In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

# ridge

In [3]:
```
data=pd.read_csv("/home/placement/Desktop/nio/fiat500.csv")
# data reading from the file
```

In [4]:
```
data.describe()# describing the data
```

Out[4]:

| | 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 [5]:
```
data
```

Out[5]:

| | 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 [6]:
```
data=pd.get_dummies(data) # to get all the dummies presentt in the dATA
```

In [7]:
```
data1=data.drop(['lat','lon','ID'],axis=1) # DROP OR ELIMINATES ALL THE DESIRED COLS
```

In [8]:
```
data1
```

Out[8]:

| | 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 [9]:
```
list(data)
```

Out[9]:
```
['ID',
 'engine_power',
 'age_in_days',
 'km',
 'previous_owners',
 'lat',
 'lon',
 'price',
 'model_lounge',
 'model_pop',
 'model_sport']
```

In [10]:
```
list(data1)
```

Out[10]:
```
['engine_power',
 'age_in_days',
 'km',
 'previous_owners',
 'price',
 'model_lounge',
 'model_pop',
 'model_sport']
```

In [11]:
```
data1=pd.get_dummies(data1) # GET THE ALL DUMMIES change string into integers
```

In [12]:
```
data1
```

Out[12]:

| | 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 [13]:
```
 data1.shape
```

Out[13]:
```
(1538, 8)
```

In [14]:
```
data1.groupby(['previous_owners']).count()
```

Out[14]:

| previous_owners | engine_power | age_in_days | km | price | model_lounge | model_pop | model_sport |
|---|---|---|---|---|---|---|---|
| **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 [15]:
```
y=data1["price"]
```

In [16]:
```
x=data1.drop('price',axis=1)
```

In [17]:
```
y
```

Out[17]:
```
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
```

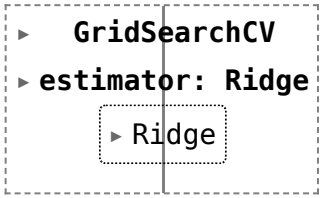
In [19]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
#THE DATA WILLL BE SPITTED INTO INO TWO TYPES TRAINING DATA AND TESTING DATA
#66% TRAINING DATA 33% TESTING DATA
```

In [20]:
```
# RIDGE MODEL FOR THE ABOVE DATA
```

In [ ]:
```
```

In [25]:
```
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[25]:




In [22]:
```
ridge_regressor.best_params_
```

Out[22]:
```
{'alpha': 30}
```

In [26]:
```
ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

In [32]:
```
from sklearn.metrics import mean_squared_error
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[32]: 579521.7970897449

In [28]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)
```

Out[28]: 0.8421969385523054

In [29]:
```
data
```

Out[29]:

| | 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 [30]:
```
Results=pd.DataFrame(columns=['actual','predicted'])# CREATING  A DATA FRAME AND INSERING COLS
Results['actual']=y_test
Results['predicted']=y_pred_ridge
Results=Results.reset_index()
Results['ID']=Results.index
Results.head(25)
```

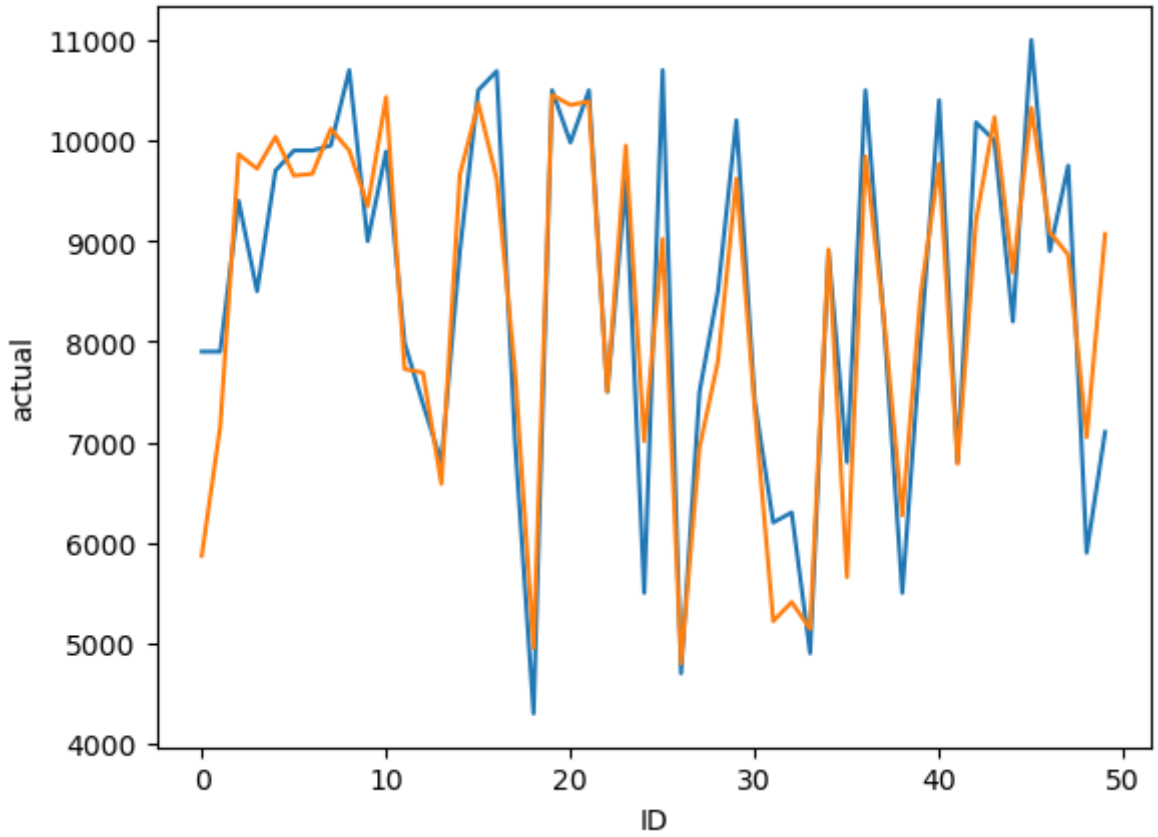
Out[30]:

| | 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 |

| | index | actual | predicted | ID |
|---|---|---|---|---|
| **20** | 344 | 9980 | 10353.107796 | 20 |
| **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 [31]:
```
sns.lineplot(x='ID',y="actual",data=Results.head(50))
sns.lineplot(x='ID',y='predicted',data=Results.head(50))
plt.plot()#THE PLOT WILL BE SHOWN FOR THE ACUAL AND PREDICTED
```

Out[31]: []




In [ ]: