**Data Science with Python**

Dept. of Computer Science & Information Systems

Bradley university, Peoria, IL, USA

Submitted to,

Dr. **Christos Nikolopoulos**, Ph.D.

By,

Sai Prasad Ashila

January Interim - 2019

**ACKNOWLEDGEMNET**

It Brings me Great Pleasure for an opportunity to work & submit my Course Report on Introduction to data science with python.

For this I deeply Indebt & sincerely Thankful to **Christos Nikolopoulos** for his help, valuable guidance and elating encouragement throughout the course.

Finally, I am deeply thankful to all the sources my seniors and peers who helped and inspired me in completing this course.

Table of Contents

[1. INTRODUCTION 5](#_Toc535630466)

[1.1 Data Science 5](#_Toc535630467)

[1.2 Python 5](#_Toc535630468)

[1.3 Building a Predictive Model in Python 5](#_Toc535630469)

[2. Installation setup 6](#_Toc535630470)

[2.1 Anaconda Installation 6](#_Toc535630471)

[3. Simple Linear Regression 6](#_Toc535630472)

[3.1 Analysis of the Problem 6](#_Toc535630473)

[3.2 Graphical View of data 7](#_Toc535630474)

[3.3 Importing the data set 8](#_Toc535630475)

[3.4 Splitting the data into training and test Sets 9](#_Toc535630476)

[3.4 Fitting Simple Linear Regression to the Training set 9](#_Toc535630477)

[3.6 Predicting the Test set Results 10](#_Toc535630478)

[3.7 Visualizing the Training set results 11](#_Toc535630479)

[4.Multiple linear Regression 11](#_1._INTRODUCTION)

[4.1 Analysis of the Problem 11](#_Toc535630481)

[12](#_Toc535630482)

[4.2 Dummy Variables 13](#_Toc535630483)

[4.3 Building A Model 14](#_Toc535630484)

[4.10 Building the optimal model using Backward Elimination 19](#_Toc535630485)

# 1. INTRODUCTION

## 1.1 Data Science

Data science is an [interdisciplinary](https://en.wikipedia.org/wiki/Interdisciplinary) field that uses scientific methods, processes, algorithms and systems to extract [knowledge](https://en.wikipedia.org/wiki/Knowledge) and insights from [data](https://en.wikipedia.org/wiki/Data) in various forms, both structured and unstructured, similar to [data mining](https://en.wikipedia.org/wiki/Data_mining). Data science is a "concept to unify statistics, data analysis, machine learning and their related methods" in order to "understand and analyze actual phenomena" with data. It employs techniques and theories drawn from many fields within the context of [mathematics](https://en.wikipedia.org/wiki/Mathematics), [statistics](https://en.wikipedia.org/wiki/Statistics), [information science](https://en.wikipedia.org/wiki/Information_science), and [computer science](https://en.wikipedia.org/wiki/Computer_science).

## 1.2 Python

Python is an increasingly popular tool for data analysis. In recent years, a number of libraries have reached maturity, allowing R and Stata users to take advantage of the beauty, flexibility, and performance of Python without sacrificing the functionality these older programs have accumulated over the years. Python is a very powerful programming language used for many different applications. Over time, the huge community around this open source language has created quite a few tools to efficiently work with Python. In recent years, a number of tools have been built specifically for data science. As a result, analyzing data with Python has never been easier. Python is a general-purpose programming language that is becoming more and more popular for doing data science. Companies worldwide are using Python to harvest insights from their data and get a competitive edge. Unlike any other Python tutorial, this course focuses on Python specifically for data science.

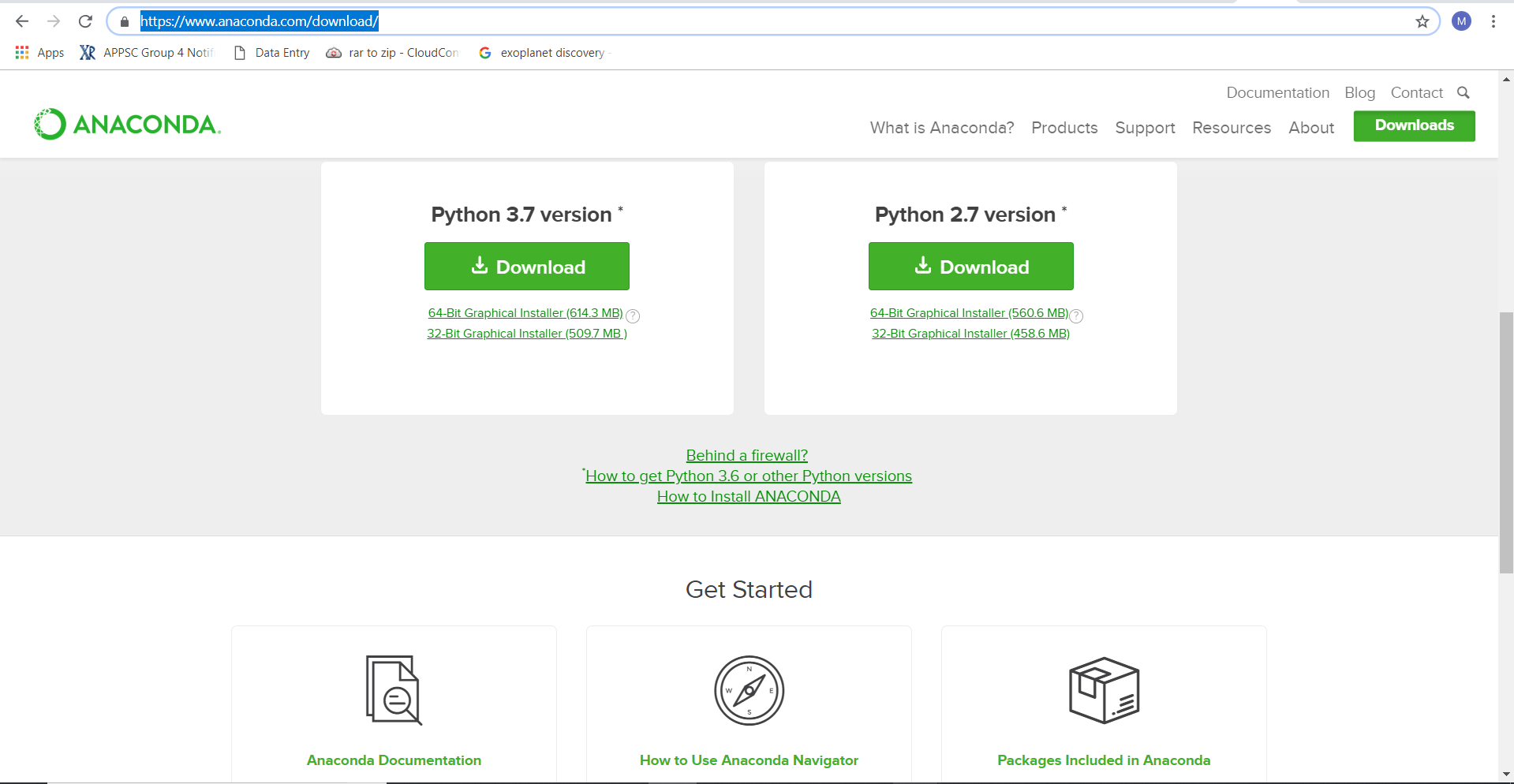
## 1.3 Building a Predictive Model in Python

Few Predicting algorithms are made using python which are the concepts of Machine learning and predictive analytics

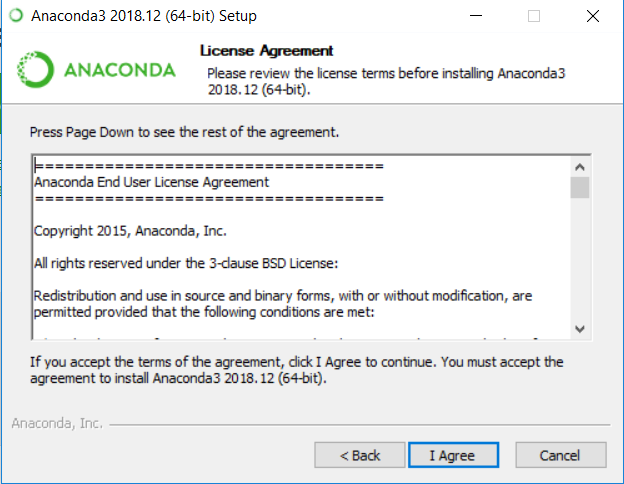
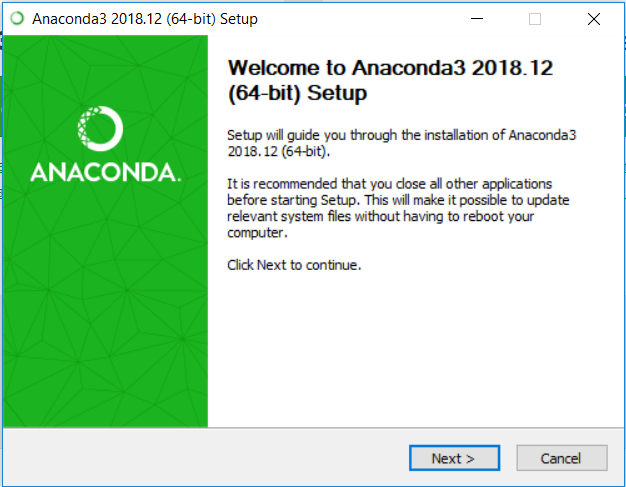
* Simple Linear Regression.
* Multiple linear Regression.
* Polynomial Regression.
* Support Vector Regression.
* Decision Tree Regression.
* Random Forest Regression.

# 2. Installation setup

## 2.1 Anaconda Installation

****

<https://www.anaconda.com/download/>

****

# 3. Simple Linear Regression

## 3.1 Analysis of the Problem

**Problem Statement:** To estimate the salary of the employee for a give particular years of experience based on the trend followed.

**Given:** Data of level of experience and corresponding salaries as shown below.

**Expectation of the regression:**  To estimate the salary for any given experience level.

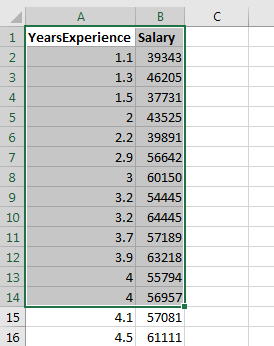
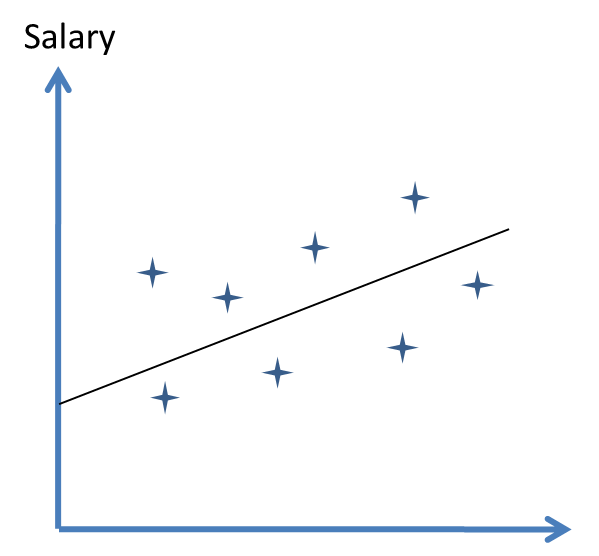
****

Figure 3.1

## 3.2 Graphical View of data



**Figure 3.1.2**

Salary = m(years of experience) + C

m is the regression coefficient

C the intercept indicates the salary of an entry level employee

## 3.3 Importing the data set

Importing the data Set using Pandas Library

1. **import** numpy as np
2. **import** pandas as pd
3. #Importing the data
4. dataset = pd.read\_csv('Salary\_Data.csv')
5. X = dataset.iloc[:,:-1].values
6. Y = dataset.iloc[:,1].values

Above code is used to import the libraries and import the data to the data sets variable and categorizing the dependent and independent variables

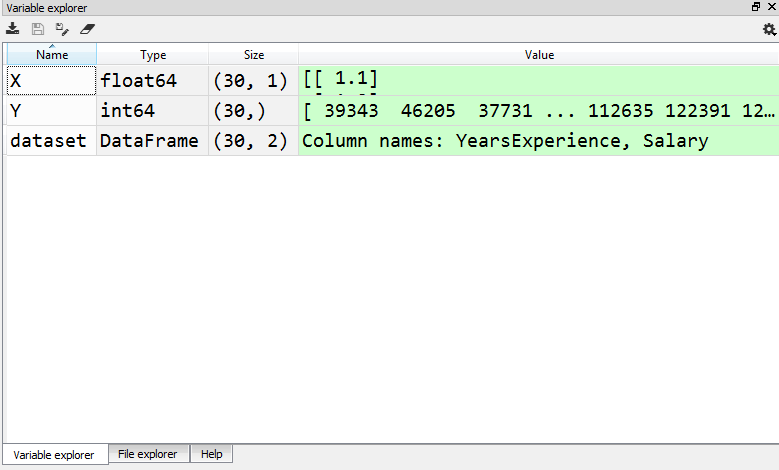
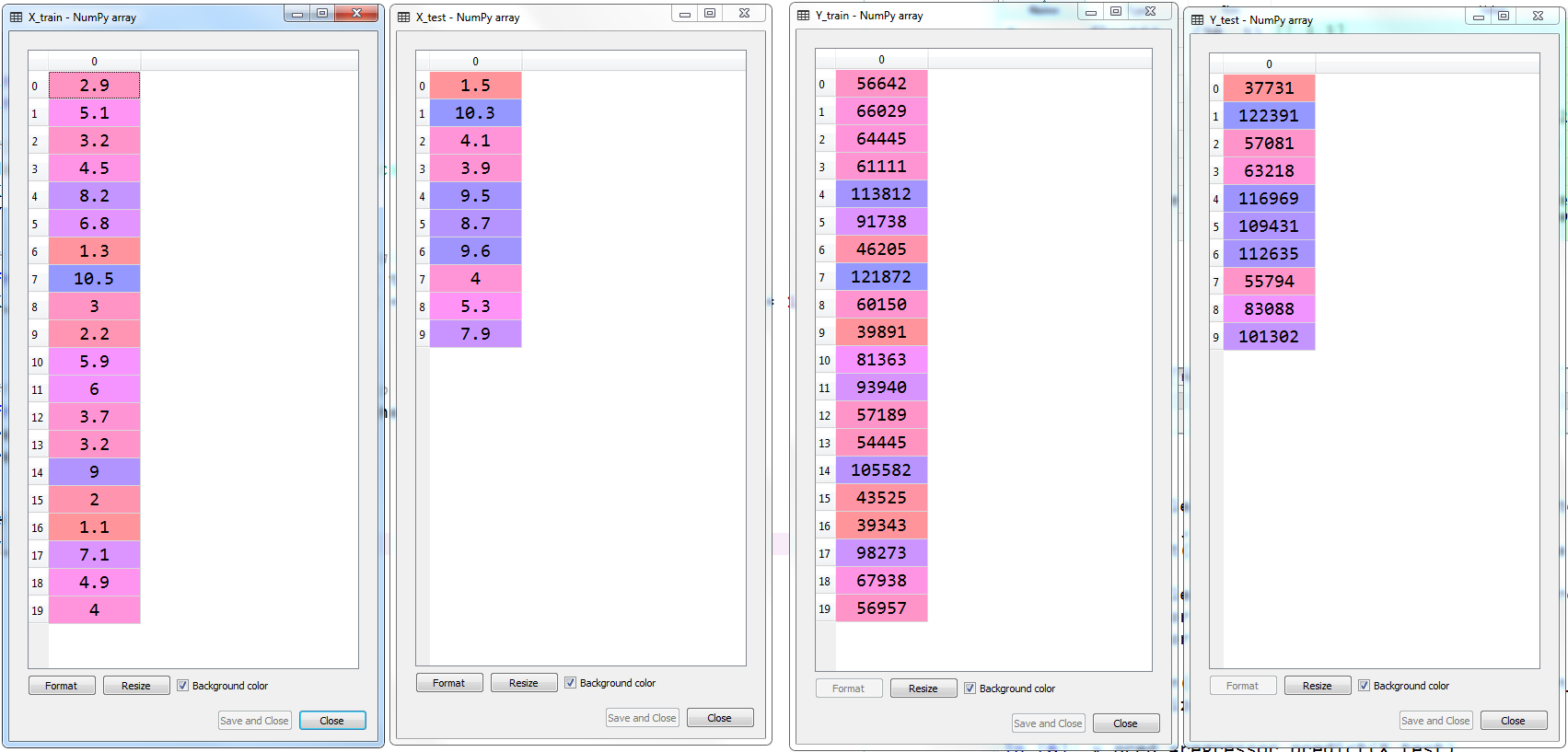


Fig: 3.1.3

## 3.4 Splitting the data into training and test Sets

1. # Splitting the dataset into the training
2. **from** sklearn.model\_selection **import** train\_test\_split
3. X\_train , X\_test , Y\_train , Y\_test =
4. train\_test\_split(X,Y, test\_size = 1/3, random\_state = 0)

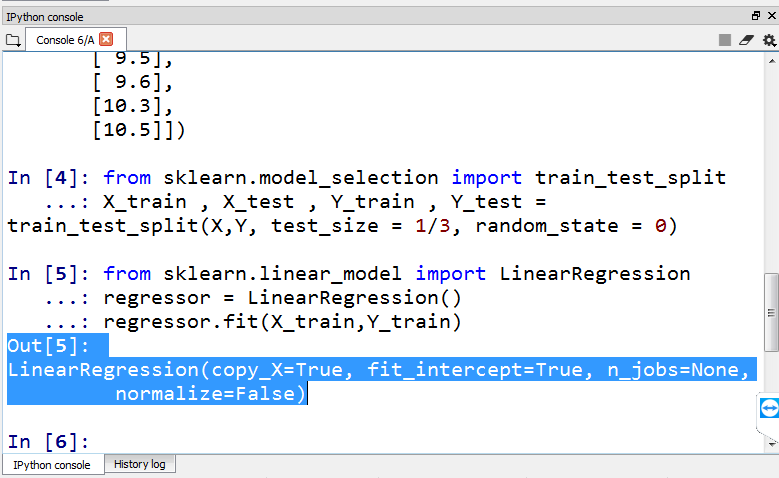
Importing the training set and splitting the given data into training set and test sets in some ratio here (1/3) to test set and 2/3 to training set



## 3.4 Fitting Simple Linear Regression to the Training set

1. #Fitting Simple Linear Regression to the
2. **from** sklearn.linear\_model **import** LinearRegression
3. regressor = LinearRegression()
4. regressor.fit(X\_train,Y\_train)

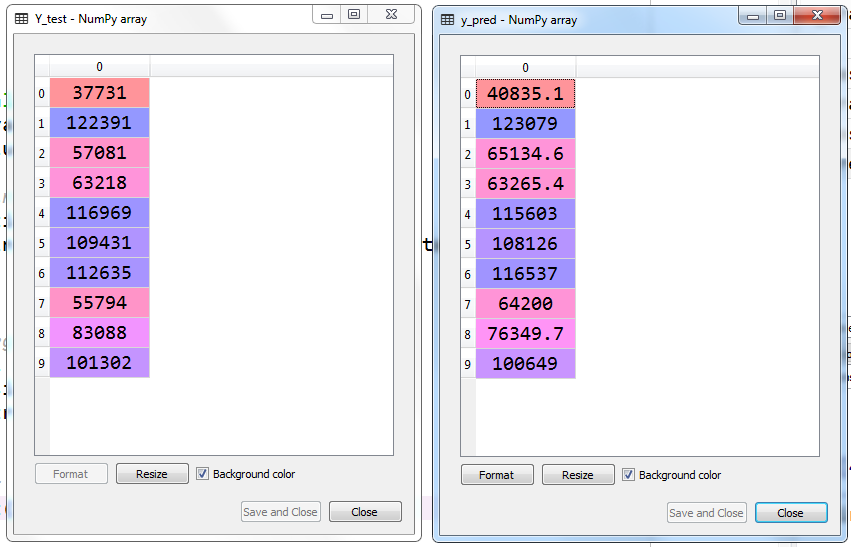
Linear Regress or is set to the training set



## 3.6 Predicting the Test set Results

1. # Predecting the Test set Results
2. y\_pred =regressor.predict(X\_test)

Predict method on the trained regreessor is called and Test set is set as the augment to get the predicted result set and the difference between the actual and predicted values are as below.

****

## 3.7 Visualizing the Training set results

1. #visualising the Training set results
2. plt.scatter(X\_train, Y\_train, color='red')
3. plt.plot(X\_train, regressor.predict(X\_train), color = 'blue')
4. plt.title('Salary vs Experience (Training Set)')
5. plt.xlabel('years of experience')
6. plt.ylabel('Salary')
7. plt.show()

The data processed is shown in the form of a graph using the plot form matplotlib.pyplot as plt libraries. The points shown with the red dots and the line that best fits the points with least possible error.

****

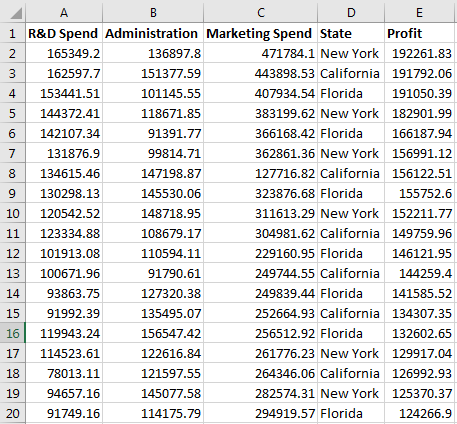
# 4.Multiple linear Regression

## 4.1 Analysis of the Problem

**Problem Statement:** For a venture capitalist to decide which type of investments from a given set of investments would lead to the highest profits.

**Given:** data Set of 50 startup companies established at different locations with their corresponding expenditures and profits.

**Expectation of the regression:** To develop a model which would help the investor to decide which investments would always lead him to the best profits and which would also help him to determine what amount of profit he would gain with the type of investment he decides on.



y = a+bx1+cx2+dx3

Where x1, x2, x3 are the independent variable and y is dependent variable.

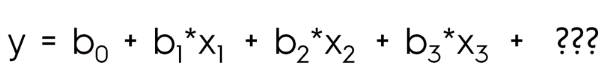
***Assumptions Of a Linear Regression***:

1. Linearity
2. Homoscedasticity
3. Multivariate normality
4. Independence of errors
5. Lack of Multicollinearity

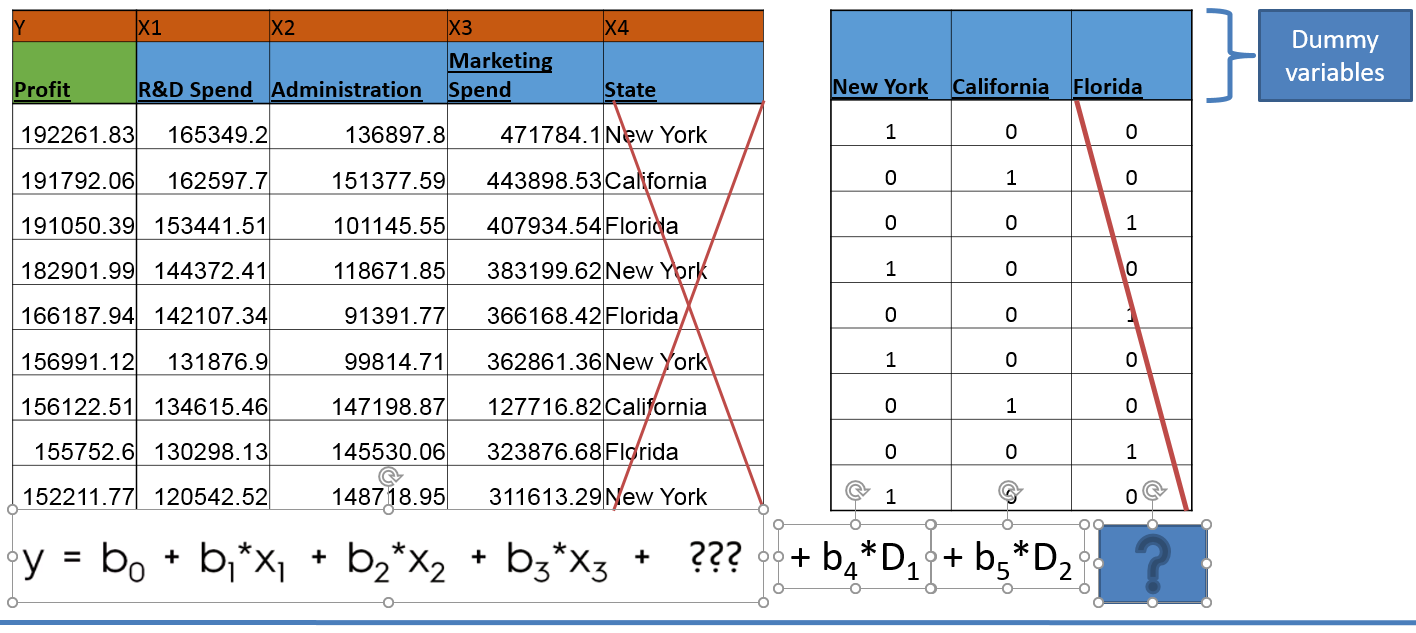
## 4.2 Dummy Variables

In [statistics](https://en.wikipedia.org/wiki/Statistics) and [econometrics](https://en.wikipedia.org/wiki/Econometrics), particularly in [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis), a dummy variable (also known as an indicator variable, design variable, [one-hot encoding](https://en.wikipedia.org/wiki/One-hot_encoding), Boolean indicator, binary variable, or qualitative variable) is one that takes the value 0 or 1 to indicate the absence or presence of some categorical effect that may be expected to shift the outcome. Dummy variables are used as devices to sort data into [mutually exclusive](https://en.wikipedia.org/wiki/Mutually_exclusive_events) categories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Y | X1 | X2 | X3 | X4 |
| **Profit** | **R&D Spend** | **Administration** | **Marketing Spend** | **State** |
| 192261.83 | 165349.2 | 136897.8 | 471784.1 | New York |
| 191792.06 | 162597.7 | 151377.59 | 443898.53 | California |
| 191050.39 | 153441.51 | 101145.55 | 407934.54 | Florida |
| 182901.99 | 144372.41 | 118671.85 | 383199.62 | New York |
| 166187.94 | 142107.34 | 91391.77 | 366168.42 | Florida |
| 156991.12 | 131876.9 | 99814.71 | 362861.36 | New York |
| 156122.51 | 134615.46 | 147198.87 | 127716.82 | California |
| 155752.6 | 130298.13 | 145530.06 | 323876.68 | Florida |
| 152211.77 | 120542.52 | 148718.95 | 311613.29 | New York |

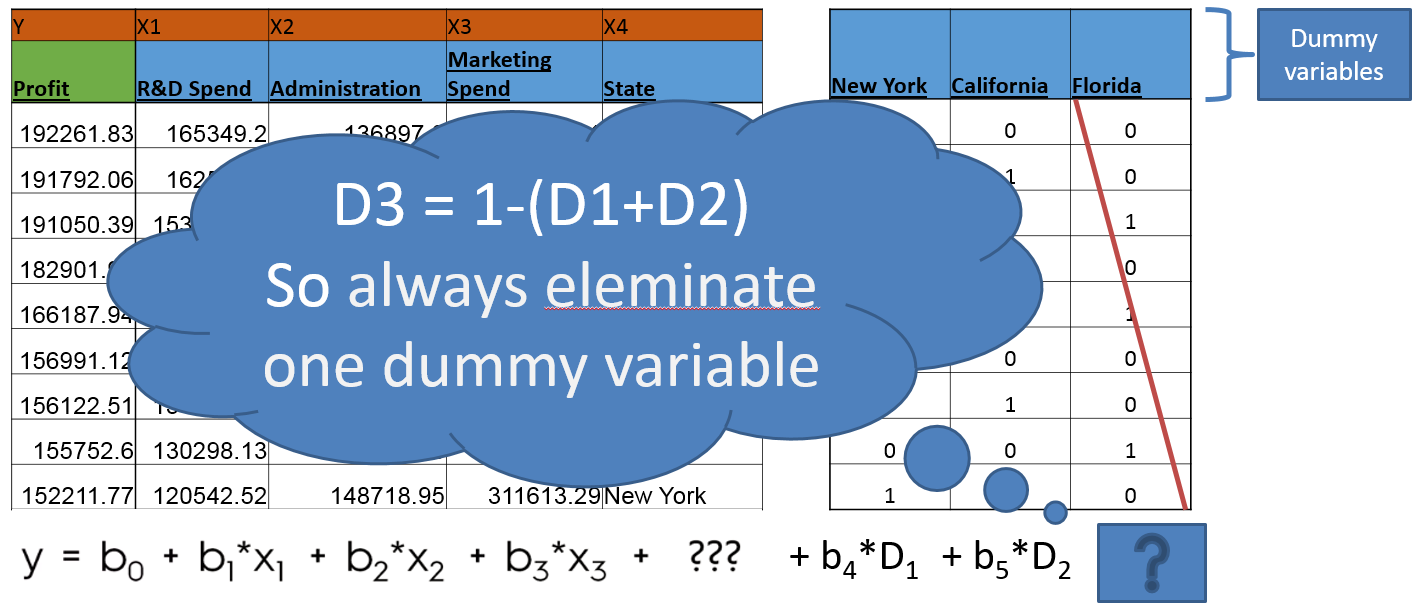


* State: Categorical variable
* Dummy variables to replace the categorical variables.



Here the State is the categorical variable and this categorical variables are given the dummy values , but at the same time care should be taken to see that the analysis doesn’t fall into dummy variable trap.

To avoid this dummy variable trap, there should always be the **n-1** dummy variables for categories of the categorical variables



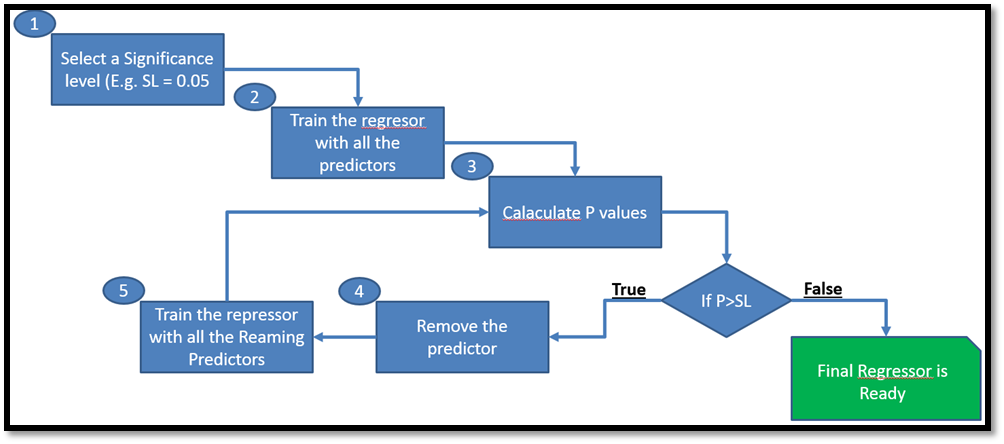
This is because the we divide the weight of the variables into n categories and the last category is added in the other variable parameters.

## 4.3 Building A Model

Methods to Build the models

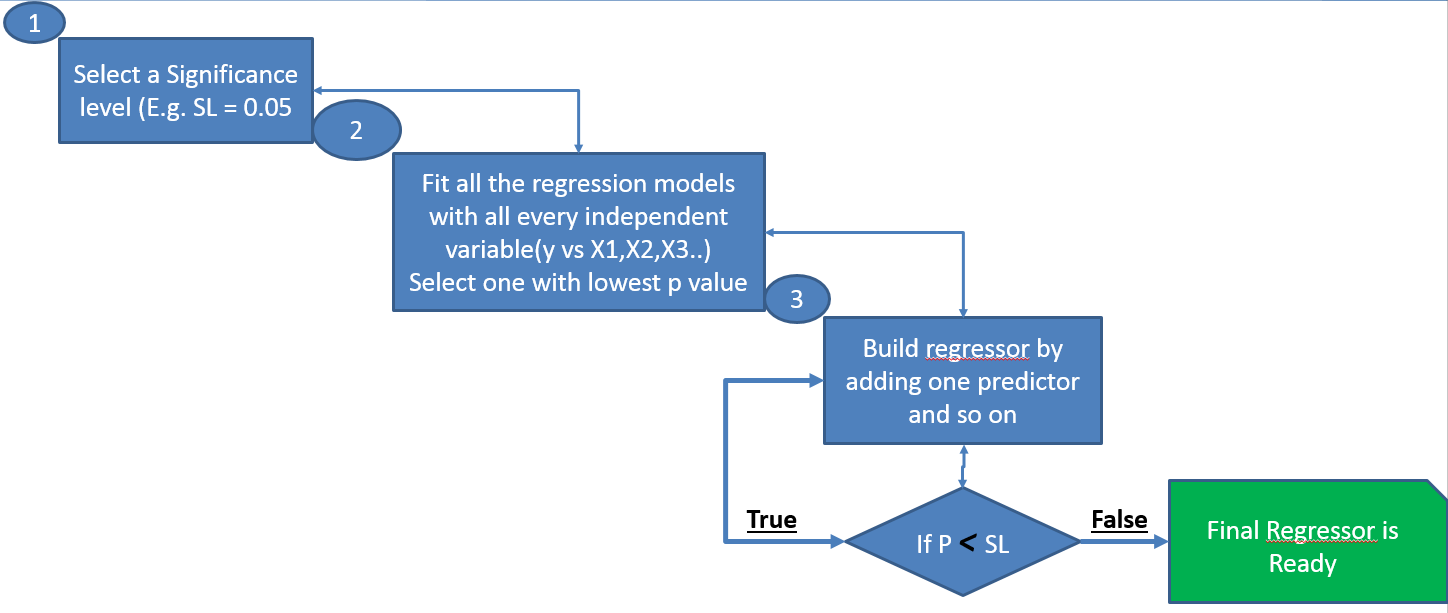
* All –in
* Step wise Regression Models
* Backward Elimination
* Forward Selection
* Bidirectional Elimination
* Score Comparison

4.4 Backward Elimination



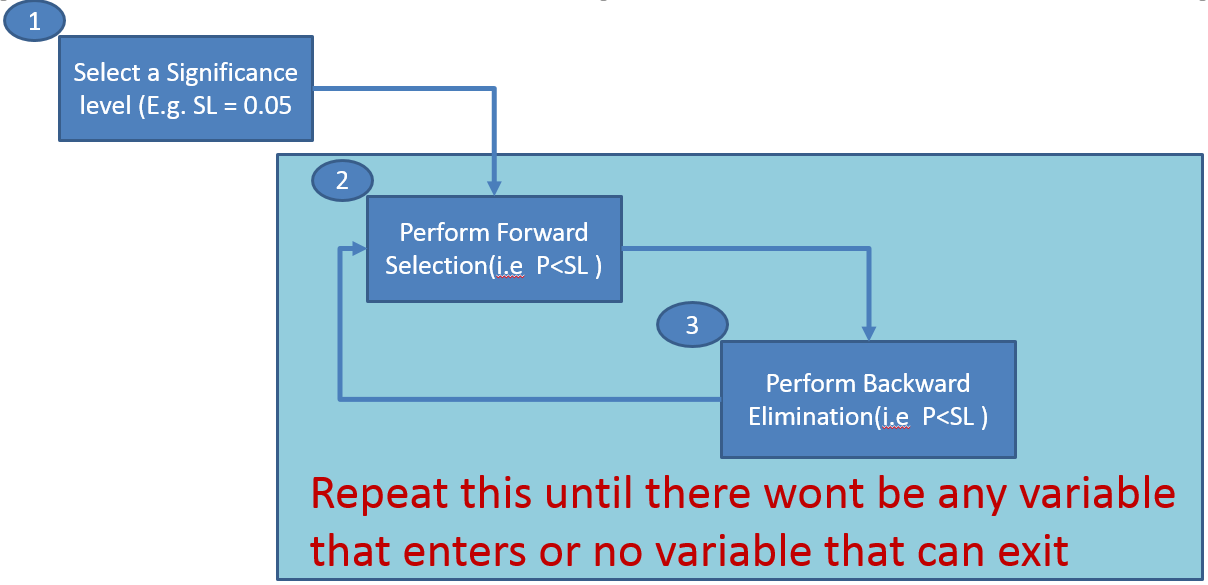
Select a significance level and then training the repressor followed by calculation of p values for all corresponding sets. After calculating the P values if P>SL remove the predictor or the independent variables and then train the regressor with reaming independent variables and calculate the P values and again check for the P validations with SL and if P< SL we will reach the final Regressor.

4.5 Forward Selection



Forward selection doesn’t involve any elimination of the variables, but it starts with adding the variables or predictors to the regressor. The Process starts with fitting the regression model with all the independent variables, selecting one with the lowest P value and then building the regressor by adding one predictor each and then checking the equality of the P with SL and following the corresponding steps that are in the model shown above.

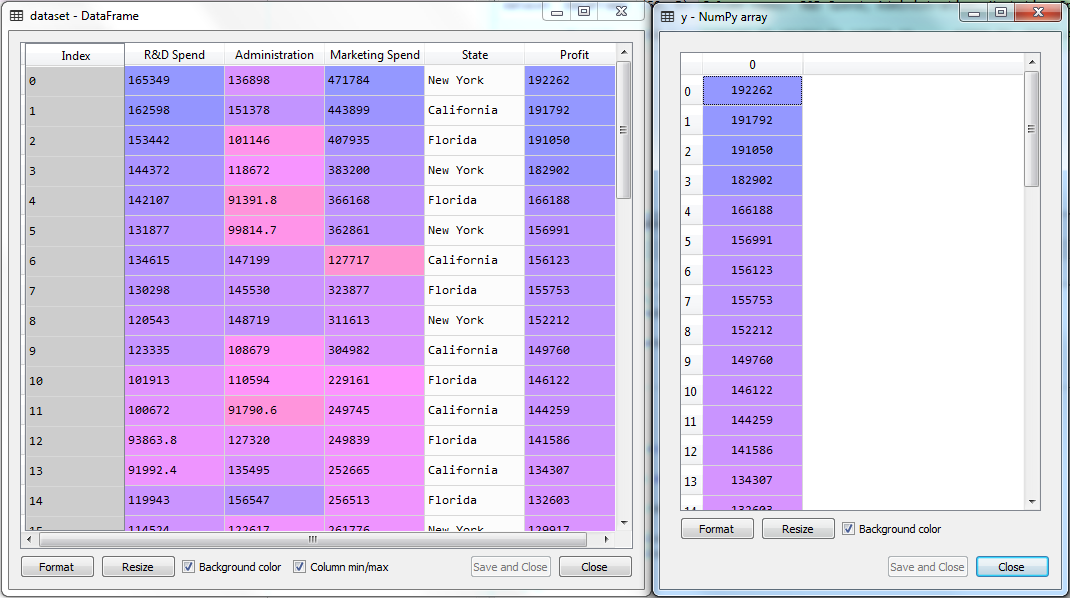
4.6 Bi-Directional Elimination



This involves both the forward Selection and backward elimination simultaneously and repeating it iteratively until no variable enters and leaves the regressor.

Reaching back the problem that has to be set to the regressor: here are the steps followed

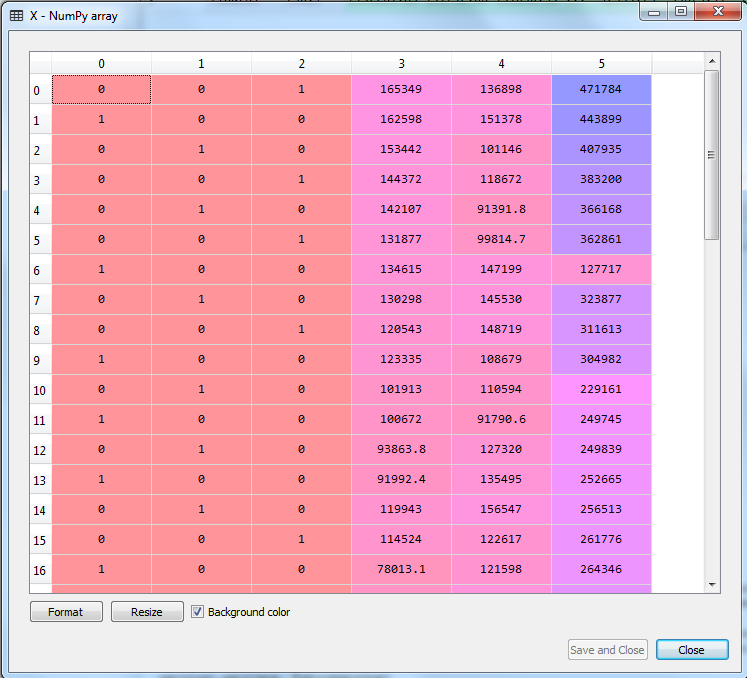
1. # Importing the libraries
2. **import** numpy as np
4. **import** pandas as pd
6. # Importing the dataset
7. dataset = pd.read\_csv('50\_Startups.csv')
8. X = dataset.iloc[:, :-1].values
9. y = dataset.iloc[:, 4].values



4.7 Encoding categorical data

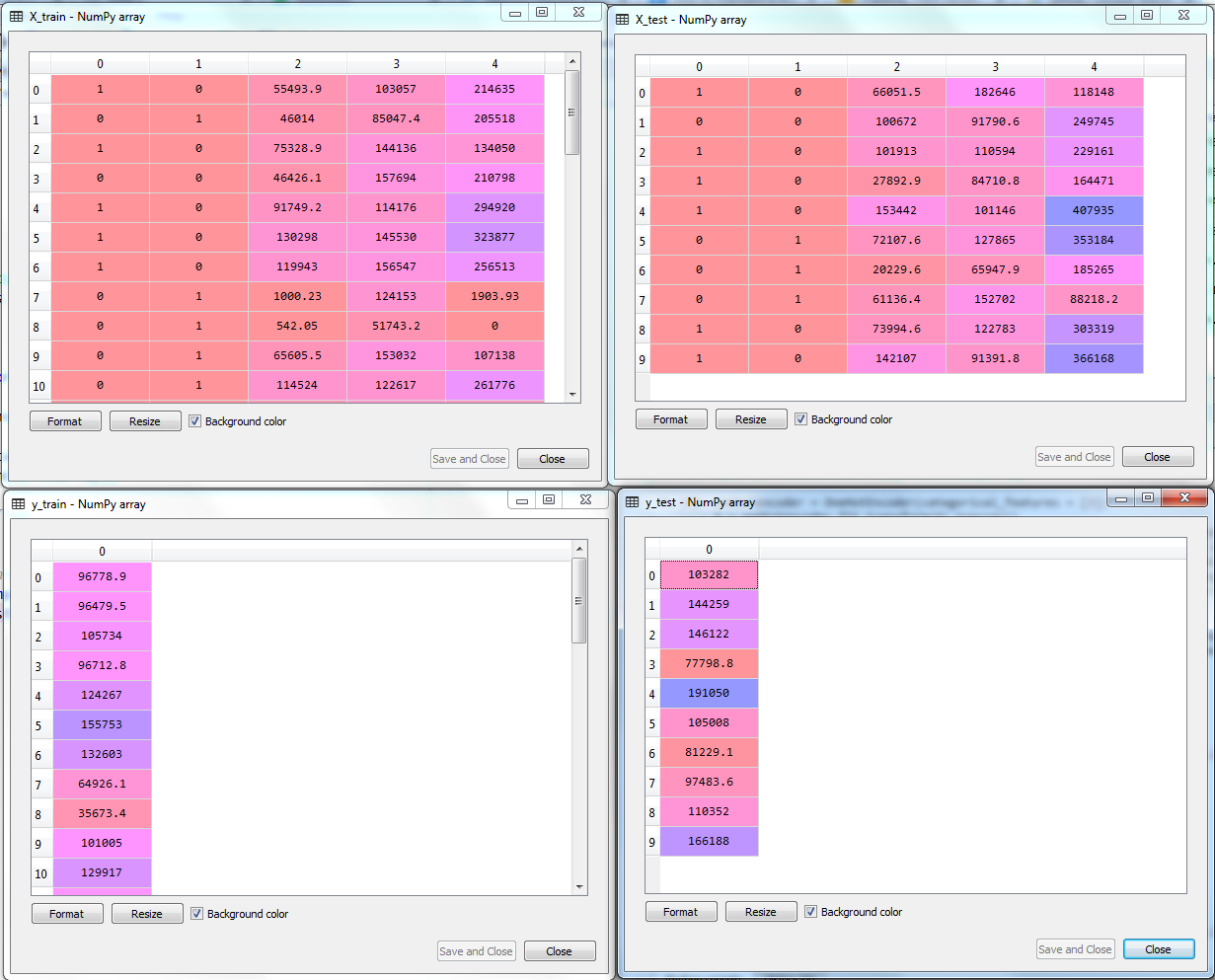
1. # Encoding categorical data
2. **from** sklearn.preprocessing **import** LabelEncoder, OneHotEncoder
3. labelencoder = LabelEncoder()
4. X[:, 3] = labelencoder.fit\_transform(X[:, 3])
5. onehotencoder = OneHotEncoder(categorical\_features = [3])
6. X = onehotencoder.fit\_transform(X).toarray()

The categorical variable is encoded using the one hot encoder.



4.8 Splitting the dataset into the Training set and Test set

1. # Splitting the dataset into the Training set and Test set
2. **from** sklearn.model\_selection **import** train\_test\_split
3. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)



4.9 Fitting Multiple Linear Regression to the Training set

1. # Fitting Multiple Linear Regression to the Training set
2. **from** sklearn.linear\_model **import** LinearRegression
3. regressor = LinearRegression()
4. regressor.fit(X\_train, y\_train)

## 4.10 Building the optimal model using Backward Elimination

1. # Building the optimal model using Backward Elimination
2. **import** statsmodels.formula.api as sm
3. X = np.append(arr = np.ones((50, 1)).astype(int), values = X, axis = 1)
4. X\_opt = X[:, [0, 1, 2, 3, 4, 5]]
5. regressor\_OLS = sm.OLS(endog = y, exog = X\_opt).fit()
6. regressor\_OLS.summary()