ProblemStatement: Which model is suitable(bestfit) for the given dataset

In [1]:

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

DataCollection

In [2]:

```
traindf=pd.read_csv(r"C:\Users\kunam\Downloads\Data_Train.csv")
traindf
```

Out[2]:

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

10683 rows × 11 columns

In [3]:

```
testdf=pd.read_csv(r"C:\Users\magam\Downloads\Test_set26.csv")
testdf
```

Out[3]:

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

2671 rows × 10 columns

Data preprocessing

In [4]:

```
traindf.describe()
```

Out[4]:

	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 [5]:

```
testdf.describe()
```

Out[5]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	D
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	
freq	897	144	1145	1145	624	62	113	

In [6]:

```
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 [7]:

```
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 [8]:

```
traindf.columns
```

Out[8]:

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

In [9]:

```
testdf.columns
```

Out[9]:

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

In [10]:

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

Out[10]:

```
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 [11]:

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

In [12]:

```
testdf.isnull().sum()
```

Out[12]:

```
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 [13]:

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

Out[13]:

```
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 [14]:

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

Out[14]:

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

In [15]:

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

Out[15]:

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

In [16]:

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

Out[16]:

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

In [17]:

```
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[17]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Du
0	1	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	
1	2	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	
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	
4	1	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	
...	...	...	...	...	...	...	...	
10678	6	9/04/2019	Kolkata	Banglore	CCU ? BLR	19:55	22:25	
10679	2	27/04/2019	Kolkata	Banglore	CCU ? BLR	20:45	23:20	
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	
10682	2	9/05/2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	

10682 rows × 11 columns

In [18]:

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

Out[18]:

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

10682 rows × 11 columns

In [19]:

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

Out[19]:

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

10682 rows × 11 columns

In [20]:

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

Out[20]:

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

10682 rows × 11 columns

In [21]:

```
traindf
```

Out[21]:

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

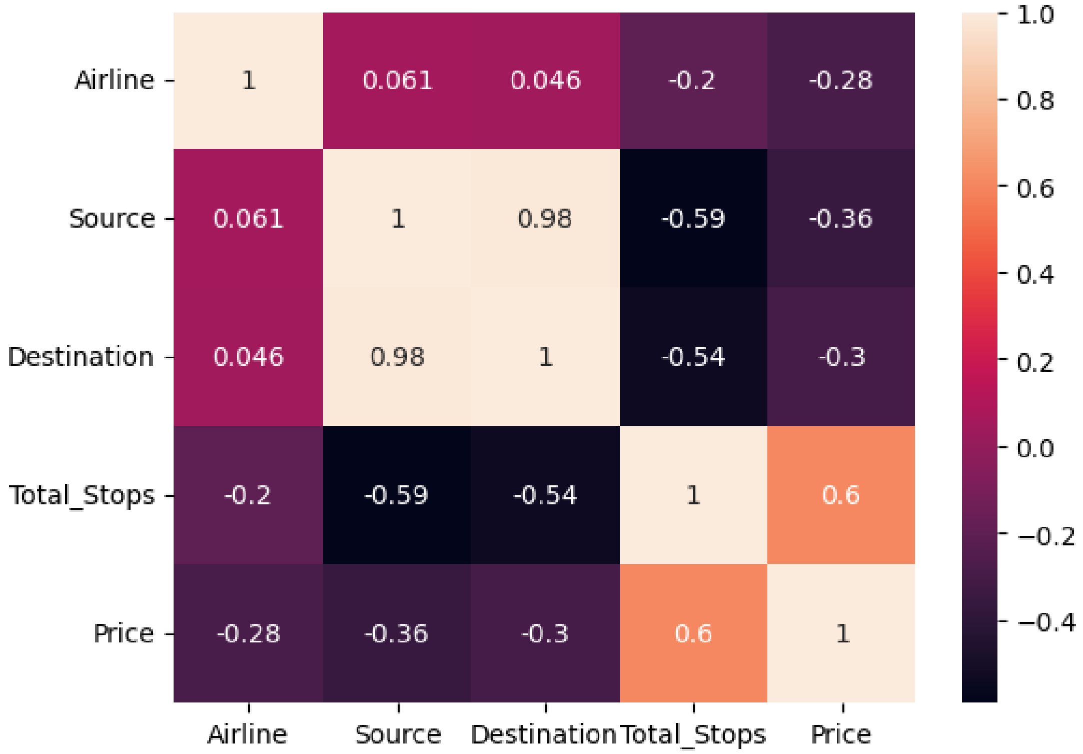
10682 rows × 11 columns

In [22]:

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

Out[22]:

```
<Axes: >
```



In [23]:

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

LINEAR REGRESSION

In [24]:

```
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 [25]:

```
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[25]:

	coefficient
Airline	-418.483922
Source	-3275.073380
Destination	2505.480291
Total_Stops	3541.798053

In [26]:

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

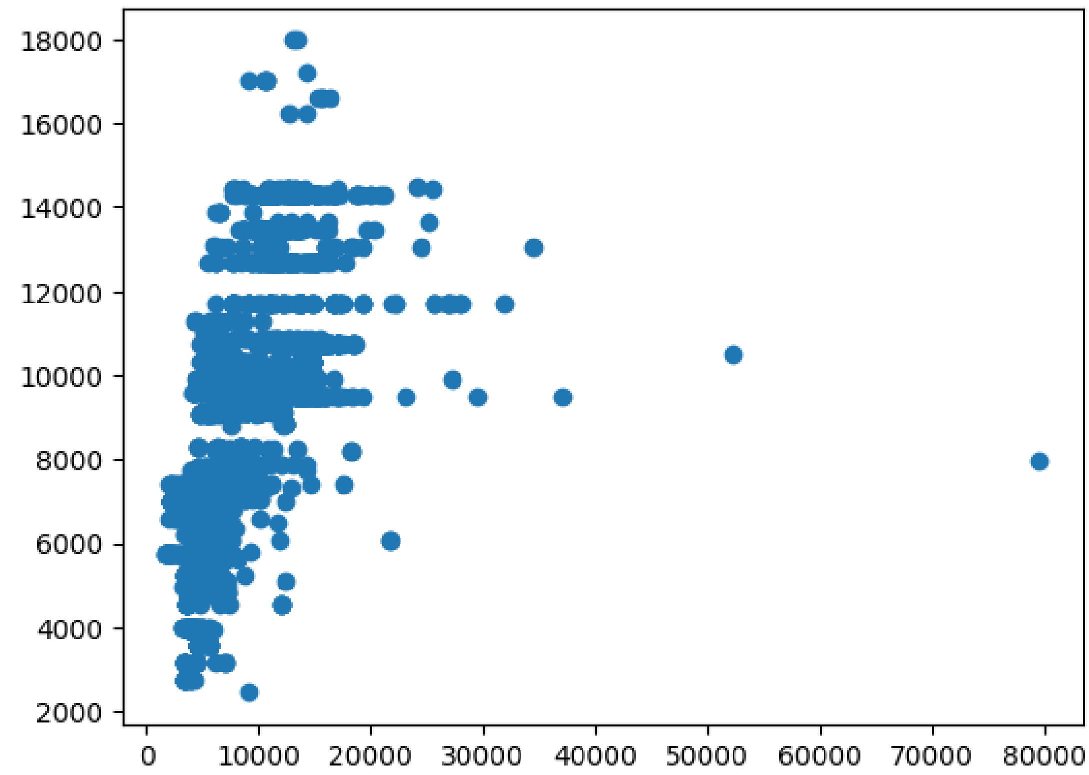
```
0.4108304890928348
```

In [27]:

```
predictions=regr.predict(X_test)
plt.scatter(y_test,predictions)
```

Out[27]:

```
<matplotlib.collections.PathCollection at 0x19bfe0cb710>
```



In [47]:

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

In [48]:

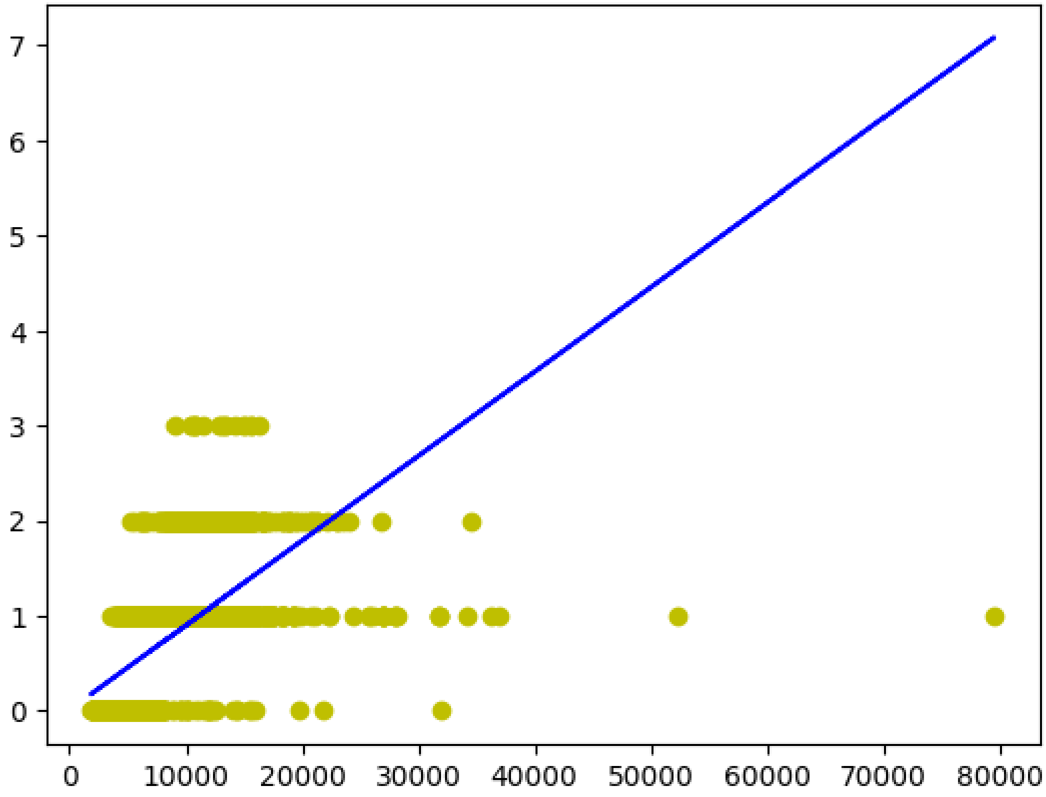
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

Out[48]:

```
▾ LinearRegression
LinearRegression()
```

In [49]:

```
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 [53]:

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

In [55]:

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

```
C:\Users\magam\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\utils\validation.py:1143: DataConversionWarning: A column-vec
tor 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[55]:

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

In [56]:

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

```
0.7160686427457098
```

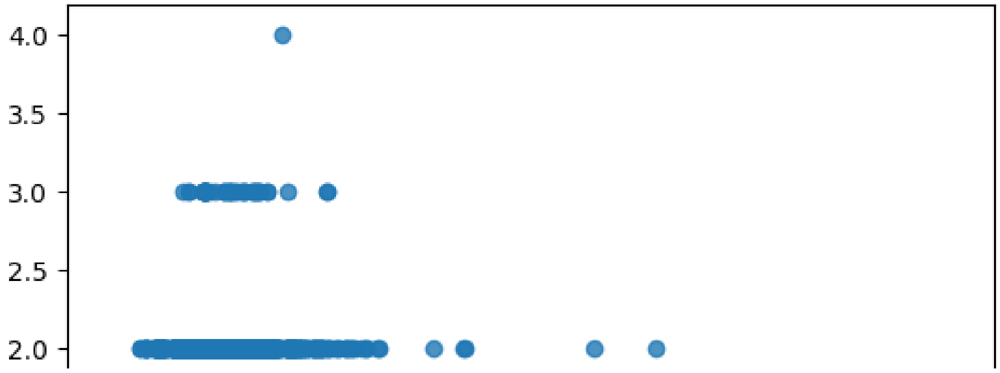
In [57]:

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

```
C:\Users\magam\AppData\Local\Programs\Python\Python311\Lib\site-pa
ckages\statsmodels\genmod\families\links.py:198: RuntimeWarning: o
verflow encountered in exp
  t = np.exp(-z)
```

Out[57]:

```
<Axes: >
```



Decision Tree

In [35]:

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

Out[35]:

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

In [58]:

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

```
0.9369734789391576
```

Random Classifier

In [37]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(X_train,y_train)
```

```
C:\Users\magam\AppData\Local\Temp\ipykernel_17416\4104924521.py:3: Data
ConversionWarning: A column-vector y was passed when a 1d array was exp
ected. Please change the shape of y to (n_samples,), for example using
ravel().
  rfc.fit(X_train,y_train)
```

Out[37]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [38]:

```
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 [39]:

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

In [40]:

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

```
ckages\sklearn\model_selection\_validation.py:686: DataConversionW
arning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ra
vel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\magam\AppData\Local\Programs\Python\Python311\Lib\site-pa
ckages\sklearn\model_selection\_validation.py:686: DataConversionW
arning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ra
vel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\magam\AppData\Local\Programs\Python\Python311\Lib\site-pa
ckages\sklearn\model_selection\_validation.py:686: DataConversionW
arning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ra
vel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\magam\AppData\Local\Programs\Python\Python311\Lib\site-pa
ckages\sklearn\model_selection\_validation.py:686: DataConversionW
arning: A column-vector y was passed when a 1d array was expected.
```

In [41]:

```
grid_search.best_score_
```

Out[41]:

```
0.523605715699528
```

In [42]:

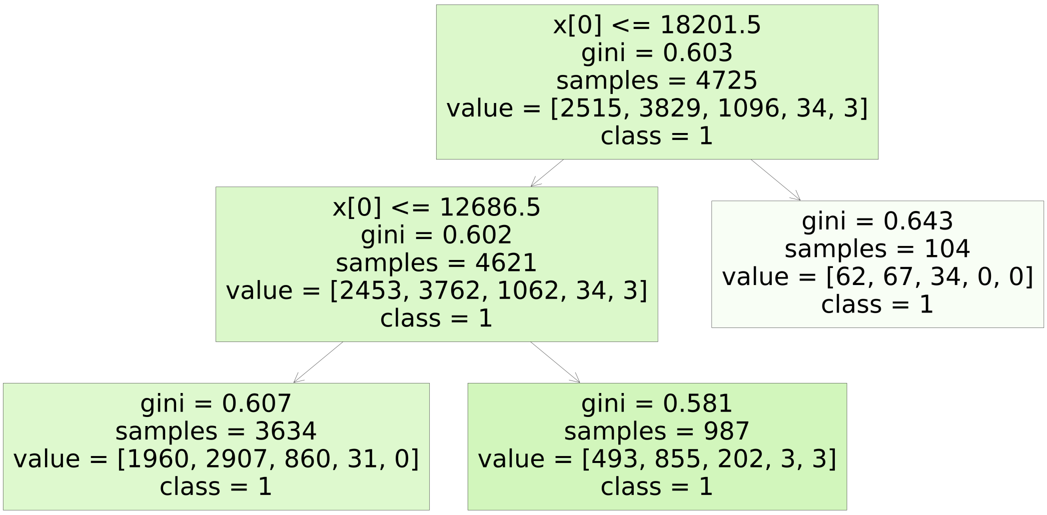
```
rf_best=grid_search.best_estimator_
rf_best
```

Out[42]:

```
▾ RandomForestClassifier
RandomForestClassifier(max_depth=2, min_samples_leaf=100, n_estimators
=10)
```

In [43]:

```
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 [44]:

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

```
0.46084243369734790
```

Conclusion

```
I analysed the data with LinearRegression, logisticRegression, DecissionTree,
RandomForest models.
I get 41% for Linear , 71% for Logistic , 93% for DecissionTree and 46% for
Randomforest.
so, I conclude that DecissionTree model is the bestfit model of remaining.
```