In [13]:
```
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
from sklearn.linear_model import Ridge,RidgeCV ,Lasso
from sklearn.preprocessing import StandardScaler
```

In [14]:
```
v=pd.read_csv(r"C:\Users\ubinl\Downloads\fiat500_VehicleSelection_Dataset.csv"
v
```

Out[14]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
0	1	lounge	51	882	25000	1	44.907242	8.611560
1	2	pop	51	1186	32500	1	45.666359	12.241890
2	3	sport	74	4658	142228	1	45.503300	11.417840
3	4	lounge	51	2739	160000	1	40.633171	17.634609
4	5	pop	73	3074	106880	1	41.903221	12.495650
...	...	...	...	...	...	...	...	...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870
1535	1536	pop	51	2223	60457	1	45.481541	9.413480
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270
1537	1538	pop	51	1766	54276	1	40.323410	17.568270

1538 rows × 9 columns

In [15]:
```
v.head()
```

Out[15]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	pric
0	1	lounge	51	882	25000	1	44.907242	8.611560	890
1	2	pop	51	1186	32500	1	45.666359	12.241890	880
2	3	sport	74	4658	142228	1	45.503300	11.417840	420
3	4	lounge	51	2739	160000	1	40.633171	17.634609	600
4	5	pop	73	3074	106880	1	41.903221	12.495650	570

In [16]:
```
v.tail()
```

Out[16]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
1533	1534	sport	51	3712	115280	1	45.069679	7.70492
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687
1535	1536	pop	51	2223	60457	1	45.481541	9.41348
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227
1537	1538	pop	51	1766	54276	1	40.323410	17.56827

In [17]:
```
v.describe()
```

Out[17]:

	ID	engine_power	age_in_days	km	previous_owners	lat
count	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
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612

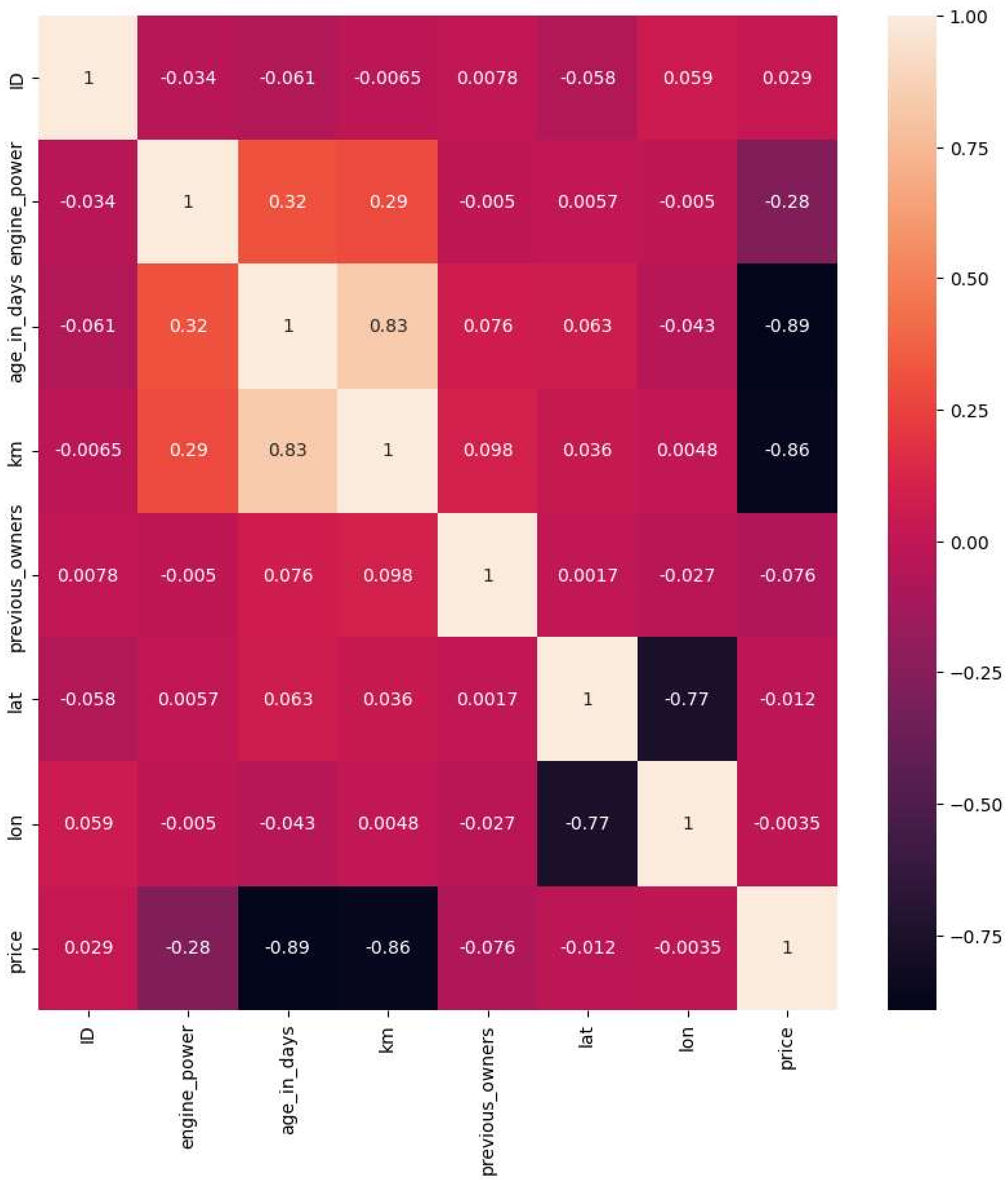
In [18]:
```
v.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1538 non-null   int64  
 1   model            1538 non-null   object 
 2   engine_power     1538 non-null   int64  
 3   age_in_days      1538 non-null   int64  
 4   km               1538 non-null   int64  
 5   previous_owners  1538 non-null   int64  
 6   lat              1538 non-null   float64
 7   lon              1538 non-null   float64
 8   price            1538 non-null   int64  
dtypes: float64(2), int64(6), object(1)
memory usage: 108.3+ KB
```

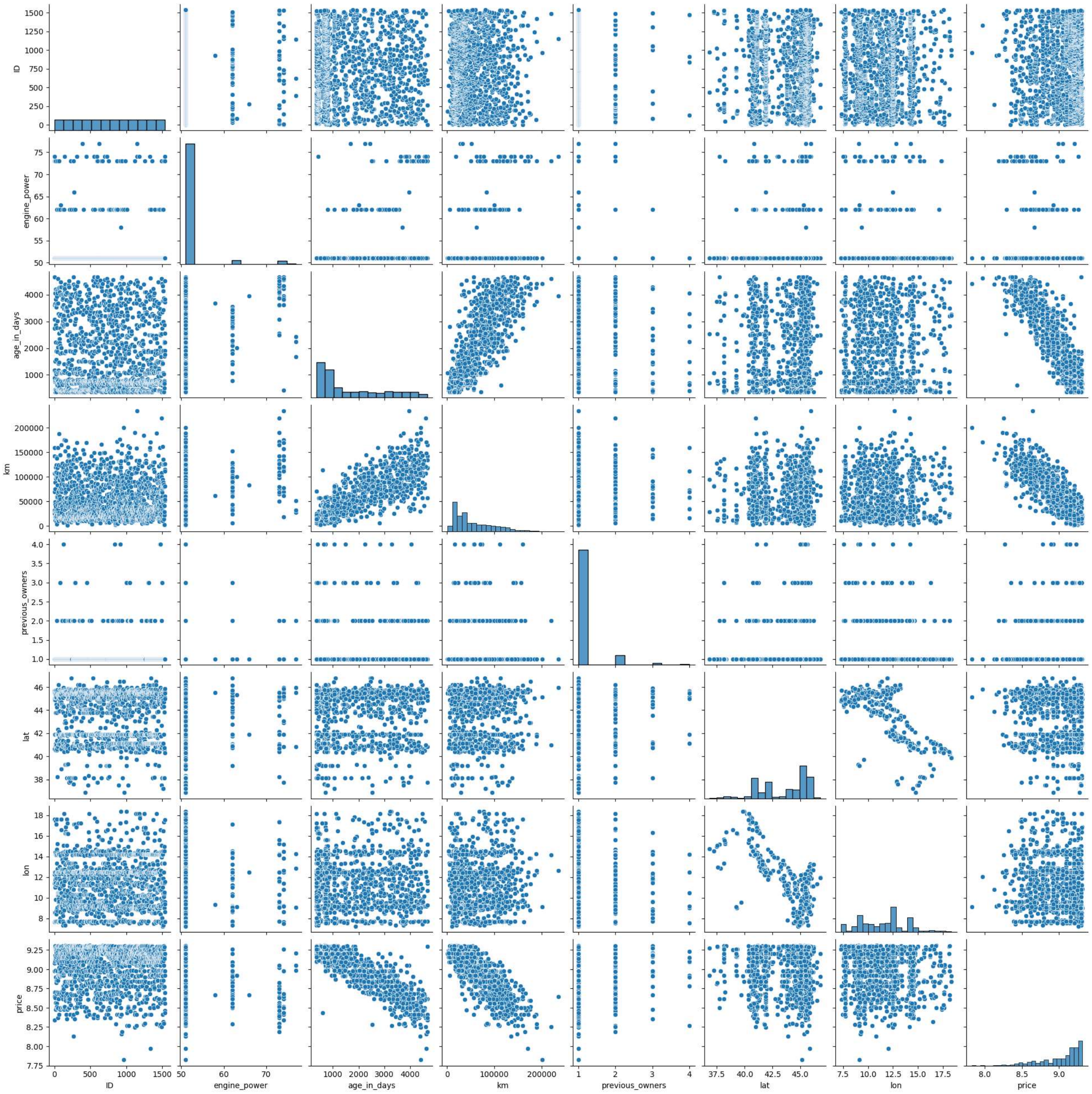
In [19]:
```
v.drop(columns = ["model"], inplace= True)
```

In [20]:
```
plt.figure(figsize=(10,10))
sns.heatmap(v.corr(), annot=True)
```

Out[20]:
```
<Axes: >
```



In [23]:
```
sns.pairplot(v)
v.price = np.log(v.price)
```



In [26]:
```
features=v.columns[0:2]
target=v.columns[-1]
x=v[features].values
y=v[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=
```

In [27]:
```
print("The dimension of x_train is {}".format(x_train.shape))
print("The dimension of x_train is {}".format(x_test.shape))
```

```
The dimension of x_train is (1076, 2)
The dimension of x_train is (462, 2)
```

In [28]:

```
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

In [29]:

```
lr=LinearRegression()
lr.fit(x_train,y_train)
actual=y_test
train_score_lr=lr.score(x_train,y_train)
test_score_lr=lr.score(x_test,y_test)
```

In [30]:

```
print("\nLinear Regression Model:\n")
print("The train score of lr model is {}".format(train_score_lr))
print("The test score of lr model is {}".format(test_score_lr))
```

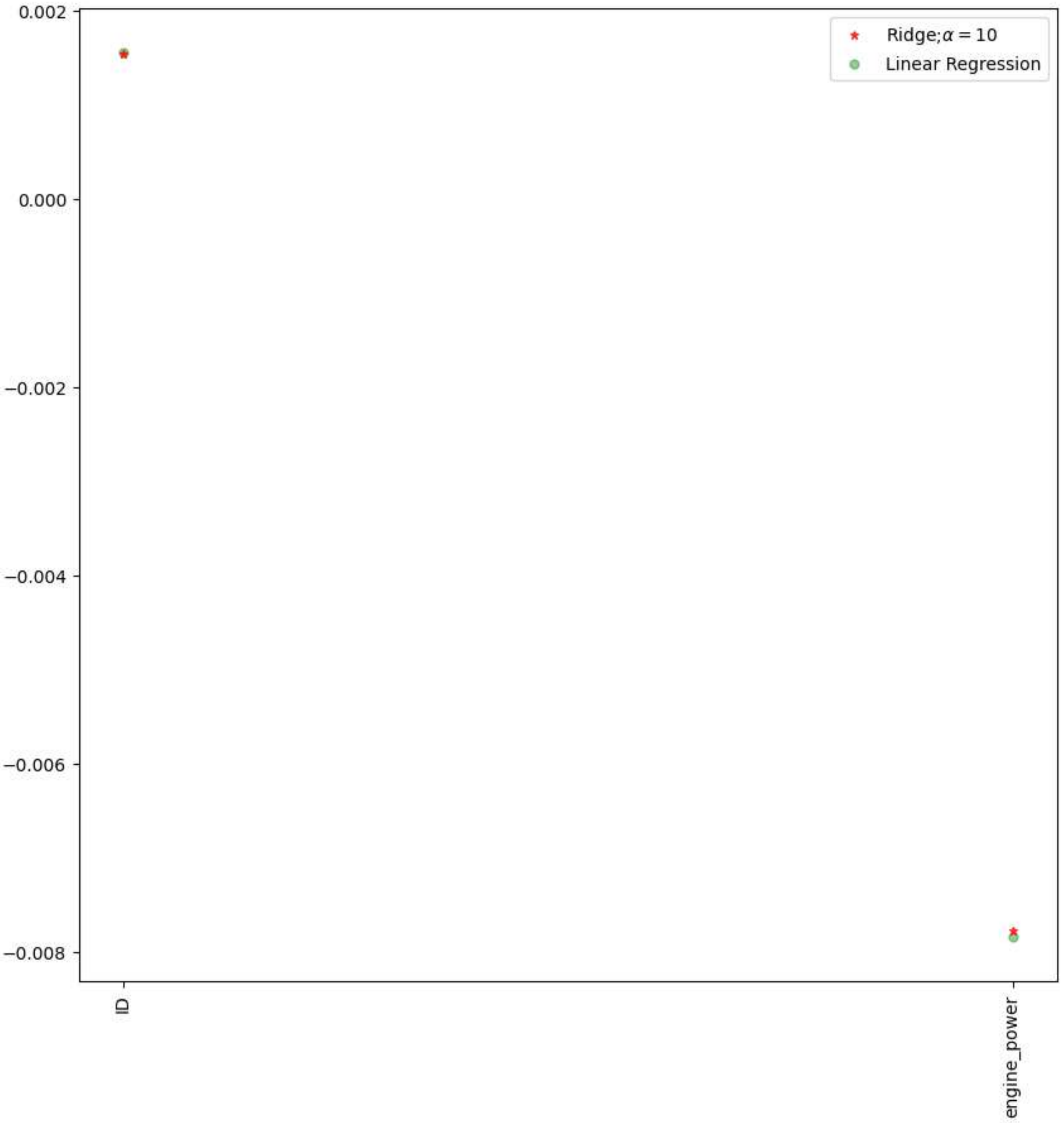
```

Linear Regression Model:

The train score of lr model is 0.0793983291169017
The test score of lr model is 0.08618385072647494
```

In [31]:

```
r=Ridge(alpha=10)
r.fit(x_train,y_train)
train_score_ridge=r.score(x_train,y_train)
test_score_ridge=r.score(x_test,y_test)
print("\nRidge model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

```

Ridge model:

The train score for ridge model is 0.0793919168454783
The test score for ridge model is 0.08585180780024182
```

In [32]:
```
plt.figure(figsize=(10,10))
plt.plot(features,r.coef_,alpha=0.7,linestyle='None',marker='*',markersize=5,c
plt.plot(features,lr.coef_,alpha=0.4,linestyle='None',marker='o',markersize=5,
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [33]:

```
l=Lasso(alpha=10)
l.fit(x_train,y_train)
train_score_ls=l.score(x_train,y_train)
test_score_ls=l.score(x_test,y_test)
print("\nRidge model:\n")
print("The train score for ridge model is {}".format(train_score_ls))
print("The test score for ridge model is {}".format(test_score_ls))
```

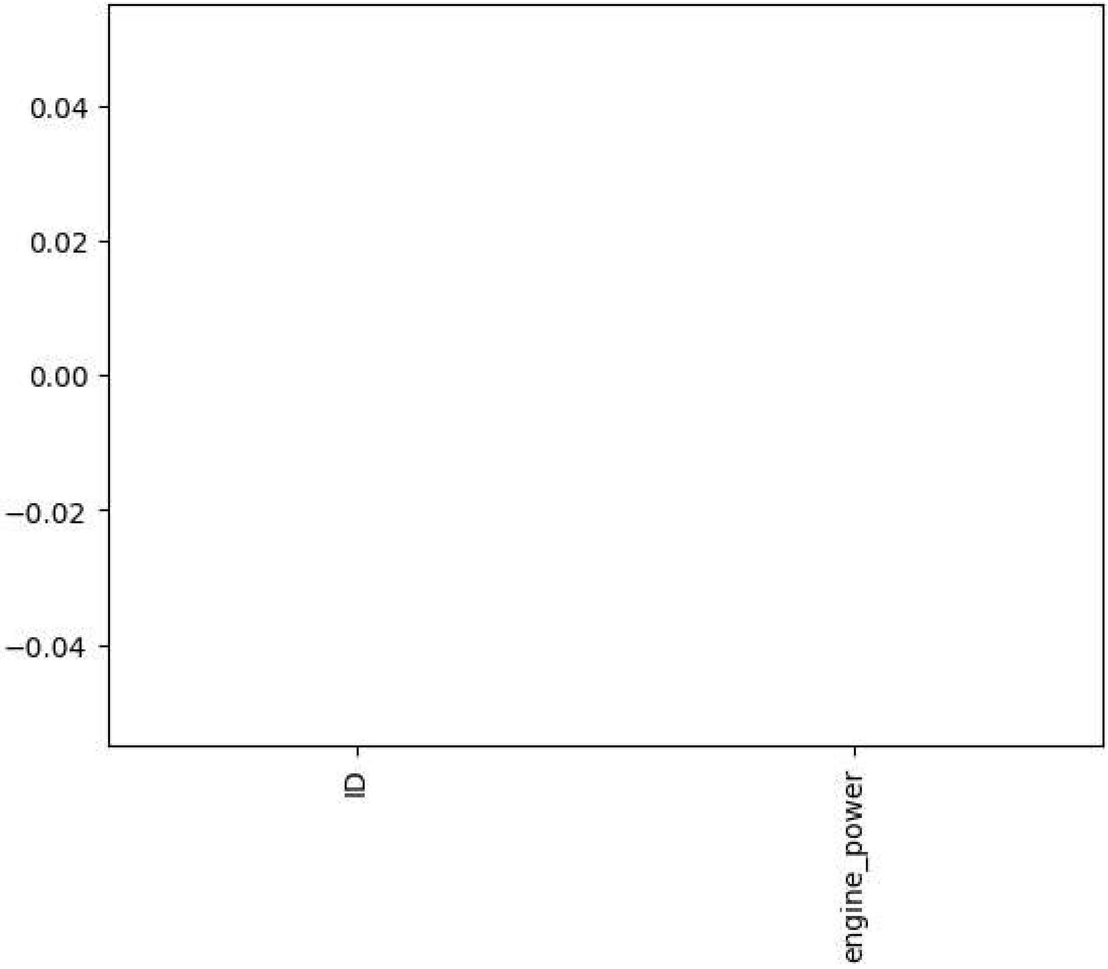
```

Ridge model:

The train score for ridge model is 0.0
The test score for ridge model is -0.0005445375127783869
```

In [34]:

```
pd.Series(l.coef_,features).sort_values(ascending=True).plot(kind='bar')
```

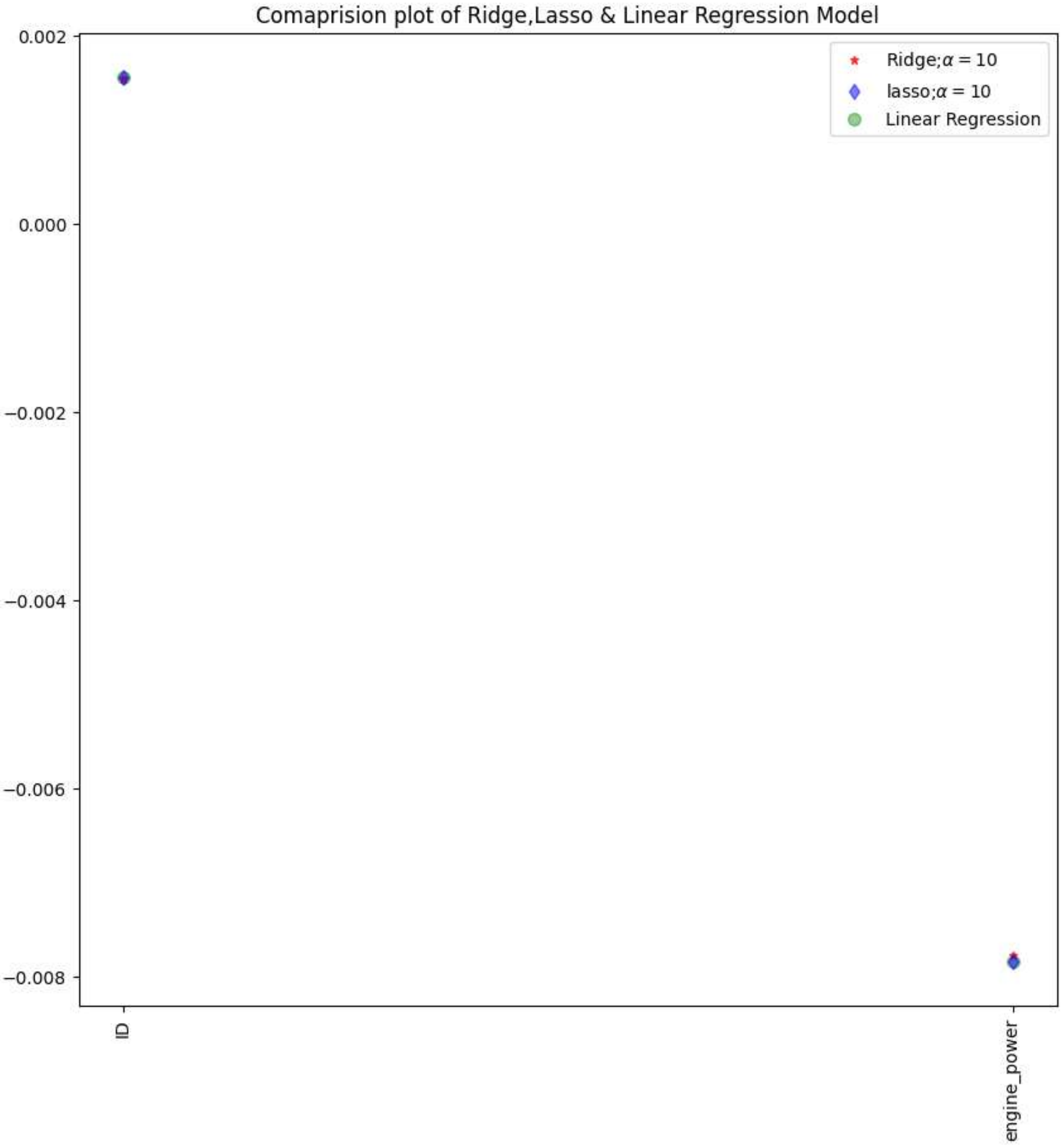
Out[34]:

```
<Axes: >
```



In [35]:
```
from sklearn.linear_model import LassoCV
lc=LassoCV(alphas=[0.0001,0.001,0.01,0.1,0,1,10],random_state=0).fit(x_train,y
print(lc.score(x_train,y_train))
print(lc.score(x_test,y_test))
```

```
0.07939832911690181
0.08618385072647528

C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: ConvergenceWarning: Objective di
d not converge. You might want to increase the number of iterations. Duality
gap: 0.34235174548073105, tolerance: 7.448129697419969e-05
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:1712: UserWarning: With alpha=0, this
algorithm does not converge well. You are advised to use the LinearRegression
estimator
  model.fit(X, y)
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:631: UserWarning: Coordinate descent
with no regularization may lead to unexpected results and is discouraged.
  model = cd_fast.enet_coordinate_descent(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:631: ConvergenceWarning: Objective di
d not converge. You might want to increase the number of iterations, check th
e scale of the features or consider increasing regularisation. Duality gap:
4.093e-01, tolerance: 8.891e-05 Linear regression models with null weight for
the l1 regularization term are more efficiently fitted using one of the solve
rs implemented in sklearn.linear_model.Ridge/RidgeCV instead.
  model = cd_fast.enet_coordinate_descent(
In [36]: plt.figure(figsize=(10,10))
         plt.plot(features,r.coef_,alpha=0.7,linestyle='None',marker='*',markersize=5,c
         plt.plot(features,lc.coef_,alpha=0.5,linestyle='None',marker='d',markersize=6,
         plt.plot(features,lr.coef_,alpha=0.4,linestyle='None',marker='o',markersize=7,
         plt.xticks(rotation=90)
         plt.title("Comaprision plot of Ridge,Lasso & Linear Regression Model")
         plt.legend()
         plt.show()
```



In [37]:

```
from sklearn.linear_model import RidgeCV
rc=RidgeCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
print("\nRidge model:\n")
print("The train score for ridge model is {}".format(rc.score(x_train,y_train)
print("The test score for ridge model is {}".format(rc.score(x_test,y_test)))
```

```

Ridge model:

The train score for ridge model is 0.07939191684547786
The test score for ridge model is 0.08585180780024249
```

In [41]:

```
from sklearn.linear_model import ElasticNet
e=ElasticNet()
e.fit(x,y)
print(e.coef_)
print(e.intercept_)
```

```
[ 0. -0.]
2.199709457887639
```

In [42]:

```
y_pred_elastic=e.predict(x_train)
```

In [43]:

```
mse=np.mean((y_pred_elastic-y_train)**2)
print("Mean squared error on test set",mse)
```

```
Mean squared error on test set 0.0008263908501133118
```

In [ ]: