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 [3]:

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

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
...	...	...	...	...	...	...	...	...	...
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 [4]:

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

In [5]:

```
data.head()
```

Out[5]:

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

In [6]:

```
data.tail()
```

Out[6]:

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

In [7]:

```
df.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 [8]:

```
df.describe()
```

Out[8]:

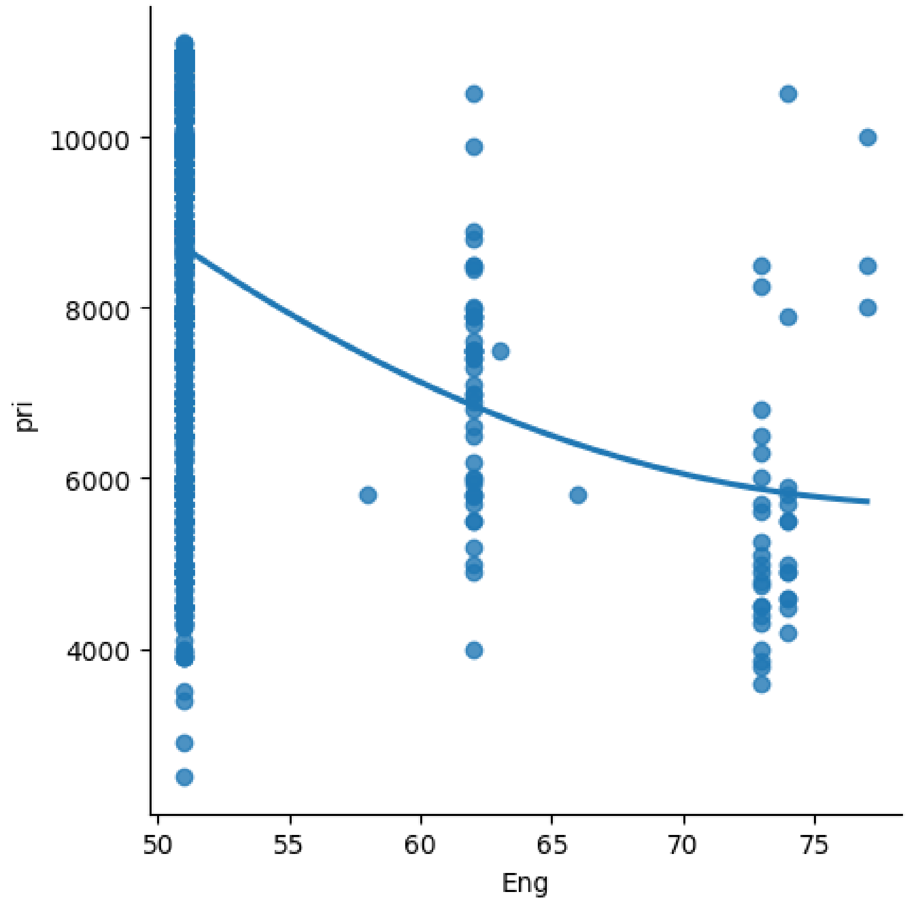
	ID	engine_power	age_in_days	km	previous_owners	lat	
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.00
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	11.56
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	2.32
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	7.24
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	9.50
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	11.86
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	12.76
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	18.36

In [9]:

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

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x21bc8e19180>
```



In [10]:

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

Out[10]:

	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 [11]:

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

In [12]:

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

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_11504\1368182302.py:1: SettingWithCopyW
arning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.o
rg/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  data.dropna(inplace=True)
```

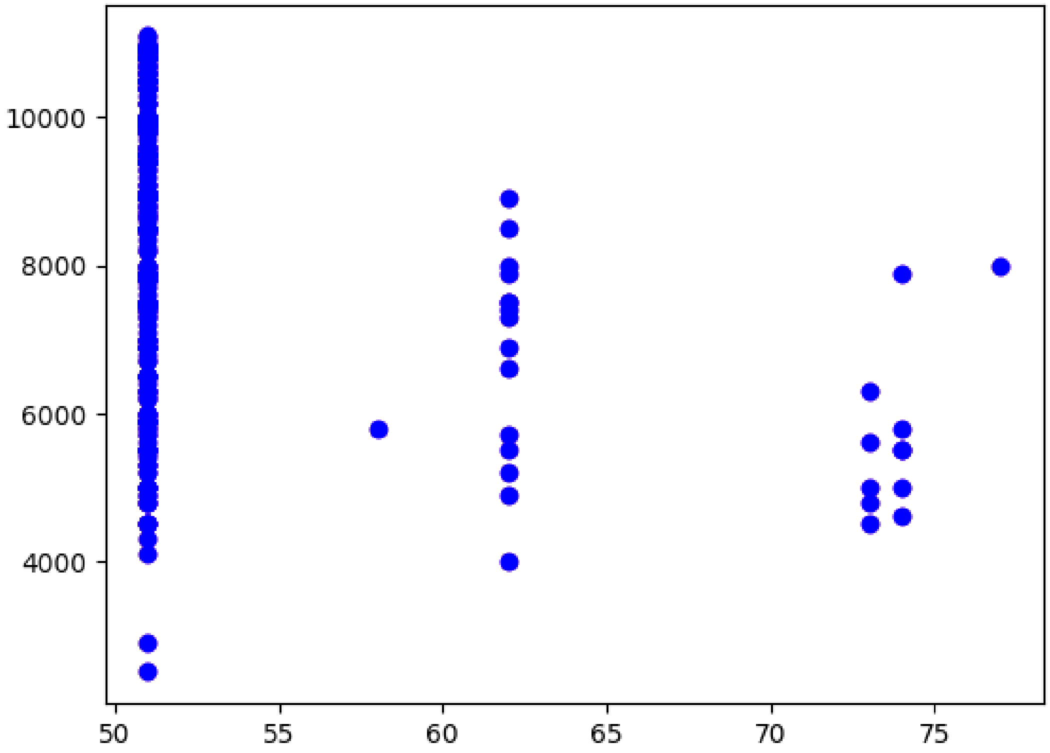
In [13]:

```
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))
```

```
0.09480072970098752
```

In [14]:

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

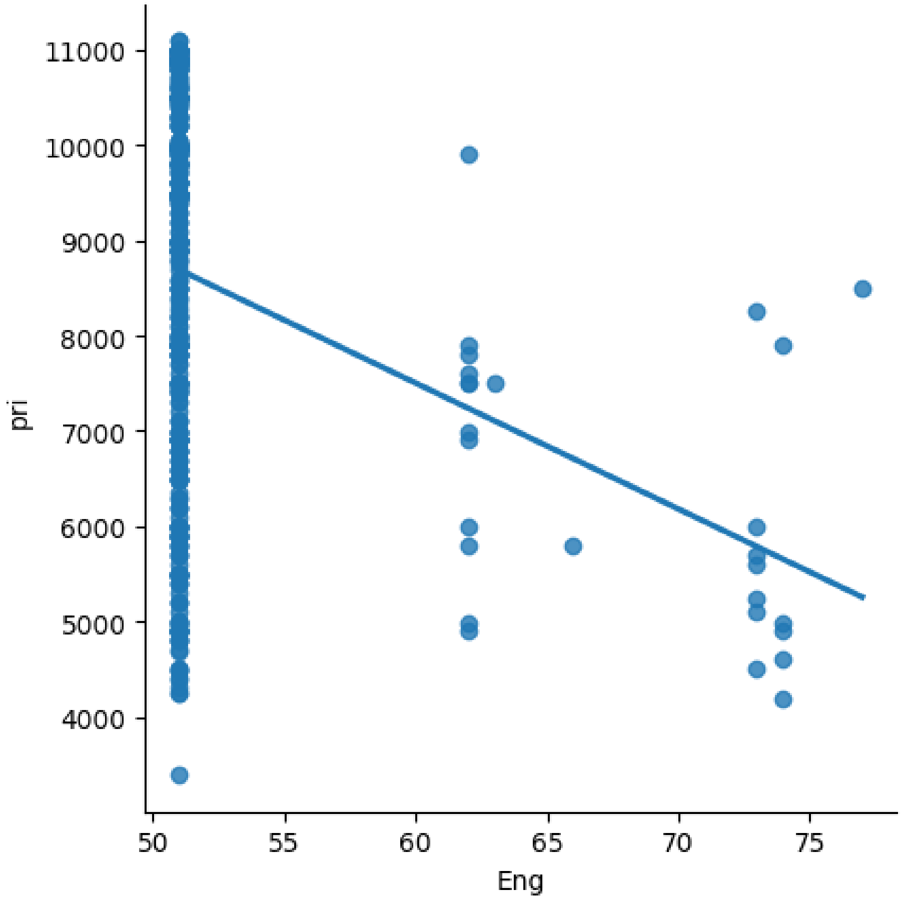


In [15]:

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

Out[15]:

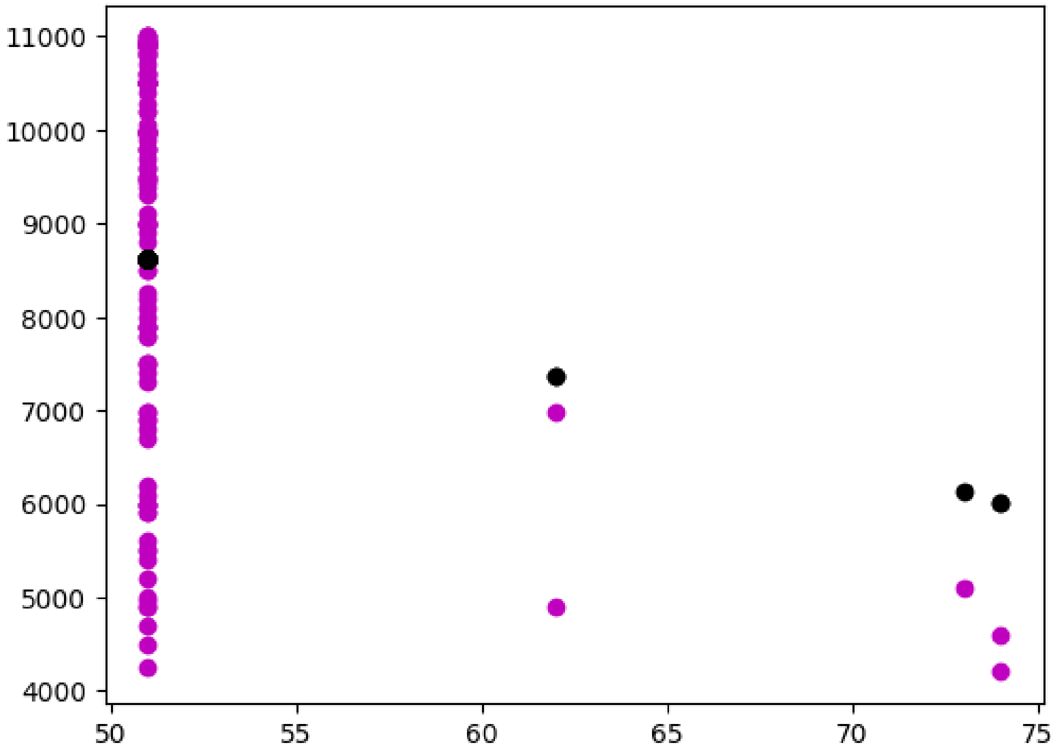
```
<seaborn.axisgrid.FacetGrid at 0x21bda696080>
```



In [17]:

```python
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)
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 = 'm')
plt.scatter(X_test, y_pred, color = 'k')
plt.show()
```

```
Regression: 0.09368490248450723
```

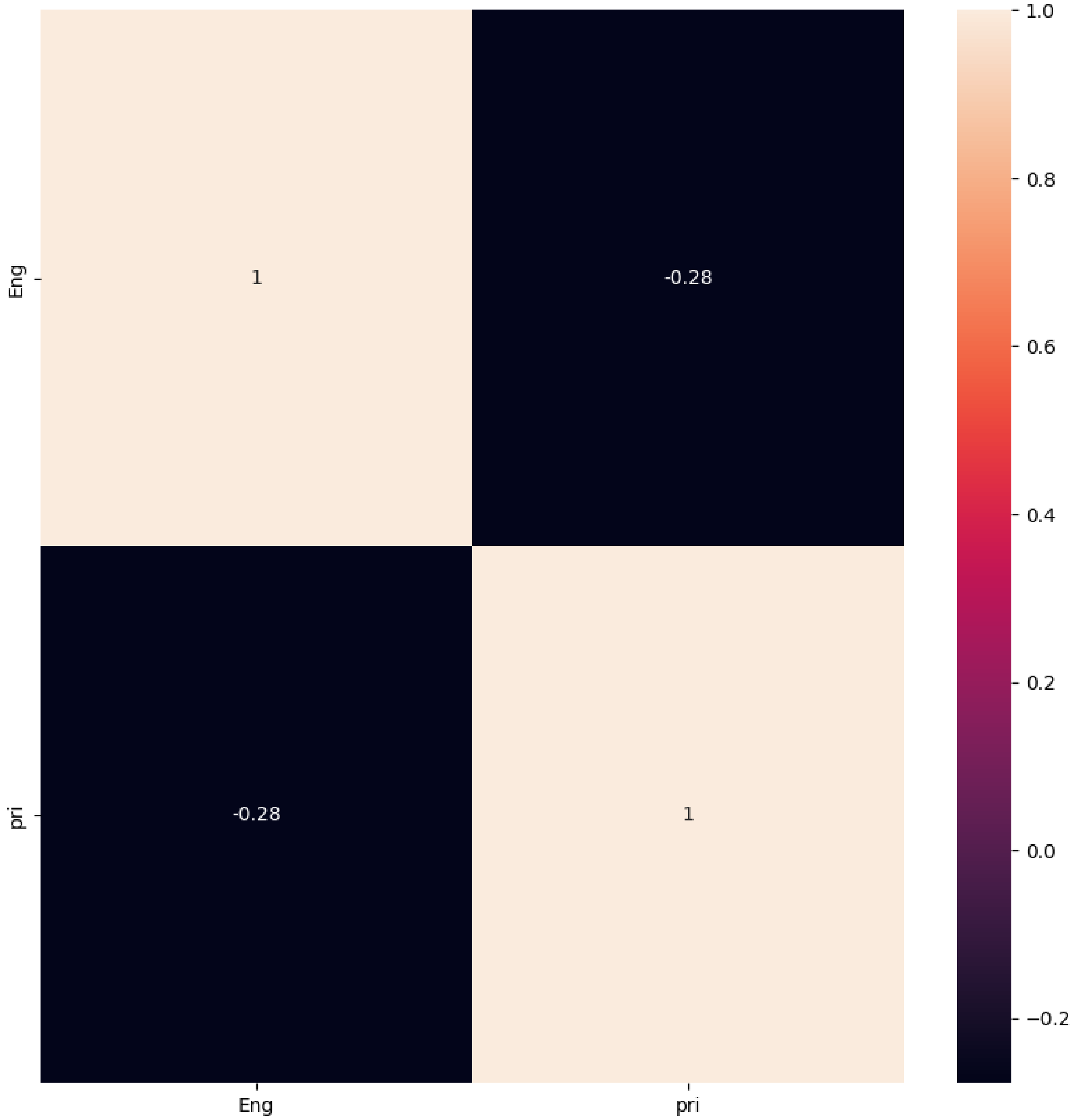


In [18]:

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

Out[18]:

```
<Axes: >
```



In [19]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
#Train the model
model = LinearRegression()
model.fit(X_train, y_train)
#Evaluating the model on the test set
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R2 score:",r2)
```

```
R2 score: 0.09368490248450723
```

In [20]:

```
#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 0.05442978352678385
The test score for lr model is 0.09368490248450723
```

In [21]:

```
#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 0.054429650752681025
The test score for ridge model is 0.09358153783492007
```

In [22]:

```
#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 0.05442832104494211
The test score for ls model is 0.09334106667257014
```

In [24]:

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

```
0.05442832104494211
0.09334106667257014
```

```
C:\Users\DELL\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\li
near_model\_coordinate_descent.py:1568: DataConversionWarning: A column-vector y w
as passed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Elastic Net Regression

In [25]:

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

```
[-128.05913739]
[15219.18170389]
```

In [26]:

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

In [27]:

```
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 4293441.03417857
```

In [ ]: