```python
#  multiple_linear_regression

# importing liberary

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

# encode categorical data
# incoding the independent varible
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
labelencoder_X = LabelEncoder()
X[:,3] = labelencoder_X.fit_transform(X[:,3])
onehotencoder = OneHotEncoder(categorical_features = [3])    # creating dummy var for state() wh
X = onehotencoder.fit_transform(X).toarray()


# avoiding dummy var trap
X = X[:,1:]



# splitting the dataset into training set and testing set
from sklearn.model_selection 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)"""


# fitting  multiple_linear_regression to trainnig dataset
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train)


# predict testing  dataset
y_pred =  regressor.predict(X_test)


# building the optimal model using the backward elimination
#{goal : to create best optimal model then we get better prediction.
#        there are some  var. higly statically significant  to model,which have great impact on
#        and some are not statically siginificant to model,if we remove non statically significa

# ols  = (ordinary least square)
   #(selecting the best column(feature) whcih impact on the dependent varible(profit)
   #then we can get   better optimal result(pred ) and remove col which are non(less) significan

import statsmodels.api as sm
# (bydefault it's not take constant(thetas 0 ,we have to put theta_0 * X0 = 1
```

```
# that's why we are creating col. of 1's and trying to put in the starting of X)
X = np.append(arr  = np.ones((50,1)).astype(int),values = X, axis = 1)# we are adding 1 extra co

# applying backword propagation

X_opt =X[:, [0,1,2,3,4,5]]   # select all col. from X, for optimal matrix of of feature  that is
regressor_OLS = sm.OLS(y,X_opt).fit() # fitting OLS
regressor_OLS.summary()

#  we have to remove col. if  not statically significant to dependen var.
# (for that we need Predictor>sginificant level)

X_opt =X[:, [0,1,3,4,5]]   # removing X2(dummmy var) cause p>SL(significant level =0.5)thats why
regressor_OLS = sm.OLS(y,X_opt).fit()  # fit          #    p>SL (0.990 > 0.05)
regressor_OLS.summary()


X_opt =X[:, [0,3,4,5]]   # remove X1(dummy var) cause P>SL(0.990 > 0.05)
regressor_OLS = sm.OLS(y,X_opt).fit()
regressor_OLS.summary()


X_opt =X[:, [0,3,5]]   #     removing X4(administration) cause P>SL (0.608 > 0.05)
regressor_OLS = sm.OLS(y,X_opt).fit()
regressor_OLS.summary()

X_opt =X[:, [0,3]]   # removing X5 (marketing spend) case P>SL(0.060 > 0.05)
regressor_OLS = sm.OLS(y,X_opt).fit()
regressor_OLS.summary()
```