In [1]:
```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
data=pd.read_csv("/home/palcement/Downloads/fiat500.csv")
data
```

Out[2]:

	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 [3]:
```
data1=data.drop(['lat','lon','ID'],axis=1)
data1
```

Out[3]:

	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 [35]:
```
data1.head(10)
```

Out[35]:

	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
5	74	3623	70225	1	7900	0	1	0
6	51	731	11600	1	10750	1	0	0
7	51	1521	49076	1	9190	1	0	0
8	73	4049	76000	1	5600	0	0	1
9	51	3653	89000	1	6000	0	0	1

In [5]:
```
data1=pd.get_dummies(data1)
data1
```

Out[5]:

	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 [6]:
```
y=data1['price']
X=data1.drop(['price'],axis=1)
```

In [7]:
```
y
```

Out[7]:
```
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 [8]:
```
X
```

Out[8]:

	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 [9]:
```
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 [10]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression() #creating of Linear Regression
reg.fit(X_train,y_train) #training and fitting LR object using training data
```

Out[10]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [11]:
```
ypred=reg.predict(X_test)
```

In [12]:
```
from sklearn.metrics import r2_score  #model efficinecy step    #y_test is the actual price    #y_predit is
r2_score(y_test,ypred)
```

Out[12]:
```
0.8415526986865394
```

In [13]:
```
from sklearn.metrics import mean_squared_error ##Calulating mean square error
mean_squared_error(y_test,ypred)
```

Out[13]:
```
581887.727391353
```

In [14]:
```
#Results=pd.DataFrame(columns=['Actual','Predicted'])
#Results['Actual']=y_test
Results=pd.DataFrame(columns=['Price','Predicted']) #price and predicted names are our wish
Results['Price']=y_test
Results['Predicted']=ypred
Results=Results.reset_index()   #This line is optional
# Results['Id']=Results.index        #This line is optional
Results.head(15)
```

Out[14]:

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

In [15]:
```
Results['diff_price']=Results.apply(lambda row:row.Price-row.Predicted,axis=1)
```

In [16]:
```
Results
```

Out[16]:

	index	Price	Predicted	diff_price
0	481	7900	5867.650338	2032.349662
1	76	7900	7133.701423	766.298577
2	1502	9400	9866.357762	-466.357762
3	669	8500	9723.288745	-1223.288745
4	1409	9700	10039.591012	-339.591012
...	...	...	...	...
503	291	10900	10032.665135	867.334865
504	596	5699	6281.536277	-582.536277
505	1489	9500	9986.327508	-486.327508
506	1436	6990	8381.517020	-1391.517020
507	575	10900	10371.142553	528.857447

508 rows × 4 columns

In [17]:
```
from sklearn.metrics import mean_squared_error ##Calulating mean square error
mean_squared_error(y_test,ypred)
```

Out[17]: 581887.727391353

Ridge Regression

In [19]:
```
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[19]:
```
GridSearchCV(estimator=Ridge(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20, 30]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [20]:
```
ridge_regressor.best_params_
```

Out[20]:
```
{'alpha': 30}
```

In [21]:
```
ridge=Ridge(alpha=30)
ridge.fit(X_train,y_train)
y_pred_ridge=ridge.predict(X_test)
```

In [22]:
```
from sklearn.metrics import mean_squared_error
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[22]:
```
579521.7970897449
```

In [23]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)
```

Out[23]:
```
0.8421969385523054
```

Elastic

In [24]:
```
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import GridSearchCV

elastic = ElasticNet()

parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20]}

elastic_regressor = GridSearchCV(elastic, parameters)

elastic_regressor.fit(X_train, y_train)
```

Out[24]:
```
GridSearchCV(estimator=ElasticNet(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [25]:
```
elastic_regressor.best_params_
```

Out[25]:
```
{'alpha': 0.01}
```

In [26]:
```
elastic=ElasticNet(alpha=0.1)
elastic.fit(X_train,y_train)
y_pred_elastic=elastic.predict(X_test)
```

In [27]:
```
from sklearn.metrics import mean_squared_error
elastic_Error=mean_squared_error(y_pred_elastic,y_test)
elastic_Error
```

Out[27]:
```
578326.9853103004
```

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

Out[28]:
```
0.8425222843073693
```

In [ ]:

FOR ALL MEAN SQUARE ERROR

In [ ]:

In [29]:
```
from sklearn.metrics import mean_squared_error ##Calulating mean square error   ##LINEAR REGRESION
mean_squared_error(y_test,ypred)
```

Out[29]: 581887.727391353

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

Out[30]: 579521.7970897449

In [31]:
```
from sklearn.metrics import mean_squared_error                                   ##ELASTIC REGRESSION
elastic_Error=mean_squared_error(y_pred_elastic,y_test)
elastic_Error
```

Out[31]: 578326.9853103004

In [ ]:

FOR ALL R2_SCORE

In [ ]:

In [32]:
```
from sklearn.metrics import r2_score  #model efficinecy step    #y_test is the actual price    #y_predit is
r2_score(y_test,ypred)
```

Out[32]: 0.8415526986865394

In [33]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)
```

Out[33]: 0.8421969385523054

In [34]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_elastic)
```

Out[34]: 0.8425222843073693

In [ ]: