PROBLEM STATEMENT:

importing required libraries

In [6]:
```
import numpy as np
import pandas as pd
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="white")
#seaborn plots
sns.set(style="whitegrid",color_codes=True)
import warnings
warnings.simplefilter (action='ignore')
```

In [17]:
```
traindf=pd.read_csv(r"C:\Users\Mastan Reddy\Downloads\Data_Train 1.csv")
train_df
```

Out[17]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	To
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 [8]:
```
testdf=pd.read_csv(r"C:\Users\Mastan Reddy\Downloads\Test_set.csv")
testdf
```

Out[8]:

	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	
...	...	...	...	...	...	...	...	...	
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 [9]:
```
testdf.head()
```

Out[9]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_S
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL ? BOM ? COK	17:30	04:25 07 Jun	10h 55m	1
1	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? MAA ? BLR	06:20	10:20	4h	1
2	Jet Airways	21/05/2019	Delhi	Cochin	DEL ? BOM ? COK	19:15	19:00 22 May	23h 45m	1
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL ? BOM ? COK	08:00	21:00	13h	1
4	Air Asia	24/06/2019	Banglore	Delhi	BLR ? DEL	23:55	02:45 25 Jun	2h 50m	nor

In [18]:
```
traindf.head()
```

Out[18]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_S
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	nor
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1

In [19]:
```
testdf.tail()
```

Out[19]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total
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	n
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 [20]:
```
traindf.describe()
```

Out[20]:

	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 [21]:
```
testdf.describe()
```

Out[21]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	To
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 [22]:
```
traindf.shape
```

Out[22]:
```
(10683, 11)
```

In [23]:
```
testdf.shape
```

Out[23]:
```
(2671, 10)
```

In [24]:
```
traindf.columns
```

Out[24]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price'],
      dtype='object')
```

In [25]:
```
testdf.columns
```

Out[25]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info'],
      dtype='object')
```

In [26]:
```
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 [27]:
```
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
```

To Find out any null or Duplicate values in DataSet

In [28]:
```
traindf.isnull().sum()
```

Out[28]:
```
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 [29]:
```
testdf.isnull().sum()
```

Out[29]:
```
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
```

To Remove Null values in DataSet

In [30]:
```
traindf.dropna(inplace=True)
```

In [31]:
```
traindf.isnull().sum()
```

Out[31]:
```
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 [32]:
```
traindf.shape
```

Out[32]:
```
(10682, 11)
```

Replacing the String values to Numerical values in given DataSet

In [33]:
```
traindf['Airline'].value_counts()
```

Out[33]:
```
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 [34]:
```
traindf['Destination'].value_counts()
```

Out[34]:
```
Cochin       4536
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: Destination, dtype: int64
```

In [35]:
```
traindf['Source'].value_counts()
```

Out[35]:
```
Delhi       4536
Kolkata     2871
Banglore    2197
Mumbai       697
Chennai      381
Name: Source, dtype: int64
```

In [36]:
```
traindf['Total_Stops'].value_counts()
```

Out[36]:
```
1 stop      5625
non-stop    3491
2 stops     1520
3 stops       45
4 stops        1
Name: Total_Stops, dtype: int64
```

In [37]:
```
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[37]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Tot
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 [38]:
```
city={"Source":{"Delhi":0,"Kolkata":1,"Banglore":2,
"Mumbai":3,"Chennai":4}}
traindf=traindf.replace(city)
traindf
```

Out[38]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Tota
0	1	24/03/2019	2	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	r
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	r
10679	2	27/04/2019	1	Banglore	CCU ? BLR	20:45	23:20	2h 35m	r
10680	0	27/04/2019	2	Delhi	BLR ? DEL	08:20	11:20	3h	r
10681	5	01/03/2019	2	New Delhi	BLR ? DEL	11:30	14:10	2h 40m	r
10682	2	9/05/2019	0	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20m	

10682 rows × 11 columns

In [39]:
```
dest={"Destination":{"Cochin":0,"Banglore":1,"Delhi":2,
"New Delhi":3,"Hyderabad":4,"Kolkata":5}}
traindf=traindf.replace(dest)
traindf
```

Out[39]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Tota
0	1	24/03/2019	2	3	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	r
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	r
10679	2	27/04/2019	1	1	CCU ? BLR	20:45	23:20	2h 35m	r
10680	0	27/04/2019	2	2	BLR ? DEL	08:20	11:20	3h	r
10681	5	01/03/2019	2	3	BLR ? DEL	11:30	14:10	2h 40m	r
10682	2	9/05/2019	0	0	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20m	

10682 rows × 11 columns

In [40]:
```
stops={"Total_Stops":{"non-stop":0,"1 stop":1,"2 stops":2,
"3 stops":3,"4 stops":4}}
traindf=traindf.replace(stops)
traindf
```

Out[40]:

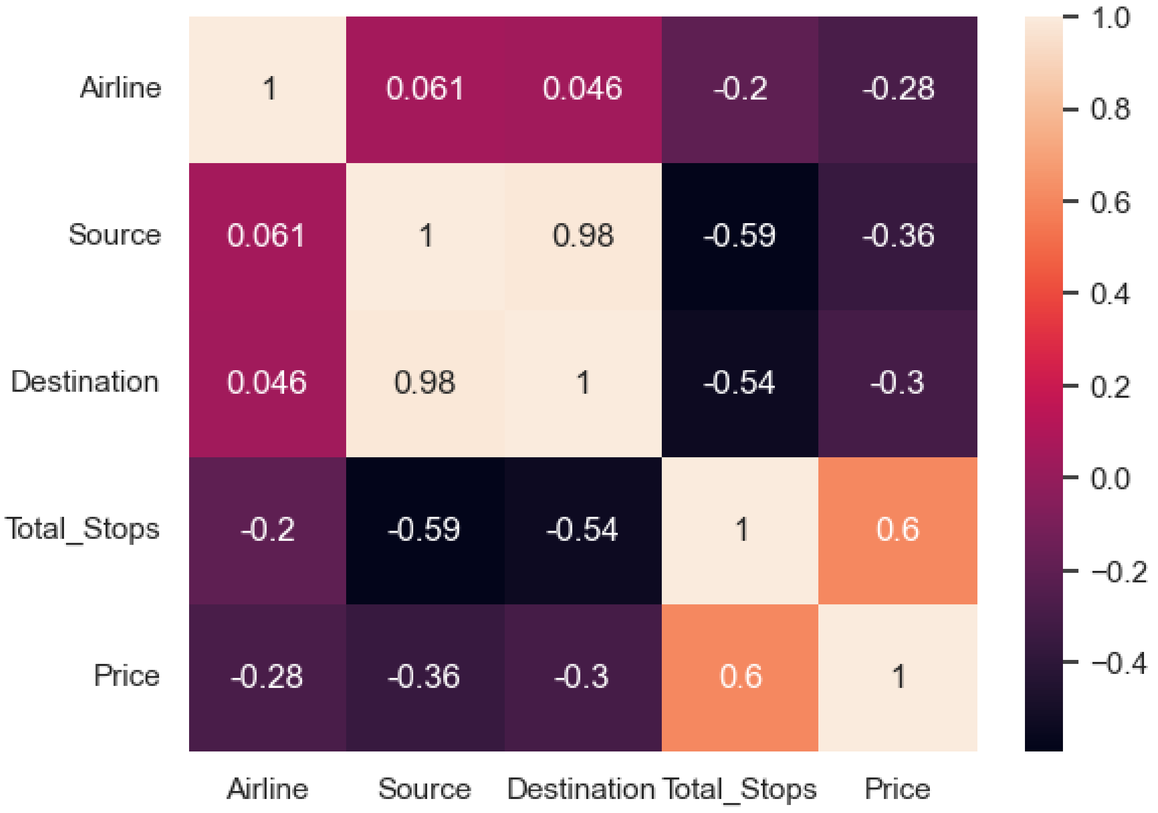
	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Tota
0	1	24/03/2019	2	3	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	
1	2	1/05/2019	1	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	
2	0	9/06/2019	0	0	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	
3	1	12/05/2019	1	1	CCU ? NAG ? BLR	18:05	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

Data visualization:-

In [41]:
```
fdf=traindf[['Airline','Source','Destination','Total_Stops','Price']]
sns.heatmap(fdf.corr(),annot=True)
```

Out[41]:
```
<AxesSubplot:>
```



Feature Scaling :- To Split the data into train data and test data

In [42]:
```
x=fdf[['Airline','Source','Destination','Total_Stops']]
y=fdf['Price']
```

In [43]:
```
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
```

Linear Regression

In [44]:
```
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[44]:

	coefficient
Airline	-418.483922
Source	-3275.073380
Destination	2505.480291
Total_Stops	3541.798053

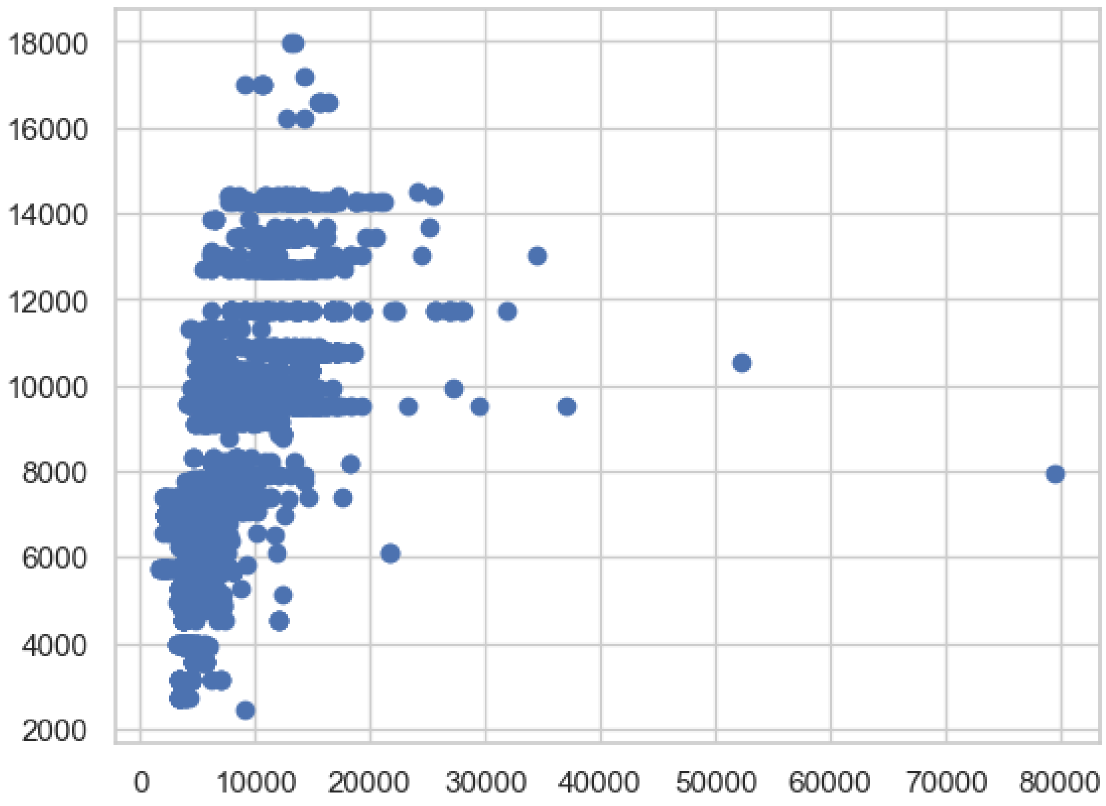
In [45]:
```
score=regr.score(X_test,y_test)
print(score)
```

```
0.4108304890928348
```

In [46]:
```
predictions=regr.predict(X_test)
```

In [47]:
```
plt.scatter(y_test,predictions)
```

Out[47]:
```
<matplotlib.collections.PathCollection at 0x44a6c94808>
```

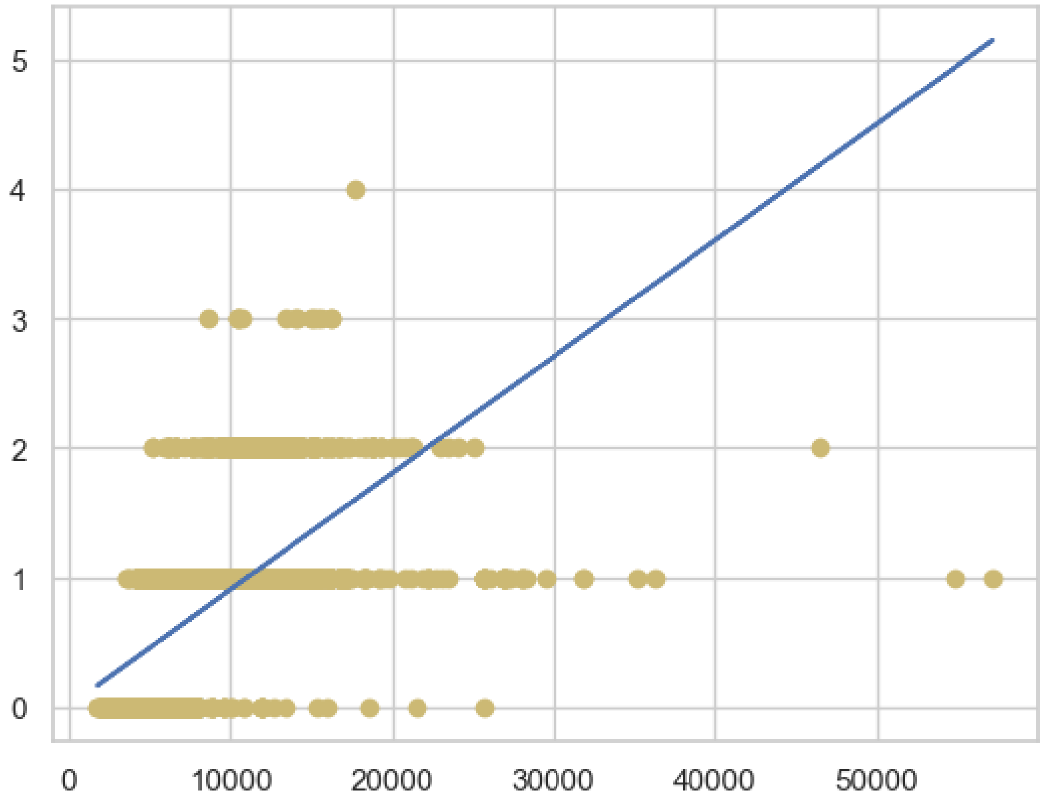


In [48]:
```
x=np.array(fdf['Price']).reshape(-1,1)
y=np.array(fdf['Total_Stops']).reshape(-1,1)
fdf.dropna(inplace=True)
```

In [49]:
```
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[49]:
```
LinearRegression()
```

In [50]:
```
y_pred=regr.predict(X_test)
plt.scatter(X_test,y_test,color='y')
plt.plot(X_test,y_pred,color='b')
plt.show()
```



Since in the above Linear regression we could not get accuracy so we can check for Logistic regression model.

Logistic Regression

In [51]:
```
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)
```

In [52]:
```
lr.fit(x_train,y_train)
```

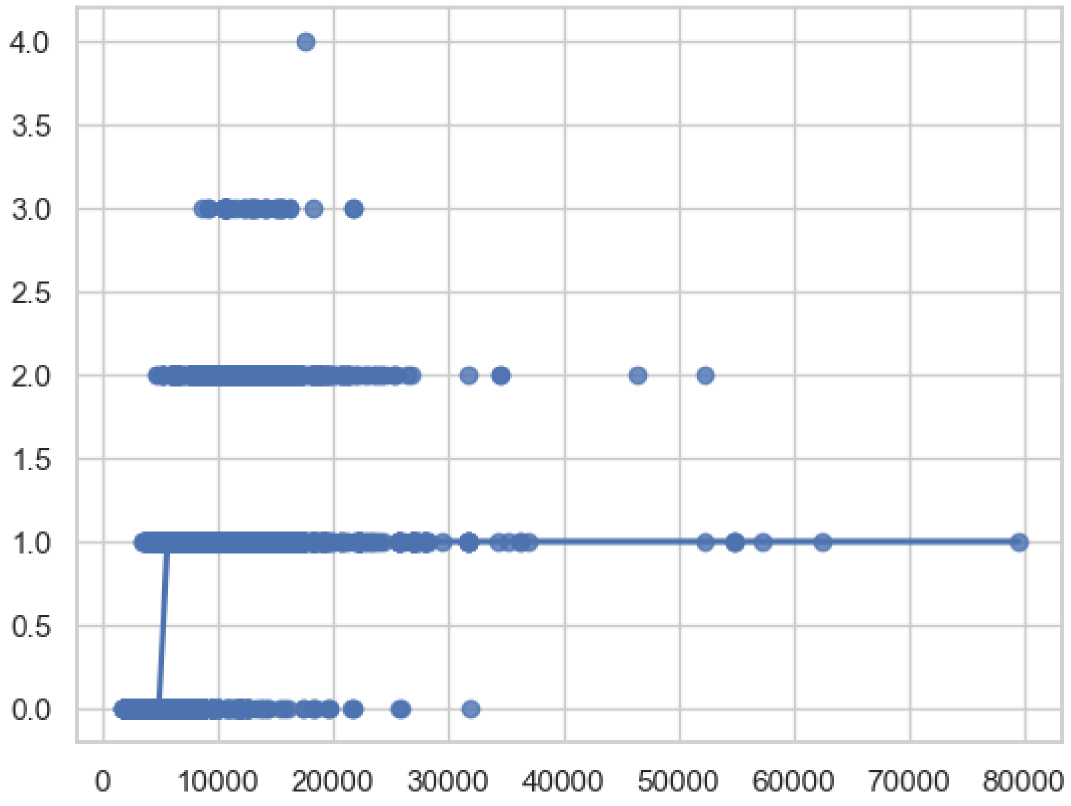
Out[52]:
```
LogisticRegression(max_iter=10000)
```

In [53]:
```
score=lr.score(x_test,y_test)
print(score)
```

```
0.7160686427457098
```

In [54]:
```
sns.regplot(x=x,y=y,data=fdf,logistic=True,ci=None)
```

Out[54]:
```
<AxesSubplot:>
```



In Logistic Regression model we could not get accuracy.we can use other models like Decision Tree and Random Forest to check the accuracy.

Decision Tree

```
In [55]: from sklearn.tree import DecisionTreeClassifier
         clf=DecisionTreeClassifier(random_state=0)
         clf.fit(x_train,y_train)

Out[55]: DecisionTreeClassifier(random_state=0)
```

```
In [56]: score=clf.score(x_test,y_test)
         print(score)

         0.9369734789391576
```

Random Forest

```
In [58]: from sklearn.ensemble import RandomForestClassifier
         rfc=RandomForestClassifier()
         rfc.fit(X_train,y_train)

Out[58]: RandomForestClassifier()
```

```
In [59]: 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 [60]: from sklearn.model_selection import GridSearchCV
         grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
```

```
In [61]: grid_search.fit(X_train,y_train)

Out[61]: GridSearchCV(cv=2, estimator=RandomForestClassifier(),
                      param_grid={'max_depth': [2, 3, 5, 10, 20],
                                  'min_samples_leaf': [5, 10, 20, 50, 100, 200],
                                  'n_estimators': [10, 25, 30, 50, 100, 200]},
                      scoring='accuracy')
```

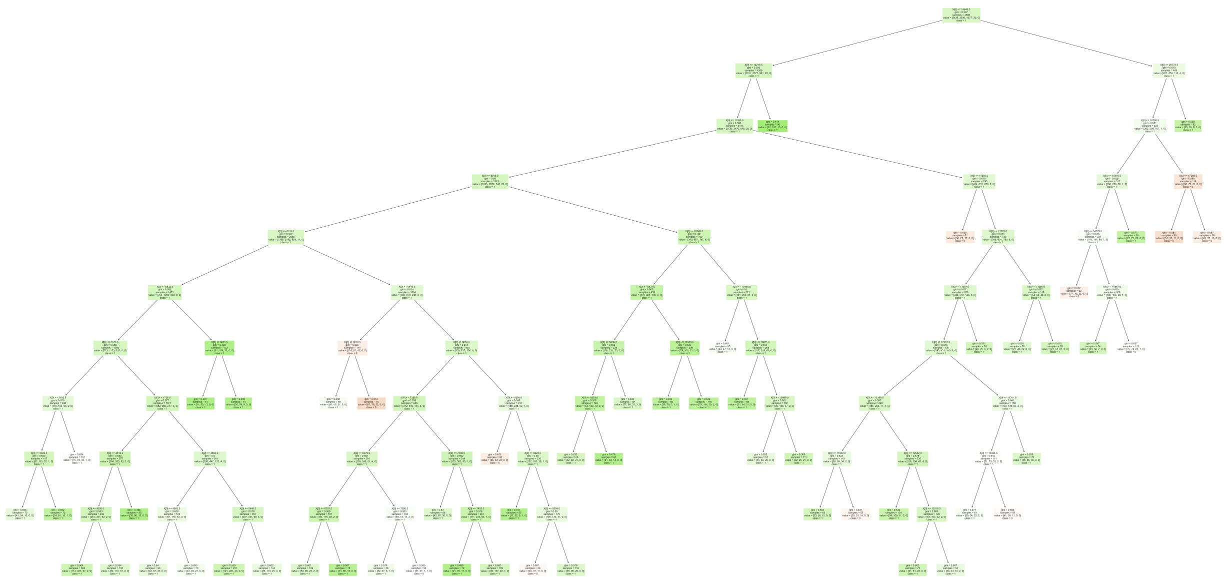
```
In [62]: grid_search.best_score_

Out[62]: 0.5253443988580163
```

```
In [63]: rf_best=grid_search.best_estimator_
         rf_best

Out[63]: RandomForestClassifier(max_depth=10, min_samples_leaf=50, n_estimators=50)
```

In [64]:
```
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 [65]:
```
score=rfc.score(x_test,y_test)
print(score)
```

```
0.44524180967238698
```

Coclusion: From the above implemented models the accuracy score is high in "Decision Tree"so it is the best model¶

In [ ]: