REDGE RESSION MODEL:

USING FIAT500.CSV

In [169]:
```
import pandas as rr
```

In [170]:
```
data=rr.read_csv("/home/placement/Desktop/ramaraju/fiat500.csv")
```

In [171]:
```
data
```

Out[171]:

	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 [172]:
```
data.describe()
```

Out[172]:

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

In [173]:
```
data.info
```

Out[173]:
```
<bound method DataFrame.info of          ID   model  engine_power  age_in_days      km  previous_owners  \
0        1  lounge            51          882   25000                1
1        2     pop            51         1186   32500                1
2        3   sport            74         4658  142228                1
3        4  lounge            51         2739  160000                1
4        5     pop            73         3074  106880                1
...    ...     ...           ...          ...     ...              ...
1533  1534   sport            51         3712  115280                1
1534  1535  lounge            74         3835  112000                1
1535  1536     pop            51         2223   60457                1
1536  1537  lounge            51         2557   80750                1
1537  1538     pop            51         1766   54276                1

            lat        lon  price
0     44.907242   8.611560   8900
1     45.666359  12.241890   8800
2     45.503300  11.417840   4200
3     40.633171  17.634609   6000
4     41.903221  12.495650   5700
...         ...        ...    ...
1533  45.069679   7.704920   5200
1534  45.845692   8.666870   4600
1535  45.481541   9.413480   7500
1536  45.000702   7.682270   5990
1537  40.323410  17.568270   7900

[1538 rows x 9 columns]>
```

In [174]:
```
data1=data.drop(["lat","lon","ID"],axis=1)
```

In [175]:
```
data1=rr.get_dummies(data1)
```

In [176]:
```
data1
```

Out[176]:

	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 [177]:
```
z=data1.loc[(data.model=="lounge")]
```

In [178]:

```
z
```

Out[178]:

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_sport
0	51	882	25000	1	8900	1	0	0
3	51	2739	160000	1	6000	1	0	0
6	51	731	11600	1	10750	1	0	0
7	51	1521	49076	1	9190	1	0	0
11	51	366	17500	1	10990	1	0	0
...	...	...	...	...	...	...	...	...
1528	51	2861	126000	1	5500	1	0	0
1529	51	731	22551	1	9900	1	0	0
1530	51	670	29000	1	10800	1	0	0
1534	74	3835	112000	1	4600	1	0	0
1536	51	2557	80750	1	5990	1	0	0

1094 rows × 8 columns

In [179]:

```
y=z['price']
x=z.drop('price',axis=1)
```

In [ ]:

```
```

In [180]:
```
y
```

Out[180]:
```
0        8900
3        6000
6       10750
7        9190
11      10990
        ...
1528     5500
1529     9900
1530    10800
1534     4600
1536     5990
Name: price, Length: 1094, dtype: int64
```

In [181]:
```
x
```

Out[181]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
0	51	882	25000	1	1	0	0
3	51	2739	160000	1	1	0	0
6	51	731	11600	1	1	0	0
7	51	1521	49076	1	1	0	0
11	51	366	17500	1	1	0	0
...	...	...	...	...	...	...	...
1528	51	2861	126000	1	1	0	0
1529	51	731	22551	1	1	0	0
1530	51	670	29000	1	1	0	0
1534	74	3835	112000	1	1	0	0
1536	51	2557	80750	1	1	0	0

1094 rows × 7 columns

In [182]:

```
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)
```

In [183]:

```
y_train.head(5)
```

Out[183]:

```
441      8980
701     10300
695      5880
1415    10490
404      9499
Name: price, dtype: int64
```

In [184]:

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

Out[184]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
441	51	762	36448	1	1	0	0
701	51	701	27100	1	1	0	0
695	51	3197	51083	1	1	0	0
1415	51	670	33000	1	1	0	0
404	51	456	14000	1	1	0	0

In [185]:
```
data1=data.drop(["lat","lon","ID"],axis=1)
data1
```

Out[185]:

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

1538 rows × 6 columns

In [186]:
```
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)
```

```
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.1816e-27): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.23704e-27): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.18103e-27): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.21179e-27): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.23074e-27): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.1816e-22): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.23704e-22): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.18103e-22): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.21179e-22): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
```

```
ll-conditioned matrix (rcond=1.23074e-22): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.1816e-20): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.23704e-20): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.18103e-20): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.21179e-20): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=1.23074e-20): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

Out[186]:

```
▸ GridSearchCV
▸ estimator: Ridge
  ▸ Ridge
```

In [187]:

```
ridge_regressor.best_params_
```

Out[187]:

```
{'alpha': 30}
```

In [188]:

```
ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

In [189]:

```
from sklearn.metrics import mean_squared_error
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[189]:

```
519771.8129989745
```

In [190]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)
```

Out[190]: 0.8373030813683994

In [191]:
```
Results=rr.DataFrame(columns=['price','predicted'])
Results['price']=y_test
Results['predicted']=y_pred_ridge
Reesults=Results.reset_index()
Results['ID']=Results.index
Results.head(15)
```

Out[191]:

	price	predicted	ID
676	10250	10045.347779	676
215	9790	9989.171535	215
146	5500	4769.099603	146
1319	9900	10048.683238	1319
1041	8900	9813.944798	1041
1425	9500	8678.143561	1425
409	10450	10173.797921	409
617	9790	10180.627008	617
1526	9300	9107.315259	1526
1010	4600	5625.007407	1010
1301	10000	10565.711088	1301
923	6900	6776.128155	923
1200	9500	9677.360191	1200
845	10500	10348.971360	845
799	7450	8049.201047	799

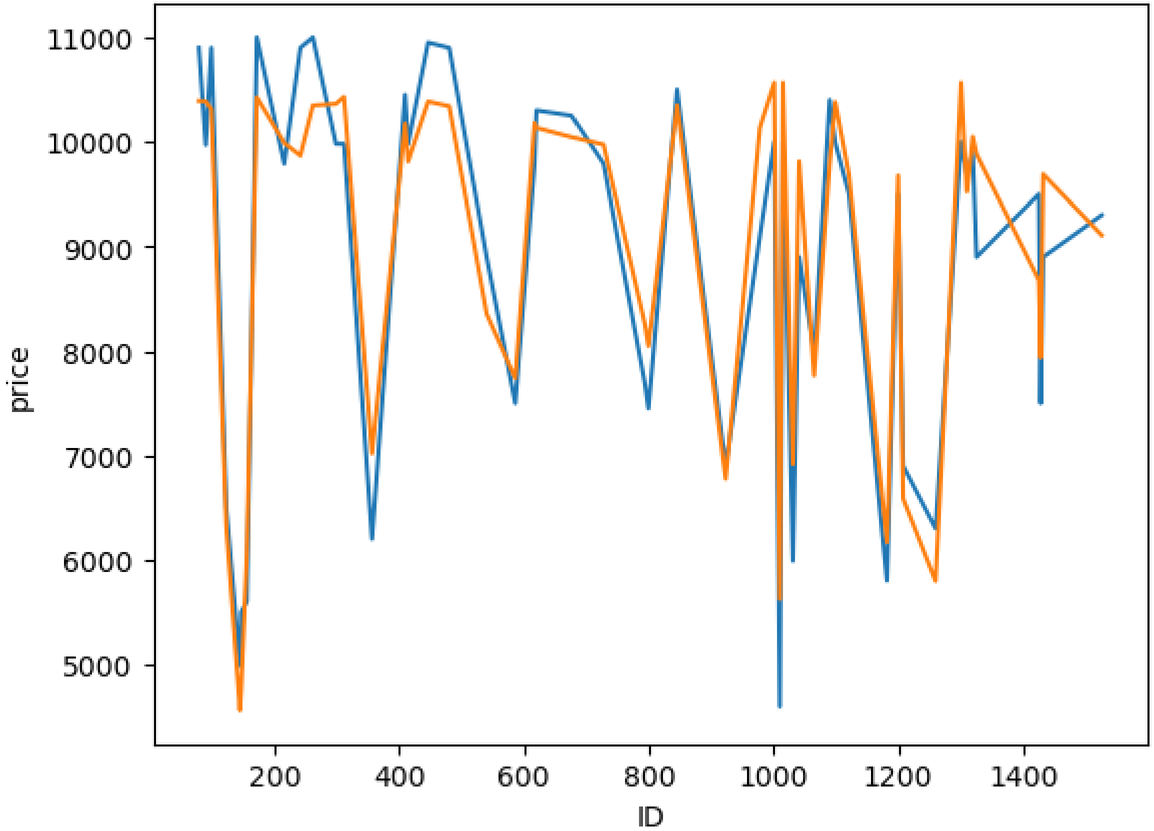
In [192]:
```
#extract column syntax.

#data2=x.loc[:,"model_lounge"]
```

In [196]:
```
#data2
```

In [ ]:
```
#data2=rr.get_dummies(data2)  =>produce("model_longue")
```

In [198]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='ID',y='price',data=Results.head(50))
sns.lineplot(x='ID',y='predicted',data=Results.head(50))
plt.plot()
```

Out[198]: []



In [ ]: