```
# Multiple Linear Regression

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

import os

print(os.getcwd())

os.chdir('E:\\Locker\\Sai\\SaiHCourseNait\\DecBtch\\R_Datasets\\')

# Importing the dataset
dataset = pd.read_csv('VC_Startups.csv')
dataset

dataset
X = dataset.iloc[:, :-1].values
X

y = dataset.iloc[:, 4].values
y

import matplotlib.pyplot as plt
x = dataset.iloc[:, 0].values
plt.scatter(x,y,label='',color='k',s=100)
plt.xlabel('R&D')
plt.ylabel('Profit')
plt.title('Profit vs R&D Spend')
plt.legend()
plt.show()


x = dataset.iloc[:, 1].values
plt.scatter(x,y,label='',color='k')
plt.xlabel('Admin')
plt.ylabel('Profit')
plt.title('Profit vs Admin Spend')
plt.legend()
plt.show()

x = dataset.iloc[:, 2].values
```

```
plt.scatter(x,y,label='',color='k')
plt.xlabel('Marketing')
plt.ylabel('Profit')
plt.title('Profit vs Marketing Spend')
plt.legend()
plt.show()

'''x = dataset.iloc[:, 3].values
plt.scatter(x,y,label='',color='k')
plt.xlabel('State')
plt.ylabel('Profit')
plt.title('Profit vs State')
plt.legend()
plt.show()'''

df = dataset.iloc[:, 3:5]
df.boxplot(column='Profit',by='State')




# Dummy Vars & Encoders
from numpy import array
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

dataset2 = ['Pizza','Burger','Bread','Bread','Bread','Burger','Pizza','Burger']

values = array(dataset2)
print(values)


label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values)
print(integer_encoded)

onehot = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded),1)
onehot_encoded = onehot.fit_transform(integer_encoded)
print(onehot_encoded)

#inverted_result = label_encoder.inverse_transform([argmax(onehot_encoded[0,:])])
#print(inverted_result)
```

```
# Encoding categorical data
#from sklearn.preprocessing import LabelEncoder, OneHotEncoder

labelencoder = LabelEncoder()
X

X[:, 3] = labelencoder.fit_transform(X[:, 3])
X

onehotencoder = OneHotEncoder(categorical_features = [3])
X = onehotencoder.fit_transform(X).toarray()
X

# Avoiding the Dummy Variable Trap
X = X[:, 1:]
X

# 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)"""

# 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)
y_pred

#***
```

```
X # (50,5)

arr = np.ones((50,1))
arr

# y= mx + c ie c*1 v r ignoring c
X = np.append(arr=np.ones((50,1)).astype(int), values=X,axis=1)
X # (50,6)

# for statsmodels needs the additional col as bias
# it is internally used for computing the pvals
# and stats significance for Adj R2

import statsmodels.formula.api as sm

X
X_opt = X[:,[0,1,2,3,4,5]]
X_opt

regressor_OLS = sm.OLS(endog=y, exog=X_opt).fit()
regressor_OLS.summary()
# pval is the prob
# the lower the pval the more the singif the ind vars r
# x1 & x2 r dummy vars for state
# x3 is R&D Spend, x4 is Admin Spend, x5 is Mktng Spend
# x2 is the highest pval
# rm it
'''
                    OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.951
Model:                            OLS   Adj. R-squared:                  0.945
'''

X_opt = X[:,[0,1,3,4,5]]
regressor_OLS = sm.OLS(endog=y, exog=X_opt).fit()
regressor_OLS.summary()
"""
                    OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.951
```

```
Model:                          OLS   Adj. R-squared:                 0.946
"""



X_opt = X[:,[0,3,4,5]]
regressor_OLS = sm.OLS(endog=y, exog=X_opt).fit()
regressor_OLS.summary()
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                    y   R-squared:                       0.951
Model:                          OLS   Adj. R-squared:                 0.948
"""


X_opt = X[:,[0,3,5]]
regressor_OLS = sm.OLS(endog=y, exog=X_opt).fit()
regressor_OLS.summary()
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                    y   R-squared:                       0.950
Model:                          OLS   Adj. R-squared:                 0.948
"""




X_opt = X[:,[0,3]]
regressor_OLS = sm.OLS(endog=y, exog=X_opt).fit()
regressor_OLS.summary()
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                    y   R-squared:                       0.947
Model:                          OLS   Adj. R-squared:                 0.945
"""

'''
```

```
X_opt = X[:,[0,1,2,3,4,5]]
#R-squared:                       0.951
#Adj. R-squared:                  0.945

X_opt = X[:,[0,1,3,4,5]]
R-squared:                       0.951
Adj. R-squared:                  0.946

X_opt = X[:,[0,3,4,5]]
 R-squared:                       0.951
Adj. R-squared:                  0.948

X_opt = X[:,[0,3,5]]
R-squared:                       0.950
Adj. R-squared:                  0.948

X_opt = X[:,[0,3]]
R-squared:                       0.947
Adj. R-squared:                  0.945
'''



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

Interpretation
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
const       4.698e+04   2689.933     17.464      0.000      4.16e+04  5.24e+04
x1             0.7966      0.041     19.266      0.000         0.713     0.880
x2             0.0299      0.016      1.927      0.060        -0.001     0.061

'''
```