Redge regression

In [1]:

```
import numpy as np
import pandas as pd
```

reading csv file

In [2]:
```
kcd=pd.read_csv("/home/placement/Downloads/fiat500.csv")
kcd.info
```

Out[2]:
```
<bound method DataFrame.info of         ID   model  engine_power  age_in_days      km  previous_o
wners  \
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 [3]:
```
kcd
```

Out[3]:

	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 [4]:
```
a=kcd.groupby(['model']).count()
a
```

Out[4]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
model								
lounge	1094	1094	1094	1094	1094	1094	1094	1094
pop	358	358	358	358	358	358	358	358
sport	86	86	86	86	86	86	86	86

removing id,lat,lon columns

In [5]:

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

Out[5]:

	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

converting strings into integers

In [6]:

```
drop['model']=drop['model'].map({'lounge':1,'pop':2,'sport':3})
drop
```

Out[6]:

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

1538 rows × 6 columns

In [7]:

```
cor1=drop.corr()
```

importing seaborn and matplot

In [8]:

```
import seaborn as sb
import matplotlib.pyplot as mp
```

extracting price column

In [9]:

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

Out[9]:

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

1538 rows × 5 columns

In [10]:

```
y
```

Out[10]:

```
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
```

test and train

In [11]:

```
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 [12]:

```
x_train.head(10)
```

Out[12]:

	model	engine_power	age_in_days	km	previous_owners
527	1	51	425	13111	1
129	1	51	1127	21400	1
602	2	51	2039	57039	1
331	1	51	1155	40700	1
323	1	51	425	16783	1
1358	1	51	762	29378	1
522	1	51	425	18443	1
584	1	51	397	11997	1
1236	1	51	2162	66900	1
535	3	51	609	35000	1

In [13]:

```
y_train.head(10)
```

Out[13]:

```
527      9990
129      9500
602      7590
331      8750
323      9100
1358    10900
522     10800
584      9999
1236     8500
535     10500
Name: price, dtype: int64
```

In [14]:
```
x_test.head(10)
```

Out[14]:

	model	engine_power	age_in_days	km	previous_owners
481	2	51	3197	120000	2
76	2	62	2101	103000	1
1502	1	51	670	32473	1
669	1	51	913	29000	1
1409	1	51	762	18800	1
1414	1	51	762	39751	1
1089	1	51	882	33160	1
1507	1	51	701	17324	1
970	1	51	701	29000	1
1198	1	51	1155	38000	1

In [15]:
```
y_test.head(10)
```

Out[15]:
```
481      7900
76       7900
1502     9400
669      8500
1409     9700
1414     9900
1089     9900
1507     9950
970     10700
1198     8999
Name: price, dtype: int64
```

```
!pip3 install scikit-learn
```

test and train

In [16]:

```
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 [17]:

```
x_test.head(10)
```

Out[17]:

	model	engine_power	age_in_days	km	previous_owners
481	2	51	3197	120000	2
76	2	62	2101	103000	1
1502	1	51	670	32473	1
669	1	51	913	29000	1
1409	1	51	762	18800	1
1414	1	51	762	39751	1
1089	1	51	882	33160	1
1507	1	51	701	17324	1
970	1	51	701	29000	1
1198	1	51	1155	38000	1

Ridge regression

In [18]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
```

In [19]:

```
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[19]:

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

In [20]:

```
ridge_regressor.best_params_
```

Out[20]:

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

In [21]:

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

Out[21]:

```
▾     Ridge
Ridge(alpha=30)
```

In [22]:
```
y_pred_ridge=ridge.predict(x_test)
y_pred_ridge
```

Out[22]:
```
array([ 5987.68298442,  7272.49041922,  9839.84769665,  9696.77540486,
       10012.04086199,  9628.2868526 ,  9646.94516016, 10090.96059178,
        9877.09434131,  9326.08898223, 10405.82324867,  7711.38762323,
        7676.22360235,  6686.18582795,  9637.23663747, 10345.19967839,
        9401.55433479,  7697.05744783,  4753.93014699, 10426.80946613,
       10340.86050971, 10363.5164179 ,  7639.33873562,  9925.71907421,
        7121.95942411,  9113.15929334,  4919.83330015,  6932.90675142,
        7788.26023667,  9598.07860192,  7313.64648609,  5213.57849239,
        5560.81625893,  5182.12147596,  8921.18592134,  5650.16555828,
        9932.50093896,  8231.7985179 ,  6263.1588039 ,  8570.95039919,
        9747.36874214,  6885.53376523,  8986.75029841, 10319.07112826,
        8669.87879661, 10300.47496653,  9180.81858681,  8843.8523145 ,
        7035.74653423,  9049.29996554,  9389.06587617, 10265.97017653,
       10044.71397804,  6868.062338  ,  9759.8967412 ,  9466.53179916,
        9309.97063726, 10414.69338808,  9768.80893086,  7315.85828673,
       10080.53675451,  6989.45302904,  9827.12942945,  7128.10041765,
        6398.29947219,  9969.70216375,  9754.65880446,  8628.96661395,
        8433.68258496,  6480.90271862,  7746.39132368,  6638.20569737,
        8325.17631535, 10410.63578515,  7336.78464334,  8539.37888515,
        9794.15939834, 10008.22703295,  7350.56181694,  9531.96814348,
```

mean_squared error

In [23]:
```
from sklearn.metrics import mean_squared_error#mean_squared error
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[23]:
```
590569.9121697355
```

finding the efficieny

In [24]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)#finding the efficieny
```

Out[24]:
```
0.8391885506165899
```

In [25]:
```
a=drop.loc[drop.model==1]
a
```

Out[25]:

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

1094 rows × 6 columns

In [26]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
```

In [27]:
```
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[27]:
```
▸   GridSearchCV
▸ estimator: Ridge
      ▸ Ridge
```

```
In [28]: ridge_regressor.best_params_

Out[28]: {'alpha': 30}
```

```
In [29]: ridge=Ridge(alpha=30)
         ridge.fit(x_train,y_train)
         y_pred_ridge=ridge.predict(x_test)
```

```
In [30]: from sklearn.metrics import mean_squared_error#mean_squared error
         Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
         Ridge_Error

Out[30]: 590569.9121697355
```

```
In [31]: from sklearn.metrics import r2_score
         r2_score(y_test,y_pred_ridge)#finding the efficieny

Out[31]: 0.8391885506165899
```

In [32]:

```
results=pd.DataFrame(columns=['price','predicted'])
results['price']=y_test
results['predicted']=y_pred_ridge
results=results.reset_index()
results['ID']=results.index
results.head(10)
```

Out[32]:

	index	price	predicted	ID
0	481	7900	5987.682984	0
1	76	7900	7272.490419	1
2	1502	9400	9839.847697	2
3	669	8500	9696.775405	3
4	1409	9700	10012.040862	4
5	1414	9900	9628.286853	5
6	1089	9900	9646.945160	6
7	1507	9950	10090.960592	7
8	970	10700	9877.094341	8
9	1198	8999	9326.088982	9

In [33]:

```
results['actual price']=results.apply(lambda column:column.price-column.predicted,axis=1)
results['actual price']=results.apply(lambda column:column.price-column.predicted,axis=1)
results
```

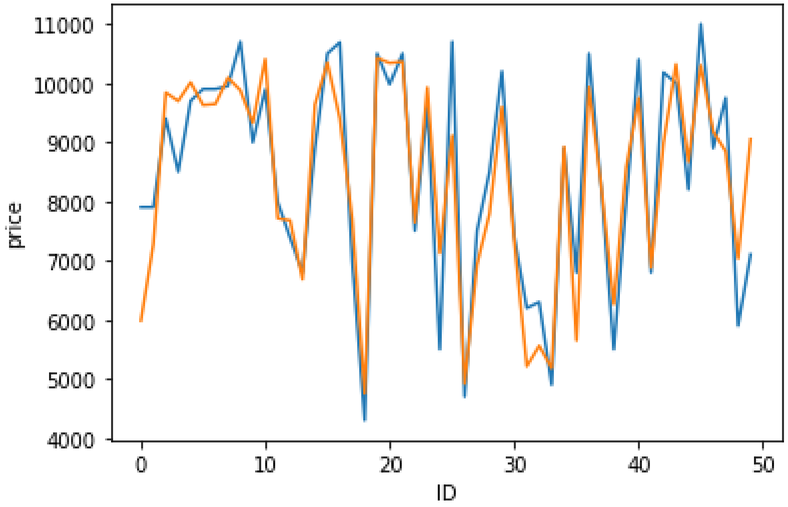
Out[33]:

	index	price	predicted	ID	actual price
0	481	7900	5987.682984	0	1912.317016
1	76	7900	7272.490419	1	627.509581
2	1502	9400	9839.847697	2	-439.847697
3	669	8500	9696.775405	3	-1196.775405
4	1409	9700	10012.040862	4	-312.040862
...	...	...	...	...	...
503	291	10900	10005.311518	503	894.688482
504	596	5699	6400.852430	504	-701.852430
505	1489	9500	10096.776914	505	-596.776914
506	1436	6990	8358.743798	506	-1368.743798
507	575	10900	10343.148204	507	556.851796

508 rows × 5 columns

```
In [34]: sb.lineplot(x='ID',y='price',data=results.head(50))
         sb.lineplot(x='ID',y='predicted',data=results.head(50))
         mp.plot()

Out[34]: []
```



```
In [ ]:

In [ ]:

In [ ]:
```