```
# Multiple Linear Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('50_Startups.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values

# Encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder = LabelEncoder()
X[:, 3] = labelencoder.fit_transform(X[:, 3])
onehotencoder = OneHotEncoder(categorical_features = [3])
X = onehotencoder.fit_transform(X).toarray()

# Avoiding the Dummy Variable Trap
X = X[:, 1:] # Library will take care of it

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

# Feature Scaling
"""from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)"""

from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)


## Fitting Multiple Linear Regression to the Training set
#from sklearn.linear_model import LinearRegression
#regressor = LinearRegression()
#regressor.fit(X_train, y_train)
#
## Predicting the Test set results
#y_pred = regressor.predict(X_test)

# Bulding optimal model using backward elimination model
import statsmodels.formula.api as sm
X = np.append(arr = np.ones((50, 1)).astype(int), values = X, axis = 1)

X_opt = X[:, [0, 1, 2, 3, 4, 5]]
regressor_ols = sm.OLS(endog = y, exog = X_opt).fit()
regressor_ols.summary()

X_opt = X[:, [0, 1, 3, 4, 5]]
regressor_ols = sm.OLS(endog = y, exog = X_opt).fit()
regressor_ols.summary()

X_opt = X[:, [0, 3, 4, 5]]
regressor_ols = sm.OLS(endog = y, exog = X_opt).fit()
regressor_ols.summary()
```