In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import  matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
df=pd.read_csv(r"C:\Users\Sushma sree\Downloads\fiat500_VehicleSelection_Dataset (1).csv")
df
```

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

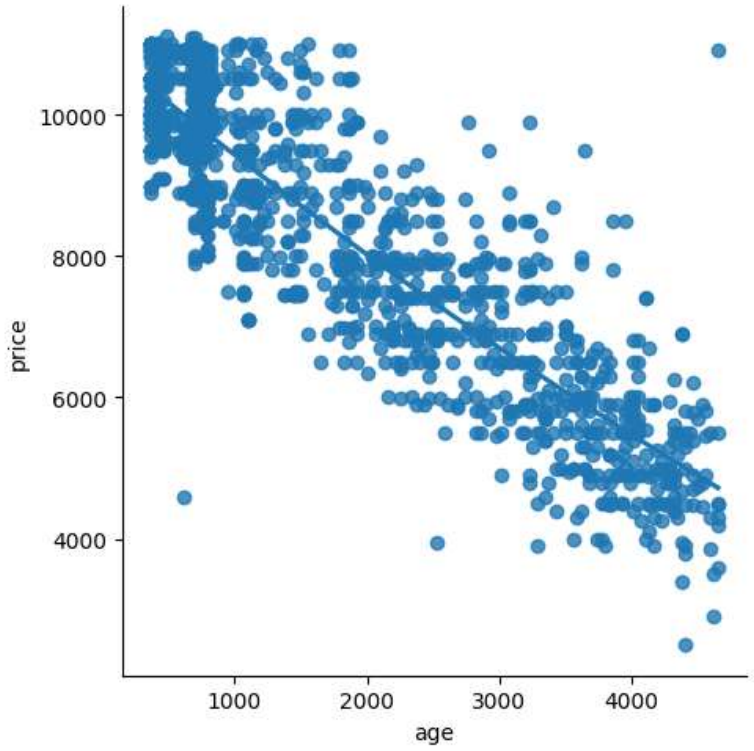
In [3]:
```
df=df[['age_in_days','price']]
df.columns=['age','price']
df.head(10)
```

Out[3]:

	age	price
0	882	8900
1	1186	8800
2	4658	4200
3	2739	6000
4	3074	5700
5	3623	7900
6	731	10750
7	1521	9190
8	4049	5600
9	3653	6000

In [4]:
```
sns.lmplot(x="age",y="price",data=df,order=2,ci=None)
```

Out[4]:
```
<seaborn.axisgrid.FacetGrid at 0x22329966d10>
```



In [5]:
```
df.describe()
```

Out[5]:

	age	price
count	1538.000000	1538.000000
mean	1650.980494	8576.003901
std	1289.522278	1939.958641
min	366.000000	2500.000000
25%	670.000000	7122.500000
50%	1035.000000	9000.000000
75%	2616.000000	10000.000000
max	4658.000000	11100.000000

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   age     1538 non-null   int64
 1   price   1538 non-null   int64
dtypes: int64(2)
memory usage: 24.2 KB
```

In [7]:
```
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\Sushma sree\AppData\Local\Temp\ipykernel_15636\4116506308.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

In [8]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['price']).reshape(-1,1)
```

In [9]:
```
df.dropna(inplace=True)
```

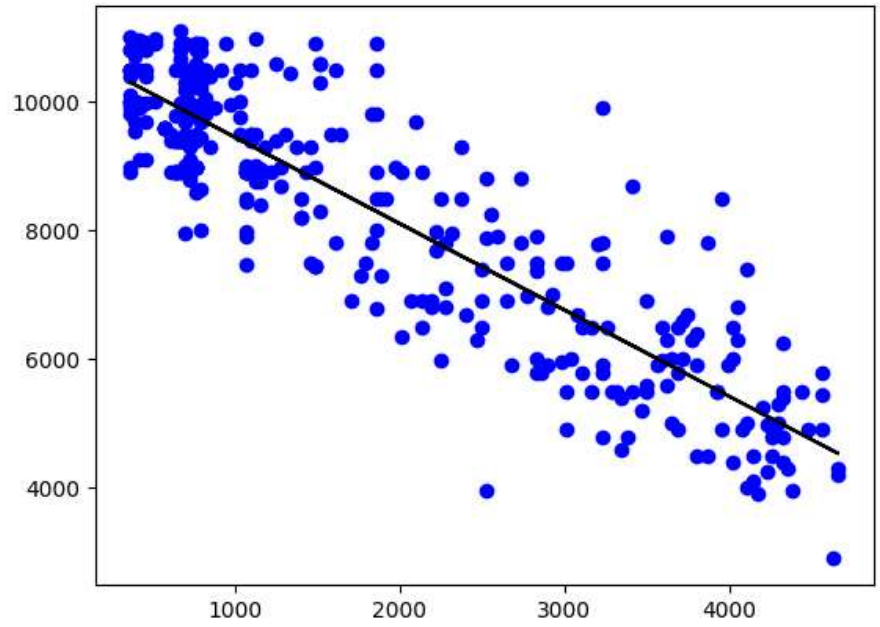
```
C:\Users\Sushma sree\AppData\Local\Temp\ipykernel_15636\1379821321.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.dropna(inplace=True)
```

In [10]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

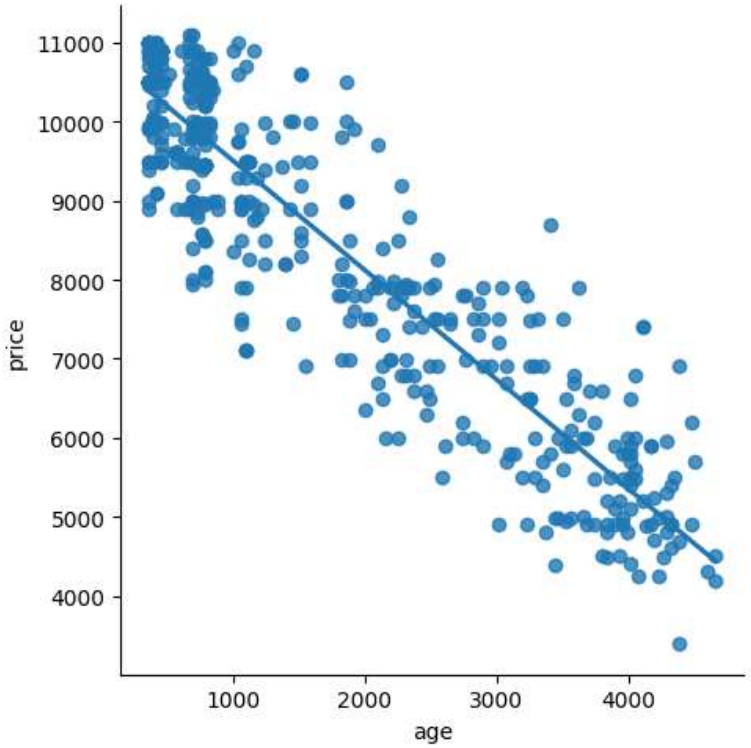
```
0.8004667612887026
```

In [11]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



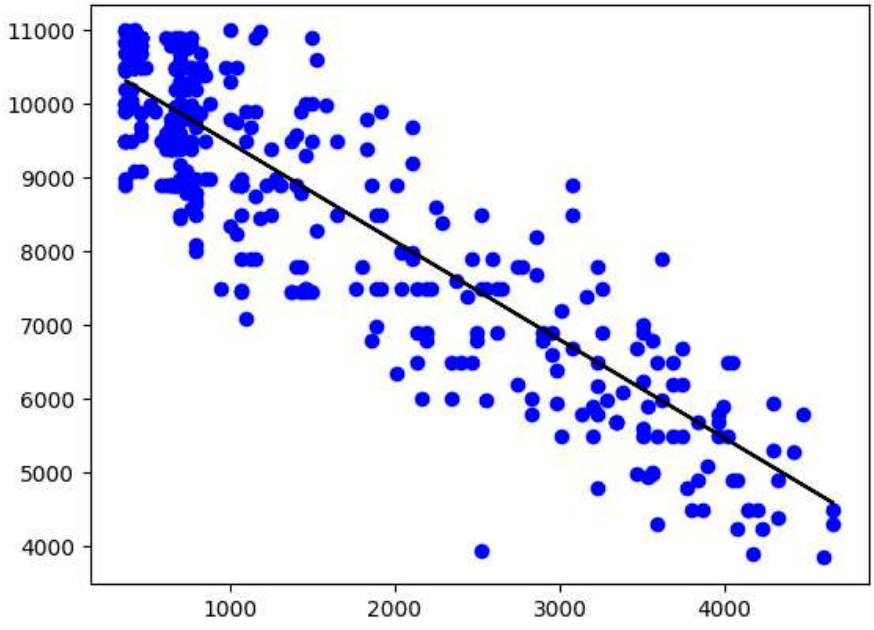
In [12]:
```
df500=df[:][:500]
sns.lmplot(x='age',y='price',data=df500,order=1,ci=None)
```

Out[12]:
```
<seaborn.axisgrid.FacetGrid at 0x2237821e5c0>
```



In [13]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression: ",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

```
Regression:  0.8014873936458661
```



In [14]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

```
R2 score: 0.8014873936458661
```

In [15]:
```
# Conclusion this is best fit LinearRegression
```

Ridge Regression Model

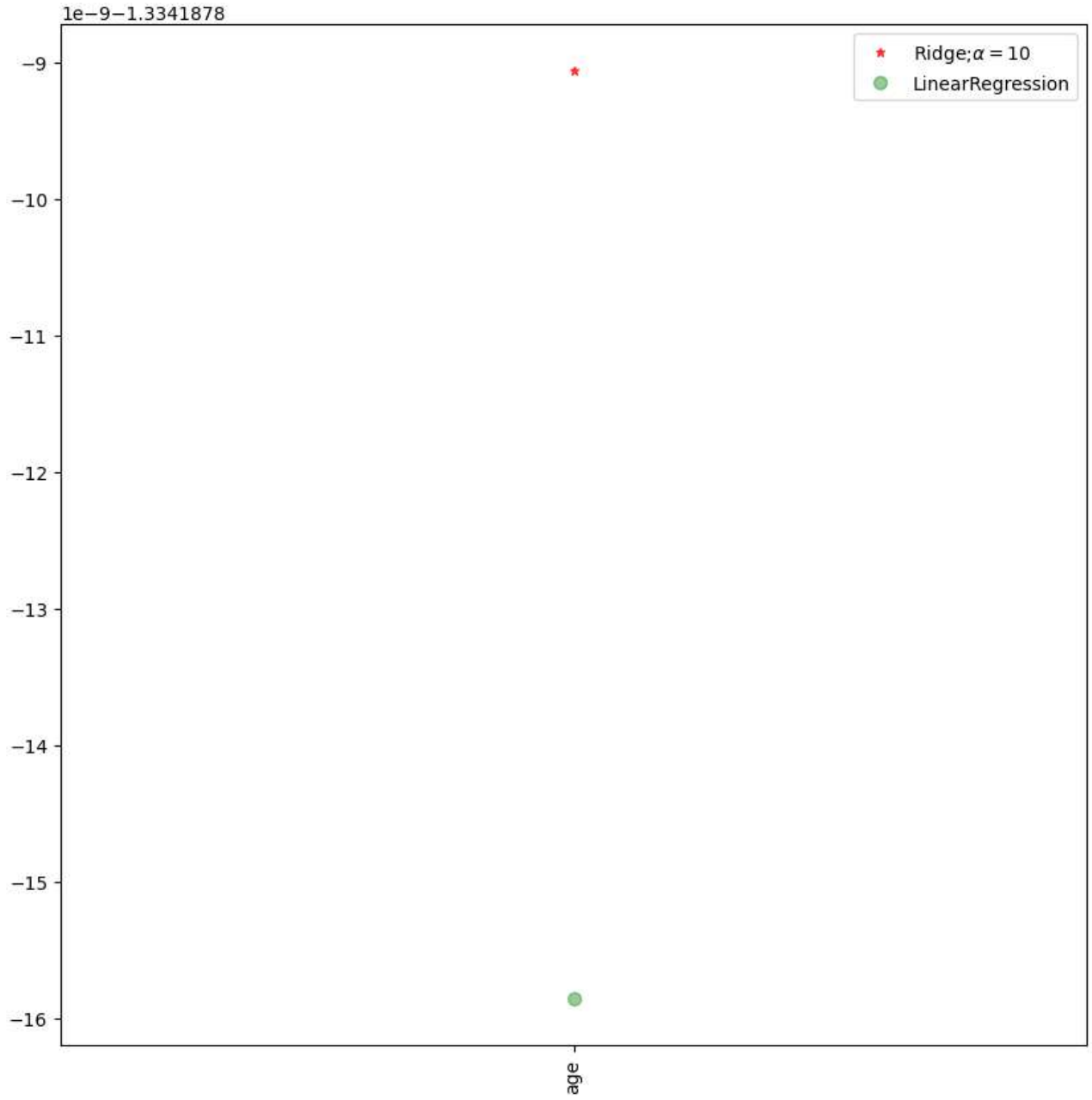
In [16]:
```
from sklearn.linear_model import Ridge,RidgeCV
from sklearn.linear_model import Lasso
from sklearn.preprocessing import StandardScaler
```

In [17]:
```
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
train_score_ridge=ridgeReg.score(x_train,y_train)
test_score_ridge=ridgeReg.score(x_test,y_test)
print('\nRidgeModel:')
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

```
RidgeModel:
The train score for ridge model is 0.7965652265935057
The test score for ridge model is 0.8014873934140825
```

In [18]:
```
features=['age']
target=['price']
```

In [20]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Rid
plt.plot(features,model.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='green',label='Linea
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

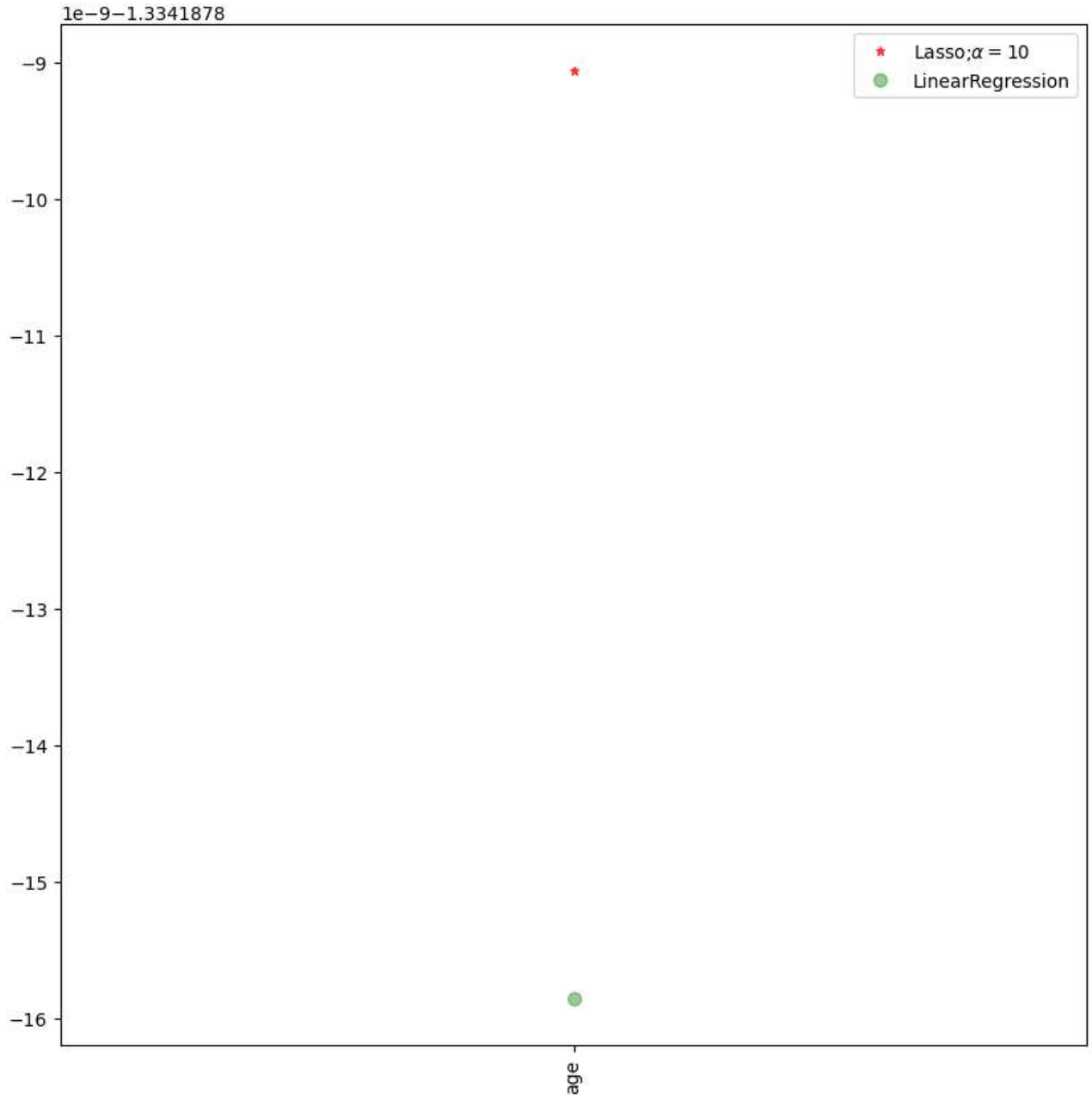


Lasso Regression Model

In [21]:
```
lassoReg=Ridge(alpha=10)
lassoReg.fit(x_train,y_train)
train_score_lasso=lassoReg.score(x_train,y_train)
test_score_lasso=lassoReg.score(x_test,y_test)
print('\nLassoModel:')
print("The train score for Lasso model is {}".format(train_score_lasso))
print("The test score for Lasso model is {}".format(test_score_lasso))
```

```
LassoModel:
The train score for Lasso model is 0.7965652265935057
The test score for Lasso model is 0.8014873934140825
```

In [23]:
```
plt.figure(figsize=(10,10))
plt.plot(features,lassoReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Las
plt.plot(features,model.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='green',label='Linea
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [24]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(x_train,y_train)
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.7965652265935057
0.8014873936438629

C:\Users\Sushma sree\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear_model\_coord
inate_descent.py:1568: DataConversionWarning: A column-vector y was passed when a 1d array was expected. P
lease change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Elastic Net

In [25]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[-1.34392217]
[10794.79318596]
```

In [26]:
```
y_pred_elastic=regr.predict(x_train)
```

In [27]:
```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print("Mean Squared Error on test set",mean_squared_error)
```

```
Mean Squared Error on test set 6885132.039918624
```

In [ ]:
```

```