

```
In [3]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [4]: data=pd.read_csv("/home/placement/Desktop/nio/fiat500.csv")
# data reading from the file
```

```
In [5]: data.describe()# describing the data
```

Out[5]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
<b>count</b>	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000
<b>mean</b>	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	11.563428	8576.003901
<b>std</b>	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	2.328190	1939.958641
<b>min</b>	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	7.245400	2500.000000
<b>25%</b>	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	9.505090	7122.500000
<b>50%</b>	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	11.869260	9000.000000
<b>75%</b>	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	12.769040	10000.000000
<b>max</b>	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	18.365520	11100.000000

In [6]: data

Out[6]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	lounge	51	882	25000	1	44.907242	8.611560	8900
1	2	pop	51	1186	32500	1	45.666359	12.241890	8800
2	3	sport	74	4658	142228	1	45.503300	11.417840	4200
3	4	lounge	51	2739	160000	1	40.633171	17.634609	6000
4	5	pop	73	3074	106880	1	41.903221	12.495650	5700
...	...	...	...	...	...	...	...	...	...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 9 columns

In [ ]:

In [7]: data=pd.get\_dummies(data) *# to get all the dummies presentt in the dATA*

In [8]: data1=data.drop(['lat','lon','ID'],axis=1) *# DROP OR ELIMINATES ALL THE DESIRED COLS*

In [9]: data1

Out[9]:

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_sport
0	51	882	25000	1	8900	1	0	0
1	51	1186	32500	1	8800	0	1	0
2	74	4658	142228	1	4200	0	0	1
3	51	2739	160000	1	6000	1	0	0
4	73	3074	106880	1	5700	0	1	0
...	...	...	...	...	...	...	...	...
1533	51	3712	115280	1	5200	0	0	1
1534	74	3835	112000	1	4600	1	0	0
1535	51	2223	60457	1	7500	0	1	0
1536	51	2557	80750	1	5990	1	0	0
1537	51	1766	54276	1	7900	0	1	0

1538 rows × 8 columns

In [10]: list(data)

Out[10]: ['ID',  
'engine\_power',  
'age\_in\_days',  
'km',  
'previous\_owners',  
'lat',  
'lon',  
'price',  
'model\_lounge',  
'model\_pop',  
'model\_sport']

```
In [11]: list(data1)
```

```
Out[11]: ['engine_power',
          'age_in_days',
          'km',
          'previous_owners',
          'price',
          'model_lounge',
          'model_pop',
          'model_sport']
```

```
In [12]: data1=pd.get_dummies(data1) # GET THE ALL DUMMIES change string into integers
```

```
In [13]: data1
```

```
Out[13]:
```

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_sport
0	51	882	25000	1	8900	1	0	0
1	51	1186	32500	1	8800	0	1	0
2	74	4658	142228	1	4200	0	0	1
3	51	2739	160000	1	6000	1	0	0
4	73	3074	106880	1	5700	0	1	0
...	...	...	...	...	...	...	...	...
1533	51	3712	115280	1	5200	0	0	1
1534	74	3835	112000	1	4600	1	0	0
1535	51	2223	60457	1	7500	0	1	0
1536	51	2557	80750	1	5990	1	0	0
1537	51	1766	54276	1	7900	0	1	0

1538 rows × 8 columns

```
In [14]: data1.shape
```

```
Out[14]: (1538, 8)
```

```
In [15]: data1.groupby(['previous_owners']).count()
```

```
Out[15]:
```

	engine_power	age_in_days	km	price	model_lounge	model_pop	model_sport
previous_owners							
1	1389	1389	1389	1389	1389	1389	1389
2	117	117	117	117	117	117	117
3	23	23	23	23	23	23	23
4	9	9	9	9	9	9	9

```
In [ ]:
```

```
In [16]: data1
```

```
Out[16]:
```

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_sport
0	51	882	25000	1	8900	1	0	0
1	51	1186	32500	1	8800	0	1	0
2	74	4658	142228	1	4200	0	0	1
3	51	2739	160000	1	6000	1	0	0
4	73	3074	106880	1	5700	0	1	0
...	...	...	...	...	...	...	...	...
1533	51	3712	115280	1	5200	0	0	1
1534	74	3835	112000	1	4600	1	0	0
1535	51	2223	60457	1	7500	0	1	0
1536	51	2557	80750	1	5990	1	0	0
1537	51	1766	54276	1	7900	0	1	0

1538 rows × 8 columns

```
In [17]: y=data1["price"]
```

```
In [18]: x=data1.drop('price',axis=1)
```

In [19]:

x

Out[19]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
0	51	882	25000	1	1	0	0
1	51	1186	32500	1	0	1	0
2	74	4658	142228	1	0	0	1
3	51	2739	160000	1	1	0	0
4	73	3074	106880	1	0	1	0
...	...	...	...	...	...	...	...
1533	51	3712	115280	1	0	0	1
1534	74	3835	112000	1	1	0	0
1535	51	2223	60457	1	0	1	0
1536	51	2557	80750	1	1	0	0
1537	51	1766	54276	1	0	1	0

1538 rows × 7 columns

In [20]:

y

```
Out[20]: 0      8900
         1      8800
         2      4200
         3      6000
         4      5700
         ...
        1533    5200
        1534    4600
        1535    7500
        1536    5990
        1537    7900
```

Name: price, Length: 1538, dtype: int64

```
In [21]: from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
#THE DATA WILL BE SPITTED INTO TWO TYPES TRAINING DATA AND TESTING DATA
#66% TRAINING DATA 33% TESTING DATA
```

In [22]: x\_train.head(5)

Out[22]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
527	51	425	13111	1	1	0	0
129	51	1127	21400	1	1	0	0
602	51	2039	57039	1	0	1	0
331	51	1155	40700	1	1	0	0
323	51	425	16783	1	1	0	0

```
In [23]: #data['model']=data['model'].map({"lounge":1,"pop":2,"sport":3})
#if getdummes not working we can work it as given in the aboveline to chage strings into integers
```



In [24]: data

Out[24]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price	model_lounge	model_pop	model_sport
0	1	51	882	25000	1	44.907242	8.611560	8900	1	0	0
1	2	51	1186	32500	1	45.666359	12.241890	8800	0	1	0
2	3	74	4658	142228	1	45.503300	11.417840	4200	0	0	1
3	4	51	2739	160000	1	40.633171	17.634609	6000	1	0	0
4	5	73	3074	106880	1	41.903221	12.495650	5700	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
1533	1534	51	3712	115280	1	45.069679	7.704920	5200	0	0	1
1534	1535	74	3835	112000	1	45.845692	8.666870	4600	1	0	0
1535	1536	51	2223	60457	1	45.481541	9.413480	7500	0	1	0
1536	1537	51	2557	80750	1	45.000702	7.682270	5990	1	0	0
1537	1538	51	1766	54276	1	40.323410	17.568270	7900	0	1	0

1538 rows × 11 columns

In [ ]:

```
In [55]: from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(x_train,y_train)
```

Out[55]:

```
▼ LinearRegression
LinearRegression()
```

```
In [56]: ypred=reg.predict(x_test)
```

In [57]: ypred

```
9686.269121 , 10463.56977746, 10133.15815395, 9762.80613855,  
9793.03056946, 6796.69068198, 9599.3262671 , 8488.31539047,  
6705.66818403, 10307.58651641, 10045.18332239, 10120.36242166,  
5836.93199112, 8772.49782933, 9680.77538859, 5719.87463854,  
8398.59735084, 9680.77538859, 4334.81943405, 10015.00600846,  
9850.72458719, 7864.73798641, 10072.71245374, 10552.64805598,  
10253.47474908, 6861.80736606, 6484.22649656, 10374.62123623,  
8426.37409382, 5447.47569851, 9914.20077691, 4687.39013431,  
7885.32100747, 5431.00822998, 9911.86294348, 10390.16991322,  
9680.84745901, 8844.57815539, 7764.08471024, 4257.54640953,  
9882.76503303, 10341.35258769, 5736.4484335 , 10179.87154436,  
9501.423448 , 7997.3181334 , 5532.33458288, 9894.57834738,  
10437.97459358, 6381.35845844, 9591.23555726, 9574.27908517,  
10322.30715736, 9501.22785499, 9789.955758 , 9593.26549752,  
6775.82788536, 7915.34831306, 10389.98590521, 10351.58343315,  
7381.32686464, 9966.53983093, 10430.87188433, 10554.43156462,  
10285.85574963, 10035.88086558, 9526.63034431, 7742.78157141,  
9297.64938364, 10051.42272678, 10004.81256571, 9985.84167026,  
9374.6573594 , 9561.57499854, 9754.94184269, 9819.85893758,  
8780.31447831. 6255.99008069. 6281.53627686. 8190.88781577.
```

In [58]: `from sklearn.metrics import r2_score`  
`r2_score(y_test,ypred)`

Out[58]: 0.8415526986865394

In [59]: `from sklearn.metrics import mean_squared_error`  
`mean_squared_error(ypred,y_test)`

Out[59]: 581887.727391353

In [60]: `print(mean_squared_error(ypred,y_test)**(1/2))# PRINTS THE SQUARE ROOT OF MEAN SQUARE ERROR`  
762.8156575420782

In [61]: ypred

```
9501.423448 , 7997.3181334 , 5532.33458288 , 9894.57834738 ,  
10437.97459358 , 6381.35845844 , 9591.23555726 , 9574.27908517 ,  
10322.30715736 , 9501.22785499 , 9789.955758 , 9593.26549752 ,  
6775.82788536 , 7915.34831306 , 10389.98590521 , 10351.58343315 ,  
7381.32686464 , 9966.53983093 , 10430.87188433 , 10554.43156462 ,  
10285.85574963 , 10035.88086558 , 9526.63034431 , 7742.78157141 ,  
9297.64938364 , 10051.42272678 , 10004.81256571 , 9985.84167026 ,  
9374.6573594 , 9561.57499854 , 9754.94184269 , 9819.85893758 ,  
8780.31447831 , 6255.99008069 , 6281.53627686 , 8190.88781577 ,  
8588.91394592 , 6566.97963218 , 6850.70237466 , 5511.29438169 ,  
8119.97866315 , 9847.74830838 , 7775.93862032 , 9875.05509733 ,  
10121.29366536 , 5791.92464084 , 9835.42728501 , 10043.91426822 ,  
8027.28015259 , 4527.22080416 , 10609.02444098 , 3808.29240951 ,  
9952.37340054 , 10511.20945172 , 5746.34019592 , 5486.40214756 ,  
10395.91036208 , 6788.47519216 , 8953.20120295 , 10442.24187982 ,  
9455.6934072 , 9976.26574762 , 8528.35753837 , 7960.77147517 ,  
10400.05054235 , 5359.97362399 , 9899.4913613 , 10203.35814213 ,  
10303.33499967 , 9507.16596227 , 9151.43928526 , 9805.06469343 ,  
5661.99787503 , 4904.40690461 , 4742.8827765 , 9663.32864144 ,  
6102.05217222 , 8070.62050125 , 10066.06016211 , 5001.21201171
```

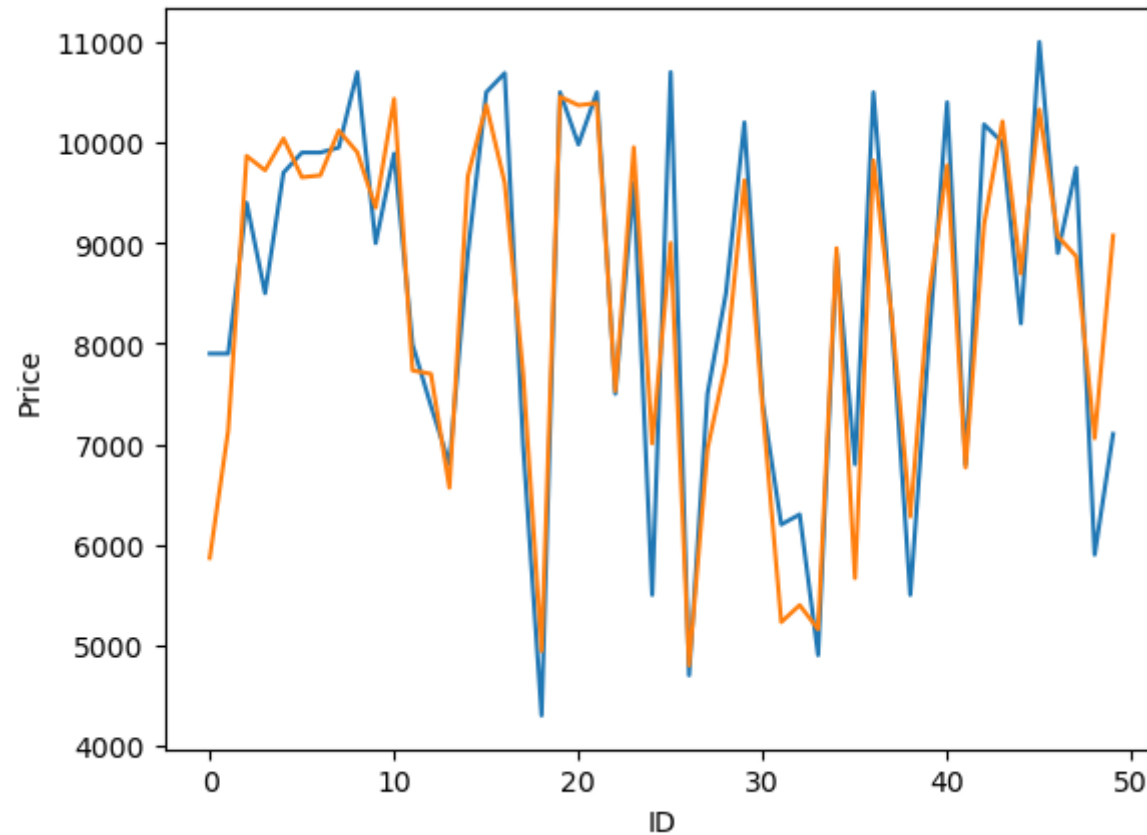
```
In [62]: results=pd.DataFrame(columns=['Price', 'Predicted'])
results['Price']=y_test
results['Predicted']=ypred
results=results.reset_index()
results['ID']=results.index
results.head(15)
```

Out[62]:

	index	Price	Predicted	ID
0	481	7900	5867.650338	0
1	76	7900	7133.701423	1
2	1502	9400	9866.357762	2
3	669	8500	9723.288745	3
4	1409	9700	10039.591012	4
5	1414	9900	9654.075826	5
6	1089	9900	9673.145630	6
7	1507	9950	10118.707281	7
8	970	10700	9903.859527	8
9	1198	8999	9351.558284	9
10	1088	9890	10434.349636	10
11	576	7990	7732.262557	11
12	965	7380	7698.672401	12
13	1488	6800	6565.952404	13
14	1432	8900	9662.901035	14

```
In [66]: sns.lineplot(x='ID',y='Price',data=results.head(50))  
sns.lineplot(x='ID',y='Predicted',data=results.head(50))  
plt.plot()
```

Out[66]: []



In [ ]:

```
In [38]: results['dif']=results.apply(lambda row: row.Price-row.Predicted,axis=1)
```

```
In [39]: results
```

```
Out[39]:
```

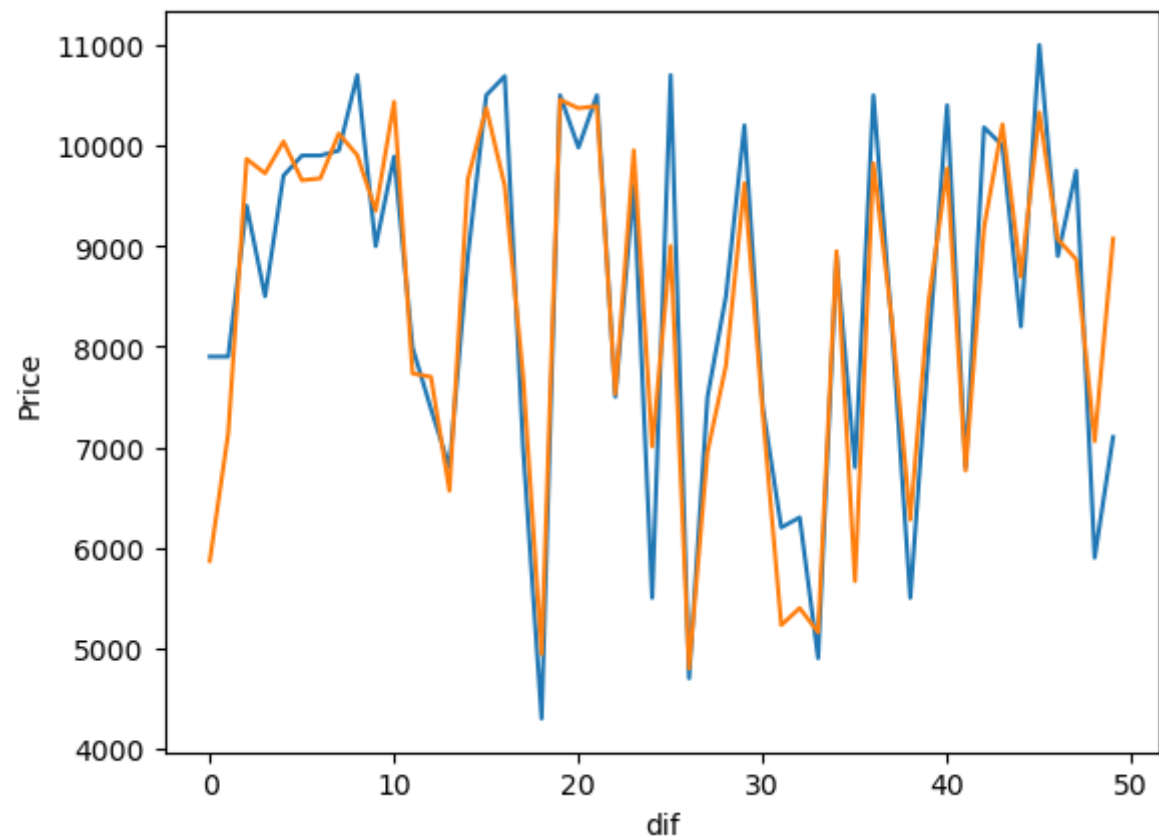
	Price	Predicted	dif
481	7900	5867.650338	2032.349662
76	7900	7133.701423	766.298577
1502	9400	9866.357762	-466.357762
669	8500	9723.288745	-1223.288745
1409	9700	10039.591012	-339.591012
...	...	...	...
291	10900	10032.665135	867.334865
596	5699	6281.536277	-582.536277
1489	9500	9986.327508	-486.327508
1436	6990	8381.517020	-1391.517020
575	10900	10371.142553	528.857447

508 rows × 3 columns

```
In [69]: results['dif']=results.index
```

```
In [70]: sns.lineplot(x='dif',y='Price',data=results.head(50))  
sns.lineplot(x='dif',y='Predicted',data=results.head(50))  
plt.plot()
```

Out[70]: []



## # RIDGE MODEL FOR THE ABOVE DATA

```
In [40]: import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge

alpha = [1e-15, 1e-10, 1e-8, 1e-4, 1e-3, 1e-2, 1, 5, 10, 20, 30]

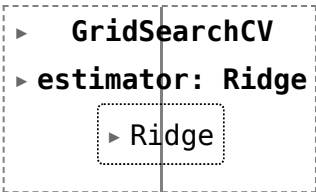
ridge = Ridge()

parameters = {'alpha': alpha}

ridge_regressor = GridSearchCV(ridge, parameters)

ridge_regressor.fit(x_train, y_train)
```

```
Out[40]:
```



```
  ▶ GridSearchCV
  ▶ estimator: Ridge
    ▶ Ridge
```

```
In [41]: ridge_regressor.best_params_
```

```
Out[41]: {'alpha': 30}
```

```
In [42]: ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

```
In [43]: Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

```
Out[43]: 579521.7970897449
```



```
In [44]: from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)
```

```
Out[44]: 0.8421969385523054
```

```
In [45]: data
```

```
Out[45]:
```

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price	model_lounge	model_pop	model_sport
0	1	51	882	25000	1	44.907242	8.611560	8900	1	0	0
1	2	51	1186	32500	1	45.666359	12.241890	8800	0	1	0
2	3	74	4658	142228	1	45.503300	11.417840	4200	0	0	1
3	4	51	2739	160000	1	40.633171	17.634609	6000	1	0	0
4	5	73	3074	106880	1	41.903221	12.495650	5700	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
1533	1534	51	3712	115280	1	45.069679	7.704920	5200	0	0	1
1534	1535	74	3835	112000	1	45.845692	8.666870	4600	1	0	0
1535	1536	51	2223	60457	1	45.481541	9.413480	7500	0	1	0
1536	1537	51	2557	80750	1	45.000702	7.682270	5990	1	0	0
1537	1538	51	1766	54276	1	40.323410	17.568270	7900	0	1	0

1538 rows × 11 columns

## # LINEAR REGRESSION FOR ONLY MODEL==LOUNGE

```
data=pd.read_csv("/home/placement/Desktop/nio/fiat500.csv")
# data reading from the file
```

```
# data2=data.loc[(data.model=='lounge')]
```

```
data2
```

```
data3=data2.drop(['lat','lon','ID'],axis=1)
```

```
# data3
```

```
data3=pd.get_dummies(data3) # to get all the dummies presentt in the dATA
```

```
data3
```

```
list(data3)
```

```
y=data3['price']
```

```
x=data3.drop(["price"],axis=1)
```

```
from sklearn.model_selection import train_test_split  
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)  
#THE DATA WILL BE SPITTED INTO TWO TYPES TRAINING DATA AND TESTING DATA  
#66% TRAINING DATA 33% TESTING DATA
```

```
y
```

```
x_train.head(5)
```

```
from sklearn.linear_model import LinearRegression  
reg = LinearRegression()  
reg.fit(x_train,y_train)
```

```
ypred=reg.predict(x_test)
```

```
ypred
```

```
from sklearn.metrics import r2_score  
r2_score(y_test,ypred)
```

```
from sklearn.metrics import mean_squared_error  
mean_squared_error(ypred,y_test)
```

```
print(mean_squared_error(ypred,y_test)**(1/2))# PRINTS THE MEAN SQUARE ERROR
```

```
# plotting a graph for the ridge predicted & actual values data
```

```
In [54]: Results=pd.DataFrame(columns=['actual','predicted'])
Results['actual']=y_test
Results['predicted']=y_pred_ridge
Results=Results.reset_index()
Results['ID']=Results.index
Results.head(25)
```

Out[54]:

	index	actual	predicted	ID
0	481	7900	5869.741155	0
1	76	7900	7149.563327	1
2	1502	9400	9862.785355	2
3	669	8500	9719.283532	3
4	1409	9700	10035.895686	4
5	1414	9900	9650.311090	5
6	1089	9900	9669.183317	6
7	1507	9950	10115.128380	7
8	970	10700	9900.241944	8
9	1198	8999	9347.080772	9
10	1088	9890	10431.237961	10
11	576	7990	7725.756431	11
12	965	7380	7691.089846	12
13	1488	6800	6583.674680	13
14	1432	8900	9659.240069	14
15	380	10500	10370.231518	15
16	754	10690	9620.427488	16
17	30	6990	7689.189244	17
18	49	4300	4954.595074	18
19	240	10500	10452.262871	19
20	344	9980	10353.107796	20

	index	actual	predicted	ID
21	354	10500	10388.635632	21
22	124	7500	7503.302407	22
23	383	9600	9948.970588	23
24	1389	5500	7009.047336	24

```
In [47]: sns.lineplot(x='ID',y="actual",data=Results.head(50))  
sns.lineplot(x='ID',y='predicted',data=Results.head(50))  
plt.plot()
```

Out[47]: []

