In [2]:
```
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 [3]:
```
#data
data=pd.read_csv(r"C:\Users\sudheer\Downloads\Advertising.csv")
data
```

Out[3]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

200 rows × 4 columns

In [4]:
```
data.head()
```

Out[4]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

In [5]:
```
data.tail()
```

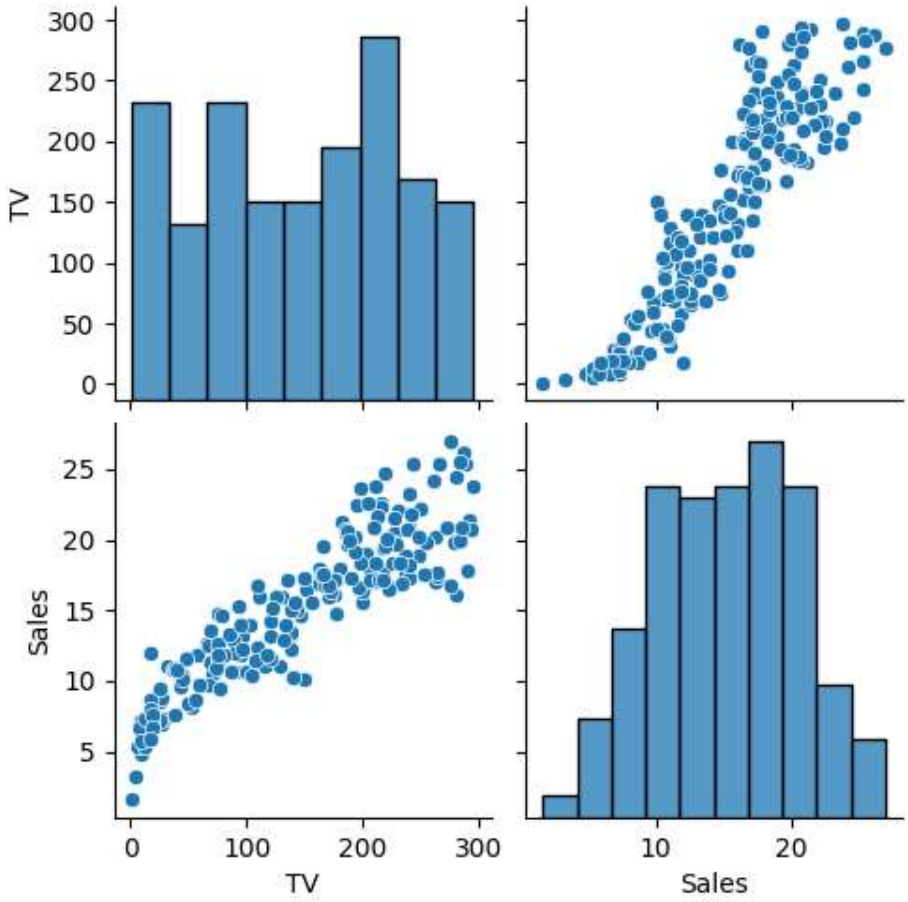
Out[5]:

	TV	Radio	Newspaper	Sales
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

In [6]:
```
plt.figure(figsize = (10, 10))
sns.heatmap(data.corr(), annot = True)
```

Out[6]: <Axes: >

In [7]:
```
data.drop(columns = ["Radio", "Newspaper"], inplace = True)
#pairplot
sns.pairplot(data)
data.Sales = np.log(data.Sales)
```



In [8]:
```
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=1
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 (140, 2)
The dimension of X_test is (60, 2)
```

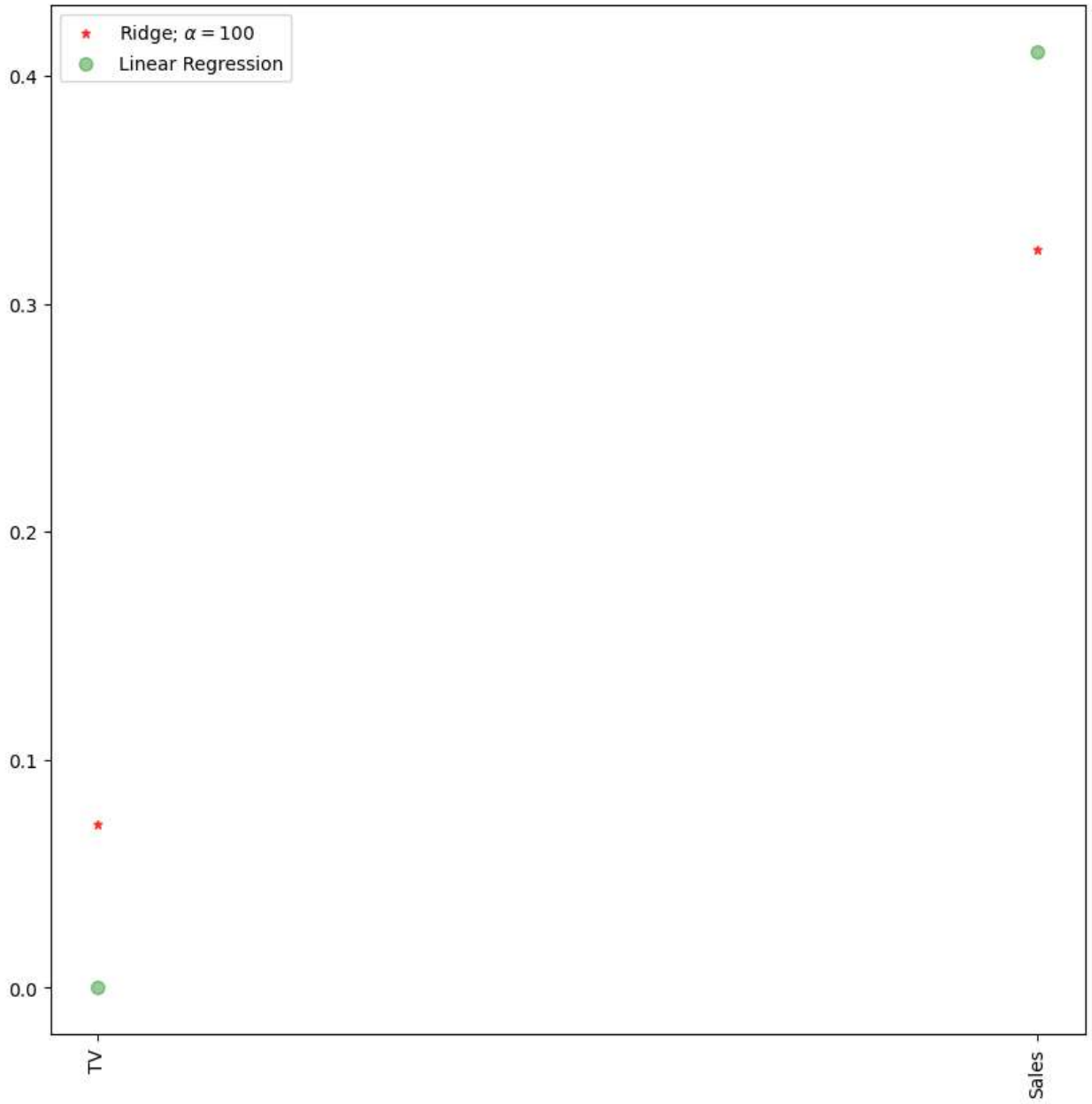
In [9]:
```
#Model
lr = LinearRegression()
#Fit model
lr.fit(X_train, y_train)
#predict
#prediction = lr.predict(X_test)
#actual
actual = y_test
train_score_lr = lr.score(X_train, y_train)
test_score_lr = lr.score(X_test, y_test)
print("\nLinear Regression Model:\n")
print("The train score for lr model 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 1.0
The test score for lr model is 1.0
```

In [10]:
```
#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))
```

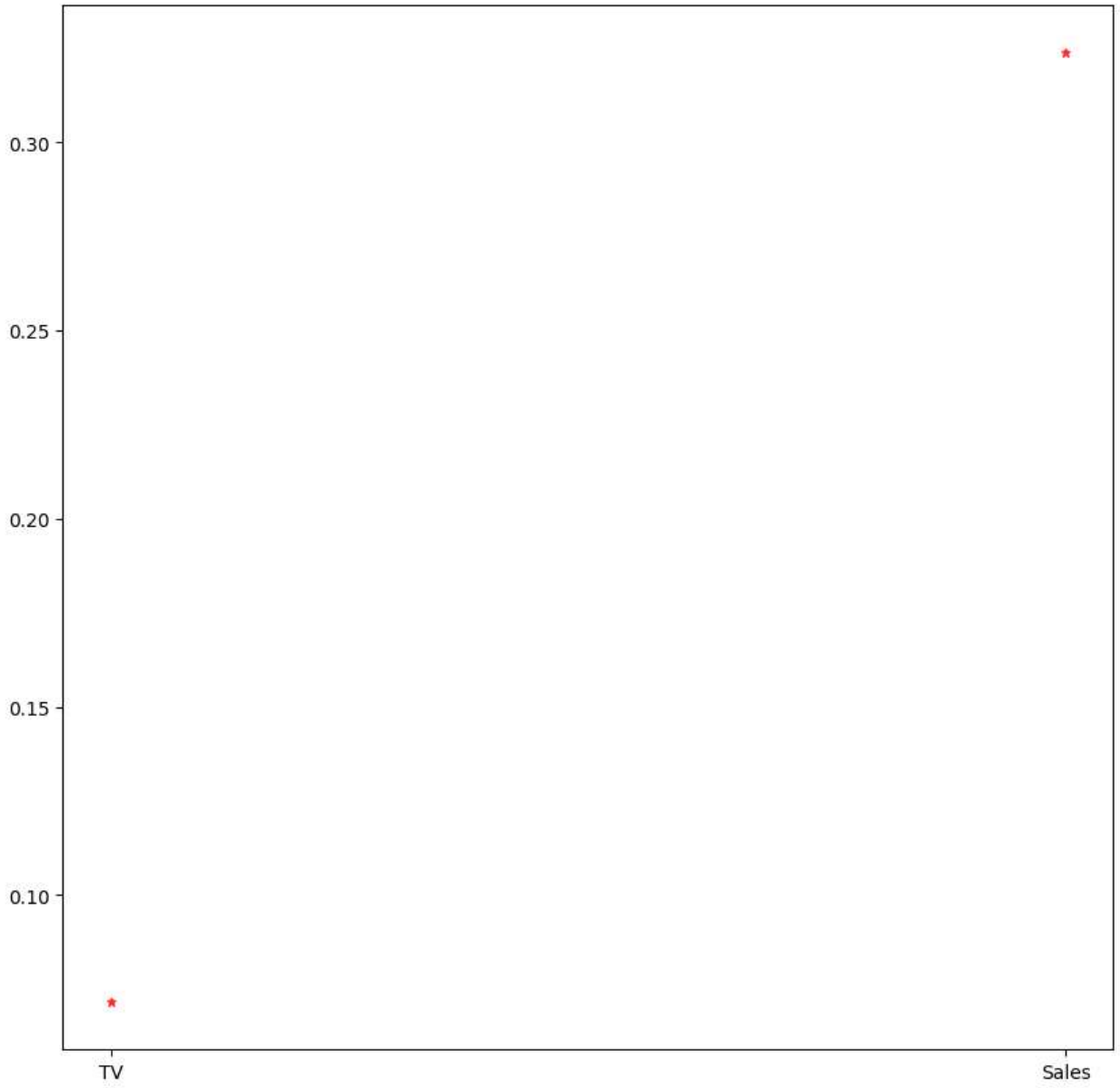
```
Ridge Model:

The train score for ridge model is 0.990287139194161
The test score for ridge model is 0.9844266285141221
```

In [11]:
```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,col
#plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',l
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gr
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



In [15]:
```
#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,col
#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='gr
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[15], line 6
      4 plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers
ize=5,color='red',label=r'Ridge; $\alpha=100$')
      5 #add plot for lasso regression
----> 6 plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,col
or='blue',label=r'lasso; $\alpha = grid$')
      7 #add plot for linear model
      8 plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,
color='green',label='Linear Regression')

NameError: name 'lasso_cv' is not defined
```

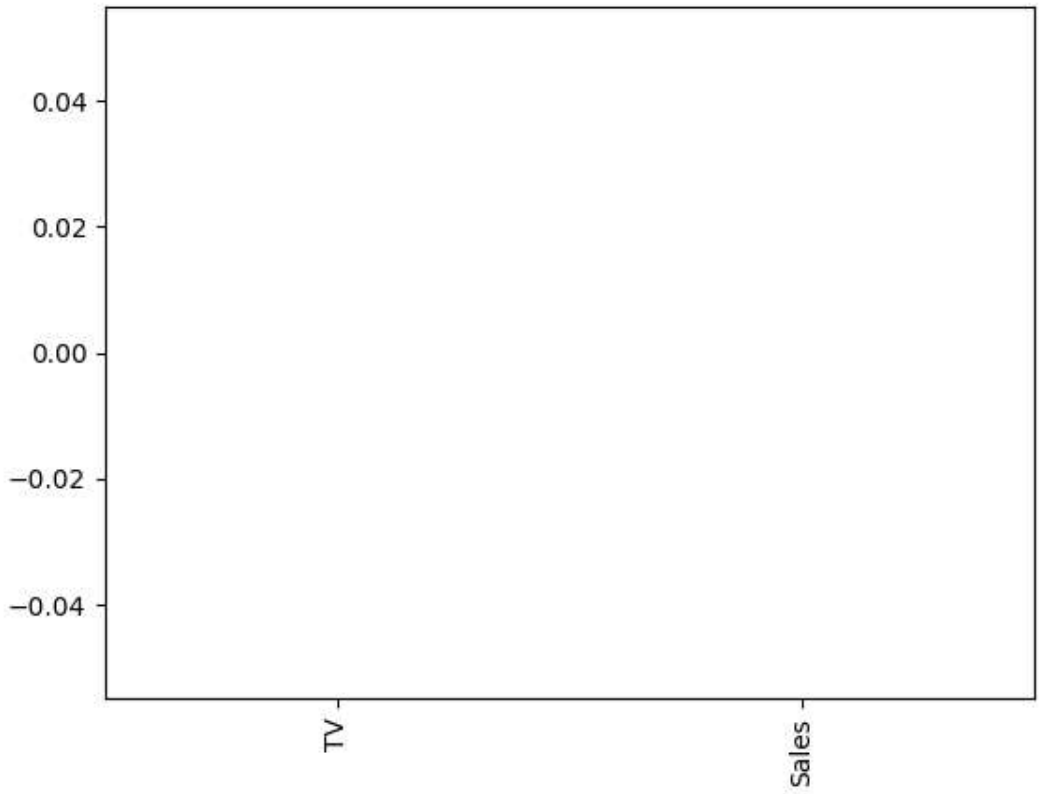


In [16]:
```
#Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
Lasso Model:

The train score for ls model is 0.0
The test score for ls model is -0.0042092253233847465
```

In [17]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[17]: <Axes: >



In [18]:
```
#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_tra
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.9999999343798134
0.9999999152638072
```

Vehicle selection

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
data=pd.read_csv(r"C:\Users\sudheer\Downloads\fiat500_VehicleSelection_Dataset (2).csv"
data
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...	...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 9 columns

In [3]:
```
data = data[['engine_power', 'price']]
data.columns=['Eng', 'pri']
```

In [4]:
```
data.head()
```

Out[4]:

	Eng	pri
0	51	8900
1	51	8800
2	74	4200
3	51	6000
4	73	5700

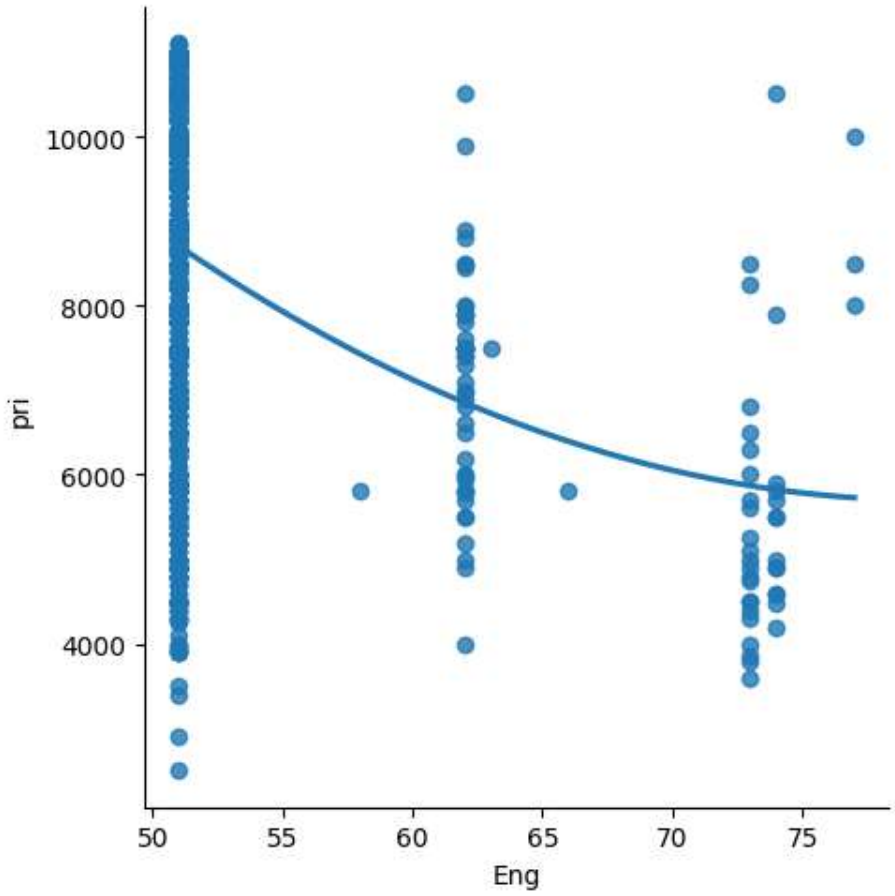
In [5]:
```
data.tail()
```

Out[5]:

	Eng	pri
1533	51	5200
1534	74	4600
1535	51	7500
1536	51	5990
1537	51	7900

In [6]:
```
sns.lmplot(x='Eng',y='pri',data=data,order=2,ci=None)
```

Out[6]:
```
<seaborn.axisgrid.FacetGrid at 0x270a1fa5870>
```



In [7]:
```
data.info()
```

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

In [8]:
```
data.describe()
```

Out[8]:

	Eng	pri
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 [9]:
```
data.fillna(method='ffill')
```

Out[9]:

	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 [10]:
```
x=np.array(data['Eng']).reshape(-1,1)
y=np.array(data['pri']).reshape(-1,1)
```

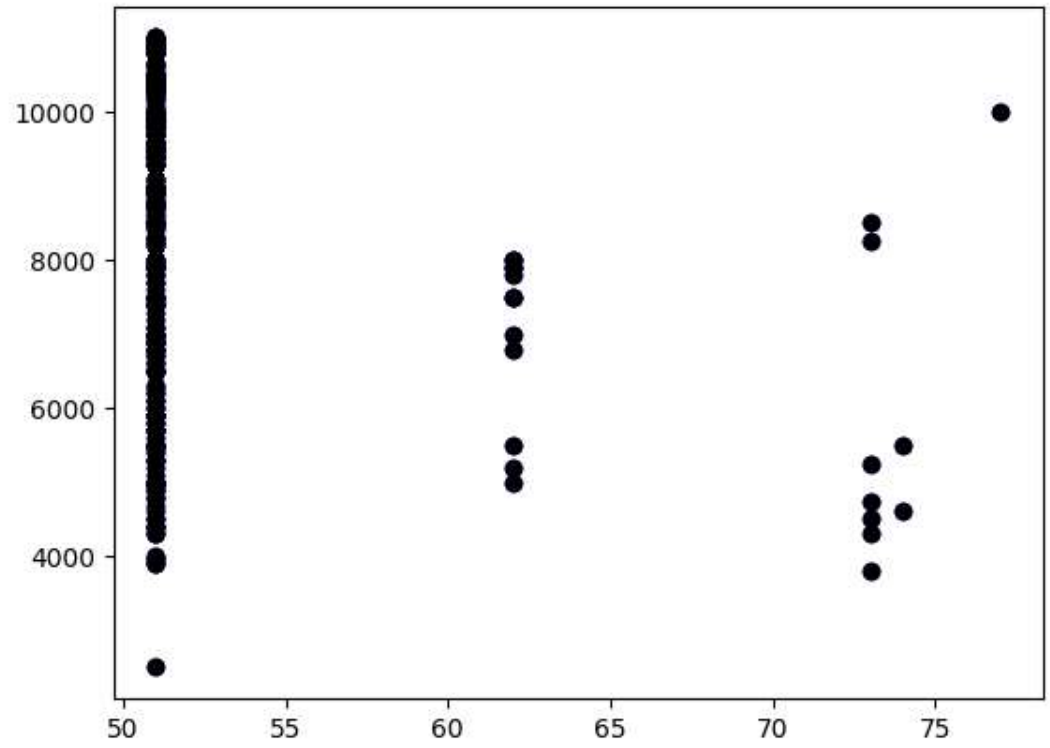
In [11]:
```
data.dropna(inplace=True)
```

```
C:\Users\sudheer\AppData\Local\Temp\ipykernel_14200\1580077326.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/use
r_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 [12]:
```
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))
```

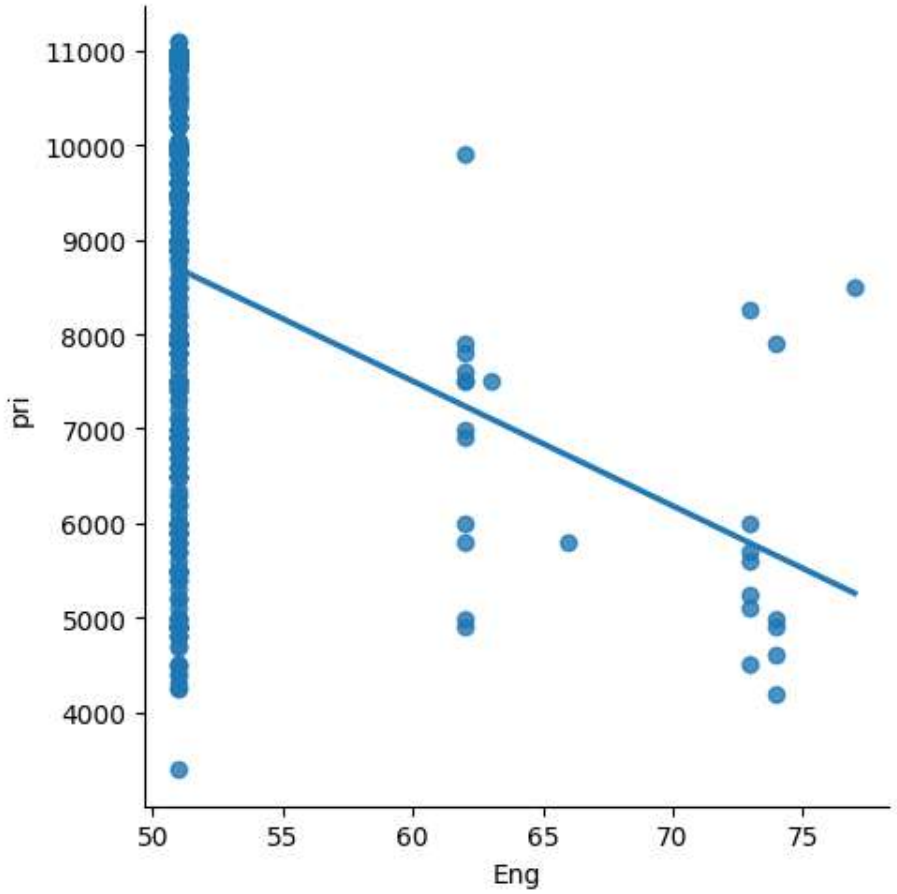
```
0.06427439865414497
```

In [13]:
```
y_pred=regr.predict(X_test)
plt.scatter(X_test, y_test, color = 'b')
plt.scatter(X_test, y_test, color = 'k')
plt.show()
```



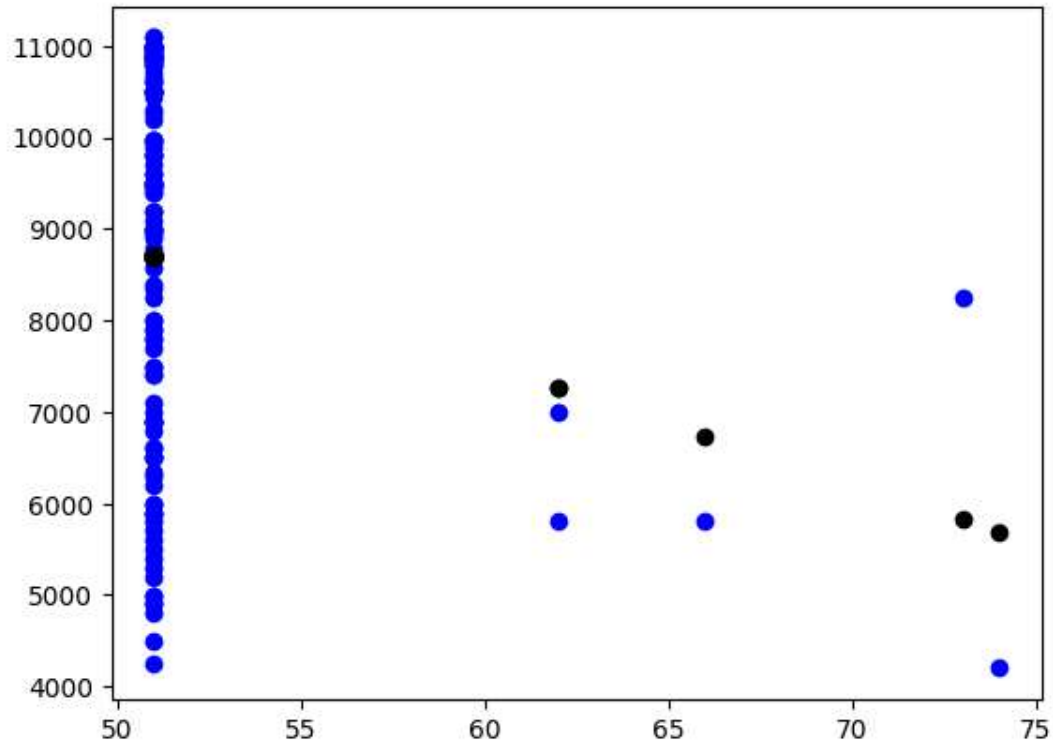
In [14]:
```
df500 = data[:][:500]
# Selecting the 1st 500 rows of teh data
sns.lmplot(x = "Eng", y = "pri", data = df500, order = 1, ci = None)
```

Out[14]: <seaborn.axisgrid.FacetGrid at 0x2708fbc0550>



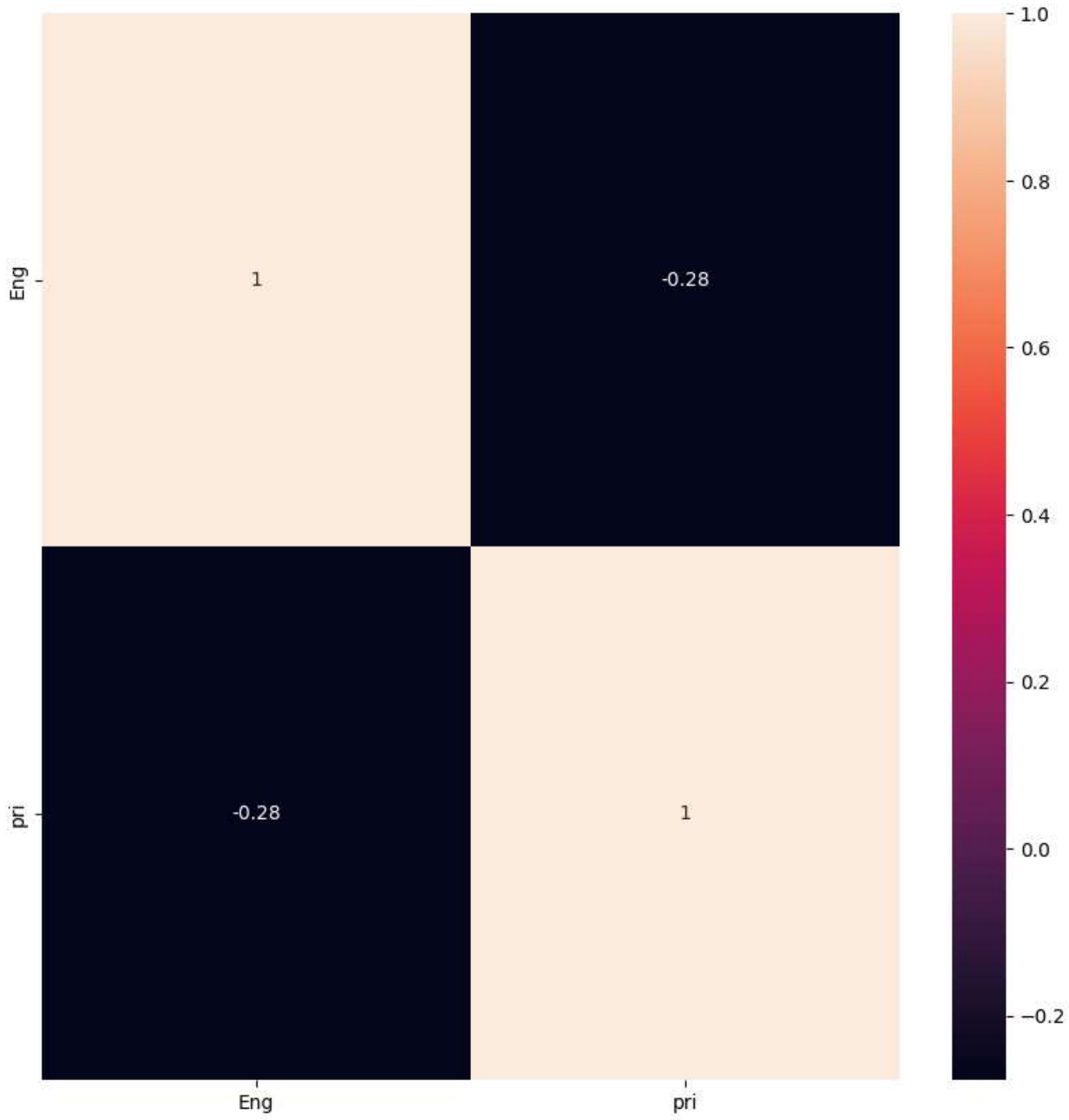
In [15]:
```
df500.fillna(method='ffill',inplace=True)
x=np.array(df500['Eng']).reshape(-1,1)
y=np.array(df500['pri']).reshape(-1,1)
df500.dropna(inplace=True)
X_train,X_test,y_train, y_test = train_test_split(x, y, test_size = 0.25)
# Splitting the data into training data and test data
regr= LinearRegression()
regr.fit(X_train, y_train)
print("Regression:",regr.score(X_test,y_test))
y_pred=regr.predict(X_test)
plt.scatter(X_test, y_test, color = 'b')
plt.scatter(X_test, y_pred, color = 'k')
plt.show()
```

```
Regression: 0.05228398076602292
```



In [16]:
```
plt.figure(figsize = (10, 10))
sns.heatmap(data.corr(), annot = True)
```

Out[16]: <Axes: >



In [17]:
```
#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 0.07526429590089345
The test score for lr model is 0.05228398076602292
```

In [18]:
```
#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 0.07526412778450209
The test score for ridge model is 0.05227522794569095
```

In [19]:
```
#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 0.0752629050977659
The test score for ls model is 0.05225824306507554
```

In [ ]: