In [32]:

```
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 [33]:

```
df=pd.read_csv(r"C:\Users\rubin\Downloads\fiat500_VehicleSelection_Dataset.xls.csv")
df
```

Out[33]:

	#NAME?	model	engine_power	age_in_days	km	previous_owners	lat	lon	pri
0	1	lounge	51	882	25000	1	44.907242	8.611560	89
1	2	pop	51	1186	32500	1	45.666359	12.241890	88
2	3	sport	74	4658	142228	1	45.503300	11.417840	42
3	4	lounge	51	2739	160000	1	40.633171	17.634609	60
4	5	pop	73	3074	106880	1	41.903221	12.495650	57
...	...	...	...	...	...	...	...	...	
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	52
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	46
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	75
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	59
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	79

1538 rows × 9 columns

In [34]:

```
df=df[['engine_power','price']]
df.columns=['Engine','price']
```

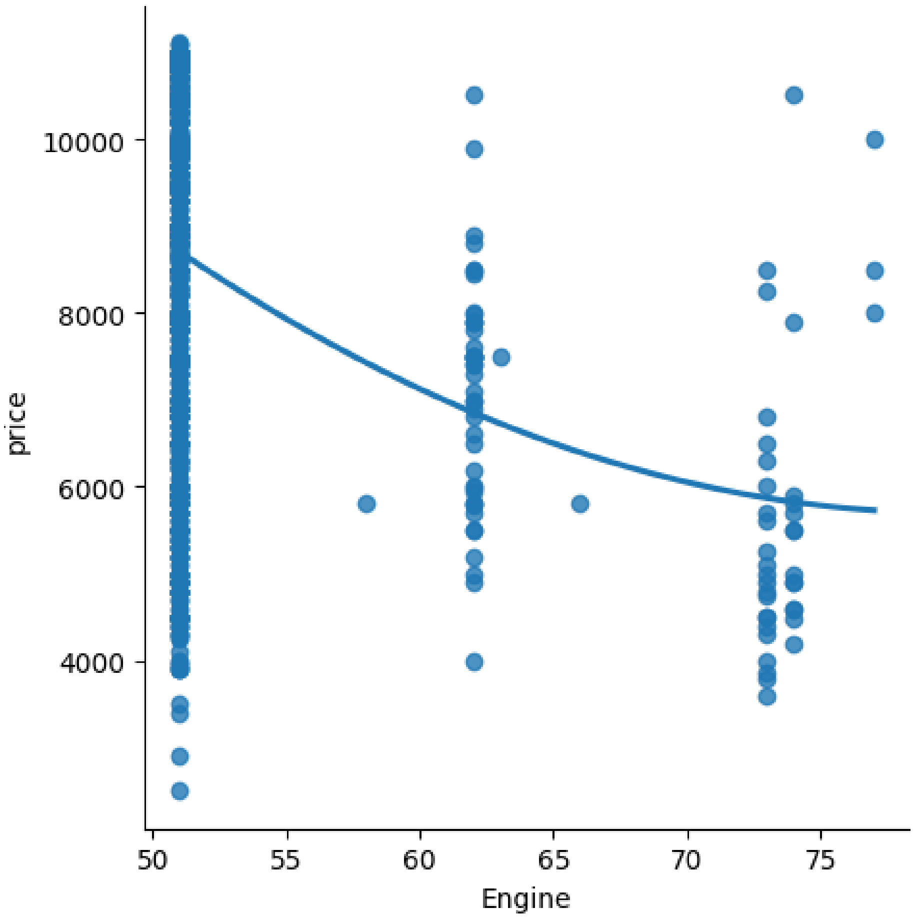
In [35]:
```
df.head(10)
```

Out[35]:

	Engine	price
0	51	8900
1	51	8800
2	74	4200
3	51	6000
4	73	5700
5	74	7900
6	51	10750
7	51	9190
8	73	5600
9	51	6000

In [36]:
```
sns.lmplot(x='Engine',y='price',data=df,order=2,ci=None)
```

Out[36]:
```
<seaborn.axisgrid.FacetGrid at 0x14c5601bac0>
```



In [37]:
```
df.info()
```

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

In [38]:
```
df.describe()
```

Out[38]:

	Engine	price
count	1538.000000	1538.000000
mean	51.904421	8576.003901
std	3.988023	1939.958641
min	51.000000	2500.000000
25%	51.000000	7122.500000
50%	51.000000	9000.000000
75%	51.000000	10000.000000
max	77.000000	11100.000000

In [39]:
```
df.fillna(method='ffill')
```

Out[39]:

	Engine	price
0	51	8900
1	51	8800
2	74	4200
3	51	6000
4	73	5700
...	...	...
1533	51	5200
1534	74	4600
1535	51	7500
1536	51	5990
1537	51	7900

1538 rows × 2 columns

In [40]:

```
x=np.array(df['Engine']).reshape(-1,1)
y=np.array(df['price']).reshape(-1,1)
```

In [41]:

```
df.dropna(inplace=True)
```

```
C:\Users\rubin\AppData\Local\Temp\ipykernel_3672\1379821321.py:1: SettingWithCopyWa
rning:
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)
```

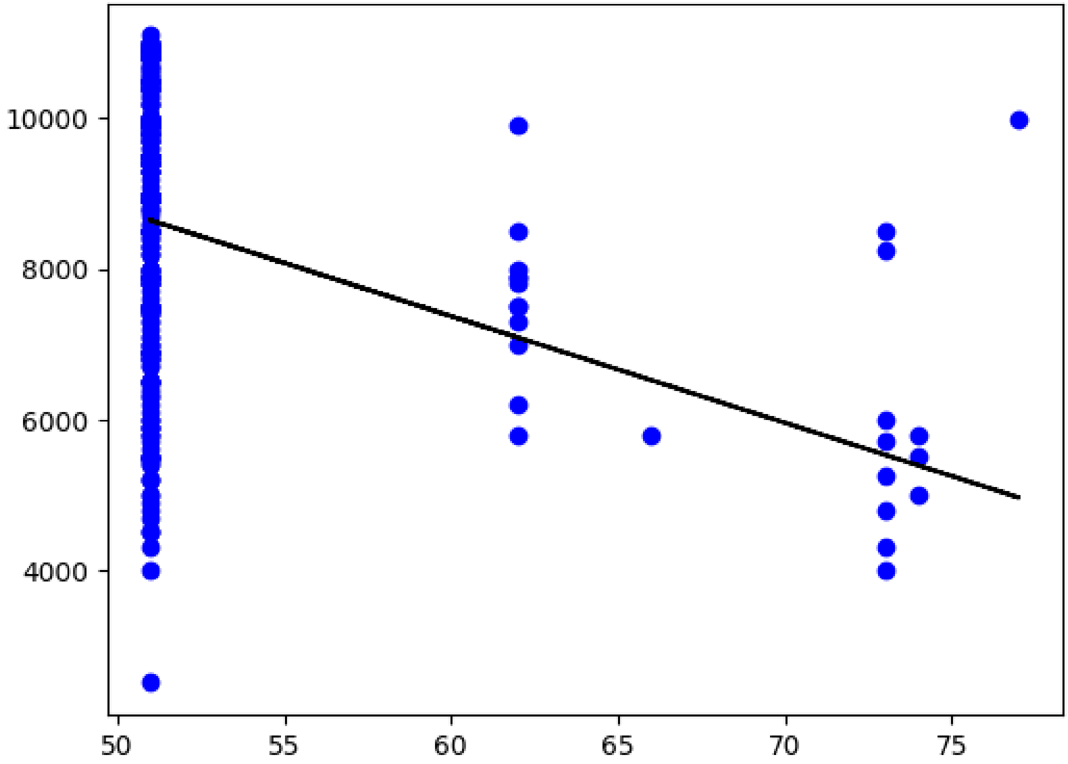
In [42]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
reg=LinearRegression()
reg.fit(x_train,y_train)
print(reg.score(x_test,y_test))
```

```
0.06391406640882291
```

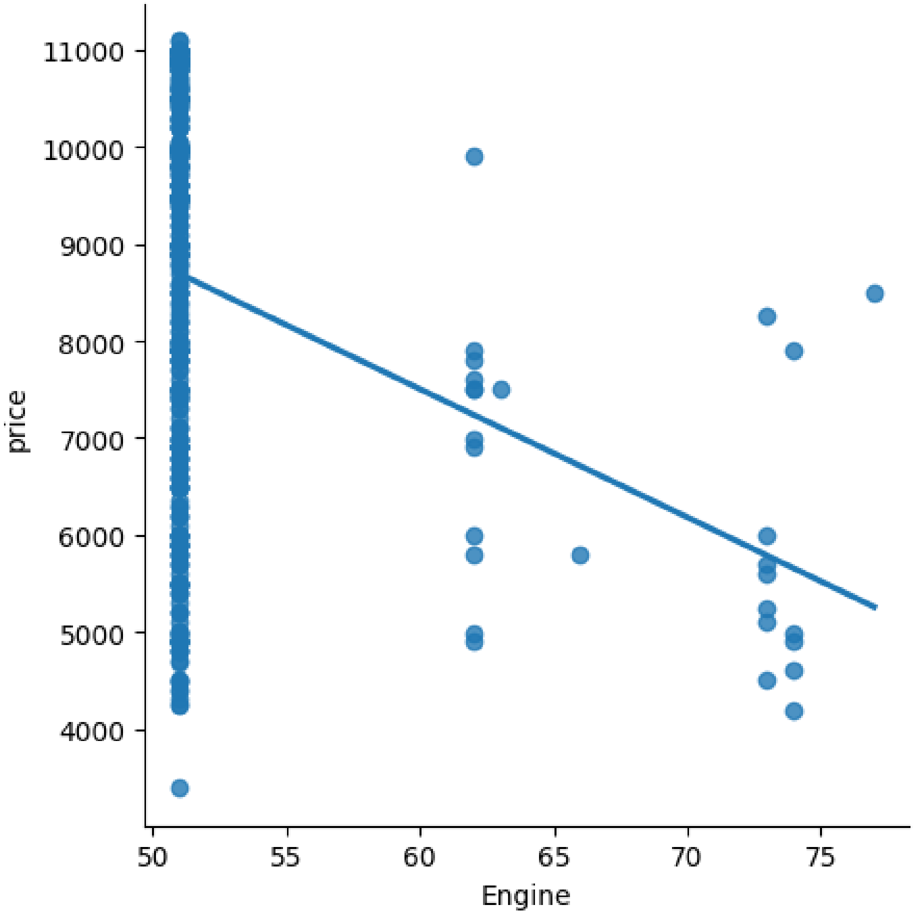
In [43]:

```
y_pred=reg.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [44]:
```
df500=df[:][:500]
sns.lmplot(x="Engine",y="price",data=df500,order=1,ci=None)
```

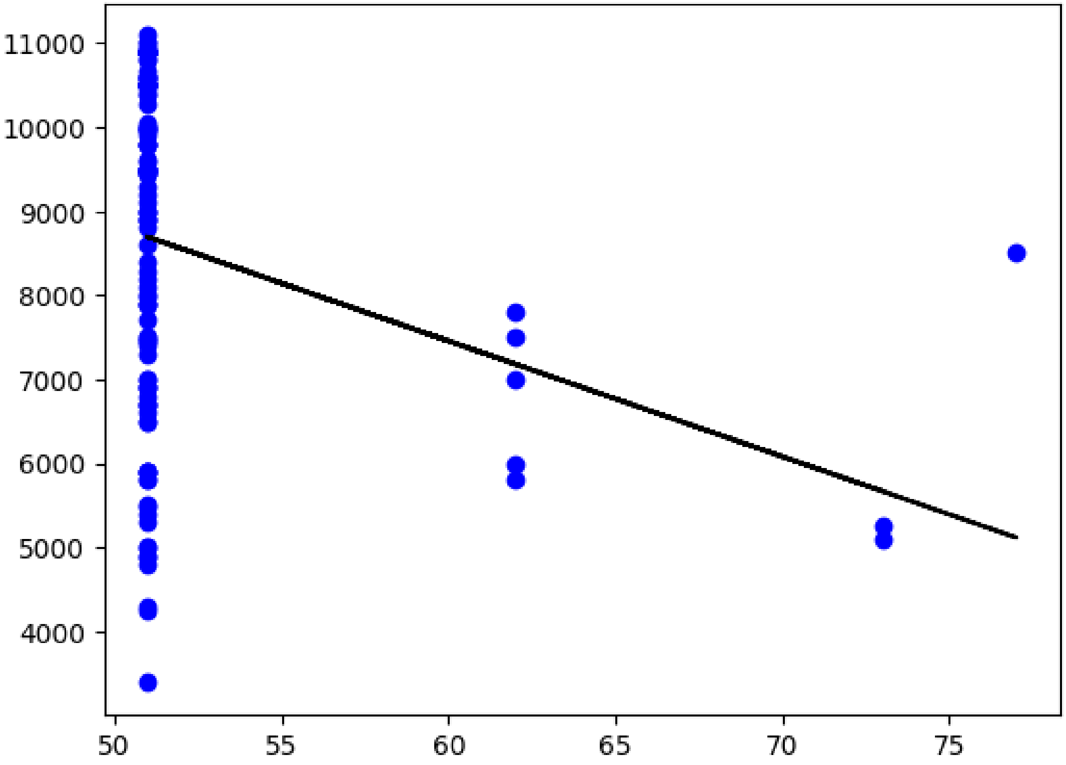
Out[44]:
```
<seaborn.axisgrid.FacetGrid at 0x14c562556c0>
```



In [45]:

```
df500.fillna(method='ffill',inplace=True)
X=np.array(df500['Engine']).reshape(-1,1)
y=np.array(df500['price']).reshape(-1,1)
df500.dropna(inplace=True)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
reg=LinearRegression()
reg.fit(X_train,y_train)
print("Regression:",reg.score(X_test,y_test))
y_pred=reg.predict(X_test)
plt.scatter(X_test,y_test,color='b')
plt.plot(X_test,y_pred,color='k')
plt.show()
```

```
Regression: 0.054419015942126725
```



In [46]:

```
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)
```

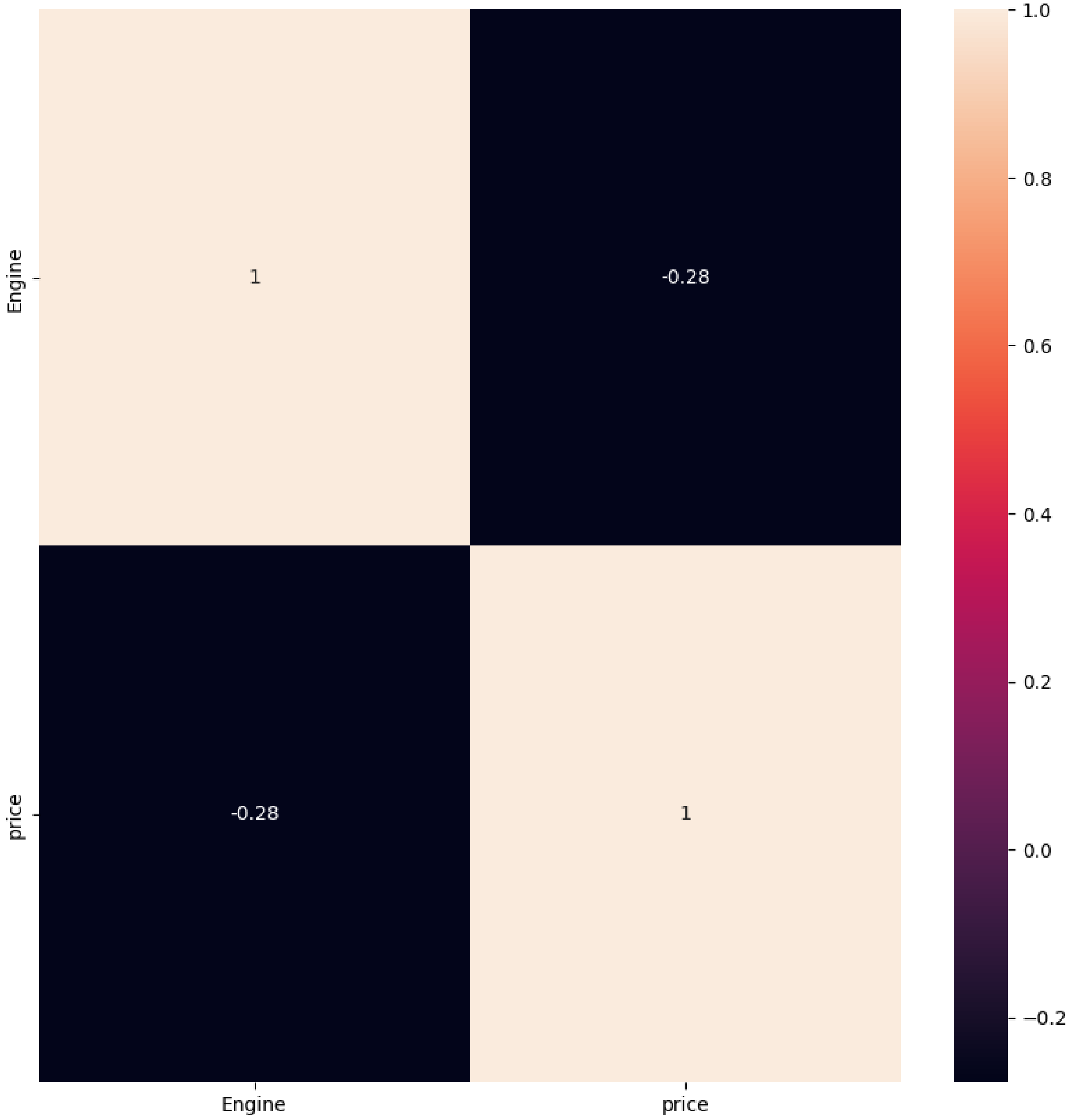
```
R2 score: 0.054419015942126725
```

In [49]:

```
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
```

```
In [50]: plt.figure(figsize=(10,10))
         sns.heatmap(df.corr(),annot=True)
```

```
Out[50]: <Axes: >
```



In [51]:

```
features = df.columns[0:2]
target = df.columns[-1]
#X and y values
X = df[features].values
y = df[target].values
#splot
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_stat
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 [52]:

```
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))
```

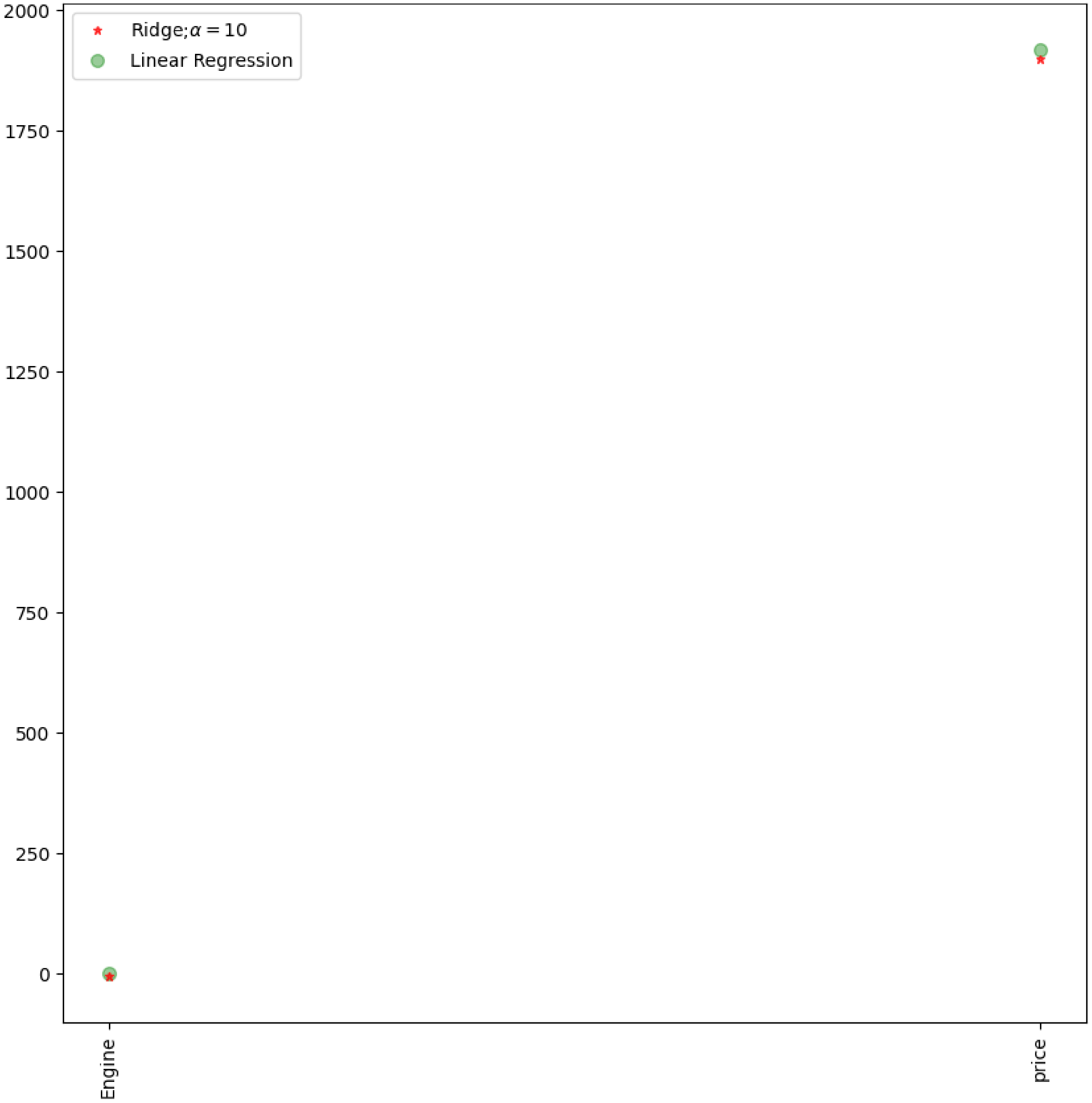
```

Linear Regression Model:

The train score for lr model is 1.0
The test score for lr model is 1.0
```

In [53]:

```
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("\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 0.9999088581979684
The test score for ridge model is 0.9999100853681022
```

```
In [54]: plt.figure(figsize=(10,10))
         plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,
         #plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue
         plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color=
         plt.xticks(rotation=90)
         plt.legend()
         #plt.title("comparison plot of Ridge,Lasso and Linear regression model")
         plt.show()
```



In [55]:

```
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))
```

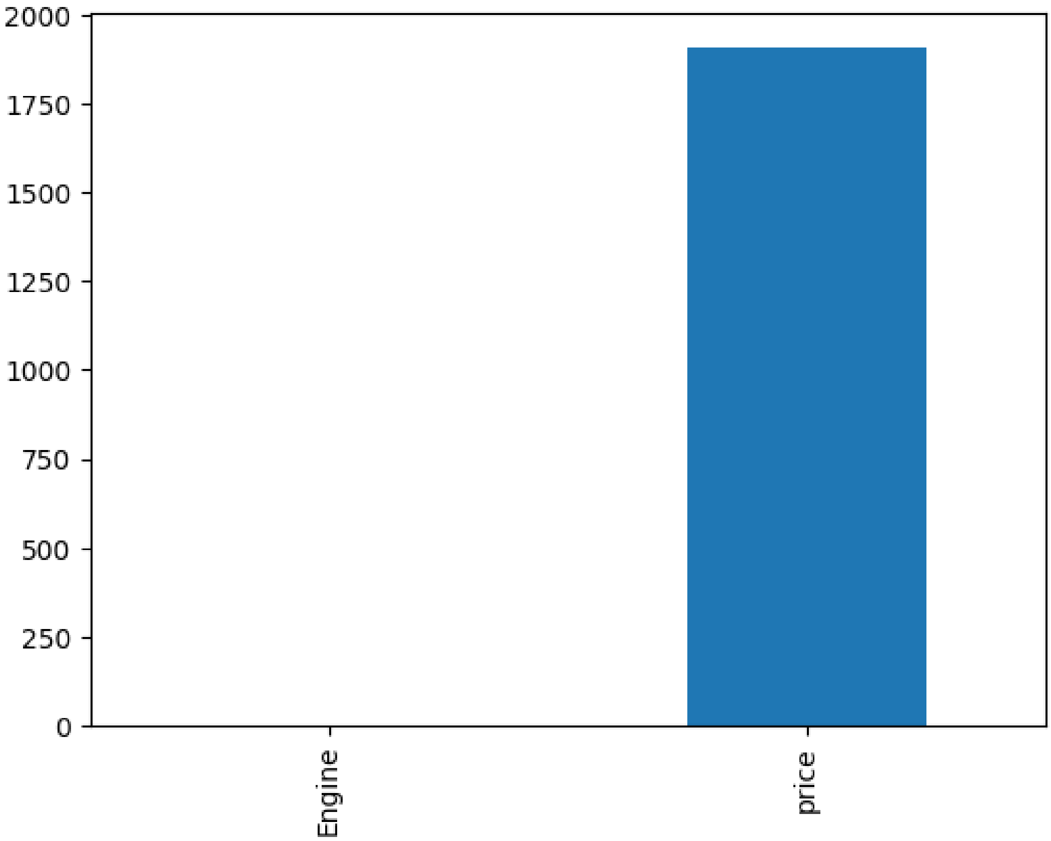
```
Lasso Model: 

The train score for ls model is 0.9999728562194999
The test score for ls model is 0.9999728508562553
```

In [56]:

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

Out[56]:

```
<Axes: >
```



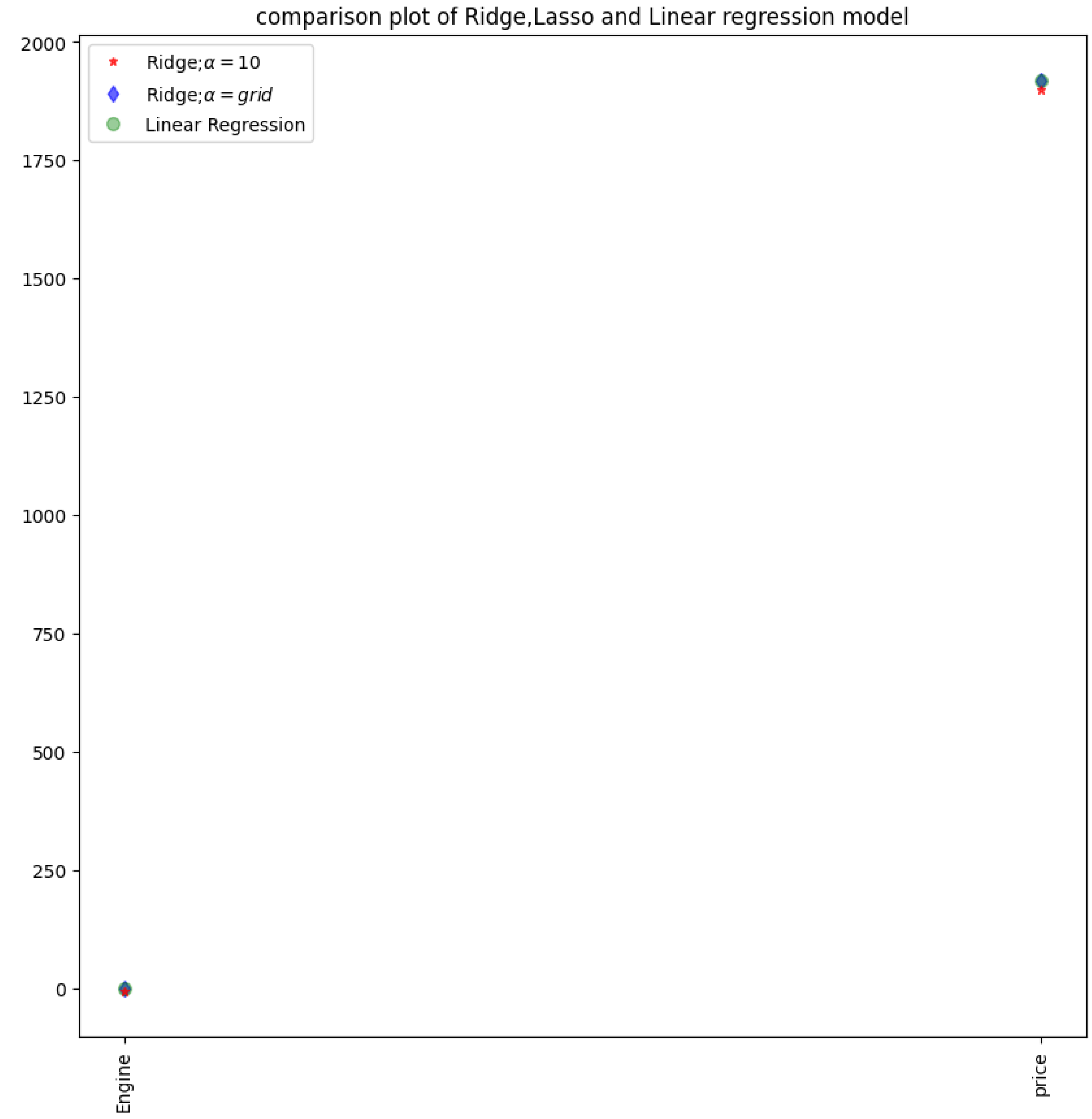
In [57]:

```
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_
print(lasso_cv.score(X_train,y_train))
print(lasso_cv.score(X_test,y_test))
```

```
0.9999999999501757
0.9999999999638806
```

In [58]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,
plt.plot(lasso_cv.coef_,alpha=0.6,linestyle='none',marker='d',markersize=6,color='bl
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color=
plt.xticks(rotation=90)
plt.legend()
plt.title("comparison plot of Ridge,Lasso and Linear regression model")
plt.show()
```



In [59]:

```
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 0.9999999999999897
The train score for ridge model is 0.9999999999999899
```

Elastic net

In [60]:

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

```
[-0.          0.99999973]
0.002280249860632466
```

In [61]:

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

In [62]:

```
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 77371869.93693778
```

In [ ]: