In [1]:

```
import pandas as pd
import numpy as np
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 [2]:

```
data=pd.read_csv(r"C:\Users\shaha\OneDrive\Desktop\Excel\vehical.csv")
data
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...	.
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	520
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	460
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	750
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	599
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	790

1538 rows × 9 columns

In [3]:

```
data.head()
```

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

In [4]:

```
data.tail()
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
1533	1534	sport	51	3712	115280	1	45.069679	7.70492	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.41348	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.56827	7900

In [5]:

```
plt.figure(figsize=(10,10))
```

Out[5]:

```
<Figure size 1000x1000 with 0 Axes>

<Figure size 1000x1000 with 0 Axes>
```

In [6]:

```
data.drop("ID",axis=1)
```

Out[6]:

	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	lounge	51	882	25000	1	44.907242	8.611560	8900
1	pop	51	1186	32500	1	45.666359	12.241890	8800
2	sport	74	4658	142228	1	45.503300	11.417840	4200
3	lounge	51	2739	160000	1	40.633171	17.634609	6000
4	pop	73	3074	106880	1	41.903221	12.495650	5700
...	...	...	...	...	...	...	...	...
1533	sport	51	3712	115280	1	45.069679	7.704920	5200
1534	lounge	74	3835	112000	1	45.845692	8.666870	4600
1535	pop	51	2223	60457	1	45.481541	9.413480	7500
1536	lounge	51	2557	80750	1	45.000702	7.682270	5990
1537	pop	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 8 columns

In [7]:

```
data.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 [10]:

```
data['model'].value_counts()
```

Out[10]:

```
model
lounge    1094
pop        358
sport       86
Name: count, dtype: int64
```

In [11]:

```
m={"model":{"lounge":1,"pop":2,"sport":3}}
data=data.replace(m)
data
```

Out[11]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	pric
0	1	1	51	882	25000	1	44.907242	8.611560	890
1	2	2	51	1186	32500	1	45.666359	12.241890	880
2	3	3	74	4658	142228	1	45.503300	11.417840	420
3	4	1	51	2739	160000	1	40.633171	17.634609	600
4	5	2	73	3074	106880	1	41.903221	12.495650	570
...	...	...	...	...	...	...	...	...	..
1533	1534	3	51	3712	115280	1	45.069679	7.704920	520
1534	1535	1	74	3835	112000	1	45.845692	8.666870	460
1535	1536	2	51	2223	60457	1	45.481541	9.413480	750
1536	1537	1	51	2557	80750	1	45.000702	7.682270	599
1537	1538	2	51	1766	54276	1	40.323410	17.568270	790

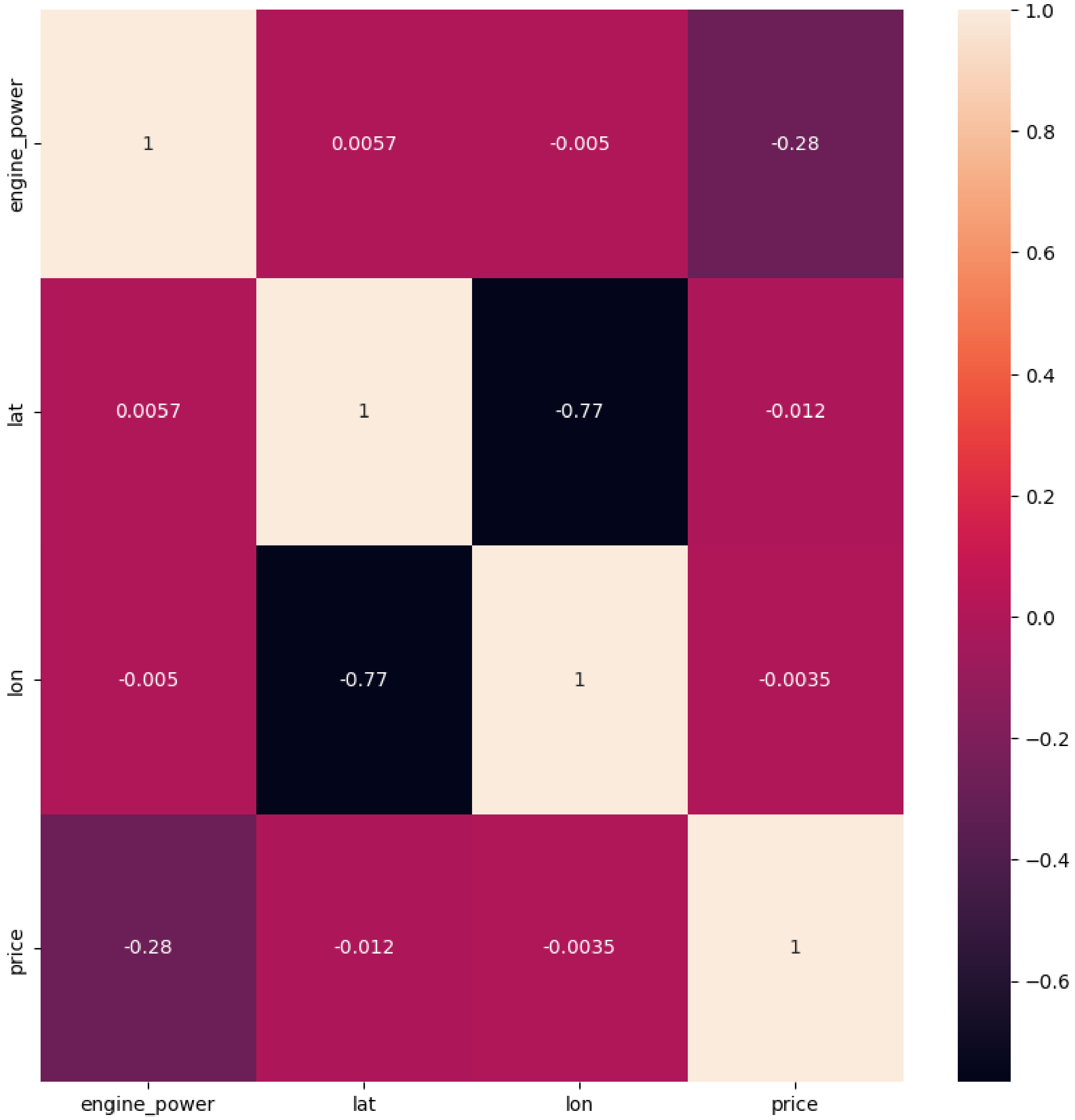
1538 rows × 9 columns

In [34]:

```
plt.figure(figsize=(10,10))
sns.heatmap(data.corr(),annot=True)
```

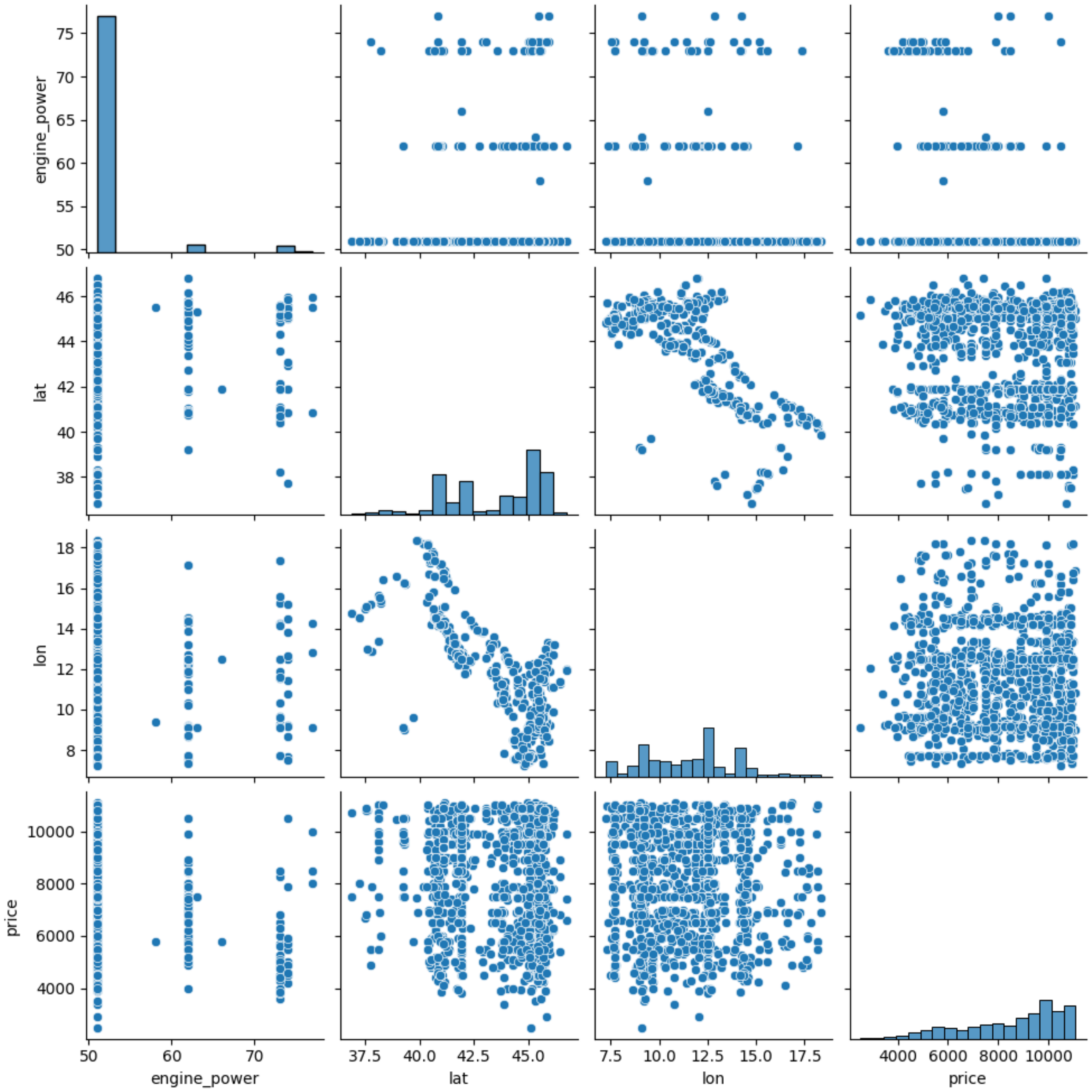
Out[34]:

```
<Axes: >
```



In [42]:

```
#pairplot
sns.pairplot(data)
data.Price=np.log(data.price)
```



In [43]:

```
features = data.columns[0:2]
target = data.columns[-1]
#x and y values
x = data[features].values
y = data[target].values
#splot
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=17)
print("The dimension of x_train is {}".format(x_train.shape))
print("The dimension of x_test is {}".format(x_test.shape))
#Scale features
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

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

In [44]:

```
#Model
lr = LinearRegression()
#fit model
lr.fit(x_train ,y_train)
#predict
#prediction = lr.predict(x_test)
#actual
actual = y_test
train_score_lr = lr.score(x_train,y_train)
test_score_lr = lr.score(x_test,y_test)
print("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model is {}".format(test_score_lr))
```

```

Linear Regression Model:

The train score for lr model is 0.07249217864832302
The test score for lr model is 0.08512644917585199
```

In [45]:

```
#Ridge Regression Model
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
#train and test score for ridge regression
train_score_ridge = ridgeReg.score(x_train,y_train)
test_score_ridge = ridgeReg.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 0.07248603871405479
The test score for ridge model is 0.08482952778985364
```

In [ ]:

```

```

In [46]:

```
#Lasso Regression model
print("\nLasso Model:\n")
lasso = Lasso(alpha = 10)
lasso.fit(x_train,y_train)
train_score_ls = lasso.score(x_train,y_train)
test_score_ls = lasso.score(x_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

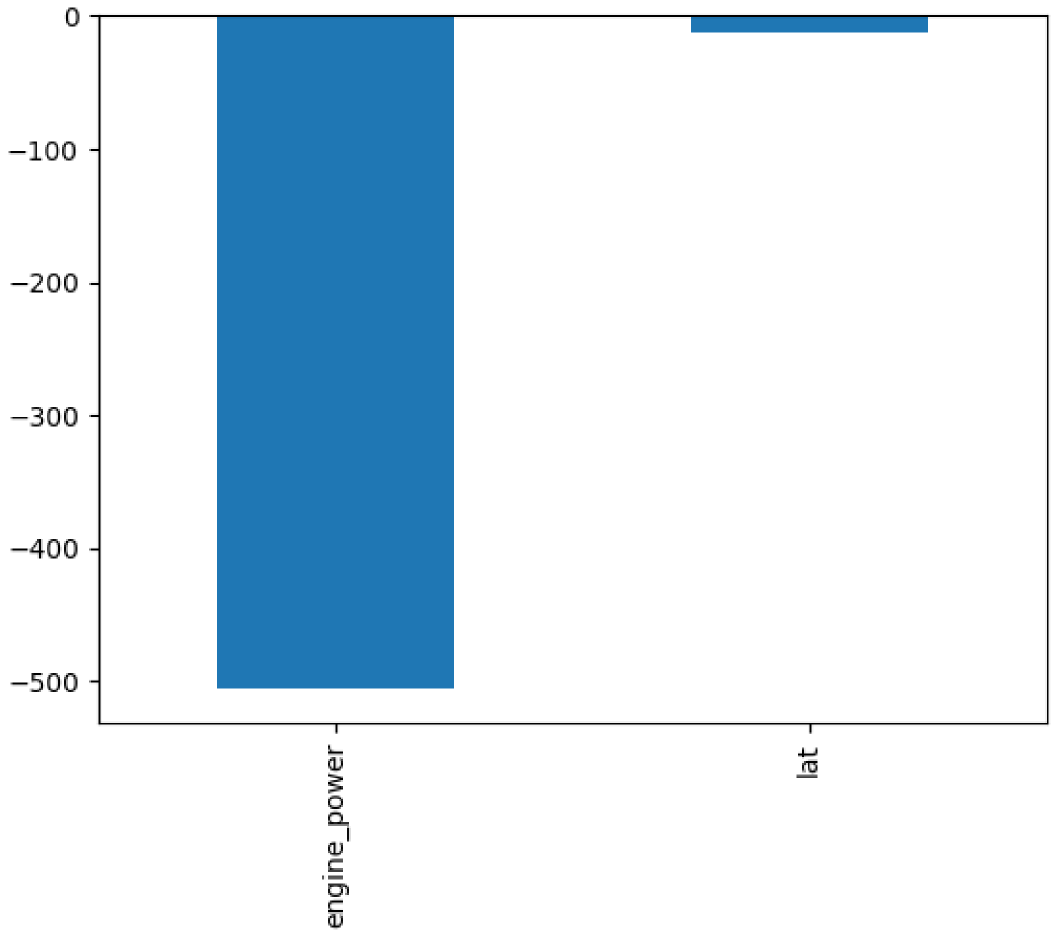
```

Lasso Model:

The train score for ls model is 0.07243823290637419
The test score for ls model is 0.0845251106284064
```

In [47]:

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

Out[47]:

```
<Axes: >
```



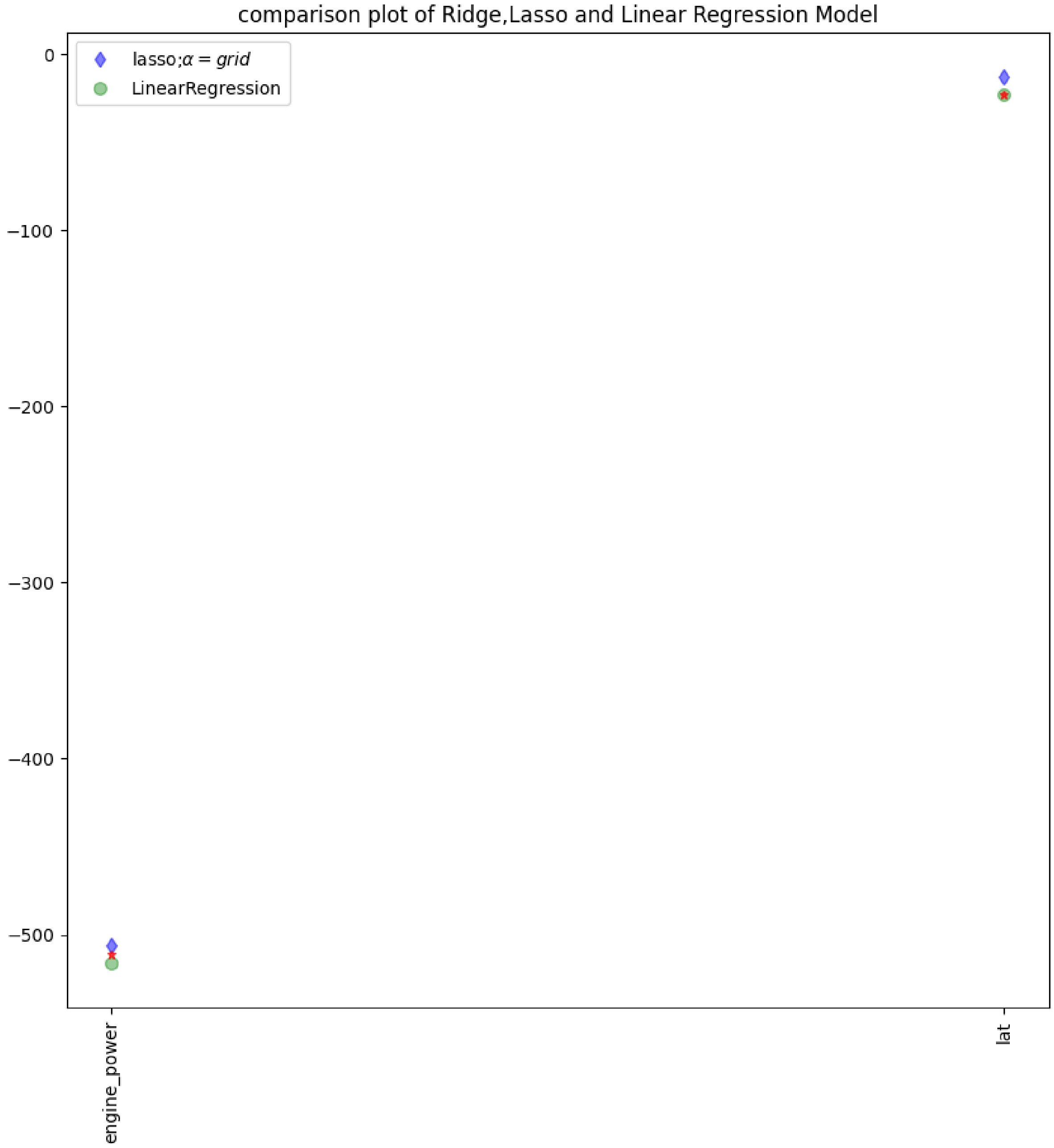
In [51]:

```
#using the linear CV model
from sklearn.linear_model import LassoCV
#Lasso Cross validation
lasso_cv = LassoCV(alphas = [0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(x_train,y_train)
#score
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.07243823290637419
0.0845251106284064
```

In [70]:

```
#plot size
plt.figure(figsize=(10,10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red'
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker = 'o',markersize=7,color='green',la
plt.xticks(rotation=90)
plt.legend()
plt.title("comparison plot of Ridge,Lasso and Linear Regression Model")
plt.show()
```



In [67]:

```
#using the linear Cv model
from sklearn.linear_model import RidgeCV
#Ridge cross validation
ridge_cv = RidgeCV(alphas = [0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print("The train score for ridge model is {}".format(ridge_cv.score(x_train,y_train)))
print("The train score for ridge model is {}".format(ridge_cv.score(x_test,y_test)))
```

```
The train score for ridge model is 0.07248603871405523
The train score for ridge model is 0.08482952778986697
```

In [63]:

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

```
[-130.69189543   -8.25401114]
15718.881987258936
```

In [64]:

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

In [65]:

```
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 54466918.9738486
```