1. Importing Libraries and reading the dataset

In [2]:
```
#importing Numerical Libraries
import pandas as pd
import numpy as np

# importing graphical plotting libraries
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

#importing Linear Regression Machine Learning Libraries
from sklearn import preprocessing
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression,Ridge,Lasso
from sklearn.metrics import r2_score


#Reading the datset
dataset=pd.read_csv(r"C:\Users\Jan Saida\Downloads\car-mpg.csv")
dataset
```

Out[2]:

	mpg	cyl	disp	hp	wt	acc	yr	origin	car_type	car_name
0	18.0	8	307.0	130	3504	12.0	70	1	0	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	0	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	0	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	0	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	0	ford torino
...	...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	1	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	1	chevy s-10

398 rows × 10 columns

In [3]:
```
dataset.head()
```

Out[3]:

	mpg	cyl	disp	hp	wt	acc	yr	origin	car_type	car_name
0	18.0	8	307.0	130	3504	12.0	70	1	0	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	0	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	0	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	0	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	0	ford torino

In [4]:
```
#Droping The car_name column from the dataset
dataset=dataset.drop(['car_name'],axis=1)
```

In [5]:
```
dataset
```

Out[5]:

	mpg	cyl	disp	hp	wt	acc	yr	origin	car_type
0	18.0	8	307.0	130	3504	12.0	70	1	0
1	15.0	8	350.0	165	3693	11.5	70	1	0
2	18.0	8	318.0	150	3436	11.0	70	1	0
3	16.0	8	304.0	150	3433	12.0	70	1	0
4	17.0	8	302.0	140	3449	10.5	70	1	0
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	1
394	44.0	4	97.0	52	2130	24.6	82	2	1
395	32.0	4	135.0	84	2295	11.6	82	1	1
396	28.0	4	120.0	79	2625	18.6	82	1	1
397	31.0	4	119.0	82	2720	19.4	82	1	1

398 rows × 9 columns

In [6]:
```
dataset.shape
```

Out[6]:
```
(398, 9)
```

In [7]:
```
#replace the origin into 1,2,3
dataset['origin']=dataset['origin'].replace({1:'America',2:'Europe',3:'India'})
dataset=pd.get_dummies(dataset,columns=['origin'])

#Replace '?' with nan
dataset=dataset.replace('?',np.nan)
```

```
#Replace all nan with median
# Fill NaN values in non-numeric columns with the mode (most frequent value)
non_numeric_cols = dataset.select_dtypes(exclude=['number'])
dataset[non_numeric_cols.columns] = non_numeric_cols.apply(lambda x:x.fillna(x.mode()[0]), axis=0)
dataset.head()
```

Out[7]:

	mpg	cyl	disp	hp	wt	acc	yr	car_type	origin_America	origin_Europe	origin_India
0	18.0	8	307.0	130	3504	12.0	70	0	True	False	False
1	15.0	8	350.0	165	3693	11.5	70	0	True	False	False
2	18.0	8	318.0	150	3436	11.0	70	0	True	False	False
3	16.0	8	304.0	150	3433	12.0	70	0	True	False	False
4	17.0	8	302.0	140	3449	10.5	70	0	True	False	False

Model Building

In [9]:
```
x=dataset.drop(['mpg'],axis=1)
y=dataset[['mpg']]
```

Scaling the data

In [11]:
```
x_s=preprocessing.scale(x)
x_s=pd.DataFrame(x_s,columns=x.columns)

y_s=preprocessing.scale(y)
y_s=pd.DataFrame(y_s,columns=y.columns)

#split the dataset into train,test
x_train,x_test,y_train,y_test=train_test_split(x_s,y_s,test_size=0.3,random_state=1)
x_train.shape
```

Out[11]:
```
(278, 10)
```

Simple Linear Regression Model

In [13]:
```
regression_model=LinearRegression()
regression_model.fit(x_train,y_train)
for idx,col_name in enumerate(x_train.columns):
    print('The coefficient for {} is {}'.format(col_name,regression_model.coef_[0][idx]))
intercept=regression_model.intercept_[0]
print('The intercept is {}'.format(intercept))
```

```
The coefficient for cyl is 0.3333912026327803
The coefficient for disp is 0.3069652035271767
The coefficient for hp is -0.14869459811305202
The coefficient for wt is -0.758617828958082
The coefficient for acc is 0.04284564002741119
The coefficient for yr is 0.3798695467907454
The coefficient for car_type is 0.3645787393442308
The coefficient for origin_America is -9790463907814.127
The coefficient for origin_Europe is -7701873918704.149
The coefficient for origin_India is -8068993212837.809
The intercept is 0.02056483049815757
```

Regularized Ridge Regression

In [15]:
```
ridge_model=Ridge(alpha=0.3) #high coefficient to low coefficient
ridge_model.fit(x_train,y_train)
print('Ridge model coef: {}'.format(ridge_model.coef_))
```

```
Ridge model coef: [[ 0.2978896   0.29770956 -0.15709228 -0.74140723  0.03573102  0.38012171
   0.36248975 -0.069183    0.04709389  0.03899154]]
```

Regularized Lasso Regression

In [17]:
```
lasso_model=Lasso(alpha=0.1) #high coefficient to 0
lasso_model.fit(x_train,y_train)
print('Lasso model coef: {}'.format(lasso_model.coef_))
```

```
Lasso model coef: [-0.         -0.         -0.01449194 -0.52122183  0.          0.28177732
  0.12814954 -0.01585551  0.          0.        ]
```

Score Comparison

In [19]:
```
print(regression_model.score(x_train,y_train))
print(regression_model.score(x_test,y_test))
print('********************')

#RIDGE
print(ridge_model.score(x_train,y_train))
print(ridge_model.score(x_test,y_test))
print('********************')

#LASSO
print(lasso_model.score(x_train,y_train))
print(lasso_model.score(x_test,y_test))
```

```
0.8322025914416533
0.8500494643118208
********************
0.8329123140194363
0.8525473444953984
********************
0.7938023543978145
0.8372658397283955
```

Model Parameter Tuning

In [21]:
```
data_train_test=pd.concat([x_train,y_train],axis=1)
data_train_test.head()
```

Out[21]:

	cyl	disp	hp	wt	acc	yr	car_type	origin_America	origin_Europe	origin_India	mpg
350	-0.856321	-0.849116	-1.093465	-0.893172	-0.242570	1.351199	0.941412	0.773559	-0.461968	-0.497643	1.432898
59	-0.856321	-0.925936	-1.326913	-0.847061	2.879909	-1.085858	0.941412	-1.292726	2.164651	-0.497643	-0.065919
120	-0.856321	-0.695475	0.177530	-0.121101	-0.024722	-0.815074	0.941412	-1.292726	2.164651	-0.497643	-0.578335
12	1.498191	1.983643	1.163200	0.934732	-2.203196	-1.627426	-1.062235	0.773559	-0.461968	-0.497643	-1.090751
349	-0.856321	-0.983552	-0.963772	-1.165111	0.156817	1.351199	0.941412	-1.292726	-0.461968	2.009471	1.356035

In [22]:
```
import statsmodels.formula.api as smf
ols1=smf.ols(formula='mpg ~ cyl+disp+hp+wt+acc+yr+car_type+origin_America+origin_Europe+origin_India',data=data_train_test).fi
ols1.params
```

Out[22]:
```
Intercept         0.019515
cyl               0.301916
disp              0.309542
hp               -0.156604
wt               -0.752289
acc               0.037773
yr                0.381307
car_type          0.366281
origin_America   -0.069658
origin_Europe     0.047623
origin_India      0.039062
dtype: float64
```

In [23]:
```
print(ols1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.833
Model:                            OLS   Adj. R-squared:                  0.827
Method:                 Least Squares   F-statistic:                     148.5
Date:                Tue, 07 Jan 2025   Prob (F-statistic):           9.95e-99
Time:                        15:01:24   Log-Likelihood:                -148.10
No. Observations:                 278   AIC:                             316.2
Df Residuals:                     268   BIC:                             352.5
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0195      0.025      0.771      0.441      -0.030       0.069
cyl            0.3019      0.112      2.690      0.008       0.081       0.523
disp           0.3095      0.128      2.420      0.016       0.058       0.561
hp            -0.1566      0.063     -2.470      0.014      -0.281      -0.032
wt            -0.7523      0.084     -8.967      0.000      -0.917      -0.587
acc            0.0378      0.037      1.027      0.305      -0.035       0.110
yr             0.3813      0.029     13.335      0.000       0.325       0.438
car_type       0.3663      0.066      5.537      0.000       0.236       0.497
origin_America -0.0697     0.020     -3.480      0.001      -0.109      -0.030
origin_Europe  0.0476      0.021      2.226      0.027       0.006       0.090
origin_India   0.0391      0.020      1.917      0.056      -0.001       0.079
==============================================================================
Omnibus:                       21.959   Durbin-Watson:                   2.116
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               35.142
Skew:                           0.496   Prob(JB):                     2.34e-08
Kurtosis:                       4.431   Cond. No.                     4.17e+15
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 8.93e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

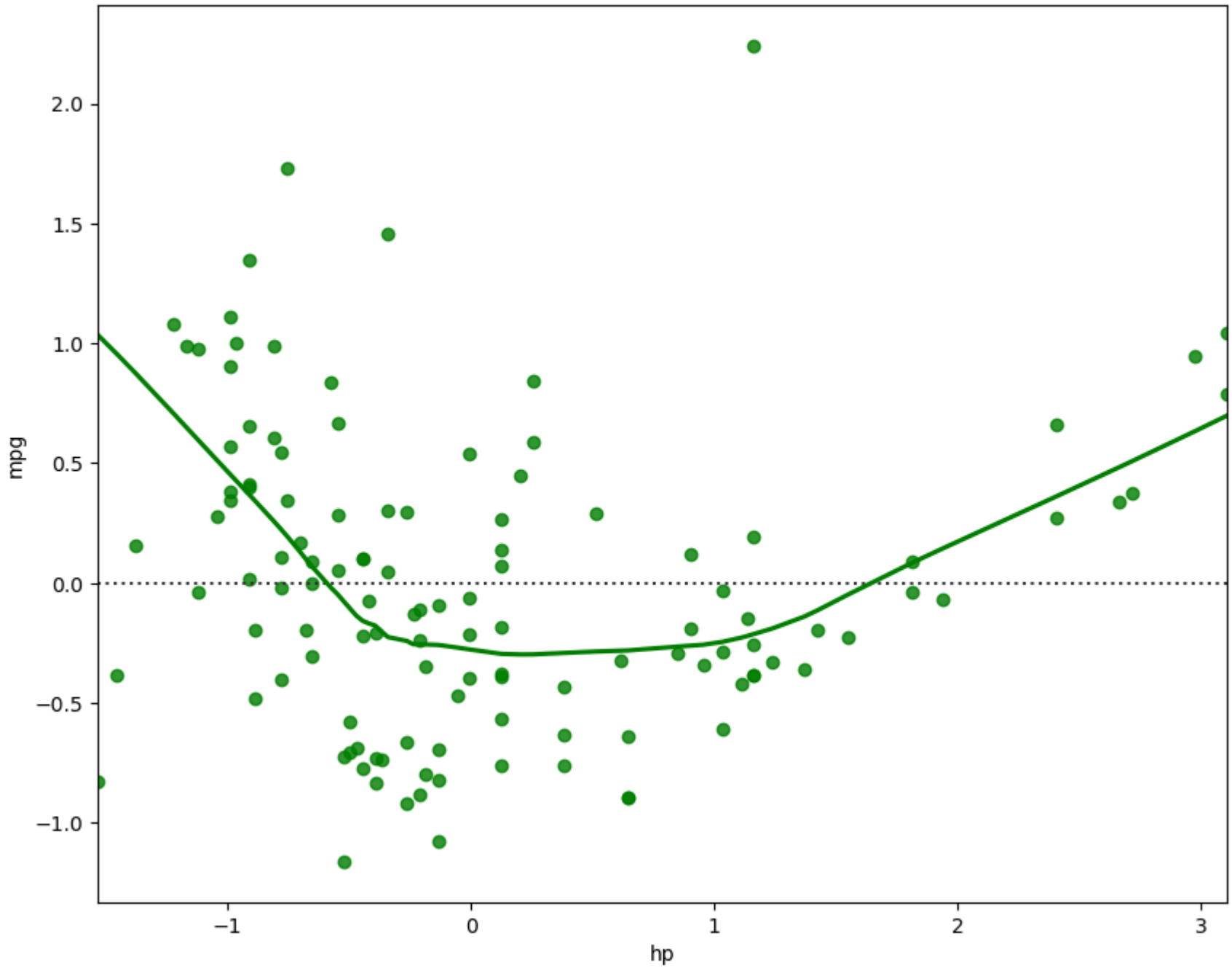
In [24]:
```
mse=np.mean((regression_model.predict(x_test)-y_test)**2)
import math
```

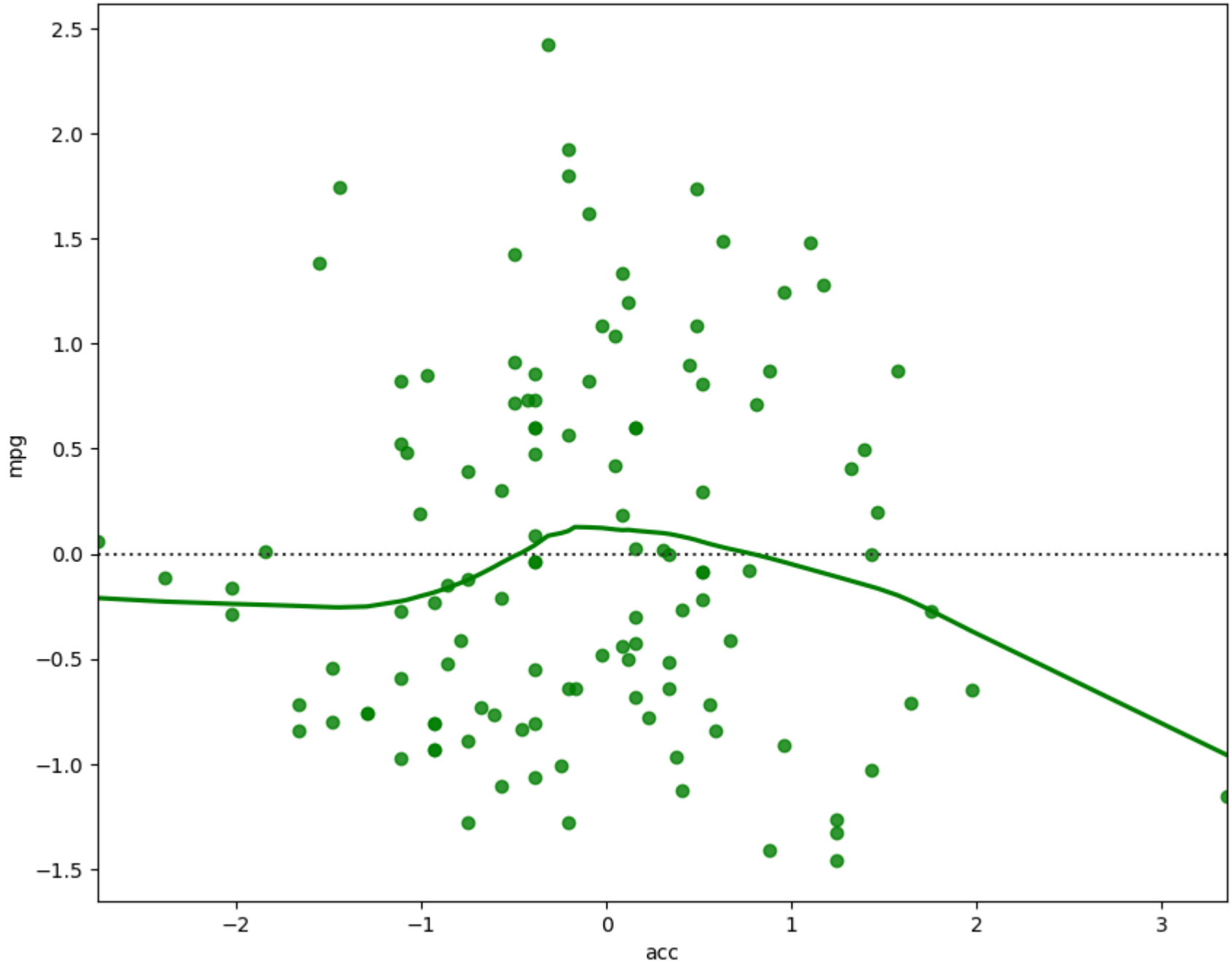
```
rmse=math.sqrt(mse)
print('Root Mean Squared Error:{}'.format(rmse))
```

```
Root Mean Squared Error:0.3793078256703554
```

In [25]:
```
import statsmodels.api as sm

# Is OLS a good model ? Lets check the residuals for some of these predictor.

fig=plt.figure(figsize=(10,8))
sns.residplot(x=x_test['hp'],y=y_test['mpg'],color='green',lowess=True)

fig=plt.figure(figsize=(10,8))
sns.residplot(x=x_test['acc'],y=y_test['mpg'],color='green',lowess=True)
```

Out[25]:
```
<Axes: xlabel='acc', ylabel='mpg'>
```


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
mpg
−1
0
1
2
3
hp


2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
mpg
−2
−1
0
1
2
3
acc

In [26]:
```
y_pred=regression_model.predict(x_test)
plt.scatter(y_test['mpg'],y_pred)
```

Out[26]: <matplotlib.collections.PathCollection at 0x235e425e8d0>