Vehicle Selection

In [1]:

```
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
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
```

In [2]:

```
#Step-2:Reading the dataset
dt=pd.read_csv(r"C:\Users\91955\Downloads\fiat500_VehicleSelection_Dataset (1).csv")
dt
```

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]:

```
dt=dt[['engine_power','age_in_days']]
#Taking only the selected two attributes from the dataset
dt.columns=['Eng','Age']
#Renaming the columns for easier writing of the code
```

In [4]:

```
dt.head(10)
#Displaying only the 1st 10 rows
```

Out[4]:

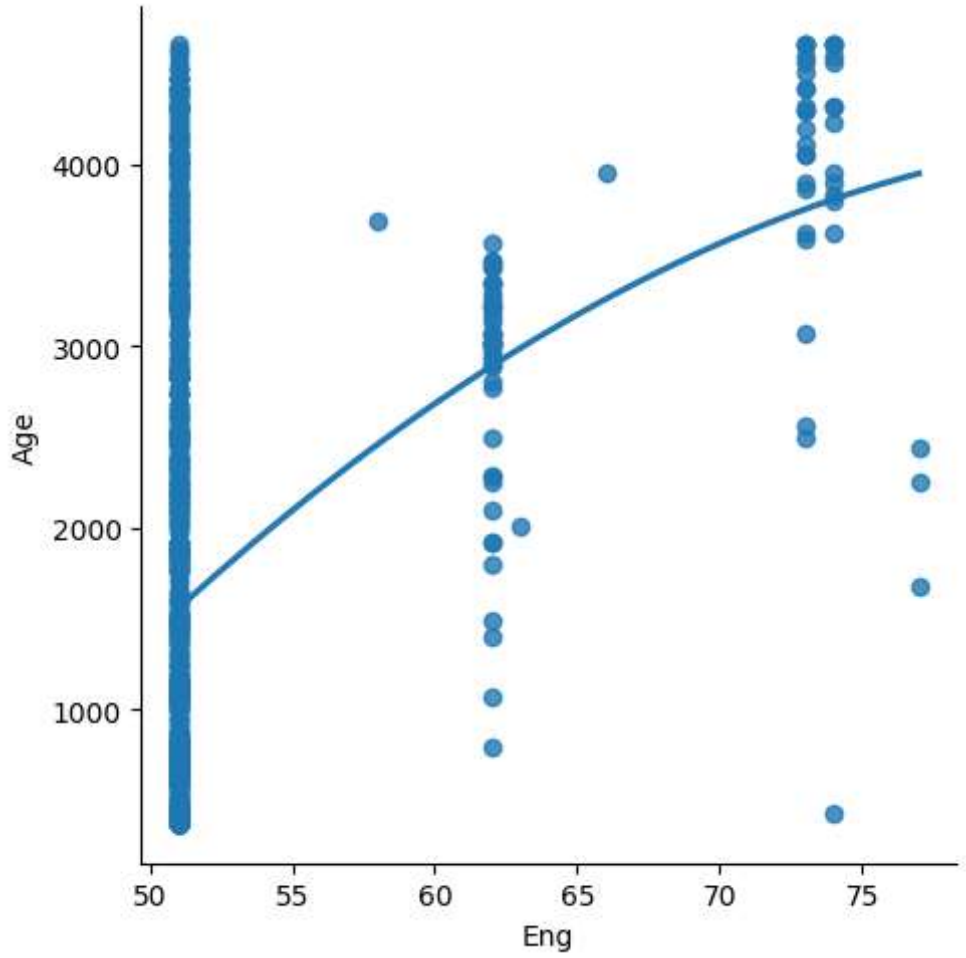
	Eng	Age
0	51	882
1	51	1186
2	74	4658
3	51	2739
4	73	3074
5	74	3623
6	51	731
7	51	1521
8	73	4049
9	51	3653

In [5]:

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

Out[5]:

```
<seaborn.axisgrid.FacetGrid at 0x1f5ef998610>
```



In [6]:

```
1 dt.describe()
```

Out[6]:

	Eng	Age
count	1538.000000	1538.000000
mean	51.904421	1650.980494
std	3.988023	1289.522278
min	51.000000	366.000000
25%	51.000000	670.000000
50%	51.000000	1035.000000
75%	51.000000	2616.000000
max	77.000000	4658.000000

In [7]:

```
1 dt.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   Age     1538 non-null   int64
dtypes: int64(2)
memory usage: 24.2 KB
```

In [8]:

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

Out[8]:

	Eng	Age
0	51	882
1	51	1186
2	74	4658
3	51	2739
4	73	3074
...	...	...
1533	51	3712
1534	74	3835
1535	51	2223
1536	51	2557
1537	51	1766

1538 rows × 2 columns

In [9]:

```
1 dt.isnull().sum()
```

Out[9]:

```
Eng    0
Age    0
dtype: int64
```

In [10]:

```
1 dt.isna().any()
```

Out[10]:

```
Eng    False
Age    False
dtype: bool
```

In [11]:

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

In [12]:

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

```
C:\Users\91955\AppData\Local\Temp\ipykernel_8716\735218168.py:1: SettingWi
thCopyWarning:
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)
  dt.dropna(inplace=True)
```

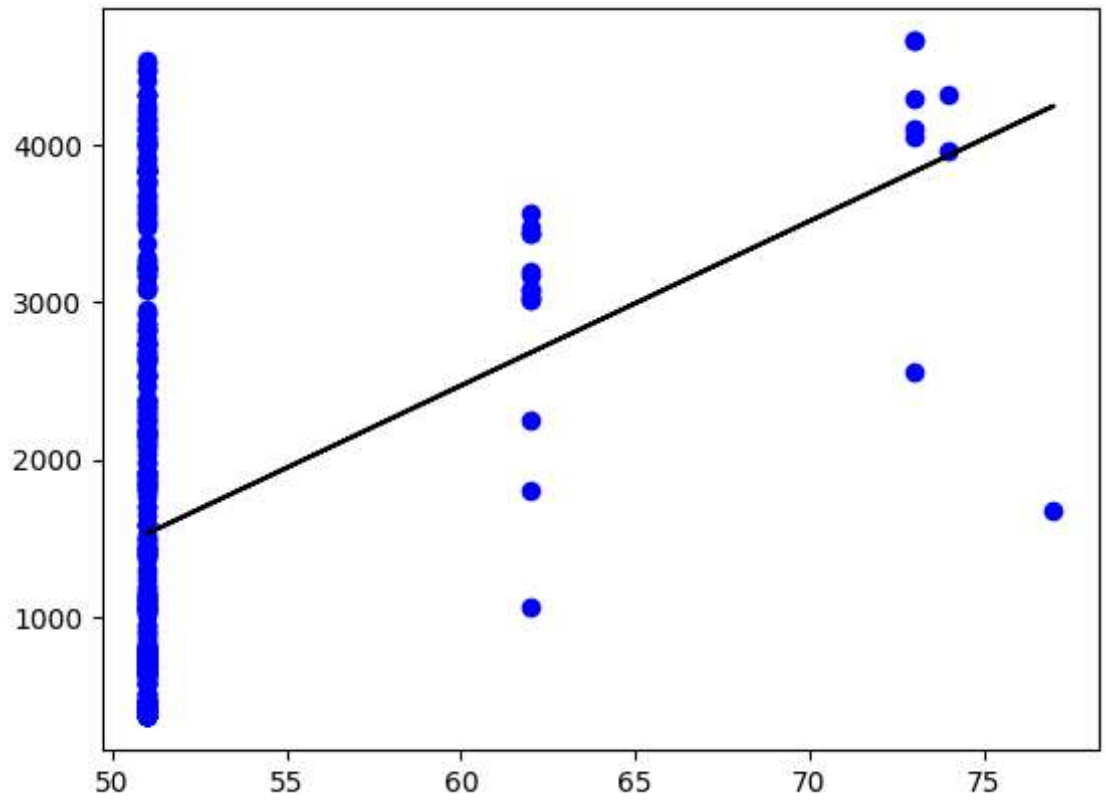
In [13]:

```
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))
```

```
0.08680240375465742
```

In [14]:

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

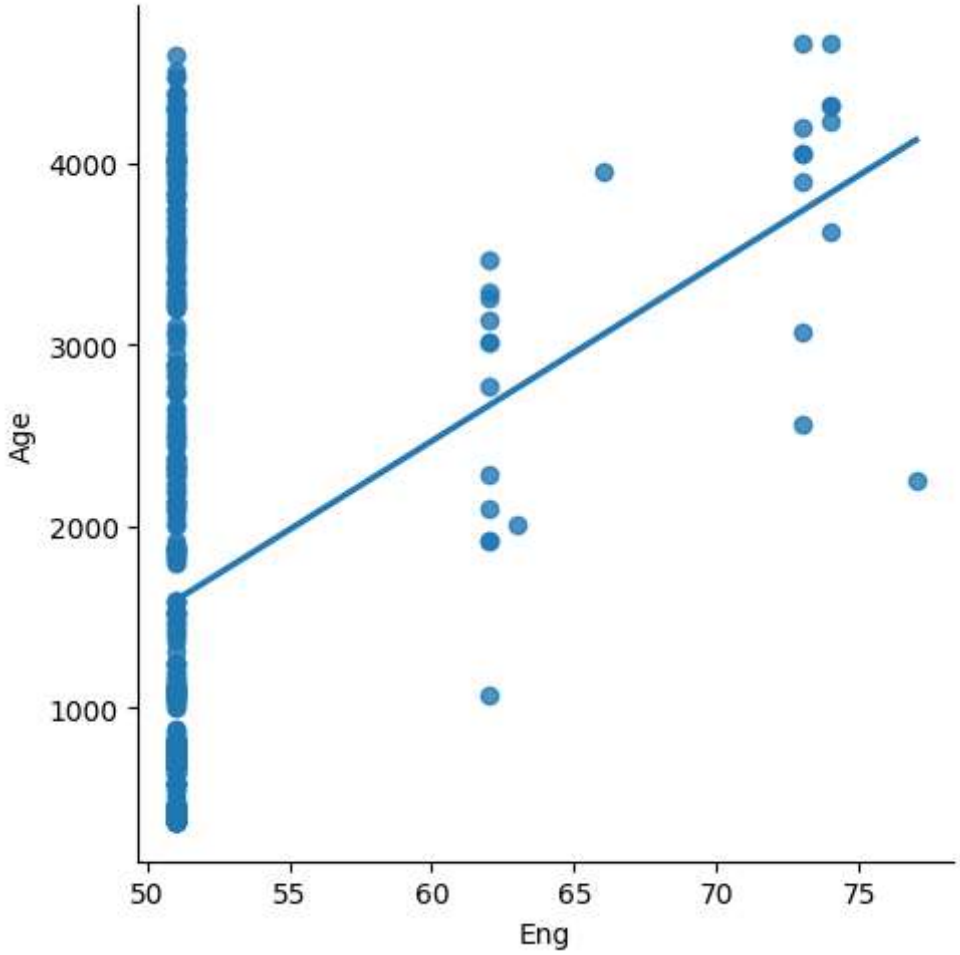


In [15]:

```
dt500=dt[:][:500]
sns.lmplot(x='Eng',y='Age',data=dt500,order=1,ci=None)
```

Out[15]:

```
<seaborn.axisgrid.FacetGrid at 0x1f5dd6fc1c0>
```



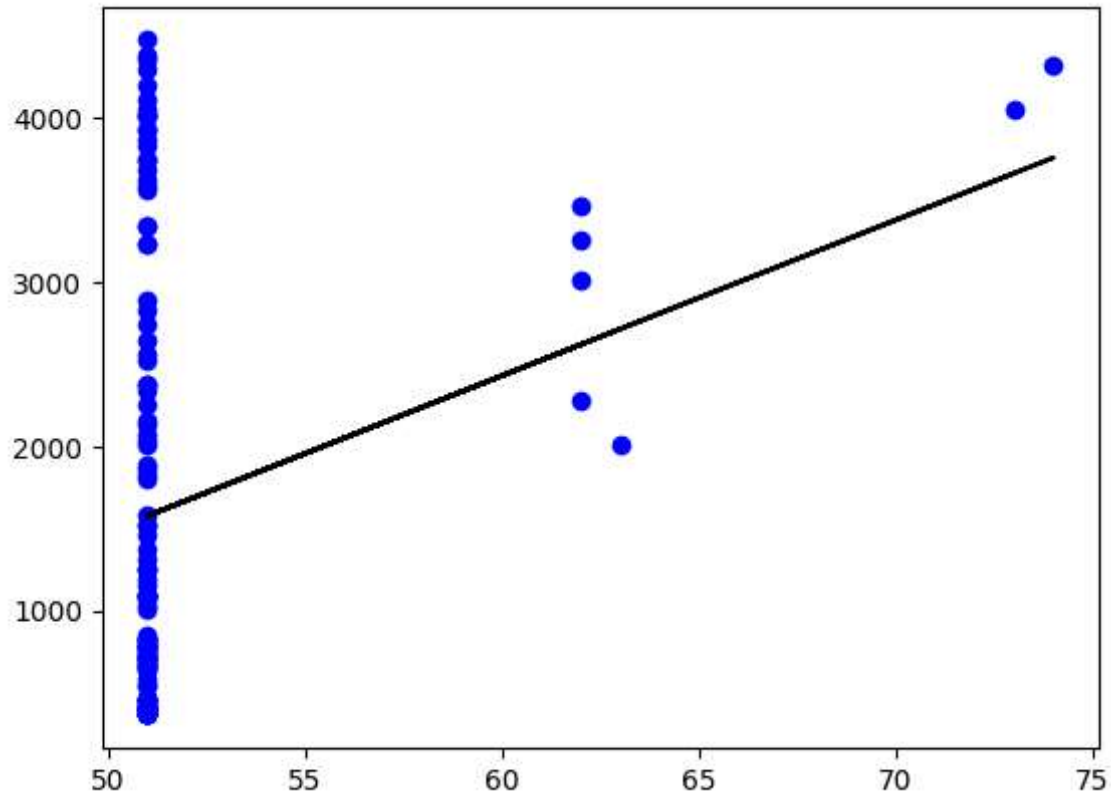
In [16]:

```
dt500.fillna(method='ffill',inplace=True)
X=np.array(dt500['Eng']).reshape(-1,1)
y=np.array(dt500['Age']).reshape(-1,1)
dt500.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: 0.08238459611477233
```

Out[16]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [17]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
mode1=LinearRegression()
mode1.fit(X_train,y_train)
y_pred=mode1.predict(X_test)
r2=r2_score(y_test,y_pred)
print("R2 score: ",r2)
```

```
R2 score:  0.08238459611477233
```

In [18]:

```
#conclusion:Linear regression is not fit for the model
```

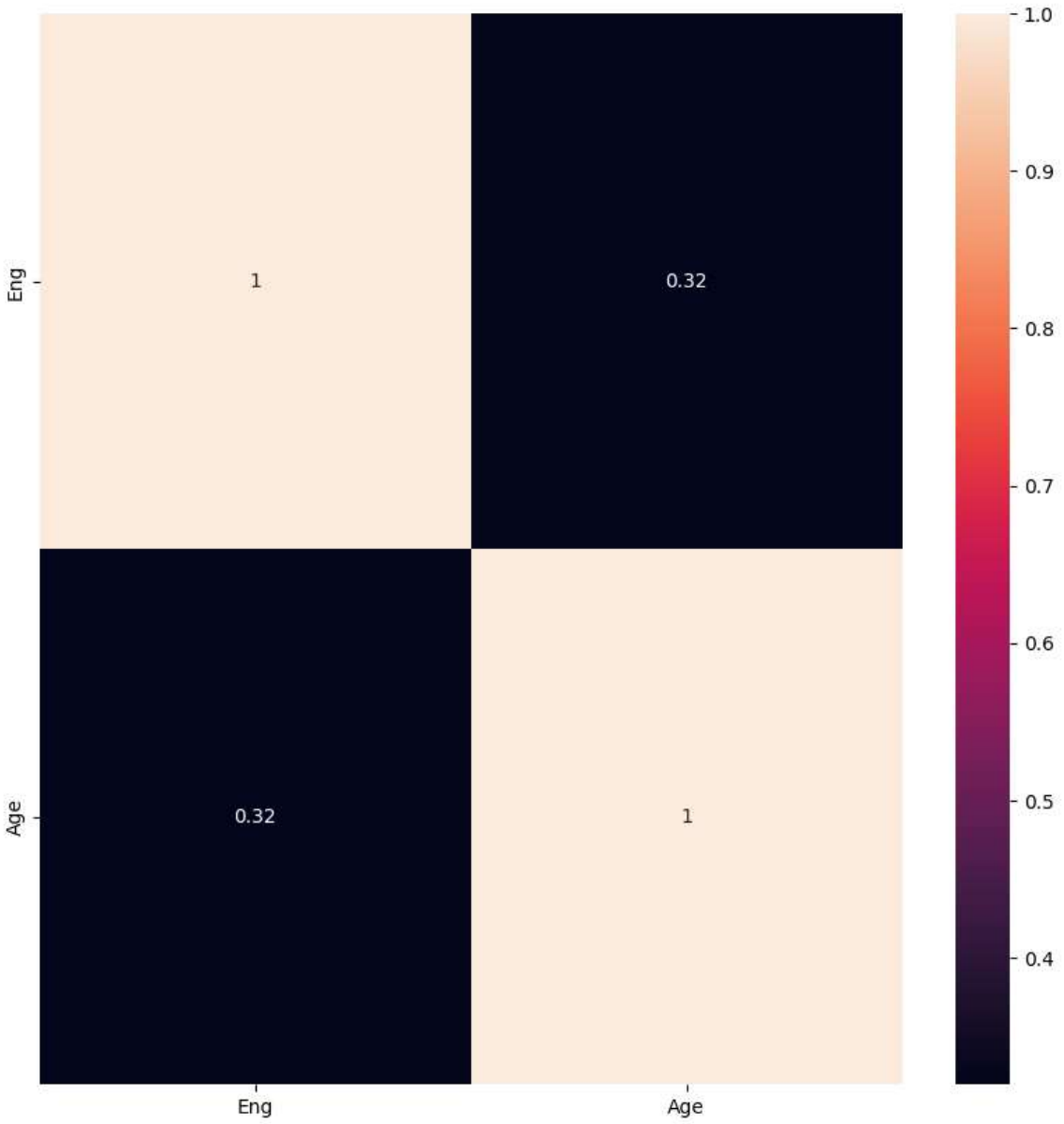
Lasso and Ridge Regression

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_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
from sklearn.preprocessing import StandardScaler
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("\nLinear Regression Model:")
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 [22]:

```
#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_trair
print("The train score for ridge model is {}".format(ridge_cv.score(X_test, y_test))
```

```
The train score for ridge model is 0.9999999999999923
The train score for ridge model is 0.9999999999999929
```

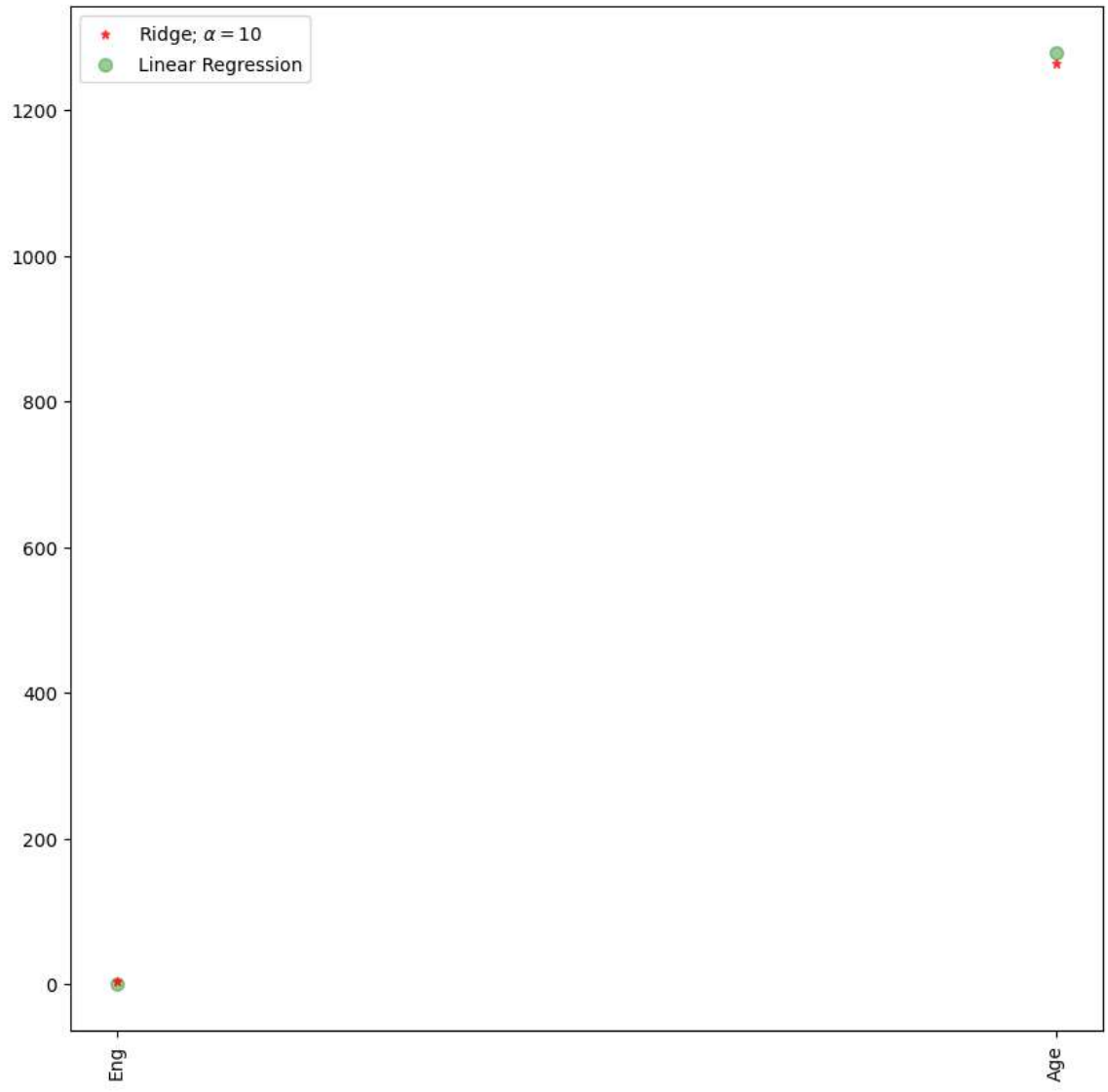
In [23]:

```
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:")
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 0.9999059589052988
The test score for ridge model is 0.9999057892122102
```

In [24]:

```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,

plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color=
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



In [25]:

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

```
0.9999999999890582
0.9999999999918023
```

In [26]:

```
#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))
```

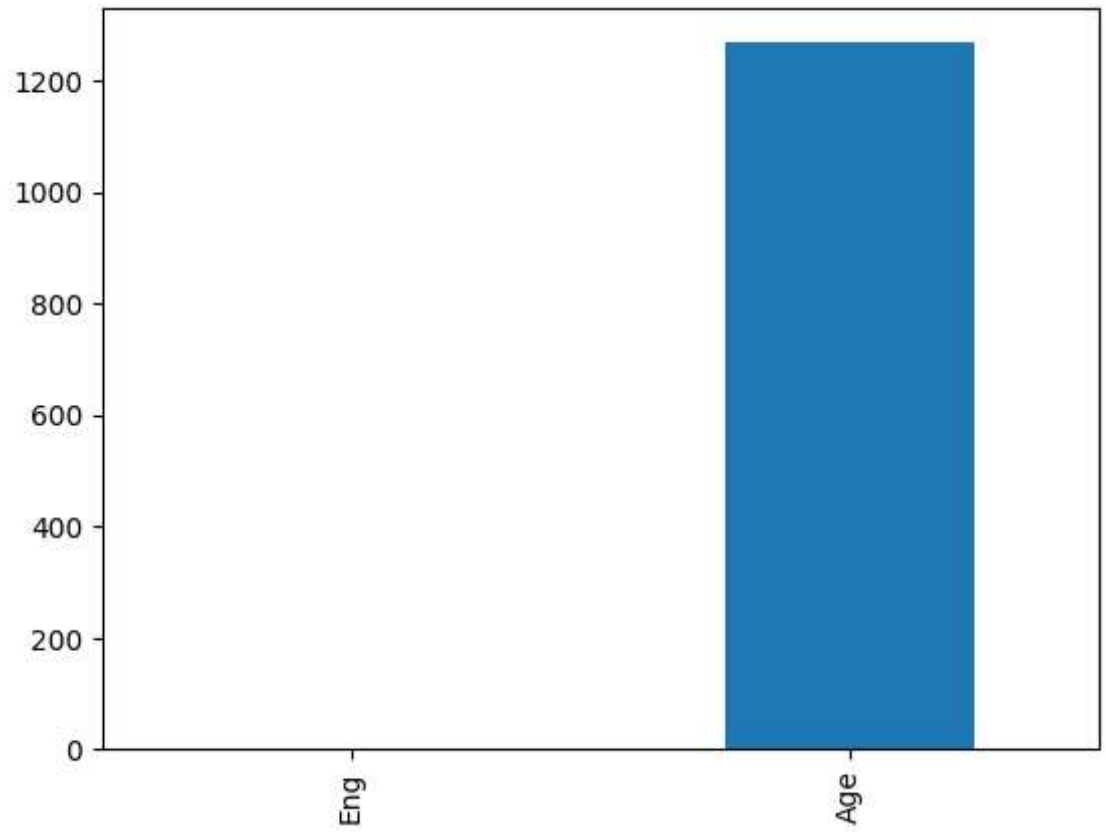
```
Lasso Model:

The train score for ls model is 0.999938742790022
The test score for ls model is 0.9999387415288635
```

In [27]:

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

Out[27]:

<Axes: >

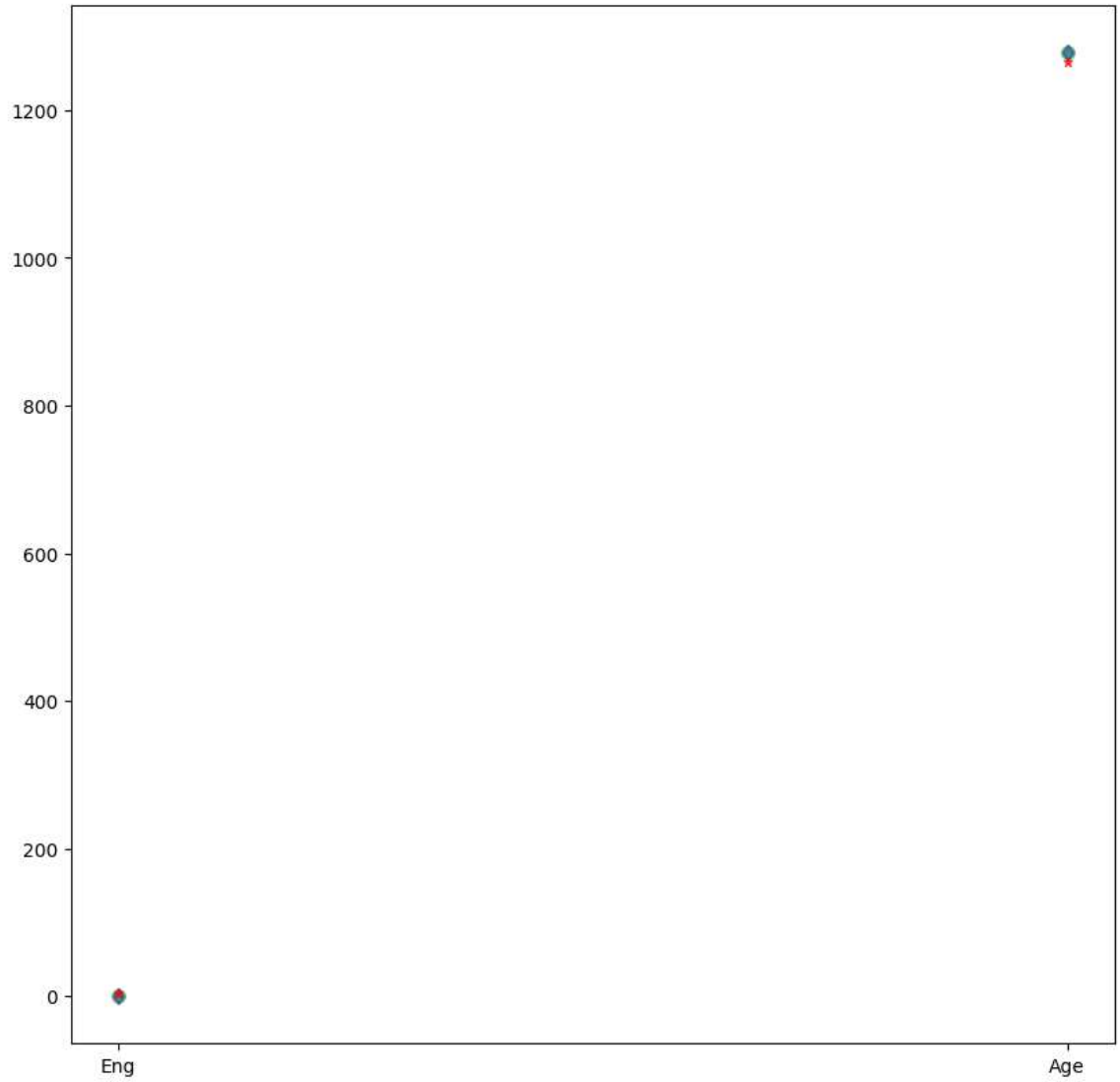


In [28]:

```
#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,
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='bl
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color=
```

Out[28]:

```
[<matplotlib.lines.Line2D at 0x1f5f5ba9c00>]
```



Elastic Regression

In [29]:

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

```
[0.        0.9999994]
0.0009934970469203108
```

In [30]:

```
y_pred_elastic=regr.predict(X_train)
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 4349723.44309672
```

In [ ]:

```

```