In [1]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\arati\anaconda3\lib\site-packages (1.4.2)
Requirement already satisfied: numpy>=1.19.5 in c:\users\arati\anaconda3\lib\site-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in c:\users\arati\anaconda3\lib\site-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\arati\anaconda3\lib\site-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\arati\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

1. Import packages and observe dataset

In [2]:
```
#Import numerical libraries
import numpy as np
import pandas as pd
```

In [3]:
```
# import graphical plotting libraries
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

#Import 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
```

In [41]:
```
data=pd.read_csv(r"C:\Users\arati\Downloads\car-mpg.csv")
data
```

Out[41]:

	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 [42]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   mpg       398 non-null    float64
 1   cyl       398 non-null    int64  
 2   disp      398 non-null    float64
 3   hp        398 non-null    object 
 4   wt        398 non-null    int64  
 5   acc       398 non-null    float64
 6   yr        398 non-null    int64  
 7   origin    398 non-null    int64  
 8   car_type  398 non-null    int64  
 9   car_name  398 non-null    object 
dtypes: float64(3), int64(5), object(2)
memory usage: 31.2+ KB
```

In [43]:

```
# Drop 'car_name' if it exists
if 'car_name' in data.columns:
    data = data.drop(['car_name'], axis=1)

# Replace numeric origin values with corresponding labels
if 'origin' in data.columns:
    data['origin'] = data['origin'].replace({1: 'america', 2: 'europe', 3: 'asia

# Apply one-hot encoding to the 'origin' column
if 'origin' in data.columns:
    data = pd.get_dummies(data, columns=['origin'], dtype=int)

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

```python
# Convert all columns to numeric where possible (non-numeric values will become
data = data.apply(lambda x: pd.to_numeric(x, errors='coerce'))

# Fill NaN values with the median of each column
data = data.apply(lambda x: x.fillna(x.median()), axis=0)
```

In [25]:
```python
data.head()
```

Out[25]:

	mpg	cyl	disp	hp	wt	acc	yr	car_type	origin_america	origin_asia	origin_
0	18.0	8	307.0	130.0	3504	12.0	70	0	1	0	
1	15.0	8	350.0	165.0	3693	11.5	70	0	1	0	
2	18.0	8	318.0	150.0	3436	11.0	70	0	1	0	
3	16.0	8	304.0	150.0	3433	12.0	70	0	1	0	
4	17.0	8	302.0	140.0	3449	10.5	70	0	1	0	

2.Model Building

Here we would like to scale the data as the columns are varied which would result in 1 column dominating the others.

First we divide the data into independent (X) and dependent data (y) then we scale it.

In [26]:
```python
X = data.drop(['mpg'],axis = 1) # independent variable
y = data[['mpg']] # dependent variable
```

In [27]:
```python
#scaling the data
X_s = preprocessing.scale(X)
X_s = pd.DataFrame(X_s,columns=X.columns) #converting scaled data into dataframe

y_s = preprocessing.scale(y)
y_s = pd.DataFrame(y_s,columns = y.columns) #ideally train, test data should be
```

In [28]:
```python
X_s
```

Out[28]:

	cyl	disp	hp	wt	acc	yr	car_type	origin_a
0	1.498191	1.090604	0.673118	0.630870	-1.295498	-1.627426	-1.062235	C
1	1.498191	1.503514	1.589958	0.854333	-1.477038	-1.627426	-1.062235	C
2	1.498191	1.196232	1.197027	0.550470	-1.658577	-1.627426	-1.062235	C
3	1.498191	1.061796	1.197027	0.546923	-1.295498	-1.627426	-1.062235	C
4	1.498191	1.042591	0.935072	0.565841	-1.840117	-1.627426	-1.062235	C
...	...	...	...	...	...	...	...	
393	-0.856321	-0.513026	-0.479482	-0.213324	0.011586	1.621983	0.941412	C
394	-0.856321	-0.925936	-1.370127	-0.993671	3.279296	1.621983	0.941412	-1
395	-0.856321	-0.561039	-0.531873	-0.798585	-1.440730	1.621983	0.941412	C
396	-0.856321	-0.705077	-0.662850	-0.408411	1.100822	1.621983	0.941412	C
397	-0.856321	-0.714680	-0.584264	-0.296088	1.391285	1.621983	0.941412	C

398 rows × 10 columns

In [29]:

```
y_s
```

Out[29]:

	mpg
0	-0.706439
1	-1.090751
2	-0.706439
3	-0.962647
4	-0.834543
...	...
393	0.446497
394	2.624265
395	1.087017
396	0.574601
397	0.958913

398 rows × 1 columns

In [30]:

```
data.shape
```

Out[30]: (398, 11)

In [31]:

```
#split into train, test set
X_train,X_test, y_train,y_test = train_test_split(X_s, y_s, test_size = 0.20, ra
```

```
X_train.shape
```

Out[31]:
```
(318, 10)
```

2.a Simple Linear Model

In [32]:
```
#Fit simple linear model and find coefficients
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_

intercept = regression_model.intercept_[0]
print('The intercept is {}'.format(intercept))
```

```
The coefficient for cyl is 0.24638776053571634
The coefficient for disp is 0.2917709209866447
The coefficient for hp is -0.18081621820393684
The coefficient for wt is -0.667553060986813
The coefficient for acc is 0.06537309205777046
The coefficient for yr is 0.3481770259426718
The coefficient for car_type is 0.3339231253960359
The coefficient for origin_america is -0.08117984631927032
The coefficient for origin_asia is 0.0698609820966492
The coefficient for origin_europe is 0.030003161242288048
The intercept is -0.01800683137092324
```

2.b Regularized Ridge Regression

In [36]:
```
#alpha factor here is lambda (penalty term) which helps to reduce the magnitude

ridge_model = Ridge(alpha = 0.3)
ridge_model.fit(X_train, y_train)

print('Ridge model coef: {}'.format(ridge_model.coef_))
#As the data has 10 columns hence 10 coefficients appear here
```

```
Ridge model coef: [[ 0.24342352  0.28293268 -0.18074242 -0.65967997  0.06398366
0.34745486
   0.33096428 -0.08087356  0.06988696  0.0295866 ]]
```

2.c Regularized Lasso Regression

In [37]:
```
#alpha factor here is lambda (penalty term) which helps to reduce the magnitude

lasso_model = Lasso(alpha = 0.1)
lasso_model.fit(X_train, y_train)

print('Lasso model coef: {}'.format(lasso_model.coef_))
#As the data has 10 columns hence 10 coefficients appear here
```

```
Lasso model coef: [-0.         -0.         -0.07247557 -0.45867691  0.
0.2698134
  0.11341188 -0.04988145  0.          0.        ]
```

3. Score Comparison

In [38]:

```python
#Model score - r^2 or coeff of determinant
#r^2 = 1-(RSS/TSS) = Regression error/TSS


#Simple Linear Model
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))
```

```
0.8373422857977738
0.8474768646673948
*************************
0.837332956087454
0.847263786646594
*************************
0.8007202116330951
0.8283046020148332
```

In [ ]:

In [ ]: