Importing libraries

In [27]:

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

In [26]:

```
traindf=pd.read_csv(r"C:\Users\sowmika\OneDrive\Desktop\Data_Train.csv")
traindf
```

Out[26]:

	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 [28]:

```
testdf=pd.read_csv(r"C:\Users\sowmika\Downloads\Copy of Test_set.csv")
testdf
```

Out[28]:

	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 cleaning

In [29]:

```
train_df.shape
```

Out[29]:

```
(10683, 11)
```

In [30]:

```
test_df.shape
```

Out[30]:

```
(2671, 10)
```

In [31]:

```
train_df.describe
```

Out[31]:

```
<bound method NDFrame.describe of            Airline Date_of_Journey    So
urce Destination
0           IndiGo      24/03/2019  Banglore   New Delhi  \
1        Air India       1/05/2019   Kolkata    Banglore
2      Jet Airways       9/06/2019     Delhi      Cochin
3           IndiGo      12/05/2019   Kolkata    Banglore
4           IndiGo      01/03/2019  Banglore   New Delhi
...            ...             ...       ...         ...
10678     Air Asia       9/04/2019   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       9/05/2019     Delhi      Cochin

                       Route Dep_Time  Arrival_Time Duration Total_Stops
0                  BLR ? DEL    22:20  01:10 22 Mar   2h 50m    non-stop
\
1      CCU ? IXR ? BBI ? BLR    05:50         13:15   7h 25m     2 stops
2      DEL ? LKO ? BOM ? COK    09:25  04:25 10 Jun      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 [34]:

```
test_df.describe
```

Out[34]:

```
<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 [35]:

```
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 [36]:

```
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 [37]:

```
test_df.describe()
```

Out[37]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Dura
count	2671	2671	2671	2671	2671	2671	2671	2
unique	11	44	5	6	100	199	704	
top	Jet Airways	9/05/2019	Delhi	Cochin	DEL ? BOM ? COK	10:00	19:00	2h
freq	897	144	1145	1145	624	62	113	

In [38]:

```
train_df.describe()
```

Out[38]:

	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

Finding missing value

In [39]:

```
train_df.isnull().sum()
```

Out[39]:

```
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 [40]:

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

In [41]:

```
train_df["Source"].value_counts()
```

Out[41]:

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

In [42]:

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

Out[42]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	IndiGo	24/03/2019	2	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 5
1	Air India	1/05/2019	1	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 2
2	Jet Airways	9/06/2019	0	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	´
3	IndiGo	12/05/2019	1	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 2
4	IndiGo	01/03/2019	2	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 4
...	...	...	...	...	...	...	...	
10678	Air Asia	9/04/2019	1	Banglore	CCU ? BLR	19:55	22:25	2h 3
10679	Air India	27/04/2019	1	Banglore	CCU ? BLR	20:45	23:20	2h 3
10680	Jet Airways	27/04/2019	2	Delhi	BLR ? DEL	08:20	11:20	
10681	Vistara	01/03/2019	2	New Delhi	BLR ? DEL	11:30	14:10	2h 4
10682	Air India	9/05/2019	0	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 2

10682 rows × 11 columns

In [43]:

```
convert={"Destination":{"Cochin":0,"Banglore":1,"Delhi":2,"New Delhi":3,"Hyderabad":4,"K
train_df=train_df.replace(convert)
train_df
```

Out[43]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	IndiGo	24/03/2019	2	3	BLR ? DEL	22:20	01:10 22 Mar	2h 5
1	Air India	1/05/2019	1	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 2
2	Jet Airways	9/06/2019	0	0	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	
3	IndiGo	12/05/2019	1	1	CCU ? NAG ? BLR	18:05	23:30	5h 2
4	IndiGo	01/03/2019	2	3	BLR ? NAG ? DEL	16:50	21:35	4h 4
...	...	...	...	...	...	...	...	
10678	Air Asia	9/04/2019	1	1	CCU ? BLR	19:55	22:25	2h 3
10679	Air India	27/04/2019	1	1	CCU ? BLR	20:45	23:20	2h 3
10680	Jet Airways	27/04/2019	2	2	BLR ? DEL	08:20	11:20	
10681	Vistara	01/03/2019	2	3	BLR ? DEL	11:30	14:10	2h 4
10682	Air India	9/05/2019	0	0	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 2

10682 rows × 11 columns

In [44]:

```
train_df=train_df[['Source','Destination']]
train_df
```

Out[44]:

	Source	Destination
0	2	3
1	1	1
2	0	0
3	1	1
4	2	3
...	...	...
10678	1	1
10679	1	1
10680	2	2
10681	2	3
10682	0	0

10682 rows × 2 columns

In [45]:

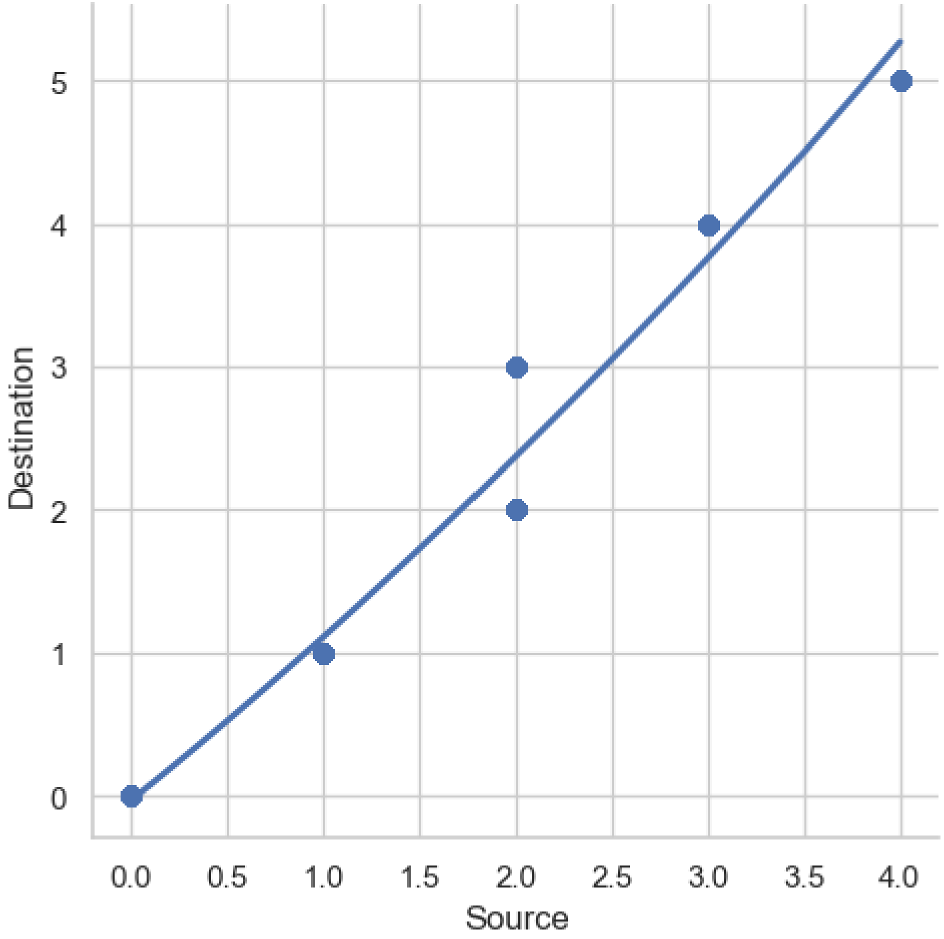
```
train_df.head(10)
```

Out[45]:

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

In [46]:

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



In [47]:

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10682 entries, 0 to 10682
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Source       10682 non-null  int64
 1   Destination  10682 non-null  int64
dtypes: int64(2)
memory usage: 250.4 KB
```

In [48]:

```
#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[48]:

	Source	Destination
0	2	3
1	1	1
2	0	0
3	1	1
4	2	3
...	...	...
10678	1	1
10679	1	1
10680	2	2
10681	2	3
10682	0	0

10682 rows × 2 columns

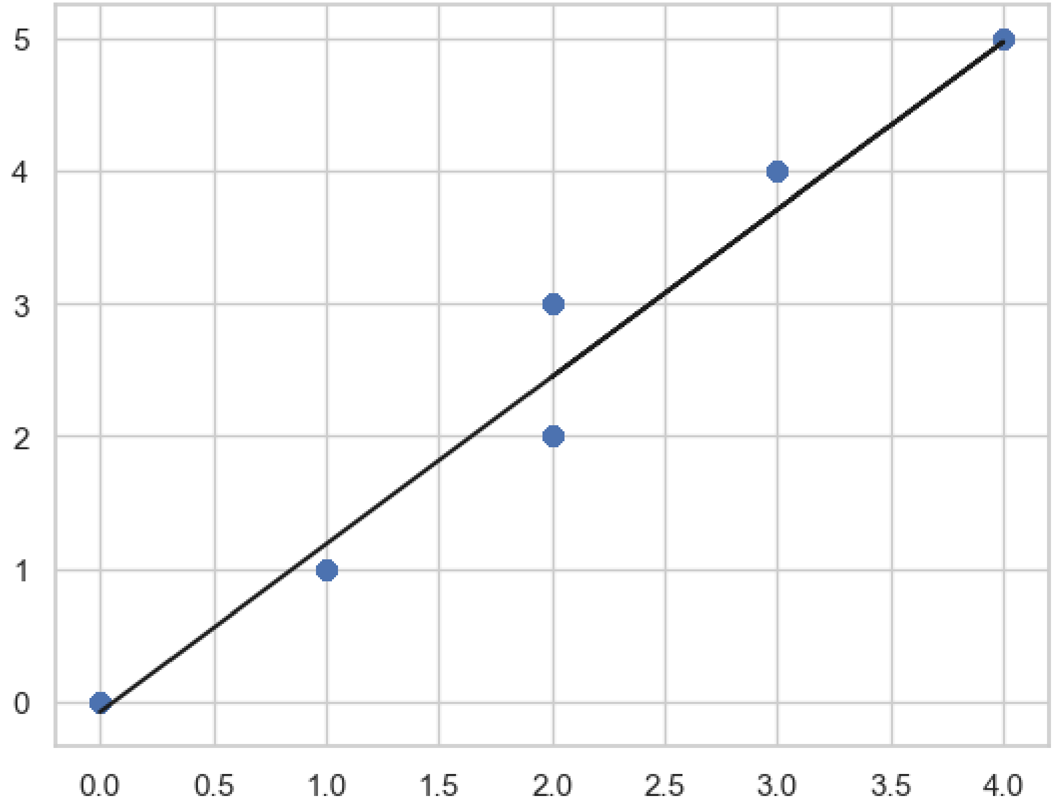
In [49]:

```
#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))
```

```
0.968457395147943
```

In [50]:

```
#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 [51]:

```
from sklearn.linear_model import Ridge, RidgeCV, Lasso
```

In [52]:

```
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
 0.965550666171003
0.9684557959564107
```

Lasso regression

In [53]:

```
#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 -0.0014072421377879785
```

Linear regression

In [61]:

```
df500=pd.read_csv(r"C:\Users\sowmika\Downloads\Copy of Test_set.csv")
df500
```

Out[61]:

	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

In [62]:

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

Out[62]:

	Source	Destination
0	2	3
1	1	1
2	0	0
3	1	1
4	2	3
...	...	...
10678	1	1
10679	1	1
10680	2	2
10681	2	3
10682	0	0

10682 rows × 2 columns

In [63]:

```
df500=df500[:][:500]
df500
```

Out[63]:

	Source	Destination
0	2	3
1	1	1
2	0	0
3	1	1
4	2	3
...	...	...
495	1	1
496	0	0
497	0	0
498	1	1
499	0	0

500 rows × 2 columns

In [65]:

```
df500=df500[['Source','Destination']]
df500
```

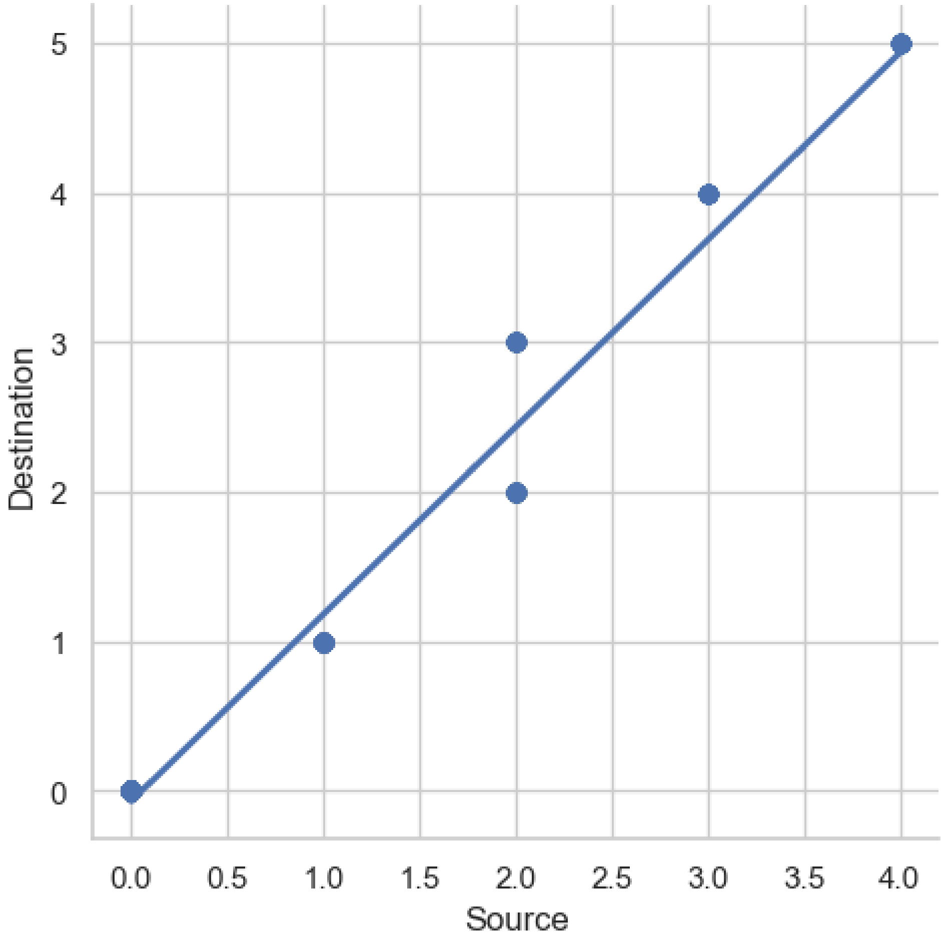
Out[65]:

	Source	Destination
0	2	3
1	1	1
2	0	0
3	1	1
4	2	3
...	...	...
495	1	1
496	0	0
497	0	0
498	1	1
499	0	0

500 rows × 2 columns

In [66]:

```
sns.lmplot(x="Source",y="Destination",data=df500,order=1,ci=None)
plt.show()
```



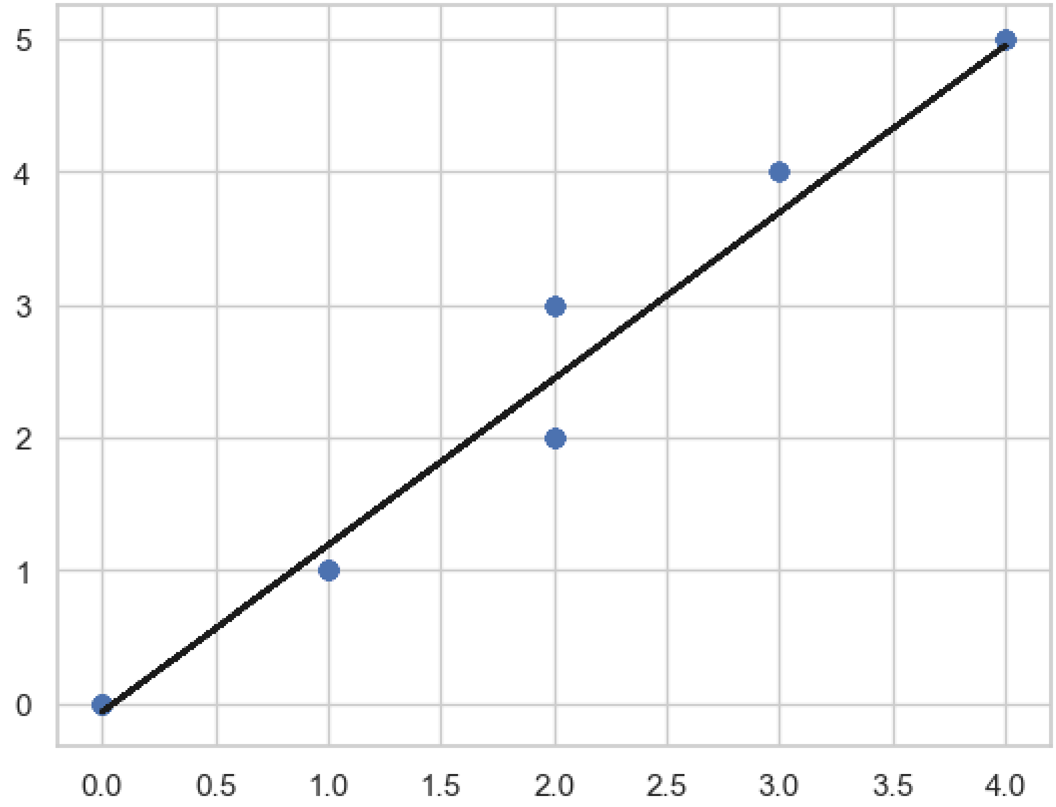
In [67]:

```
x=np.array(df500['Source']).reshape(-1,1)
y=np.array(df500['Destination']).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: 0.9732807720465518
```

In [68]:

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



Logistic regression

In [69]:

```
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 [70]:

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

Out[70]:

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

In [71]:

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

```
0.9110764430577223
```

Decision Tree

In [72]:

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

Out[72]:

```
▾        DecisionTreeClassifier

DecisionTreeClassifier(random_state=0)
```

In [73]:

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

```
0.9110764430577223
```

Random Classifier

In [74]:

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

Out[74]:

```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [75]:

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

In [76]:

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

In [77]:

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

Out[77]:

```
▸            GridSearchCV
▸ estimator: RandomForestClassifier
      ▸ RandomForestClassifier
```

In [78]:

```
grid_search.best_score_
```

Out[78]:

```
0.9134679490013939
```

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

In [ ]:

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

In [ ]:

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

In [ ]:

```
test_df["Destination"].value_counts()
```

In [ ]:

```
convert={"Destination":{"Cochin":0,"Banglore":1,"Delhi":2,"New Delhi":3,"Hyderabad":4,"K
test_df=test_df.replace(convert)
test_df
```

In [ ]:

```
test_df["Destination"].value_counts()
```

In [ ]:

```
test_df=test_df[['Source','Destination']]
test_df
```

In [ ]:

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

In [ ]:

```
test_df.describe()
```

In [ ]:

```
test_df.info()
```

In [ ]:

```
#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
```

In [ ]:

```
#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))
```

In [ ]:

```
#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 [ ]:

```
from sklearn.linear_model import Ridge, RidgeCV, Lasso
```

In [ ]:

```
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)
```

LASSO REGRESSION

In [ ]:

```
#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))
```

LOGISTIC REGRESSION:

```
            TO CHECK BEST FIT WE ARE GOING TO DO LOGISTIC REGRESSION
```

In [ ]:

```
#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 [ ]:

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

In [ ]:

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

CONCLUSION:

In [ ]:

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

In [ ]:

```

```