In [4]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
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 [5]:

```
dt=pd.read_csv(r"C:\Users\RAMADEVI SURIPAKA\Downloads\fiat500_VehicleSelection_Dataset (
dt
```

Out[5]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1	lounge	51	882	25000	1	44.907242	8.611
1	2	pop	51	1186	32500	1	45.666359	12.241
2	3	sport	74	4658	142228	1	45.503300	11.417
3	4	lounge	51	2739	160000	1	40.633171	17.634
4	5	pop	73	3074	106880	1	41.903221	12.495
...	...	...	...	...	...	...	...	
1533	1534	sport	51	3712	115280	1	45.069679	7.704
1534	1535	lounge	74	3835	112000	1	45.845692	8.666
1535	1536	pop	51	2223	60457	1	45.481541	9.413
1536	1537	lounge	51	2557	80750	1	45.000702	7.682
1537	1538	pop	51	1766	54276	1	40.323410	17.568

1538 rows × 9 columns

In [6]:

```
dt.head()
```

Out[6]:

	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	

In [7]:

```
dt.tail()
```

Out[7]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	l
1533	1534	sport	51	3712	115280	1	45.069679	7.704
1534	1535	lounge	74	3835	112000	1	45.845692	8.666
1535	1536	pop	51	2223	60457	1	45.481541	9.413
1536	1537	lounge	51	2557	80750	1	45.000702	7.682
1537	1538	pop	51	1766	54276	1	40.323410	17.568

In [8]:

```
dt.shape
```

Out[8]:

```
(1538, 9)
```

In [9]:

```
dt.describe()
```

Out[9]:

	ID	engine_power	age_in_days	km	previous_owners	l
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.0000
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.5413
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.1335
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.8558
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.8029
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.3940
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.4679
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.7956

In [10]:

```
dt.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 [11]:

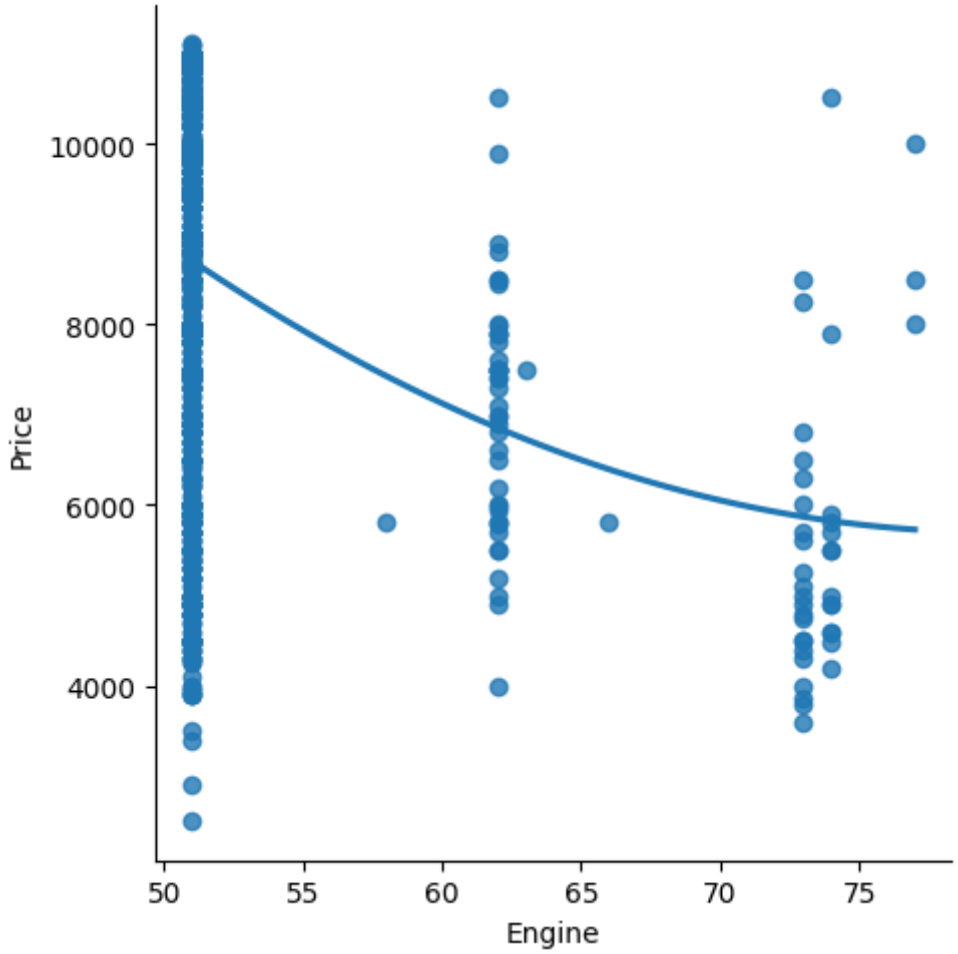
```
dt=dt[['engine_power','price']]
dt.columns=['Engine','Price']
```

In [12]:

```
sns.lmplot(x='Engine',y='Price',data=dt,order=2,ci=None)
```

Out[12]:

```
<seaborn.axisgrid.FacetGrid at 0x213a0b3ff70>
```



In [13]:

```
dt.fillna(method='ffill')
```

Out[13]:

	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 [14]:

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

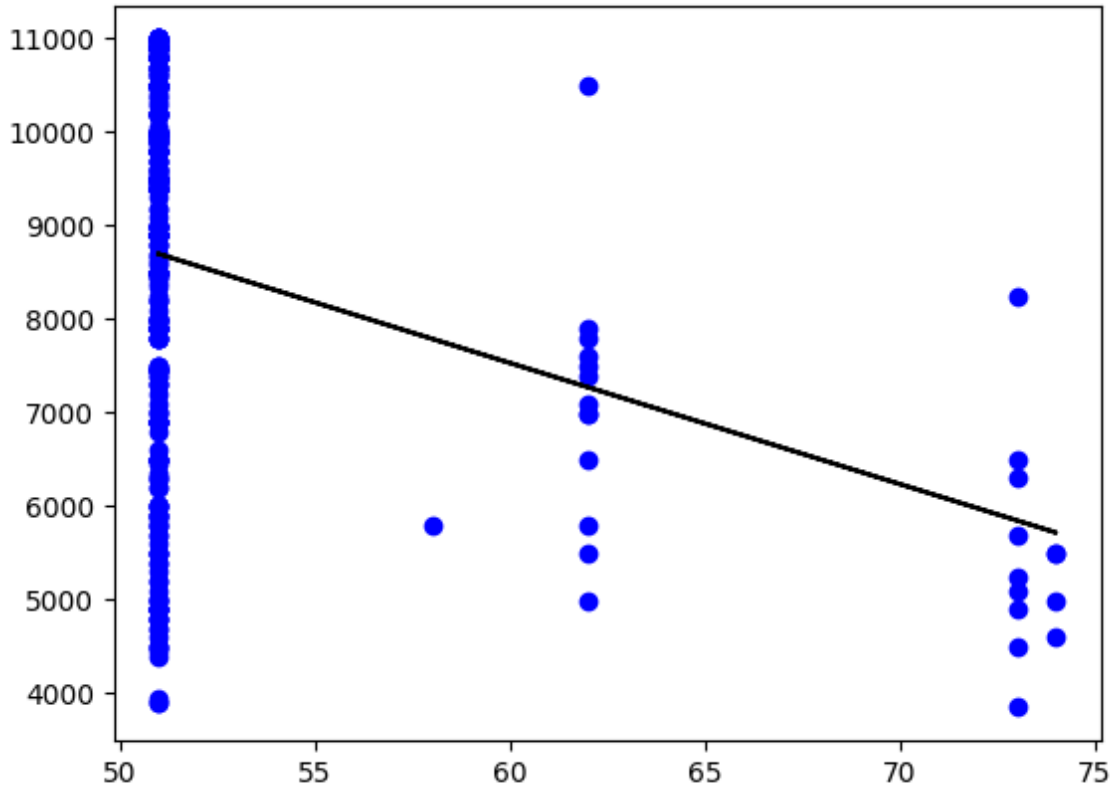
In [15]:

```
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))
```

```
0.10666917869657677
```

In [16]:

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

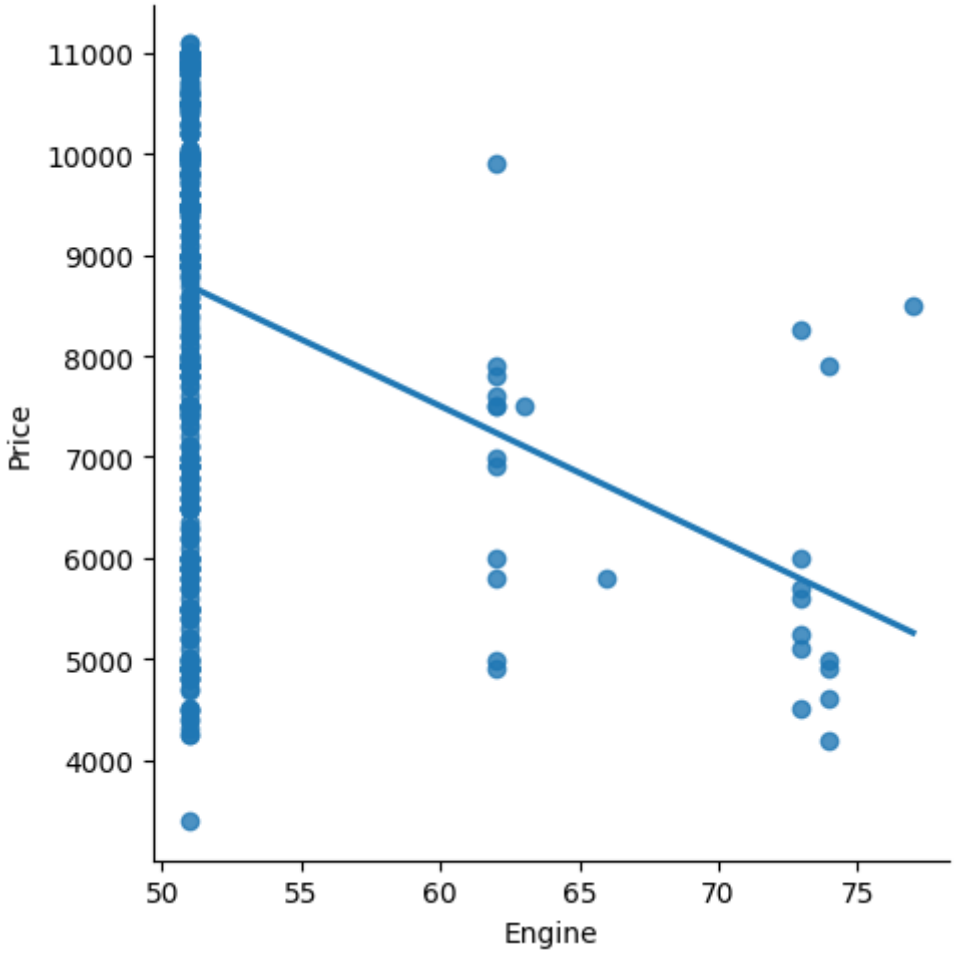


In [17]:

```
dt500=dt[:][:500]
sns.lmplot(x="Engine",y="Price",data=dt500,order=1,ci=None)
```

Out[17]:

```
<seaborn.axisgrid.FacetGrid at 0x213a426d8d0>
```



In [18]:

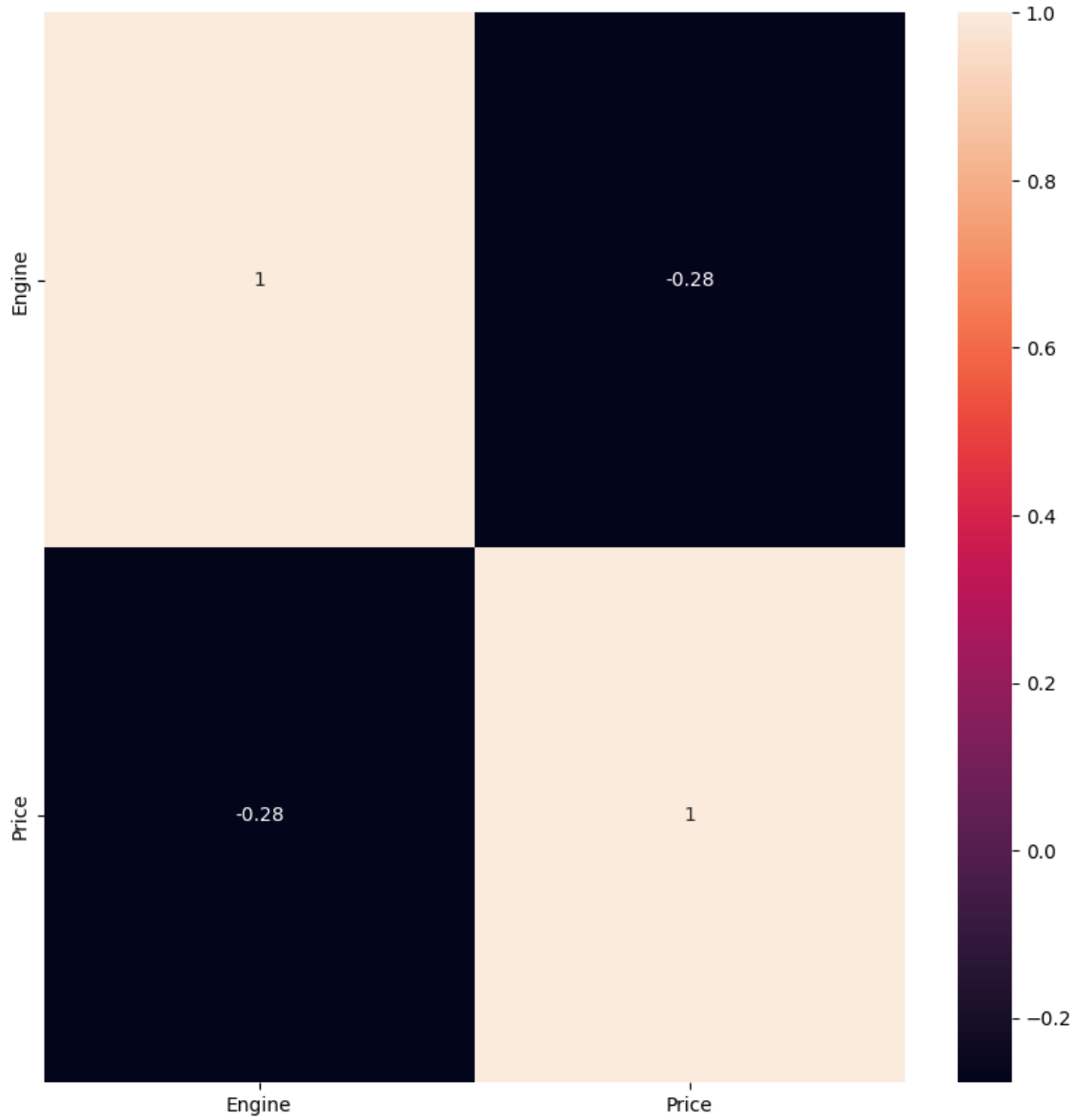
```
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
```

In [19]:

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

Out[19]:

<Axes: >



In [20]:

```
features = dt.columns[0:2]
target = dt.columns[-1]
#X and y values
X = dt[features].values
y = dt[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 [21]:

```
#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("\nRidg 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))
```

```
Ridg Model:

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

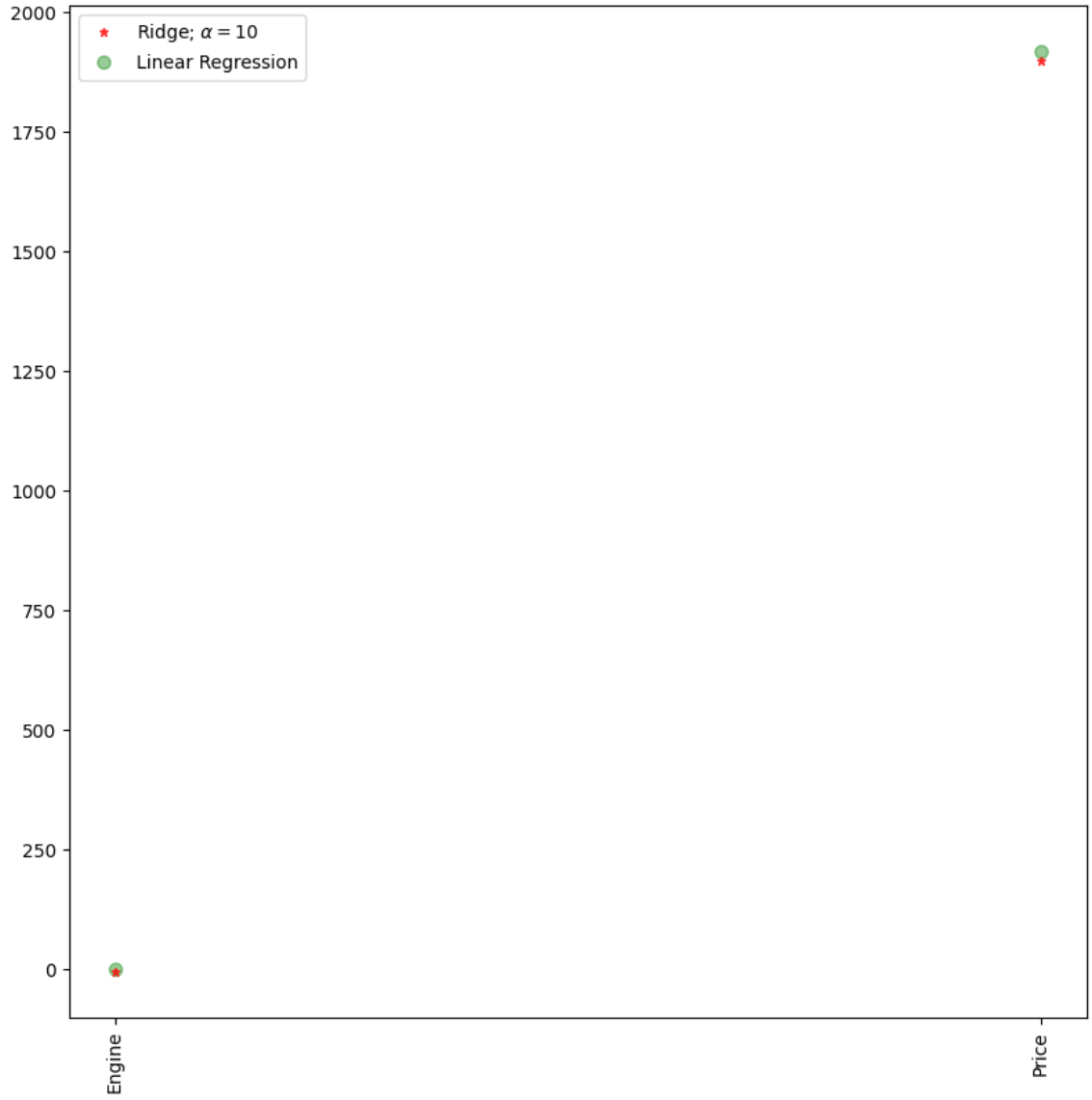
In [22]:

```
#Ridge Regression Model
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
#train and test scorefor 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 0.9999088581979684
The test score for ridge model is 0.9999100853681022
```

In [23]:

```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,colo
#lot(rr100.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='gre
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



In [24]:

```
#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))
```

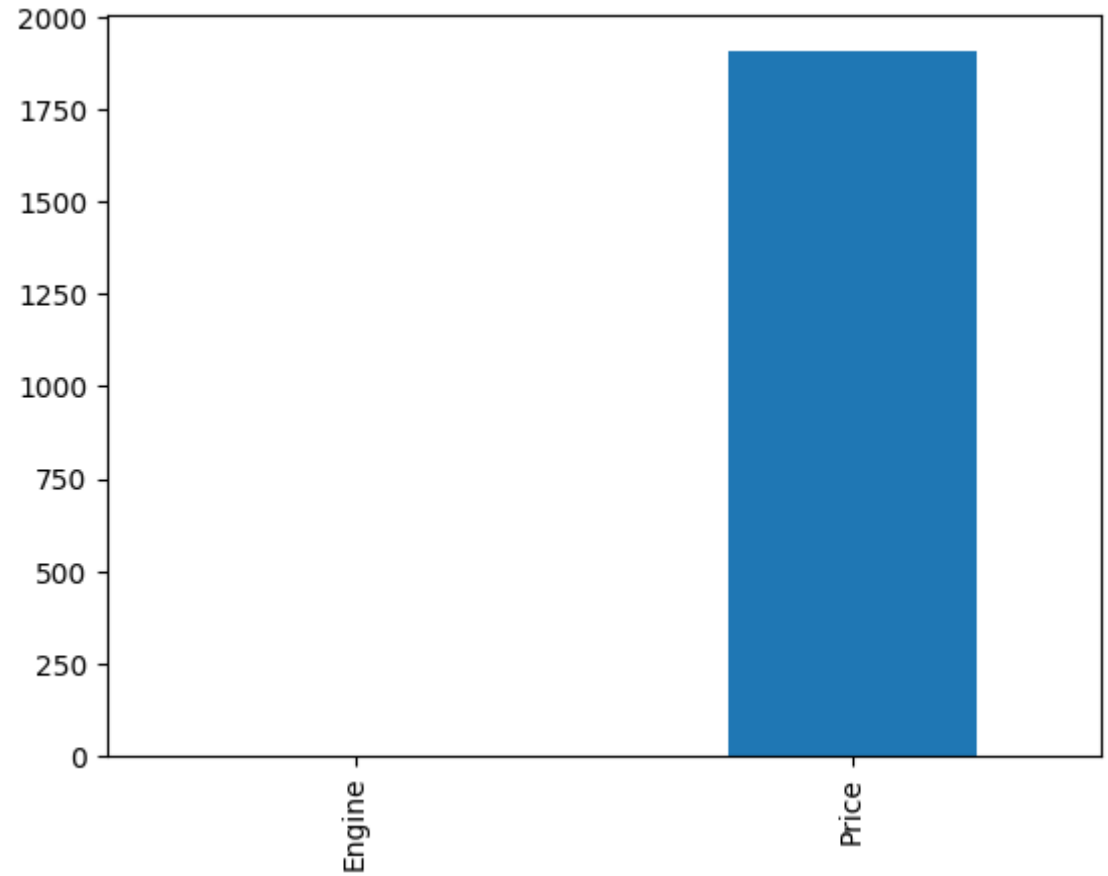
```
Lasso Model:

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

In [25]:

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

Out[25]:

```
<Axes: >
```



In [26]:

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

```
0.9999999999501757
0.9999999999638806
```

ELASTICNET

In [*]:

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

In [*]:

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

In [*]:

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

In [ ]:

```

```