Problem Statement : Which model is suitable for Flight Price Prediction

In [ ]:

```
#imporing libraries and reading the dataa
```

In [2]:

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

In [3]:

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

Out[3]:

	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 [4]:

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

Out[4]:

	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 [5]:

```
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]:

```
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]:

```
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]:

```
traindf.shape
```

Out[8]:

```
(10683, 11)
```

In [9]:

```
traindf.columns
```

Out[9]:

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

In [10]:

```
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 [11]:

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

Out[11]:

```
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 [12]:

```
traindf.dropna(inplace=True)
traindf.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
Price              0
dtype: int64
```

In [13]:

```
traindf.shape
```

Out[13]:

```
(10682, 11)
```

In [14]:

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

Out[14]:

```
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: Airline, dtype: int64
```

In [15]:

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

Out[15]:

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

In [16]:

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

Out[16]:

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

In [17]:

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

Out[17]:

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

In [18]:

```
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[18]:

	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

```
# Conversion of datatype of values from String toNumerical Values
```

In [19]:

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

Out[19]:

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

10682 rows × 11 columns

In [20]:

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

Out[20]:

	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 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	19
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 [21]:

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

Out[21]:

	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 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	19
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 [22]:

```
traindf
```

Out[22]:

	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 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	19
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

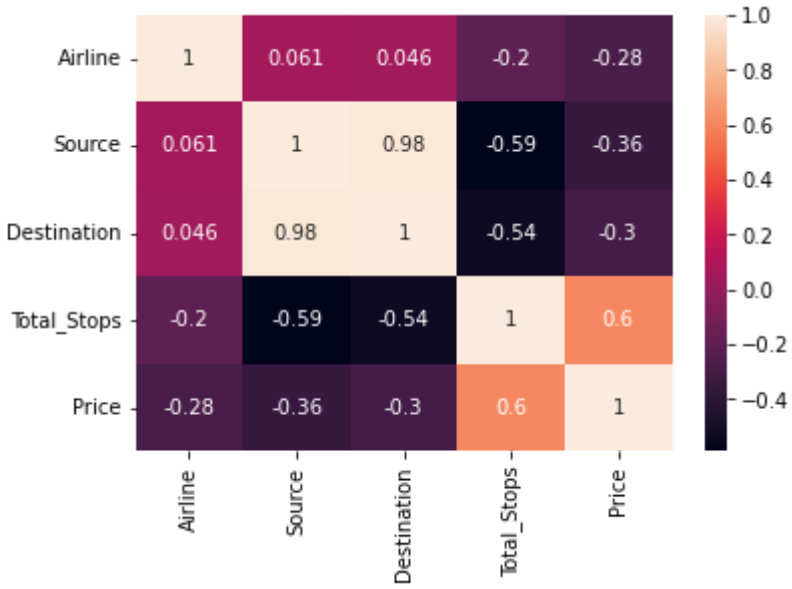
```
# Data Visualization
```

In [23]:

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

Out[23]:

<AxesSubplot:>



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

In [25]:

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

Linear Regression

In [26]:

```
#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)
from sklearn.linear_model import LinearRegression
regr=LinearRegression()
regr.fit(x_train,y_train)
#print(regr.intercept_)
```

Out[26]:

LinearRegression()

In [27]:

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

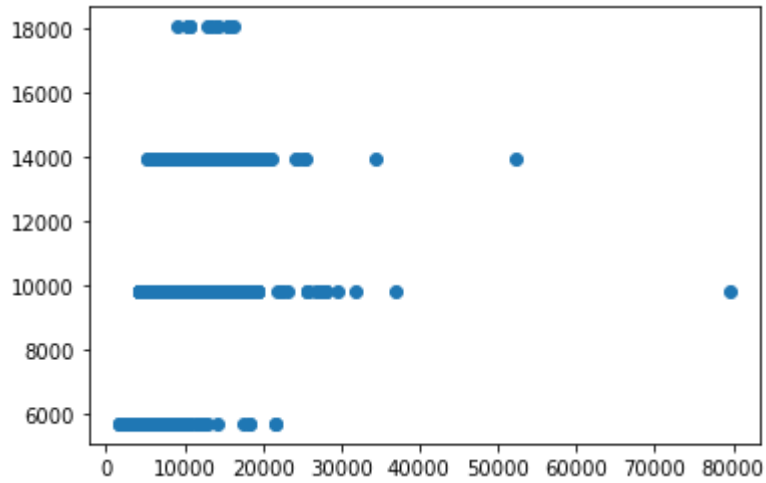
```
0.3787899241988959
```

In [28]:

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

Out[28]:

```
<matplotlib.collections.PathCollection at 0x234974e1c70>
```



In [29]:

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

```
C:\Users\shaik\AppData\Local\Temp\ipykernel_15036\521034954.py:3: SettingW
ithCopyWarning:
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 [30]:

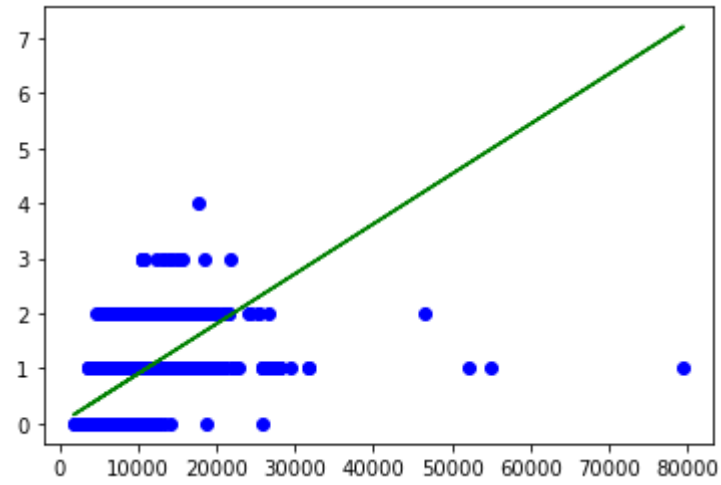
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
```

Out[30]:

```
LinearRegression()
```

In [31]:

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



In [32]:

```
"""we did not get the accuracy for LinearRegression
                          we are going to implement LogisticRegression"""
```

Out[32]:

```
'we did not get the accuracy for LinearRegression \n
we are going to implement LogisticRegression'
```

```
    # Logistic Regression
```

In [33]:

```
#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\shaik\AppData\Local\Temp\ipykernel_15036\3604832714.py:4: Setting
WithCopyWarning:
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 [34]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:99
3: 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[34]:

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

In [35]:

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

```
0.7160686427457098
```

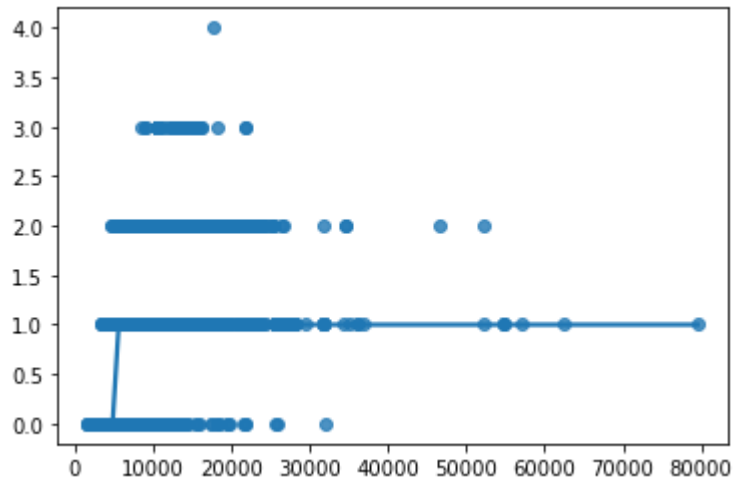
In [36]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\genmod\families\lin
ks.py:187: RuntimeWarning: overflow encountered in exp
  t = np.exp(-z)
```

Out[36]:

```
<AxesSubplot:>
```



In [37]:

```
"""we did not get the accuracy for LogisticRegression
         we are going to implement Decision Tree and Random Forest and make a comparati
                                                              for finding the best model f
```

Out[37]:

```
'we did not get the accuracy for LogisticRegression \n          we are goi
ng to implement Decision Tree and Random Forest and make a comparative stu
dy\n                                                              for findin
g the best model for the dataset'
```

Decision tree

In [38]:

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

Out[38]:

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

In [39]:

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

```
0.9369734789391576
```

Random forest classifier

In [40]:

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

```
C:\Users\shaik\AppData\Local\Temp\ipykernel_15036\1232785509.py:4: DataCon
versionWarning: 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[40]:

```
RandomForestClassifier()
```

In [41]:

```
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 [42]:

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

In [43]:

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

```
n a 1d array was expected. Please change the shape of y to (n_sample
s,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_val
idation.py:680: DataConversionWarning: A column-vector y was passed whe
n a 1d array was expected. Please change the shape of y to (n_sample
s,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_val
idation.py:680: DataConversionWarning: A column-vector y was passed whe
n a 1d array was expected. Please change the shape of y to (n_sample
s,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_val
idation.py:680: DataConversionWarning: A column-vector y was passed whe
n a 1d array was expected. Please change the shape of y to (n_sample
s,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_val
idation.py:680: DataConversionWarning: A column-vector y was passed whe
```

In [44]:

```
grid_search.best_score_
```

Out[44]:

```
0.523605715699528
```

In [45]:

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

Out[45]:

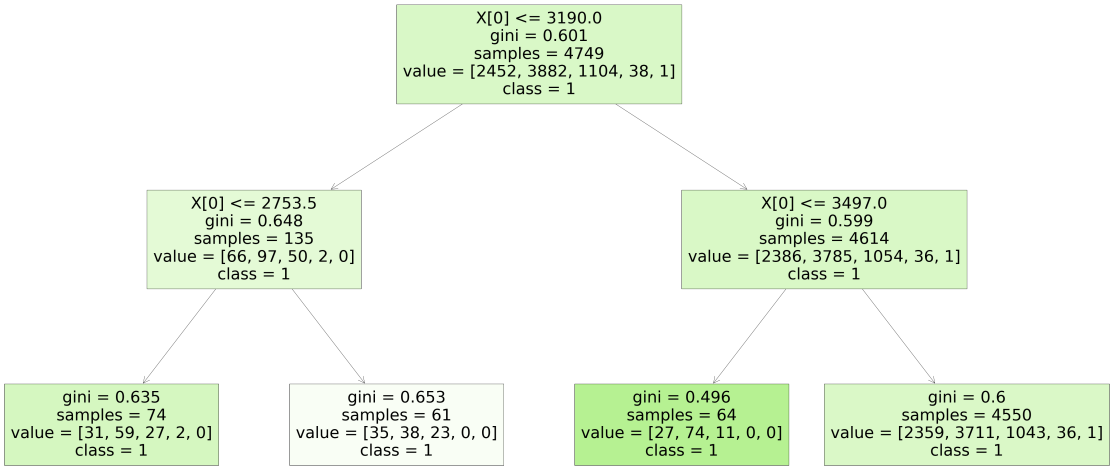
```
RandomForestClassifier(max_depth=2, min_samples_leaf=5, n_estimators=10)
```

In [46]:

```
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)
```

Out[46]:

```
[Text(0.5, 0.8333333333333334, 'X[0] <= 3190.0\ngini = 0.601\nsamples = 47
49\nvalue = [2452, 3882, 1104, 38, 1]\nclass = 1'),
 Text(0.25, 0.5, 'X[0] <= 2753.5\ngini = 0.648\nsamples = 135\nvalue = [6
6, 97, 50, 2, 0]\nclass = 1'),
 Text(0.125, 0.16666666666666666, 'gini = 0.635\nsamples = 74\nvalue = [3
1, 59, 27, 2, 0]\nclass = 1'),
 Text(0.375, 0.16666666666666666, 'gini = 0.653\nsamples = 61\nvalue = [3
5, 38, 23, 0, 0]\nclass = 1'),
 Text(0.75, 0.5, 'X[0] <= 3497.0\ngini = 0.599\nsamples = 4614\nvalue = [2
386, 3785, 1054, 36, 1]\nclass = 1'),
 Text(0.625, 0.16666666666666666, 'gini = 0.496\nsamples = 64\nvalue = [2
7, 74, 11, 0, 0]\nclass = 1'),
 Text(0.875, 0.16666666666666666, 'gini = 0.6\nsamples = 4550\nvalue = [23
59, 3711, 1043, 36, 1]\nclass = 1')]
```



In [47]:

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

```
0.46146645865834635
```

In [ ]:

```
"""compare between Decision Tree andRandom Forest,
            we can confirm that Decision Treehas more accuracy than Random Forest which
                                                                it the best model for this
```

CONCLUSION :

```
   #Based on accuracy scores of allmodels that were implemented we can conclude that,
```

```
                                "Decision Tree" is the best model for the
givendataset.
```

In [ ]:

```

```