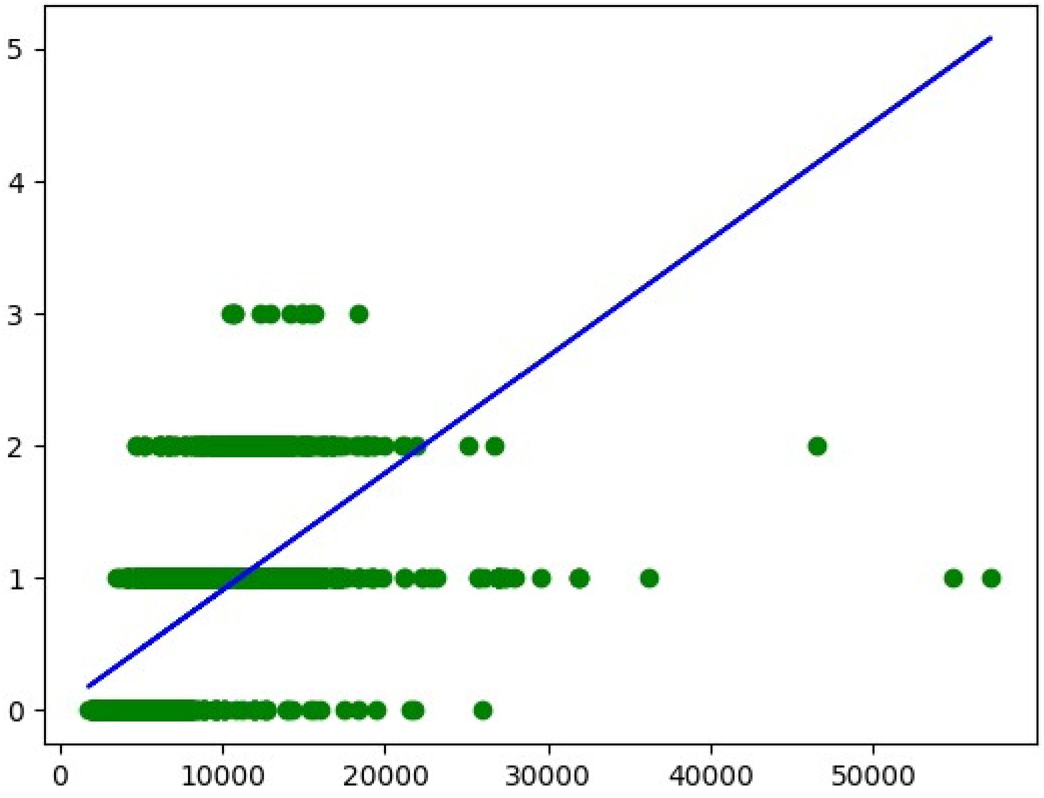
Out[176]:

```
▾ LinearRegression

LinearRegression()
```

In [177]:

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



Logistic Regression

In [178]:

```
#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_st
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(max_iter=10000)
```

```
C:\Users\chinta pavani\AppData\Local\Temp\ipykernel_5260\3604832714.py:4:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  fdf.dropna(inplace=True)
```

In [179]:

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

```
C:\Users\chinta pavani\AppData\Local\Programs\Python\Python311\Lib\site-p
ackages\sklearn\utils\validation.py:1143: DataConversionWarning: A column
-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[179]:

```
▾        LogisticRegression

LogisticRegression(max_iter=10000)
```

In [180]:

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

```
0.7160686427457098
```

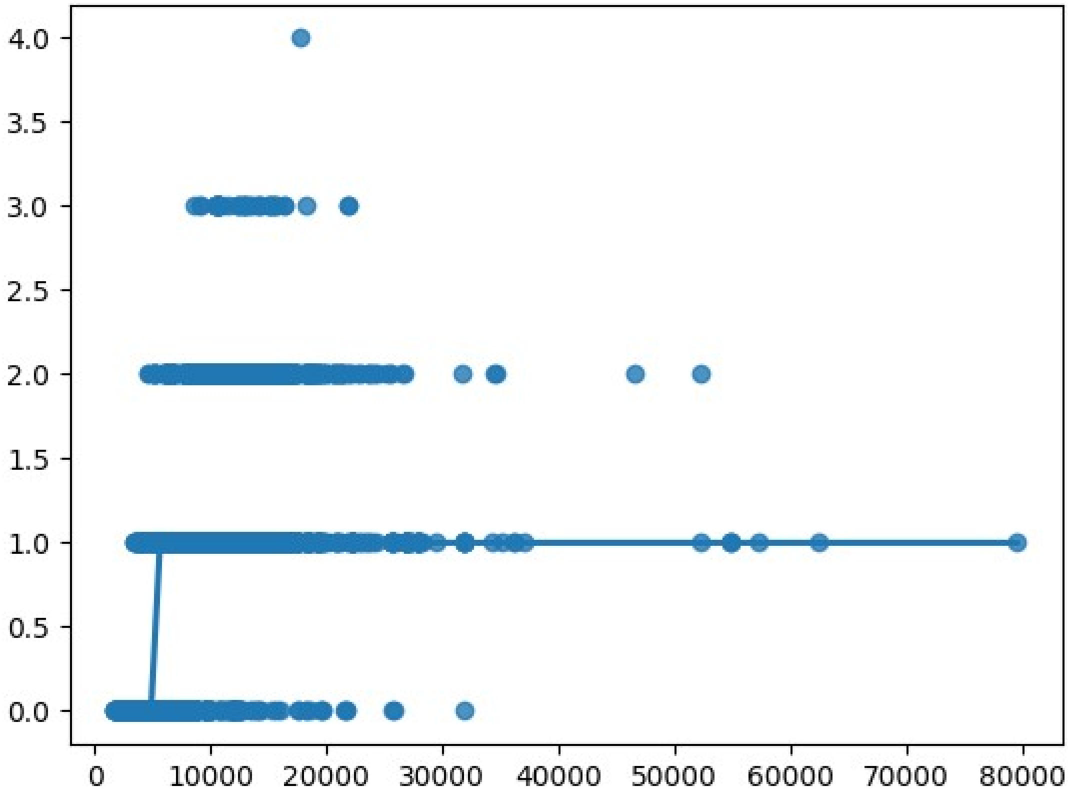
In [181]:

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

```
C:\Users\chinta pavani\AppData\Local\Programs\Python\Python311\Lib\site-p
ackages\statsmodels\genmod\families\links.py:198: RuntimeWarning: overflo
w encountered in exp
  t = np.exp(-z)
```

Out[181]:

```
<Axes: >
```



Decision Tree

In [182]:

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

Out[182]:

```
▾        DecisionTreeClassifier

DecisionTreeClassifier(random_state=0)
```

In [183]:

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

```
0.9369734789391576
```

Random Forest

In [184]:

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

```
C:\Users\chinta pavani\AppData\Local\Temp\ipykernel_5260\1232785509.py:4:
DataConversionWarning: A column-vector y was passed when a 1d array was e
xpected. Please change the shape of y to (n_samples,), for example using
ravel().
  rfc.fit(X_train,y_train)
```

Out[184]:

```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [185]:

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

In [186]:

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

In [187]:

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

```
C:\Users\chinta pavani\AppData\Local\Programs\Python\Python311\Lib\site-p
ackages\sklearn\model_selection\_split.py:700: UserWarning: The least pop
ulated class in y has only 1 members, which is less than n_splits=2.
  warnings.warn(
C:\Users\chinta pavani\AppData\Local\Programs\Python\Python311\Lib\site-p
ackages\sklearn\model_selection\_validation.py:686: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\chinta pavani\AppData\Local\Programs\Python\Python311\Lib\site-p
ackages\sklearn\model_selection\_validation.py:686: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\chinta pavani\AppData\Local\Programs\Python\Python311\Lib\site-p
ackages\sklearn\model_selection\_validation.py:686: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

In [188]:

```
grid_search.best_score_
```

Out[188]:

```
0.5245420452875429
```

In [189]:

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

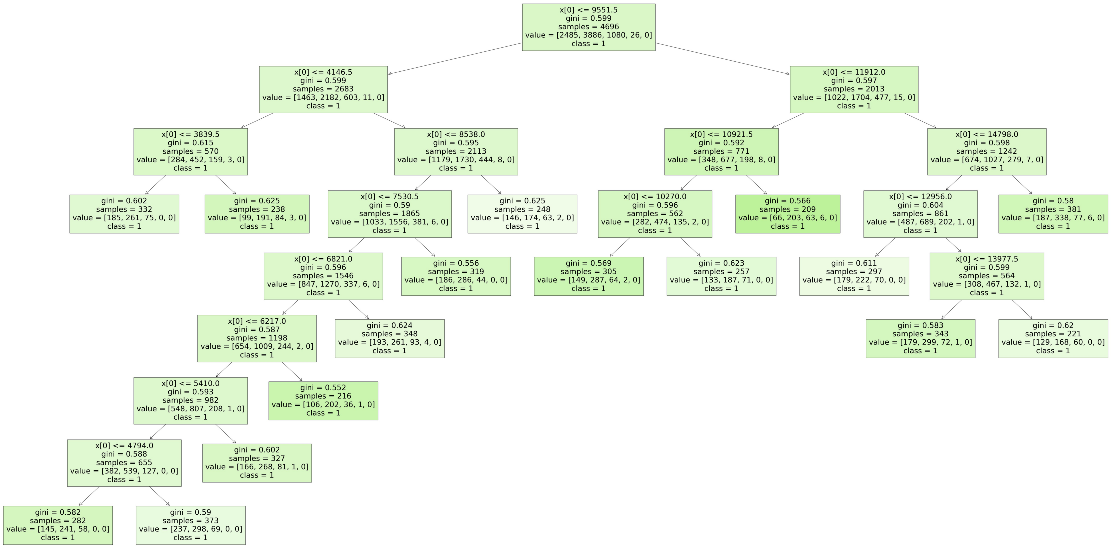
Out[189]:

```
▾                       RandomForestClassifier

RandomForestClassifier(max_depth=20, min_samples_leaf=200, n_estimators=
25)
```

In [190]:

```
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=
```



In [191]:

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

```
0.4627145085803432
```

Conclusion:

In [ ]:

```
*Here when we compare between Decision Tree and Random Forest, we
 can confirm that Decision Tree has more accuracy than Random Forest
 which makesit 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 casesRandom Forest p
* Based on accuracy scores of all models that were
 implemented we can conclude that"Decision Tree" is the best model for
 the given dataset
```