In [1]:
```
#Import numerical libraries
import pandas as pd
import numpy as np

#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 [2]:
```
data = pd.read_csv(r'C:\Users\user\Downloads\car-mpg.csv')
data.head()
```

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

In [5]:
```
data = data.drop(['car_name'], axis = 1)
data['origin'] = data['origin'].replace({1: 'america', 2: 'europe', 3: 'asia'})
data = pd.get_dummies(data,columns = ['origin'],dtype = int)
data = data.replace('?', np.nan)
```

In [7]:
```
# Step 3: Convert all columns to numeric (non-numeric -> NaN)
data = data.apply(pd.to_numeric, errors='coerce')

# Step 4: Fill missing values with median for numeric columns
numeric_cols = data.select_dtypes(include=[np.number]).columns
data[numeric_cols] = data[numeric_cols].apply(lambda x: x.fillna(x.median()))
```

In [9]:
```
data.head()
```

Out[9]:

	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	

In [11]:
```
X = data.drop(['mpg'], axis = 1) # independent variable
y = data[['mpg']] #dependent variable
```

In [13]:
```
#Scaling the data

X_s = preprocessing.scale(X)
X_s = pd.DataFrame(X_s, columns = X.columns) #converting scaled data into datafr

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

In [15]:
```
X_s
```

Out[15]:

	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	0
1	1.498191	1.503514	1.589958	0.854333	-1.477038	-1.627426	-1.062235	0
2	1.498191	1.196232	1.197027	0.550470	-1.658577	-1.627426	-1.062235	0
3	1.498191	1.061796	1.197027	0.546923	-1.295498	-1.627426	-1.062235	0
4	1.498191	1.042591	0.935072	0.565841	-1.840117	-1.627426	-1.062235	0
...	...	...	...	...	...	...	...	
393	-0.856321	-0.513026	-0.479482	-0.213324	0.011586	1.621983	0.941412	0
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	0
396	-0.856321	-0.705077	-0.662850	-0.408411	1.100822	1.621983	0.941412	0
397	-0.856321	-0.714680	-0.584264	-0.296088	1.391285	1.621983	0.941412	0

398 rows × 10 columns

In [17]:
```
y_s
```

Out[17]:

	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 [19]:
```
#Split into train, test set

X_train, X_test, y_train,y_test = train_test_split(X_s, y_s, test_size = 0.30, r
X_train.shape
```

Out[19]: (278, 10)

2.a Simple Linear Model

In [22]:
```
#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 0.321022385691611
The coefficient for disp is 0.32483430918483897
The coefficient for hp is -0.22916950059437569
The coefficient for wt is -0.7112101905072298
The coefficient for acc is 0.014713682764191237
The coefficient for yr is 0.3755811949510748
The coefficient for car_type is 0.3814769484233099
The coefficient for origin_america is -0.07472247547584178
The coefficient for origin_asia is 0.044515252035677896
The coefficient for origin_europe is 0.04834854953945386
The intercept is 0.019284116103639767
```

2.b Regularized Ridge Regression

In [25]:
```
#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: [[ 0.31649043  0.31320707 -0.22876025 -0.70109447  0.01295851
0.37447352
   0.37725608 -0.07423624  0.04441039  0.04784031]]
```

2.c Regularized Lasso Regression

In [28]:
```
#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.         -0.01690287 -0.51890013  0.
0.28138241
  0.1278489  -0.01642647  0.          0.        ]
```

3. Score Comparison

In [31]:
```
#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))
```

```
0.8343770256960538
0.8513421387780066
**************************
0.8343617931312616
0.8518882171608508
**************************
0.7938010766228453
0.8375229615977083
```

In [ ]:

In [ ]: