In [184]:
```
import pandas as pd
```

In [185]:
```
data=pd.read_csv("/home/placement/Downloads/fiat500.csv")#read data from file as csv
```

In [186]:
```
data.describe()#describe data like count,mean,max value
```

Out[186]:

	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 [187]:

```
data.tail(10)#shows last 10 rows
```

Out[187]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
1528	1529	lounge	51	2861	126000	1	43.841980	10.51531	5500
1529	1530	lounge	51	731	22551	1	38.122070	13.36112	9900
1530	1531	lounge	51	670	29000	1	45.764648	8.99450	10800
1531	1532	sport	73	4505	127000	1	45.528511	9.59323	4750
1532	1533	pop	51	1917	52008	1	45.548000	11.54947	9900
1533	1534	sport	51	3712	115280	1	45.069679	7.70492	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.41348	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.56827	7900

In [188]:

```
data1=data.drop(['lat','ID','lon'],axis=1)#remove column of lat,id,lon by using drop function
```

```
In [189]: data1
```

```
Out[189]:
        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 [190]:
```
data2=pd.get_dummies(data1)#where the lounge model it shows"1" other models it shows "0"
data2
```

Out[190]:

	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 [191]:
```
data2.shape#how many rows and columns in the data frame
```

Out[191]:
```
(1538, 8)
```

In [192]:
```
z=data2.loc[(data.model=='lounge')]#determine only for lounge cars
```

In [193]:
```
z
```

Out[193]:

	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 [194]:
```
y=z['price']#removing price from data2 and put in new data frame y
x=z.drop(['price'],axis=1)#remaining data can be put in another data fram
```

In [195]:
```
x# to get data in x
```

Out[195]:

	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 [196]:
```
y# to get data in y
```

Out[196]:
```
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 [197]:
```
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)#dividing training data ar
```

In [198]:
```
x_test.head(5)#display top 5 data in testing data
```

Out[198]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
676	51	762	18609	1	1	0	0
215	51	701	25000	1	1	0	0
146	51	4018	152900	1	1	0	0
1319	51	731	20025	1	1	0	0
1041	51	640	38231	1	1	0	0

In [199]:
```
y_test.head(5)#display top 5 data in testing data price dataframe
```

Out[199]:
```
676     10250
215      9790
146      5500
1319     9900
1041     8900
Name: price, dtype: int64
```

In [200]:
```
x_train.head(5)#display top 5 data in training data
```

Out[200]:

	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 [201]:
```
y_train.head(5)#display top 5 data in training data price dataframe
```

Out[201]:
```
441      8980
701     10300
695      5880
1415    10490
404      9499
Name: price, dtype: int64
```

In [202]:
```
from sklearn.model_selection import GridSearchCV#ridge  regression
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[202]:
```
GridSearchCV(estimator=Ridge(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20, 30]})
```

In [203]:
```
ridge_regressor.best_params_#alpha value
```

Out[203]:
```
{'alpha': 30}
```

In [204]:
```
ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

In [205]:
```
from sklearn.metrics import mean_squared_error#mean_squared error
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[205]:
```
519771.8129989742
```

In [206]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)#finding the efficieny
```

Out[206]:
```
0.8373030813683995
```

In [220]:

```
Results=pd.DataFrame(columns=['Actual','predicted'])##creating a data frame called results for given price a
Results['Actual']=y_test
Results['predicted']=y_pred_ridge
Results=Results.reset_index()
Results['ID']=Results.index
Results.head(10)
```

Out[220]:

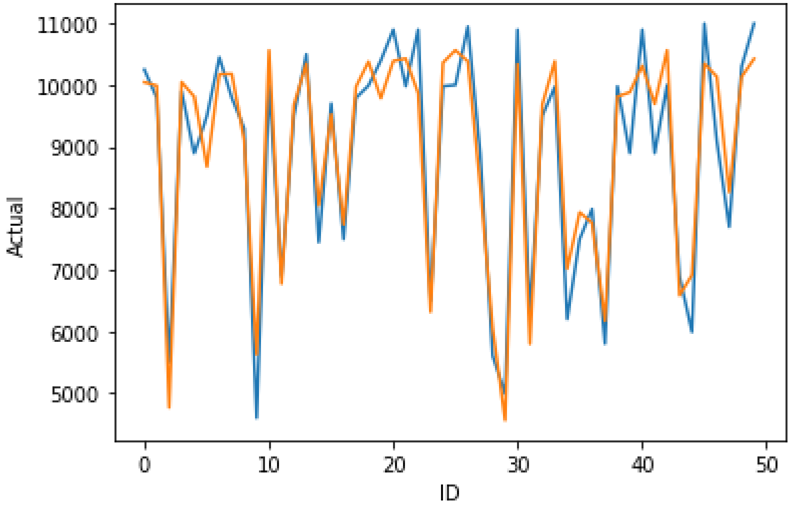
	index	Actual	predicted	ID
0	676	10250	10045.347779	0
1	215	9790	9989.171535	1
2	146	5500	4769.099603	2
3	1319	9900	10048.683238	3
4	1041	8900	9813.944798	4
5	1425	9500	8678.143561	5
6	409	10450	10173.797921	6
7	617	9790	10180.627008	7
8	1526	9300	9107.315259	8
9	1010	4600	5625.007407	9

In [221]:

```
import seaborn as sns#putting the graoh for actual price and predicted price
import matplotlib.pyplot as plt
sns.lineplot(x='ID',y='Actual',data=Results.head(50))
sns.lineplot(x='ID',y='predicted',data=Results.head(50))
plt.plot()
```

Out[221]:

```
[]
```

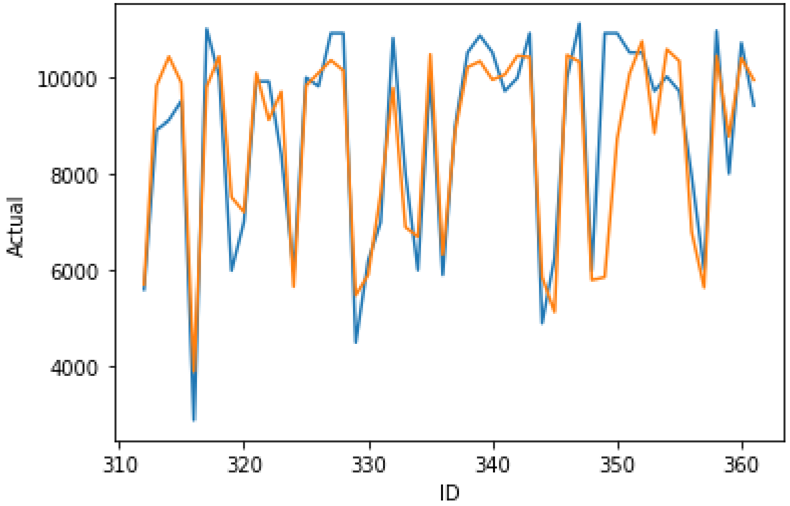


In [222]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='ID',y='Actual',data=Results.tail(50))
sns.lineplot(x='ID',y='predicted',data=Results.tail(50))
plt.plot()
```

Out[222]:

```
[]
```



In [ ]:

```

```