Problem Statement:predicting which model best suits the dataset

1.Data Collection

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
traindf=pd.read_csv(r"C:\Users\DELL\Downloads\Data_Train.csv")
traindf
```

Out[1]:

	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 [2]:
```
testdf=pd.read_csv(r"C:\Users\DELL\Downloads\Test_set.csv")
testdf
```

Out[2]:

	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 [3]:
```
traindf.head()
```

Out[3]:

	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 [4]:
```
testdf.head()
```

Out[4]:

	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 [5]:
```python
traindf.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]:
```python
testdf.tail()
```

Out[6]:

	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 [7]:
```
traindf.describe()
```
Out[7]:

	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 [8]:
```
testdf.describe()
```
Out[8]:

	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 [9]:
```
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 [10]: 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 [11]: traindf.shape
```

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

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

```
Out[12]: (2671, 10)
```

```
In [13]: traindf.duplicated().sum()
```

```
Out[13]: 220
```

```
In [14]: testdf.duplicated().sum()
```

```
Out[14]: 26
```

```
In [15]: traindf.isnull().sum()
```

```
Out[15]: 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 [16]:
```
testdf.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
dtype: int64
```
In [17]:
```
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]:
```
testdf.columns
```
Out[18]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info'],
      dtype='object')
```
In [19]:
```
traindf.dropna(inplace=True)
```
In [20]:
```
traindf.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]:
```
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
         "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	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 [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	Duration
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 [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	Duration
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 [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	Duration
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]:
```
traindf
```

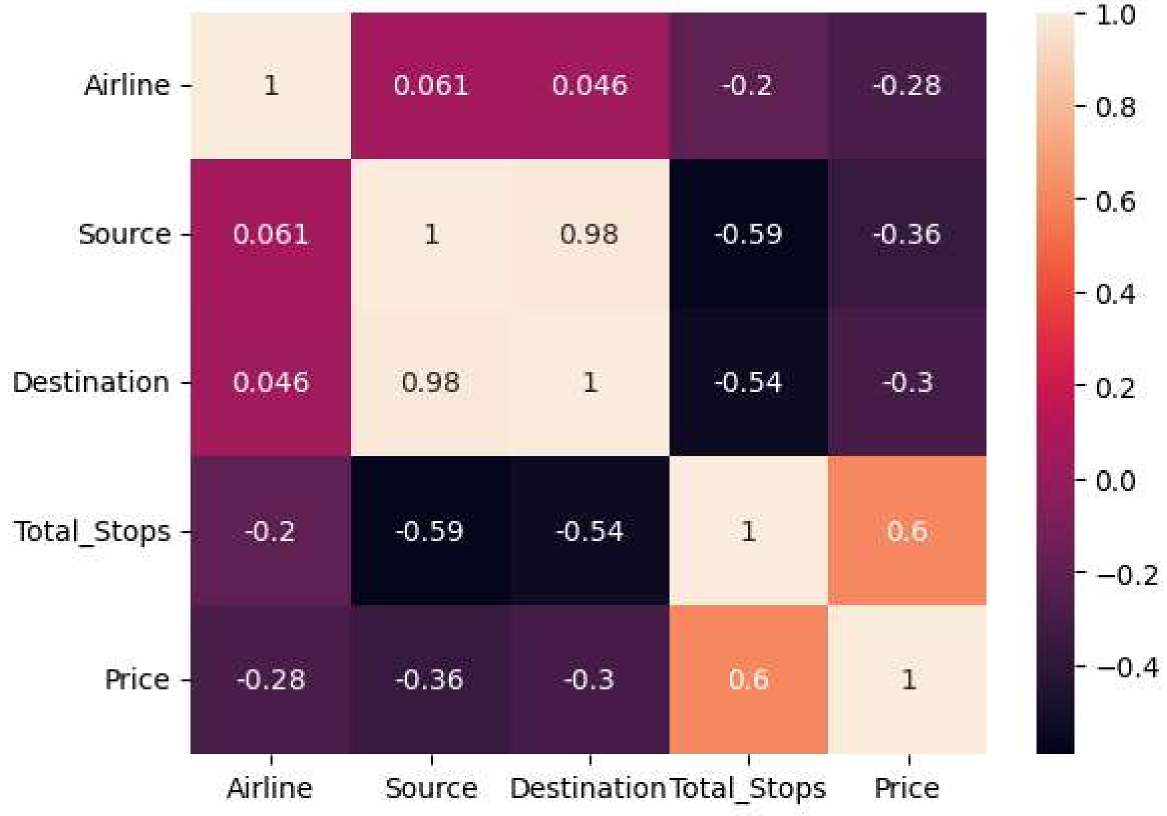
Out[29]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration
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]:
```
#EDA
fdf=traindf[['Airline','Source','Destination','Total_Stops','Price']]
sns.heatmap(fdf.corr(),annot=True)
```

Out[30]:
```
<Axes: >
```



In [31]:
```
x=fdf[['Airline','Source','Destination','Total_Stops']]
y=fdf['Price']
```

1.Linear Regression

In [32]:
```
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 [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

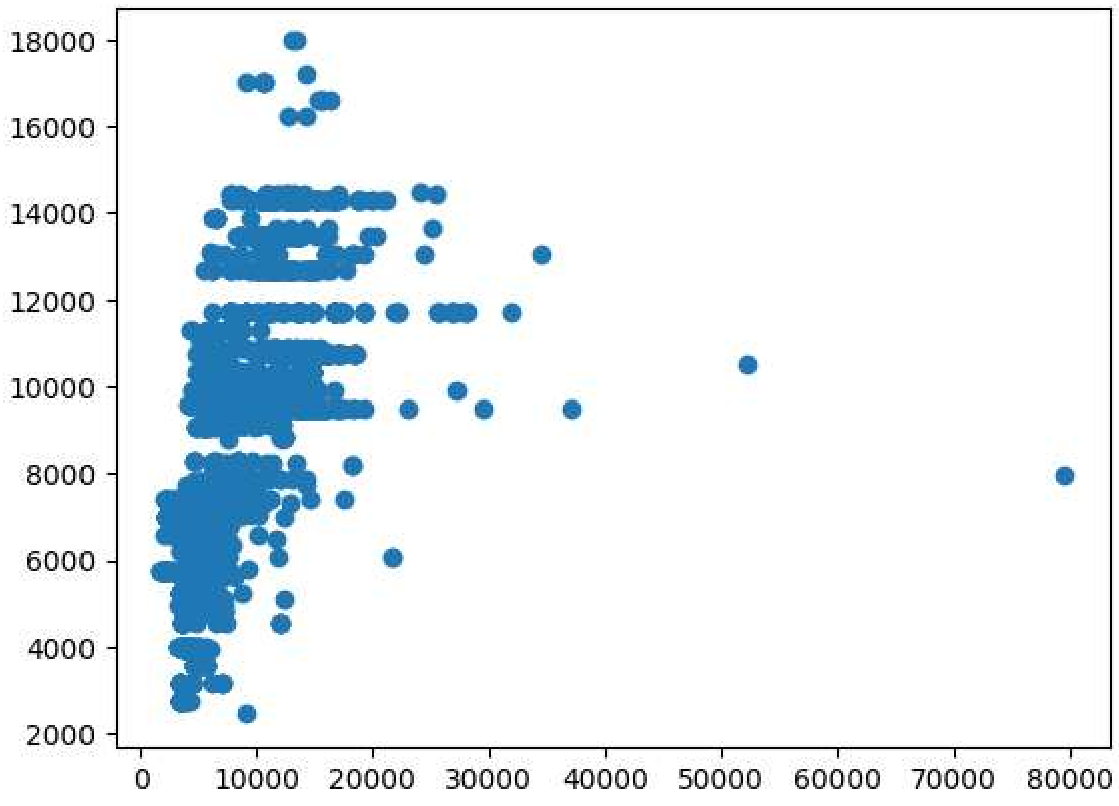
In [34]:
```
score=regr.score(X_test,y_test)
print(score)
```

```
0.41083048909283415
```

In [35]:
```
predictions=regr.predict(X_test)
```

In [36]:
```
plt.scatter(y_test,predictions)
```

Out[36]:
```
<matplotlib.collections.PathCollection at 0x1e6ce13f250>
```



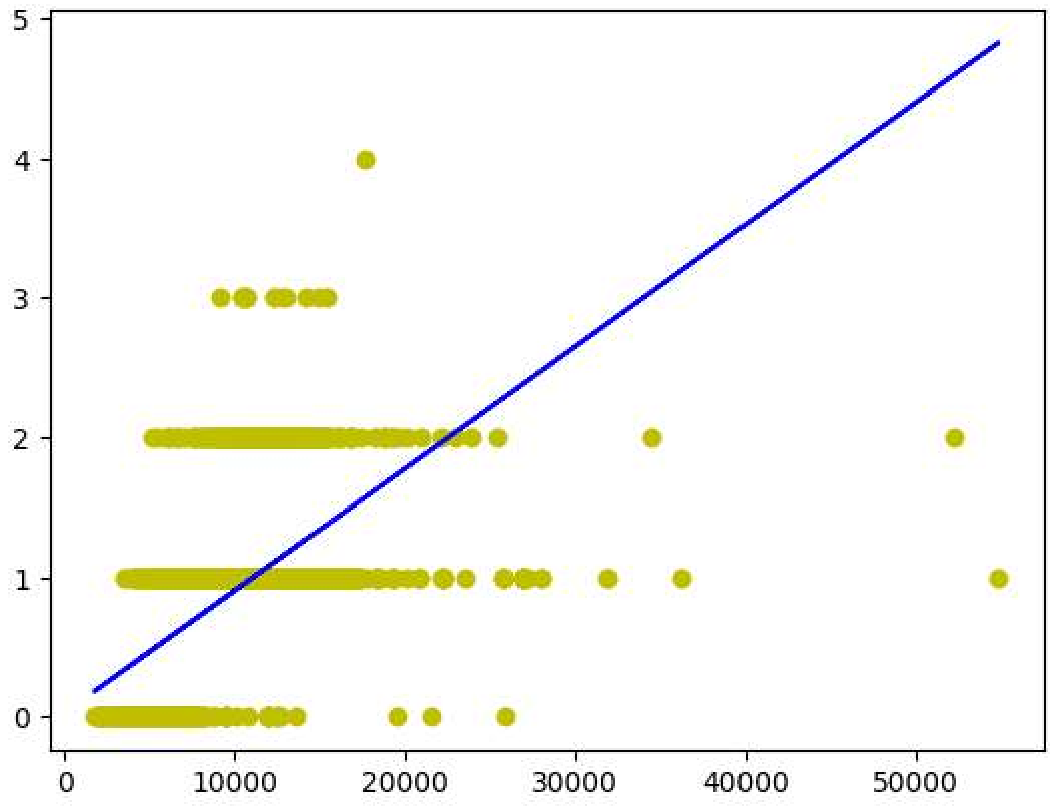
In [37]:
```
x=np.array(fdf['Price']).reshape(-1,1)
y=np.array(fdf['Total_Stops']).reshape(-1,1)
fdf.dropna(inplace=True)
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_8956\521034954.py:3: SettingWithCo
pyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-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()
```



In [54]:
```
from sklearn.metrics import r2_score
lr=LinearRegression()
r2=r2_score(y_pred,y_test)
print(r2)
```

```
-2.951376774869263
```

2.Logistic Regression

In [40]:
```
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_8956\497261869.py:3: SettingWithCo
pyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  fdf.dropna(inplace=True)
```

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

```
C:\Users\DELL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklea
rn\utils\validation.py:1143: DataConversionWarning: A column-vector y was pas
sed 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)
```

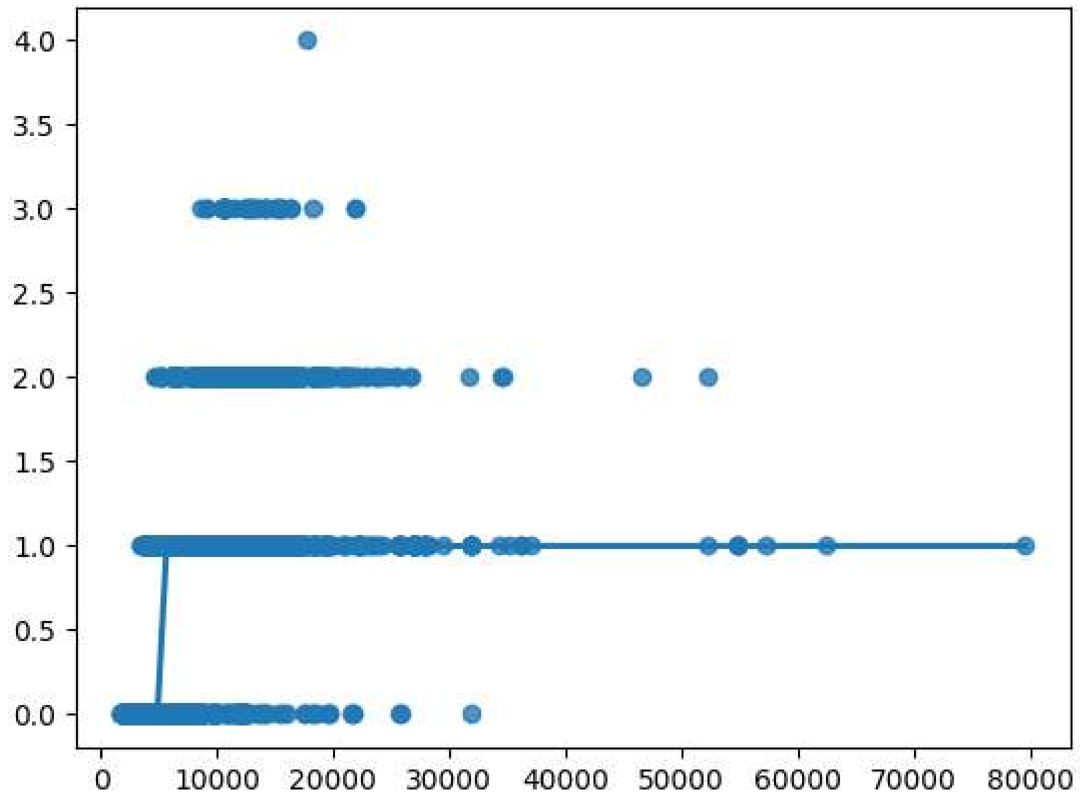
```
0.7160686427457098
```

In [ ]:
```
#conclusion:this model has 70% of accuracy.
```

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

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

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



3.Decision Trees

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
```

```
In [ ]: #conclusion:This model has 90% of accuracy.
```

4.Random Forest

```
In [46]: #Random forest classifier
         from sklearn.ensemble import RandomForestClassifier
         rfc=RandomForestClassifier()
         rfc.fit(X_train,y_train)
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_8956\1232785509.py:4: DataConversi
onWarning: 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]: 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
         grid_search.fit(X_train,y_train)
```

```
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\skl
earn\model_selection\_validation.py:686: DataConversionWarning: A column-ve
ctor 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\skl
earn\model_selection\_validation.py:686: DataConversionWarning: A column-ve
ctor 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\skl
earn\model_selection\_validation.py:686: DataConversionWarning: A column-ve
ctor 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\skl
earn\model_selection\_validation.py:686: DataConversionWarning: A column-ve
```

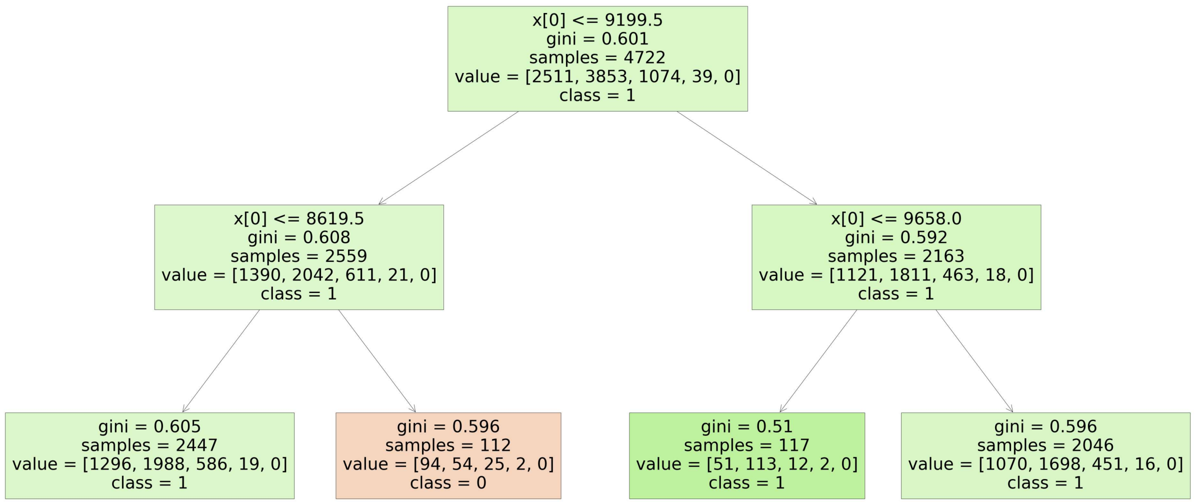
```
In [49]: grid_search.best_score_
```

```
Out[49]: 0.523605715699528
```

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

Out[50]:
```
▾                    RandomForestClassifier

RandomForestClassifier(max_depth=2, min_samples_leaf=5, n_estimators=25)
```

In [51]:
```
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 [52]:
```
score=rfc.score(x_test,y_test)
print(score)
```

```
0.4527301092043682
```

In [ ]:
```
#conclusion:This model has 40% accuracy.
```

Conclusion

```
    For the above Dataset we use different Types of
Models,For that each and every model we get different
Types of Accuracies.Based on that accuracies we can
conclude which model is best fit for my our
Dataset.
Decision Trees model will best suits the dataset.Beacuse
among all it has highest accuracy rate 90%.
```

```
In [ ]:
```