In [1]:

```
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 [2]:

```
data=pd.read_csv(r"C:\Users\monim\Downloads\fiat500_VehicleSelection_Dataset.csv")
data
```

Out[2]:

	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 [3]:

```
data.head()
```

Out[3]:

	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 [4]:

```
data.tail()
```

Out[4]:

	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 [5]:

```
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 [6]:

```
data.describe()
```

Out[6]:

	ID	engine_power	age_in_days	km	previous_owners	l
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.00000
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.54136
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.13351
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.85583
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.80299
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.39409
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.46796
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.79561

In [7]:

```
data.isnull().any()
```

Out[7]:

```
ID                  False
model               False
engine_power        False
age_in_days         False
km                  False
previous_owners     False
lat                 False
lon                 False
price               False
dtype: bool
```

In [8]:

```
data.isnull().sum()
```

Out[8]:

```
ID                  0
model               0
engine_power        0
age_in_days         0
km                  0
previous_owners     0
lat                 0
lon                 0
price               0
dtype: int64
```

In [9]:

```
data.columns
```

Out[9]:

```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owner
s',
       'lat', 'lon', 'price'],
      dtype='object')
```

In [10]:

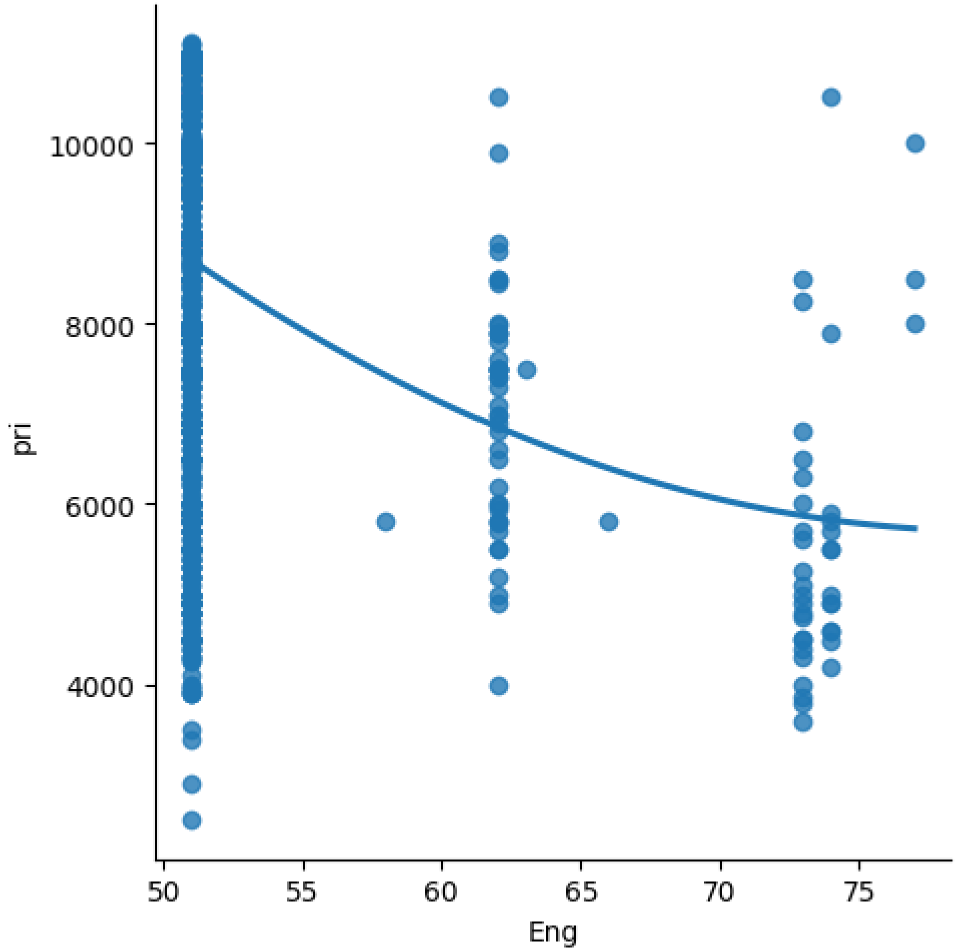
```
data = data[['engine_power','price']]
data.columns=['Eng','pri']
```

In [11]:

```
sns.lmplot(x='Eng',y='pri',data=data,order=2,ci=None)
```

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x1fa70a5a0b0>
```



In [12]:

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

Out[12]:

	Eng	pri
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 [13]:

```
x=np.array(data['Eng']).reshape(-1,1)
y=np.array(data['pri']).reshape(-1,1)
```

In [14]:

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

```
C:\Users\monim\AppData\Local\Temp\ipykernel_26768\286435216.py:1: SettingW
ithCopyWarning:
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-doc
s/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)
  data.dropna(inplace=True)
```

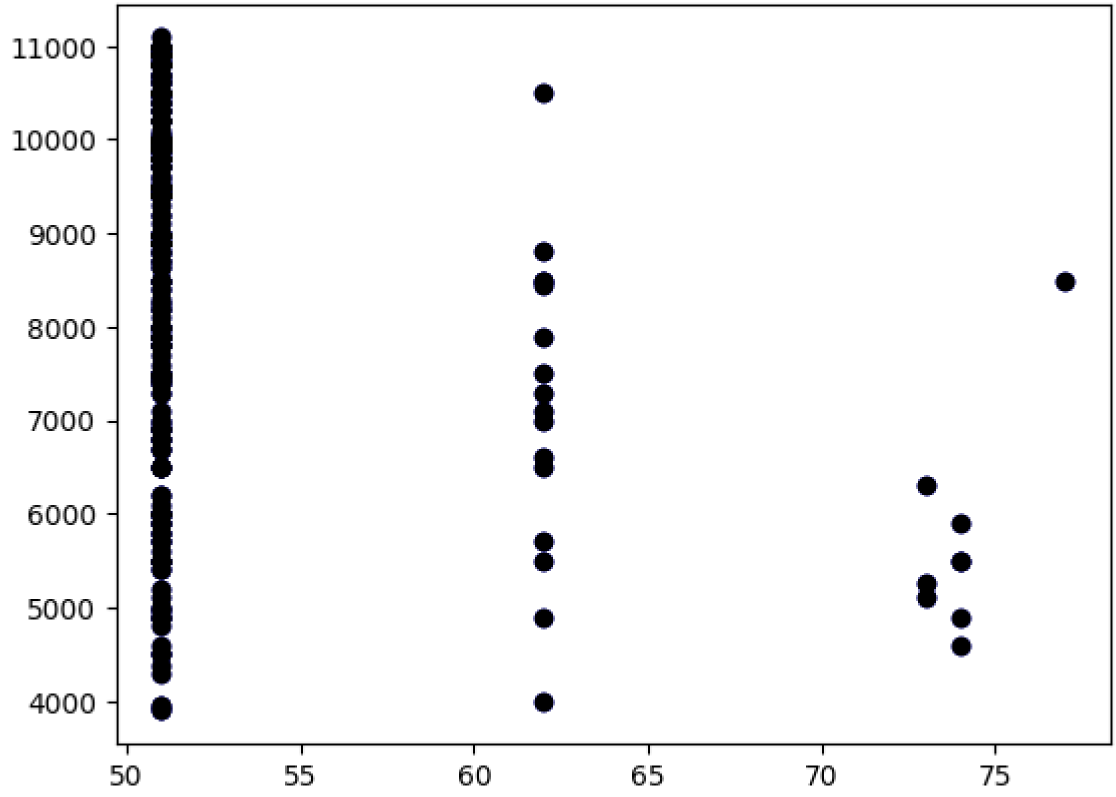
In [15]:

```
X_train,X_test,y_train,y_test = train_test_split(x, y, test_size = 0.25)
# Splitting the data into training data and test data
regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

```
0.07952662244908848
```

In [16]:

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

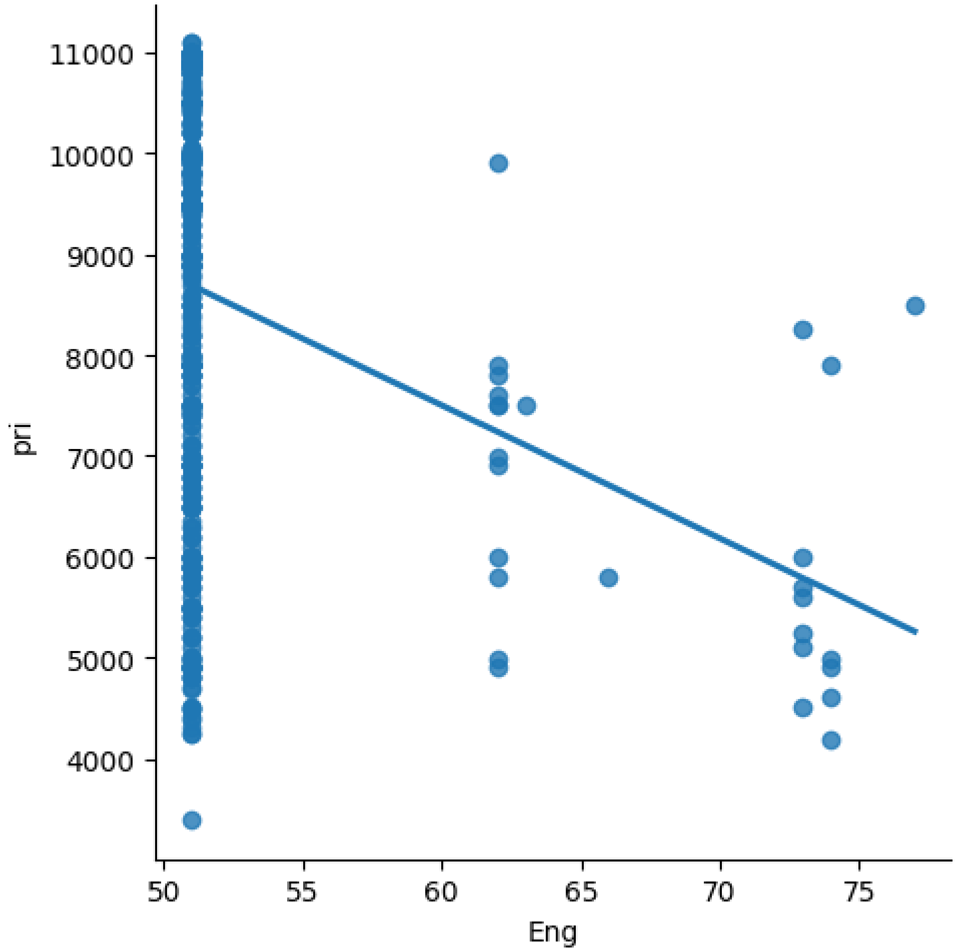


In [17]:

```
df500 = data[:][:500]
# Selecting the 1st 500 rows of teh data
sns.lmplot(x = "Eng", y = "pri", data = df500, order = 1, ci = None)
```

Out[17]:

```
<seaborn.axisgrid.FacetGrid at 0x1fa72c2bb20>
```



In [29]:

```
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("\nLinearRegression 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))
```

```

LinearRegression model:

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

In [30]:

```
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 dimensions of x_train is {}".format(x_train.shape))
print("The dimensions of x_train is {}".format(x_test.shape))
#scale features
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

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

RIDGE REGRESSION

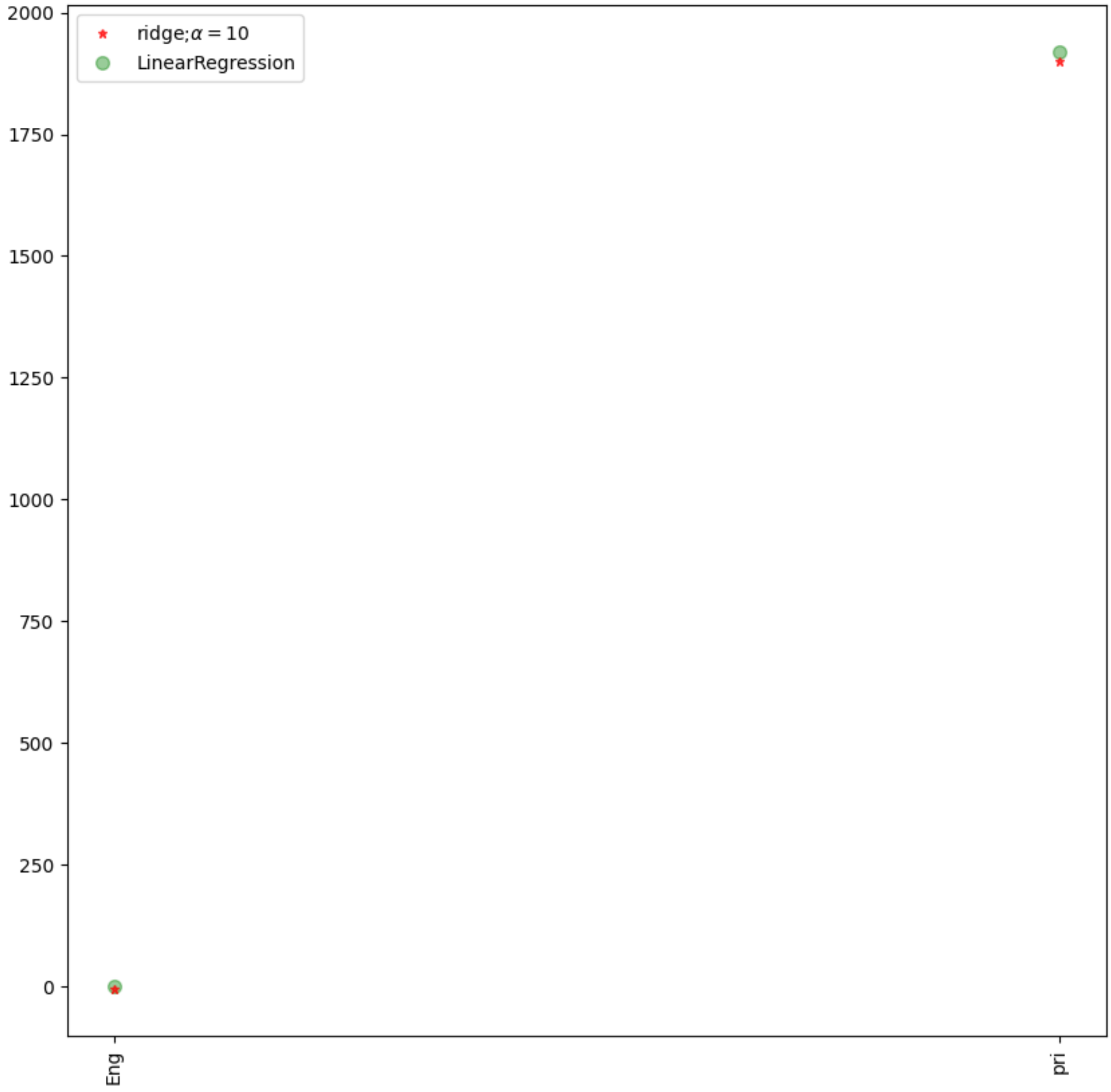
In [31]:

```
# 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")
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 [32]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,colo
         label=r'ridge;$\alpha=10$',zorder=7)
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gre
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



LASSO REGRESSION

In [33]:

```
#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))
```

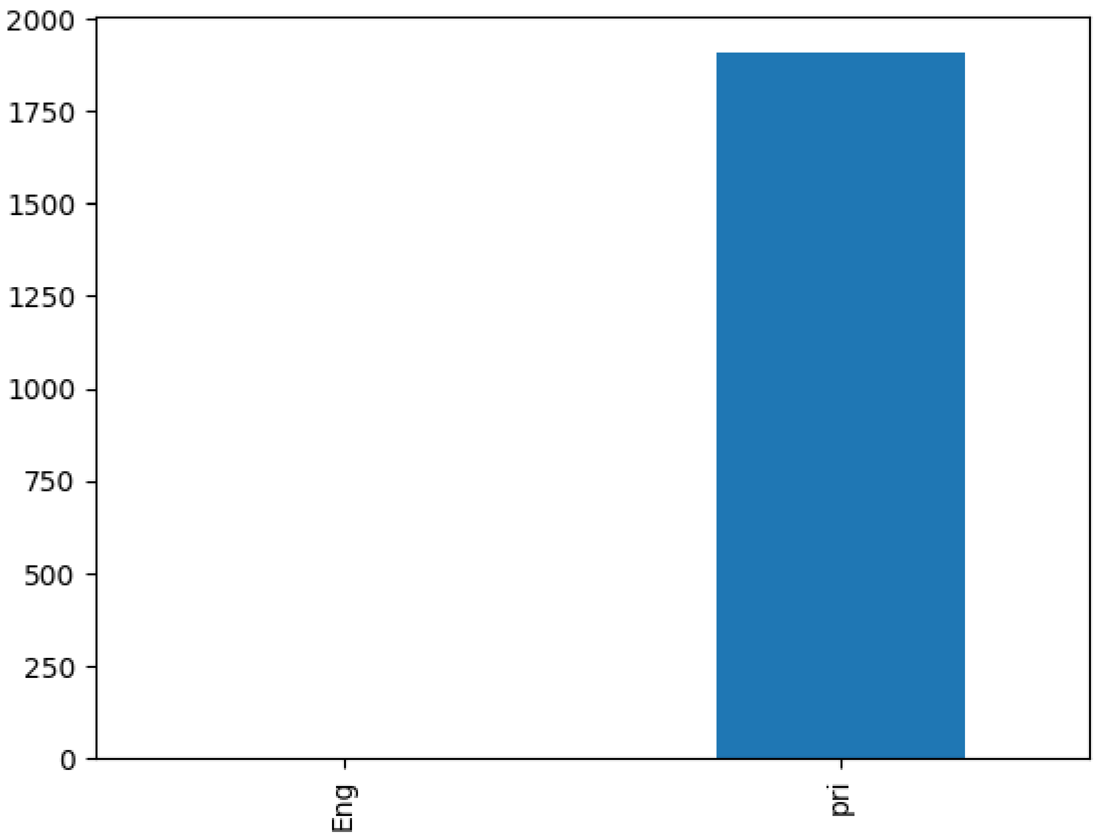
```
Lasso Model:

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

In [34]:

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

Out[34]:

```
<Axes: >
```



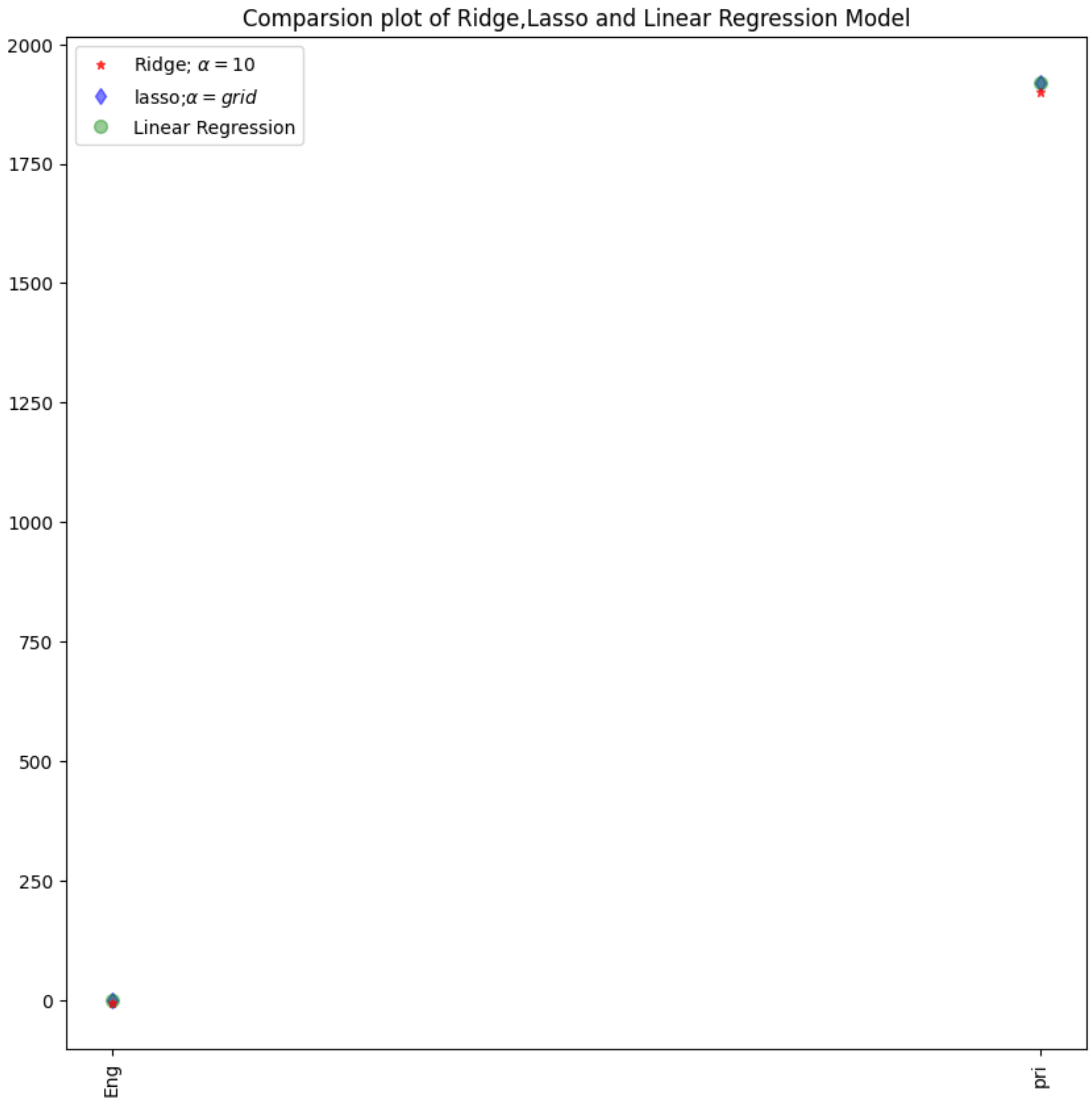
In [35]:

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

```
0.9999999999501757
0.9999999999638806
```

In [36]:

```
#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,colo
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gre
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparsion plot of Ridge,Lasso and Linear Regression Model")
plt.show()
```



In [37]:

```
#using the linear CV model
from sklearn.linear_model import RidgeCV

#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 0.9999999999999913
The train score for ridge model is 0.9999999999999917
```

ELASTICNET

In [38]:

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

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

In [39]:

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

In [40]:

```
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 [ ]: