PROBLEM STATEMENT:

which model is best suit for this dataset.

importing required libraries

In [63]:

```
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 [64]:
```
train_df=pd.read_csv(r"C:\Users\MY HOME\Downloads\Data_Train11.csv")
train_df
```

Out[64]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24-03-2019	Banglore	New Delhi	BLR ? DEL	22:20	22-03-2023 01:10	2h 50m	non-stop	No info	3897
1	Air India	01-05-2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	09-06-2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	10-06-2023 04:25	19h	2 stops	No info	13882
3	IndiGo	12-05-2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01-03-2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302
...	...	...	...	...	...	...	...	...	...	...	...
10678	Air Asia	09-04-2019	Kolkata	Banglore	CCU ? BLR	19:55	22:25	2h 30m	non-stop	No info	4107
10679	Air India	27-04-2019	Kolkata	Banglore	CCU ? BLR	20:45	23:20	2h 35m	non-stop	No info	4145
10680	Jet Airways	27-04-2019	Banglore	Delhi	BLR ? DEL	08:20	11:20	3h	non-stop	No info	7229
10681	Vistara	01-03-2019	Banglore	New Delhi	BLR ? DEL	11:30	14:10	2h 40m	non-stop	No info	12648
10682	Air India	09-05-2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20m	2 stops	No info	11753

10683 rows × 11 columns

In [65]:
```
test_df=pd.read_csv(r"C:\Users\MY HOME\Downloads\Test_set22.csv")
test_df
```

Out[65]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL ? BOM ? COK	17:30	04:25 07 Jun	10h 55m	1 stop	No info
1	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? MAA ? BLR	06:20	10:20	4h	1 stop	No info
2	Jet Airways	21/05/2019	Delhi	Cochin	DEL ? BOM ? COK	19:15	19:00 22 May	23h 45m	1 stop	In-flight meal not included
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL ? BOM ? COK	08:00	21:00	13h	1 stop	No info
4	Air Asia	24/06/2019	Banglore	Delhi	BLR ? DEL	23:55	02:45 25 Jun	2h 50m	non-stop	No info
...	...	...	...	...	...	...	...	...	...	...
2666	Air India	6/06/2019	Kolkata	Banglore	CCU ? DEL ? BLR	20:30	20:25 07 Jun	23h 55m	1 stop	No info
2667	IndiGo	27/03/2019	Kolkata	Banglore	CCU ? BLR	14:20	16:55	2h 35m	non-stop	No info
2668	Jet Airways	6/03/2019	Delhi	Cochin	DEL ? BOM ? COK	21:50	04:25 07 Mar	6h 35m	1 stop	No info
2669	Air India	6/03/2019	Delhi	Cochin	DEL ? BOM ? COK	04:00	19:15	15h 15m	1 stop	No info
2670	Multiple carriers	15/06/2019	Delhi	Cochin	DEL ? BOM ? COK	04:55	19:15	14h 20m	1 stop	No info

2671 rows × 10 columns

Data cleaning

In [66]:
```
train_df.shape
```

Out[66]: (10683, 11)

In [67]:
```
test_df.shape
```

Out[67]:
```
(2671, 10)
```

In [68]:
```
train_df.describe
```

Out[68]:
```
<bound method NDFrame.describe of            Airline Date_of_Journey    Source Destination
0           IndiGo      24-03-2019  Banglore   New Delhi  \
1        Air India      01-05-2019   Kolkata    Banglore
2      Jet Airways      09-06-2019     Delhi      Cochin
3           IndiGo      12-05-2019   Kolkata    Banglore
4           IndiGo      01-03-2019  Banglore   New Delhi
...            ...             ...       ...         ...
10678     Air Asia      09-04-2019   Kolkata    Banglore
10679    Air India      27-04-2019   Kolkata    Banglore
10680  Jet Airways      27-04-2019  Banglore       Delhi
10681      Vistara      01-03-2019  Banglore   New Delhi
10682    Air India      09-05-2019     Delhi      Cochin

                      Route Dep_Time      Arrival_Time Duration Total_Stops
0                 BLR ? DEL    22:20  22-03-2023 01:10   2h 50m    non-stop  \
1      CCU ? IXR ? BBI ? BLR    05:50             13:15   7h 25m     2 stops
2      DEL ? LKO ? BOM ? COK    09:25  10-06-2023 04:25      19h     2 stops
3            CCU ? NAG ? BLR    18:05             23:30   5h 25m      1 stop
4            BLR ? NAG ? DEL    16:50             21:35   4h 45m      1 stop
...                      ...      ...               ...      ...         ...
10678             CCU ? BLR    19:55             22:25   2h 30m    non-stop
10679             CCU ? BLR    20:45             23:20   2h 35m    non-stop
10680             BLR ? DEL    08:20             11:20       3h    non-stop
10681             BLR ? DEL    11:30             14:10   2h 40m    non-stop
10682  DEL ? GOI ? BOM ? COK    10:55             19:15   8h 20m     2 stops

      Additional_Info  Price
0             No info   3897
1             No info   7662
2             No info  13882
3             No info   6218
4             No info  13302
...               ...    ...
10678         No info   4107
10679         No info   4145
10680         No info   7229
10681         No info  12648
10682         No info  11753

[10683 rows x 11 columns]>
```

In [69]:
```
test_df.describe
```

Out[69]:
```
<bound method NDFrame.describe of                 Airline Date_of_Journey    Source Destination
0           Jet Airways       6/06/2019     Delhi      Cochin  \
1                IndiGo      12/05/2019   Kolkata    Banglore
2           Jet Airways      21/05/2019     Delhi      Cochin
3     Multiple carriers      21/05/2019     Delhi      Cochin
4              Air Asia      24/06/2019  Banglore       Delhi
...                 ...             ...       ...         ...
2666          Air India       6/06/2019   Kolkata    Banglore
2667             IndiGo      27/03/2019   Kolkata    Banglore
2668        Jet Airways       6/03/2019     Delhi      Cochin
2669          Air India       6/03/2019     Delhi      Cochin
2670  Multiple carriers      15/06/2019     Delhi      Cochin

                Route Dep_Time  Arrival_Time Duration Total_Stops
0     DEL ? BOM ? COK    17:30  04:25 07 Jun  10h 55m      1 stop  \
1     CCU ? MAA ? BLR    06:20         10:20       4h      1 stop
2     DEL ? BOM ? COK    19:15  19:00 22 May  23h 45m      1 stop
3     DEL ? BOM ? COK    08:00         21:00      13h      1 stop
4           BLR ? DEL    23:55  02:45 25 Jun   2h 50m    non-stop
...               ...      ...           ...      ...         ...
2666  CCU ? DEL ? BLR    20:30  20:25 07 Jun  23h 55m      1 stop
2667        CCU ? BLR    14:20         16:55   2h 35m    non-stop
2668  DEL ? BOM ? COK    21:50  04:25 07 Mar   6h 35m      1 stop
2669  DEL ? BOM ? COK    04:00         19:15  15h 15m      1 stop
2670  DEL ? BOM ? COK    04:55         19:15  14h 20m      1 stop

                  Additional_Info
0                         No info
1                         No info
2     In-flight meal not included
3                         No info
4                         No info
...                           ...
2666                      No info
2667                      No info
2668                      No info
2669                      No info
2670                      No info

[2671 rows x 10 columns]>
```

In [70]:
```
train_df.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 [71]:
```
test_df.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 [72]:
```
test_df.describe()
```

Out[72]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
count	2671	2671	2671	2671	2671	2671	2671	2671	2671	2671
unique	11	44	5	6	100	199	704	320	5	6
top	Jet Airways	9/05/2019	Delhi	Cochin	DEL ? BOM ? COK	10:00	19:00	2h 50m	1 stop	No info
freq	897	144	1145	1145	624	62	113	122	1431	2148

In [73]:
```
train_df.describe()
```

Out[73]:

	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

To find missing values

In [74]:
```
train_df.isnull().sum()
```

Out[74]:
```
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 [76]:
```
train_df.dropna(inplace=True)
```

In [77]:
```
train_df["Source"].value_counts()
```

Out[77]:
```
Source
Delhi       4536
Kolkata     2871
Banglore    2197
Mumbai       697
Chennai      381
Name: count, dtype: int64
```

In [87]:
```
convert={"Source":{"Delhi":0,"Kolkata":1,"Banglore":2,"Mumbai":3,"Chennai":4}}
train_df=train_df.replace(convert)
train_df
```

Out[87]:

	Source	Destination
0	0	0
1	1	1
2	0	0
3	0	0
4	2	2
...	...	...
2666	1	1
2667	1	1
2668	0	0
2669	0	0
2670	0	0

2671 rows × 2 columns

In [88]:
```
convert={"Destination":{"Cochin":0,"Banglore":1,"Delhi":2,"New Delhi":3,"Hyderabad":4,"Kolkata":5}}
train_df=train_df.replace(convert)
train_df
```

Out[88]:

	Source	Destination
0	0	0
1	1	1
2	0	0
3	0	0
4	2	2
...	...	...
2666	1	1
2667	1	1
2668	0	0
2669	0	0
2670	0	0

2671 rows × 2 columns

In [89]:
```
train_df=train_df[['Source','Destination']]
train_df
```

Out[89]:

	Source	Destination
0	0	0
1	1	1
2	0	0
3	0	0
4	2	2
...	...	...
2666	1	1
2667	1	1
2668	0	0
2669	0	0
2670	0	0

2671 rows × 2 columns

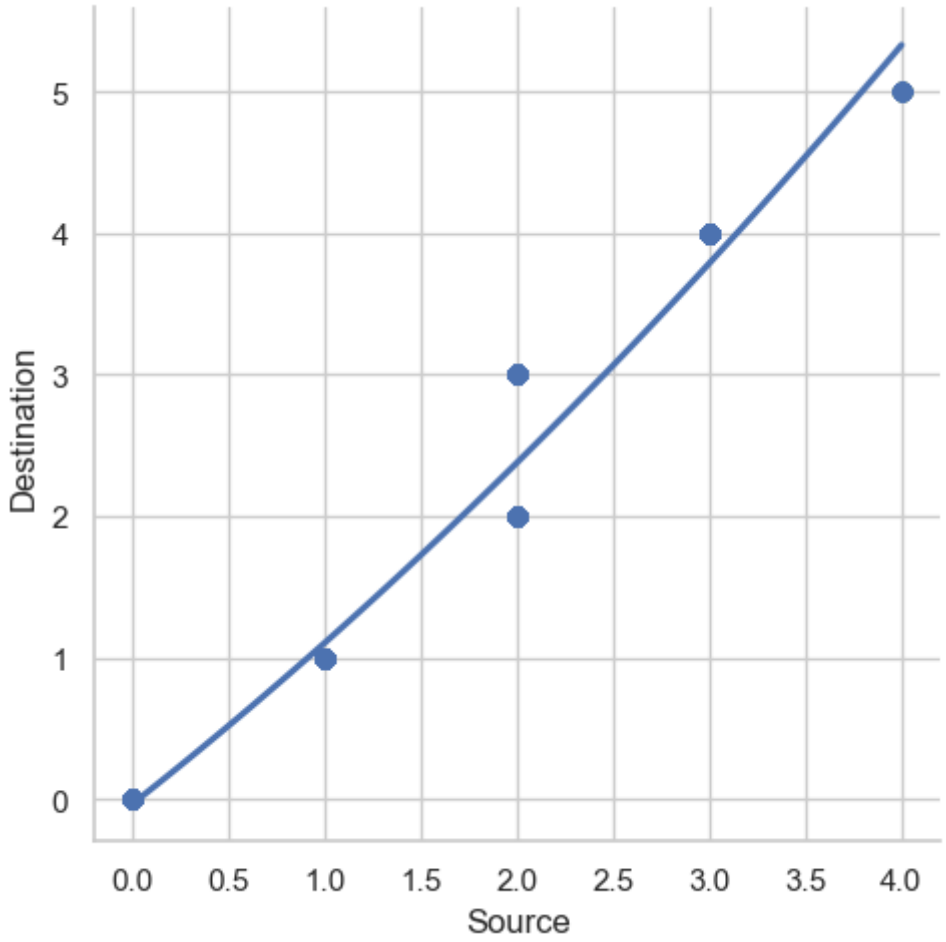
In [90]:
```
train_df.head(10)
```

Out[90]:

	Source	Destination
0	0	0
1	1	1
2	0	0
3	0	0
4	2	2
5	0	0
6	2	3
7	1	1
8	1	1
9	1	1

In [ ]:

In [91]:
```
sns.lmplot(x="Source",y="Destination",order=2,data=train_df,ci=None)
plt.show()
```



In [92]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2671 entries, 0 to 2670
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Source       2671 non-null   int64
 1   Destination  2671 non-null   int64
dtypes: int64(2)
memory usage: 41.9 KB
```

In [93]:
```
#Separating data into independent & dependent variables
#Now each dataframe contains only one coloumn
x=np.array(train_df['Source']).reshape(-1,1)
y=np.array(train_df['Destination']).reshape(-1,1)
#Dropping any rows with Nan values
train_df.dropna(inplace=True)
train_df
```

Out[93]:

	Source	Destination
0	0	0
1	1	1
2	0	0
3	0	0
4	2	2
...	...	...
2666	1	1
2667	1	1
2668	0	0
2669	0	0
2670	0	0

2671 rows × 2 columns

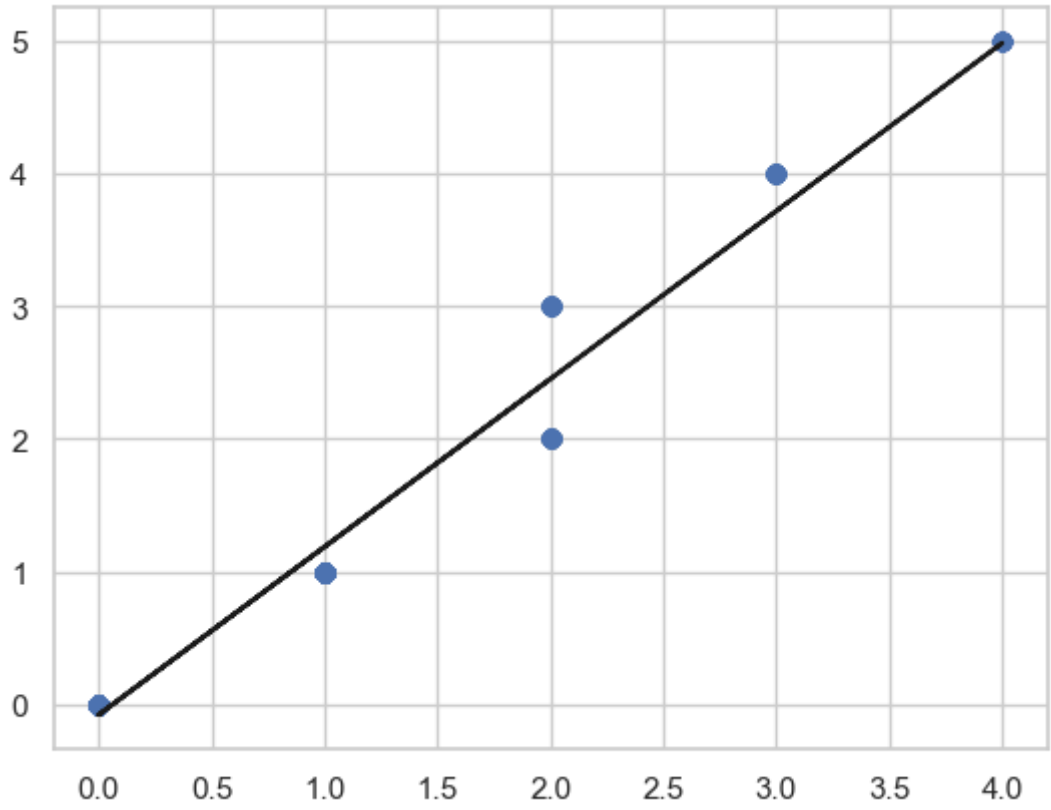
In [94]:

```
#Splitting the data into training and testing data
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.9697896482947828
```

In [95]:
```
#Data scatter to predict the values

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



RIDGE REGRESSION

In [ ]:
```
-------->To check better accuracy/fit.
```

In [96]:
```
from sklearn.linear_model import Ridge, RidgeCV, Lasso
```

In [97]:
```
ridge=Ridge(alpha=2)
ridge.fit(x_train,y_train)
train_score_ridge=ridge.score(x_train,y_train)
test_score_ridge=ridge.score(x_test,y_test)
print("\nLinearRegression\n",(train_score_ridge))
print(test_score_ridge)
```

```
LinearRegression
 0.9629209899323132
0.9697859490617774
```

LASSO REGRESSION

In [ ]:
```
-------->To check for better model.
```

In [98]:
```
#Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(x_train,y_train)
train_score_ls =lasso.score(x_train,y_train)
test_score_ls =lasso.score(x_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
Lasso Model:

The train score for ls model is 0.0
The test score for ls model is -7.948032513360737e-05
```

To check best accurcay we are going to do Linear Regression for this Dataset by taking different fields.

In [25]:
```
#first 500 rows
```

In [26]:
```
df500=pd.read_csv(r"C:\Users\MY HOME\Downloads\Data_Train11.csv")
df500
```

Out[26]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24-03-2019	Banglore	New Delhi	BLR ? DEL	22:20	22-03-2023 01:10	2h 50m	non-stop	No info	3897
1	Air India	01-05-2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	09-06-2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	10-06-2023 04:25	19h	2 stops	No info	13882
3	IndiGo	12-05-2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01-03-2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302
...	...	...	...	...	...	...	...	...	...	...	...
10678	Air Asia	09-04-2019	Kolkata	Banglore	CCU ? BLR	19:55	22:25	2h 30m	non-stop	No info	4107
10679	Air India	27-04-2019	Kolkata	Banglore	CCU ? BLR	20:45	23:20	2h 35m	non-stop	No info	4145
10680	Jet Airways	27-04-2019	Banglore	Delhi	BLR ? DEL	08:20	11:20	3h	non-stop	No info	7229
10681	Vistara	01-03-2019	Banglore	New Delhi	BLR ? DEL	11:30	14:10	2h 40m	non-stop	No info	12648
10682	Air India	09-05-2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20m	2 stops	No info	11753

10683 rows × 11 columns

In [27]:
```
convert={"Source":{"Delhi":0,"Kolkata":1,"Banglore":2,"Mumbai":3,"Chennai":4}}
df500=train_df.replace(convert)
df500
```

Out[27]:

	Source	Price
0	2	3897
1	1	7662
2	0	13882
3	1	6218
4	2	13302
...	...	...
10678	1	4107
10679	1	4145
10680	2	7229
10681	2	12648
10682	0	11753

10683 rows × 2 columns

In [28]:
```
#selecting the first 500 rows
df500=df500[:][:500]
df500
```

Out[28]:

	Source	Price
0	2	3897
1	1	7662
2	0	13882
3	1	6218
4	2	13302
...	...	...
495	1	9663
496	0	14714
497	0	5406
498	1	8610
499	0	14237

500 rows × 2 columns

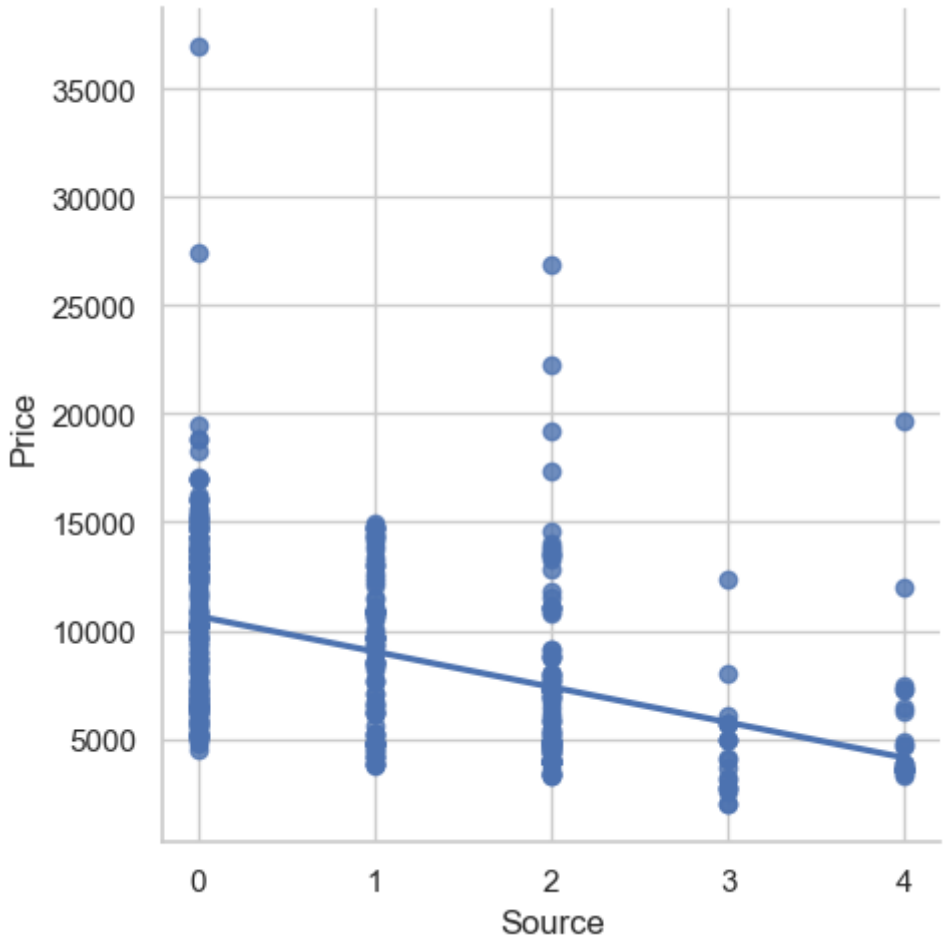
In [29]:
```
#Renamin columns for easier process(OPTIONAL)
df500=df500[['Source','Price']]
df500
```

Out[29]:

	Source	Price
0	2	3897
1	1	7662
2	0	13882
3	1	6218
4	2	13302
...	...	...
495	1	9663
496	0	14714
497	0	5406
498	1	8610
499	0	14237

500 rows × 2 columns

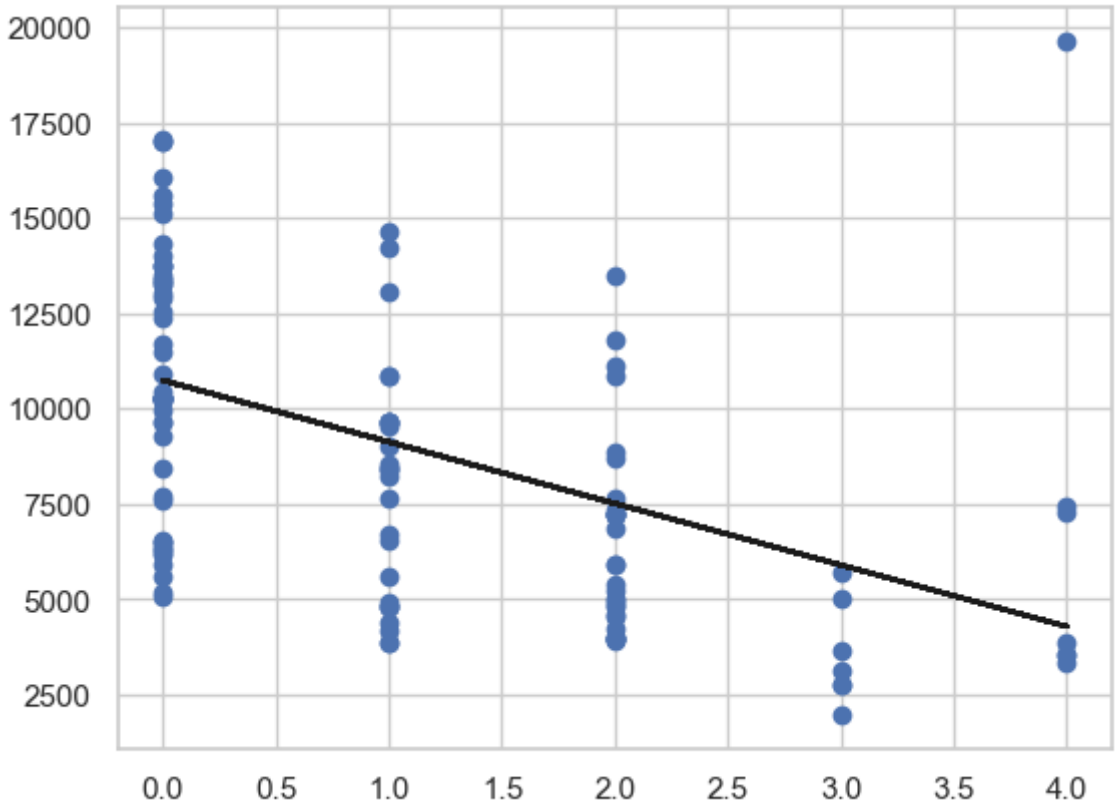
In [30]:
```
sns.lmplot(x="Source",y="Price",data=df500,order=1,ci=None)
plt.show()
```



In [31]:
```
x=np.array(df500['Source']).reshape(-1,1)
y=np.array(df500['Price']).reshape(-1,1)
df500.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print("regression:",regr.score(x_test,y_test))
```

```
regression: 0.2449336058162136
```

In [32]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



by taking Source and price we got 24% accuracy.

logistic Regression

In [ ]:
```
--------->To check the better model we are going to do logistic regression.
```

In [100]:
```
#Logistic Regression
x=np.array(train_df['Source']).reshape(-1,1)
y=np.array(train_df['Destination']).reshape(-1,1)
train_df.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 [101]:
```
lr.fit(x_train,y_train)
```

Out[101]:
```
LogisticRegression(max_iter=10000)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [102]:
```
score=lr.score(x_test,y_test)
print(score)
```

```
0.9152119700748129
```

Decision Tree

In [103]:
```
from sklearn.tree import DecisionTreeClassifier
d=DecisionTreeClassifier(random_state=0)
d.fit(x_train,y_train)
```

Out[103]:
```
DecisionTreeClassifier(random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [104]:
```
score=d.score(x_test,y_test)
print(score)
```

```
0.9152119700748129
```

Random Classifier

In [105]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[105]:
```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [106]:
```
params={'max_depth':[2,8,6,15],
'min_samples_leaf':[20,50,16,200],
'n_estimators':[10,25,38,50]}
```

In [107]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
```

In [108]:
```
grid_search.fit(x_train,y_train)
```

Out[108]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [2, 8, 6, 15],
                         'min_samples_leaf': [20, 50, 16, 200],
                         'n_estimators': [10, 25, 38, 50]},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [109]:
```
grid_search.best_score_
```

Out[109]:
```
0.9090422425540199
```

conclusion:

```
        By doing Linear Regression and Logistic Regression on this Dataset.Based on the models accuracies w
e conclude the best fit/model.

    ---->we got 96% of accuracy for Linear Regression and very minimal change after doing Ridge Regression.
    so,compared to both Ridge Regression is the best suit for the insurence train_Dataset with the accuracy
of 96%.
    ---->we got only 90% accuracy for Rndom forest classification.
```

TEST -DATASET

PROBLEM STATEMENT:

to check which model is the best suit for this dataset.

In [43]:
```
test_df=pd.read_csv(r"C:\Users\MY HOME\Downloads\Test_set22.csv")
test_df
```

Out[43]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL ? BOM ? COK	17:30	04:25 07 Jun	10h 55m	1 stop	No info
1	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? MAA ? BLR	06:20	10:20	4h	1 stop	No info
2	Jet Airways	21/05/2019	Delhi	Cochin	DEL ? BOM ? COK	19:15	19:00 22 May	23h 45m	1 stop	In-flight meal not included
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL ? BOM ? COK	08:00	21:00	13h	1 stop	No info
4	Air Asia	24/06/2019	Banglore	Delhi	BLR ? DEL	23:55	02:45 25 Jun	2h 50m	non-stop	No info
...	...	...	...	...	...	...	...	...	...	...
2666	Air India	6/06/2019	Kolkata	Banglore	CCU ? DEL ? BLR	20:30	20:25 07 Jun	23h 55m	1 stop	No info
2667	IndiGo	27/03/2019	Kolkata	Banglore	CCU ? BLR	14:20	16:55	2h 35m	non-stop	No info
2668	Jet Airways	6/03/2019	Delhi	Cochin	DEL ? BOM ? COK	21:50	04:25 07 Mar	6h 35m	1 stop	No info
2669	Air India	6/03/2019	Delhi	Cochin	DEL ? BOM ? COK	04:00	19:15	15h 15m	1 stop	No info
2670	Multiple carriers	15/06/2019	Delhi	Cochin	DEL ? BOM ? COK	04:55	19:15	14h 20m	1 stop	No info

2671 rows × 10 columns

In [44]:
```
convert={"Source":{"Delhi":0,"Kolkata":1,"Banglore":2,"Mumbai":3,"Chennai":4}}
test_df=test_df.replace(convert)
test_df
```

Out[44]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
0	Jet Airways	6/06/2019	0	Cochin	DEL ? BOM ? COK	17:30	04:25 07 Jun	10h 55m	1 stop	No info
1	IndiGo	12/05/2019	1	Banglore	CCU ? MAA ? BLR	06:20	10:20	4h	1 stop	No info
2	Jet Airways	21/05/2019	0	Cochin	DEL ? BOM ? COK	19:15	19:00 22 May	23h 45m	1 stop	In-flight meal not included
3	Multiple carriers	21/05/2019	0	Cochin	DEL ? BOM ? COK	08:00	21:00	13h	1 stop	No info
4	Air Asia	24/06/2019	2	Delhi	BLR ? DEL	23:55	02:45 25 Jun	2h 50m	non-stop	No info
...	...	...	...	...	...	...	...	...	...	...
2666	Air India	6/06/2019	1	Banglore	CCU ? DEL ? BLR	20:30	20:25 07 Jun	23h 55m	1 stop	No info
2667	IndiGo	27/03/2019	1	Banglore	CCU ? BLR	14:20	16:55	2h 35m	non-stop	No info
2668	Jet Airways	6/03/2019	0	Cochin	DEL ? BOM ? COK	21:50	04:25 07 Mar	6h 35m	1 stop	No info
2669	Air India	6/03/2019	0	Cochin	DEL ? BOM ? COK	04:00	19:15	15h 15m	1 stop	No info
2670	Multiple carriers	15/06/2019	0	Cochin	DEL ? BOM ? COK	04:55	19:15	14h 20m	1 stop	No info

2671 rows × 10 columns

In [45]:
```
test_df["Destination"].value_counts()
```

Out[45]:
```
Destination
Cochin       1145
Banglore      710
Delhi         317
New Delhi     238
Hyderabad     186
Kolkata        75
Name: count, dtype: int64
```

In [46]:
```
convert={"Destination":{"Cochin":0,"Banglore":1,"Delhi":2,"New Delhi":3,"Hyderabad":4,"Kolkata":5}}
test_df=test_df.replace(convert)
test_df
```

Out[46]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
0	Jet Airways	6/06/2019	0	0	DEL ? BOM ? COK	17:30	04:25 07 Jun	10h 55m	1 stop	No info
1	IndiGo	12/05/2019	1	1	CCU ? MAA ? BLR	06:20	10:20	4h	1 stop	No info
2	Jet Airways	21/05/2019	0	0	DEL ? BOM ? COK	19:15	19:00 22 May	23h 45m	1 stop	In-flight meal not included
3	Multiple carriers	21/05/2019	0	0	DEL ? BOM ? COK	08:00	21:00	13h	1 stop	No info
4	Air Asia	24/06/2019	2	2	BLR ? DEL	23:55	02:45 25 Jun	2h 50m	non-stop	No info
...	...	...	...	...	...	...	...	...	...	...
2666	Air India	6/06/2019	1	1	CCU ? DEL ? BLR	20:30	20:25 07 Jun	23h 55m	1 stop	No info
2667	IndiGo	27/03/2019	1	1	CCU ? BLR	14:20	16:55	2h 35m	non-stop	No info
2668	Jet Airways	6/03/2019	0	0	DEL ? BOM ? COK	21:50	04:25 07 Mar	6h 35m	1 stop	No info
2669	Air India	6/03/2019	0	0	DEL ? BOM ? COK	04:00	19:15	15h 15m	1 stop	No info
2670	Multiple carriers	15/06/2019	0	0	DEL ? BOM ? COK	04:55	19:15	14h 20m	1 stop	No info

2671 rows × 10 columns

In [47]:
```
test_df["Destination"].value_counts()
```

Out[47]:
```
Destination
0    1145
1     710
2     317
3     238
4     186
5      75
Name: count, dtype: int64
```

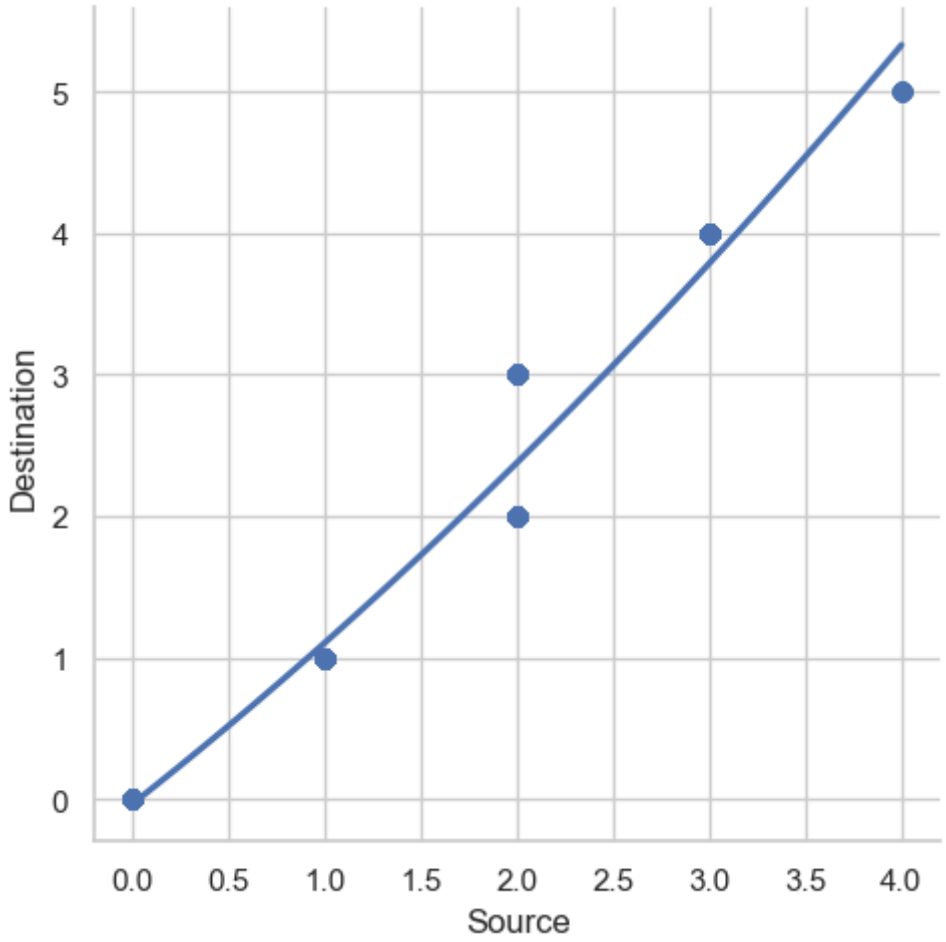
In [48]:
```
#taking selected columns from dataset
test_df=test_df[['Source','Destination']]
test_df
```

Out[48]:

	Source	Destination
0	0	0
1	1	1
2	0	0
3	0	0
4	2	2
...	...	...
2666	1	1
2667	1	1
2668	0	0
2669	0	0
2670	0	0

2671 rows × 2 columns

In [49]:
```
sns.lmplot(x="Source",y="Destination",order=2,data=test_df,ci=None)
plt.show()
```



In [50]:
```
test_df.describe()
```

Out[50]:

	Source	Destination
count	2671.000000	2671.000000
mean	1.002621	1.189442
std	1.080814	1.394497
min	0.000000	0.000000
25%	0.000000	0.000000
50%	1.000000	1.000000
75%	2.000000	2.000000
max	4.000000	5.000000

In [51]:
```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2671 entries, 0 to 2670
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Source       2671 non-null   int64
 1   Destination  2671 non-null   int64
dtypes: int64(2)
memory usage: 41.9 KB
```

In [52]:
```
#Separating data into independent & dependent variables
#Now each dataframe contains only one coloumn
x=np.array(test_df['Source']).reshape(-1,1)
y=np.array(test_df['Destination']).reshape(-1,1)
#Dropping any rows with Nan values
test_df.dropna(inplace=True)
test_df
```

Out[52]:

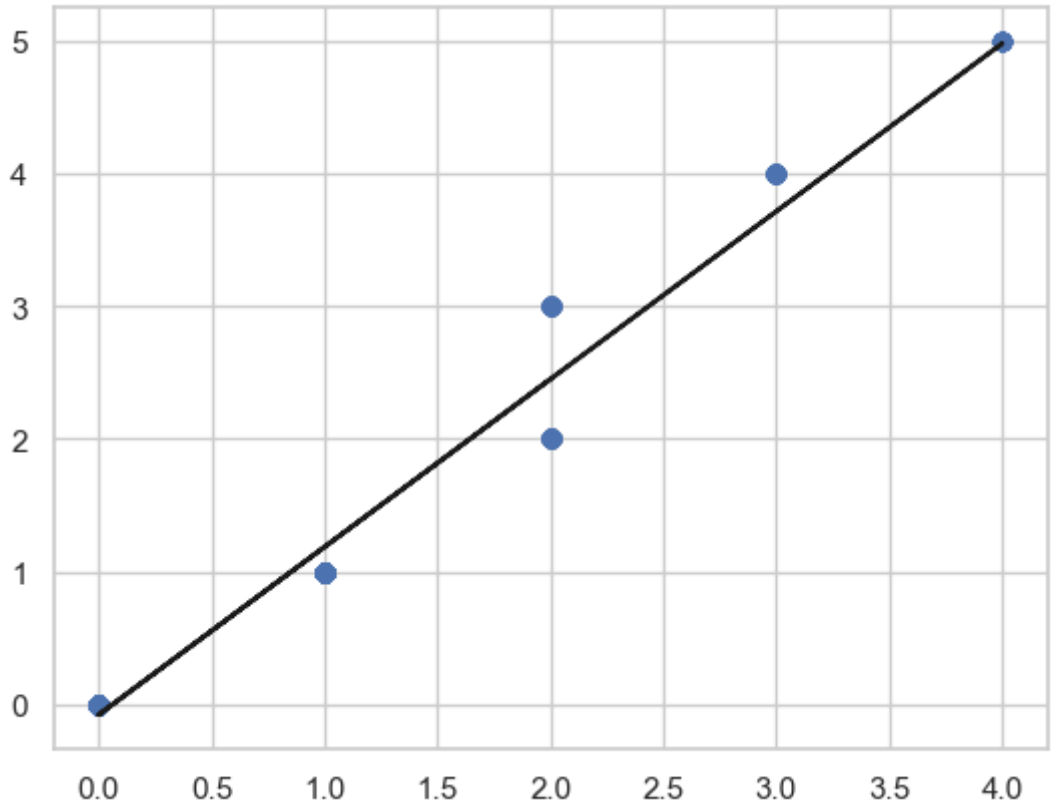
	Source	Destination
0	0	0
1	1	1
2	0	0
3	0	0
4	2	2
...	...	...
2666	1	1
2667	1	1
2668	0	0
2669	0	0
2670	0	0

2671 rows × 2 columns

In [53]:
```
#Splitting the data into training and testing data
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.9643367612678272
```

In [54]:
```
#Data scatter to predict the values
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



RIDGE REGRESSION

In [55]:
```
from sklearn.linear_model import Ridge, RidgeCV, Lasso
```

In [56]:
```
ridge=Ridge(alpha=2)
ridge.fit(x_train,y_train)
train_score_ridge=ridge.score(x_train,y_train)
test_score_ridge=ridge.score(x_test,y_test)
print("\nLinearRegression\n",(train_score_ridge))
print(test_score_ridge)
```

```
LinearRegression
 0.9649041263710476
0.9643284805867383
```

LASSO REGRESSION

In [58]:
```
#Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(x_train,y_train)
train_score_ls =lasso.score(x_train,y_train)
test_score_ls =lasso.score(x_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
Lasso Model:

The train score for ls model is 0.0
The test score for ls model is -6.384511517731895e-05
```

LOGISTIC REGRESSION:

tO CHECK BEST FIT WE ARE GOING TO DO LOGISTIC REGRESSION.

In [60]:
```
#Logistic Regression
x=np.array(test_df['Source']).reshape(-1,1)
y=np.array(test_df['Destination']).reshape(-1,1)
test_df.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 [61]:
```
lr.fit(x_train,y_train)
```

Out[61]:
```
LogisticRegression(max_iter=10000)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [62]:
```
score=d.score(x_test,y_test)
print(score)
```

```
0.0
```

CONCLUSION:

```
          For the test_dataset we got 96% accuracy for Linear Regression and for Ridge Regression we also g
ot the same accuracy with the minimal change of 0.00000002%.
          Based on the different model accuracies we conclude  the best fit.
   ---->So,Ridge Rgression is the best model/fit for the flight price pridiction test_dataset.
```

In [ ]: