**Data Science with Python**

Dept. of Computer Science & Information Systems

Bradley university, Peoria, IL, USA

Submitted to,

**Chris Nikolopoulos**,

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](#_Toc535673890)

[1.1 Data Science 5](#_Toc535673891)

[1.2 Python 5](#_Toc535673892)

[1.3 Building a Predictive Model in Python 5](#_Toc535673893)

[2. Installation setup 6](#_Toc535673894)

[2.1 Anaconda Installation 6](#_Toc535673895)

[3. Simple Linear Regression 6](#_Toc535673896)

[3.1 Analysis of the Problem 6](#_Toc535673897)

[3.2 Graphical View of data 7](#_Toc535673898)

[3.3 Importing the data set 8](#_Toc535673899)

[3.4 Splitting the data into training and test Sets 9](#_Toc535673900)

[3.4 Fitting Simple Linear Regression to the Training set 9](#_Toc535673901)

[3.6 Predicting the Test set Results 10](#_Toc535673902)

[3.7 Visualizing the Training set results 11](#_Toc535673903)

[4.Multiple linear Regression 11](#_Toc535673904)

[4.1 Analysis of the Problem 11](#_Toc535673905)

[4.2 Dummy Variables 13](#_Toc535673906)

[4.3 Building A Model 14](#_Toc535673907)

[4.10 Building the optimal model using Backward Elimination 19](#_Toc535673908)

[5.0 Polynomial Regression 21](#_Toc535673909)

[5.1 Difference and need of Polynomial Regression : 21](#_Toc535673910)

[5.2 Analysis of the Problem 21](#_Toc535673911)

[5.3 Fitting Polynomial Regression to the dataset 22](#_Toc535673912)

[5.4 Visualising the Polynomial Regression results 23](#_Toc535673913)

[5.5 Predicting a new result with Linear Regression 24](#_Toc535673914)

[6.0 Decision Tree Regression 25](#_Toc535673915)

[6.1 Splitting the scatter plot to segments 25](#_Toc535673916)

[6.2 Working of a decision Tree Algorithm 27](#_Toc535673917)

[6.3 Analysis of the Problem 28](#_Toc535673918)

[6.4 Fitting Decision Tree Regression to the dataset 29](#_Toc535673919)

[6.5 Visualizing the Decision Tree Regression results 30](#_Toc535673920)

[7.0 Random Forest Regression 30](#_Toc535673921)

[7.1 Analysis of the Problem 31](#_Toc535673922)

[7.2 Fitting Random Forest Regression to the dataset 31](#_Toc535673923)

[7.3 Predicting the value 32](#_Toc535673924)

[7.4 Visualizing the Random Forest Regression results 33](#_Toc535673925)

[8.0 Support Vector Regression 33](#_Toc535673926)

[8.1 Introduction 34](#_Toc535673927)

[8.2 Building a SVR 34](#_Toc535673928)

[8.3 Feature Scaling 35](#_Toc535673929)

[8.4 Visualizing the SVR results 35](#_Toc535673930)

# 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.

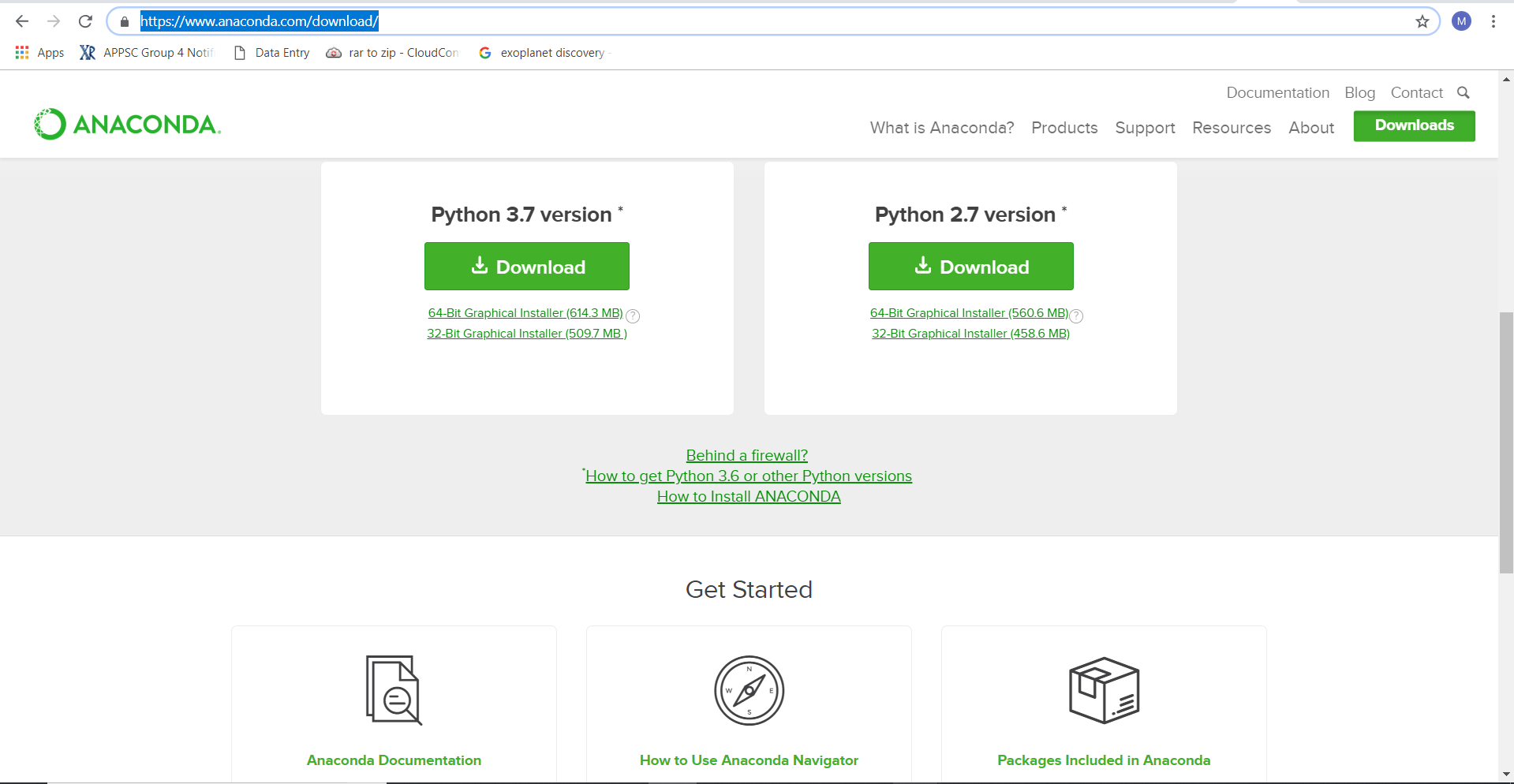
## 1.3 Building a Predictive Model in Python

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

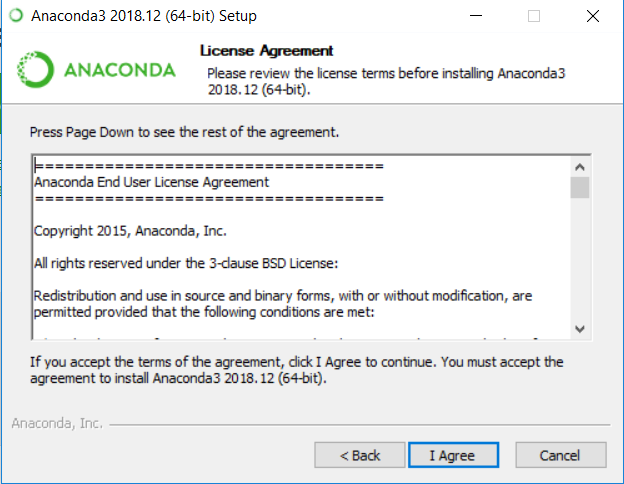
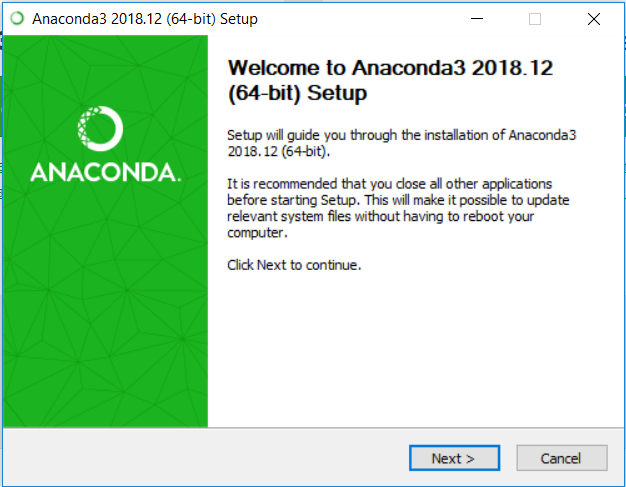
* Simple Linear Regression.
* Multiple linear Regression.
* Polynomial Regression.
* Support Vector Regression.
* Decision Tree Regression.
* Random Forest Regression.

# 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.

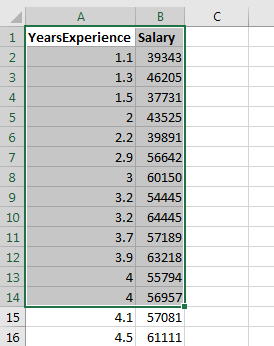
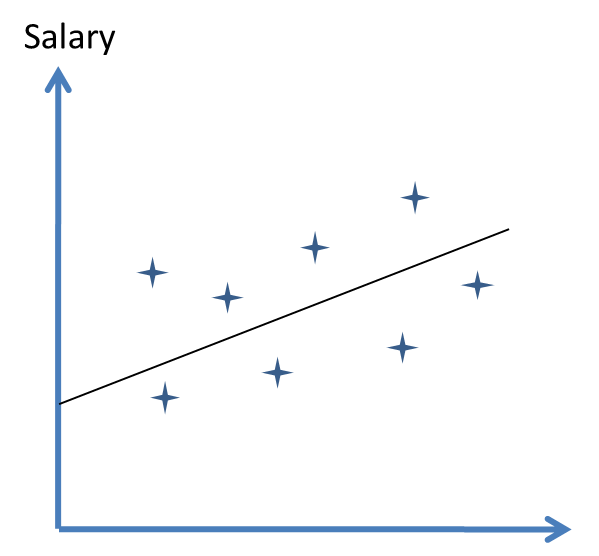
****

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

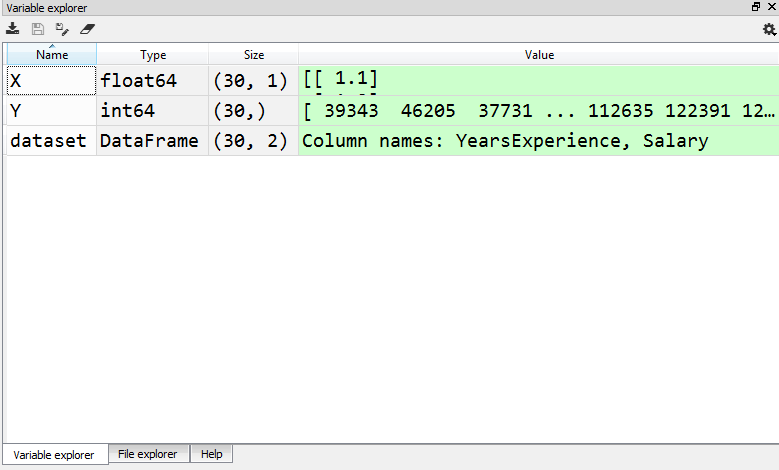
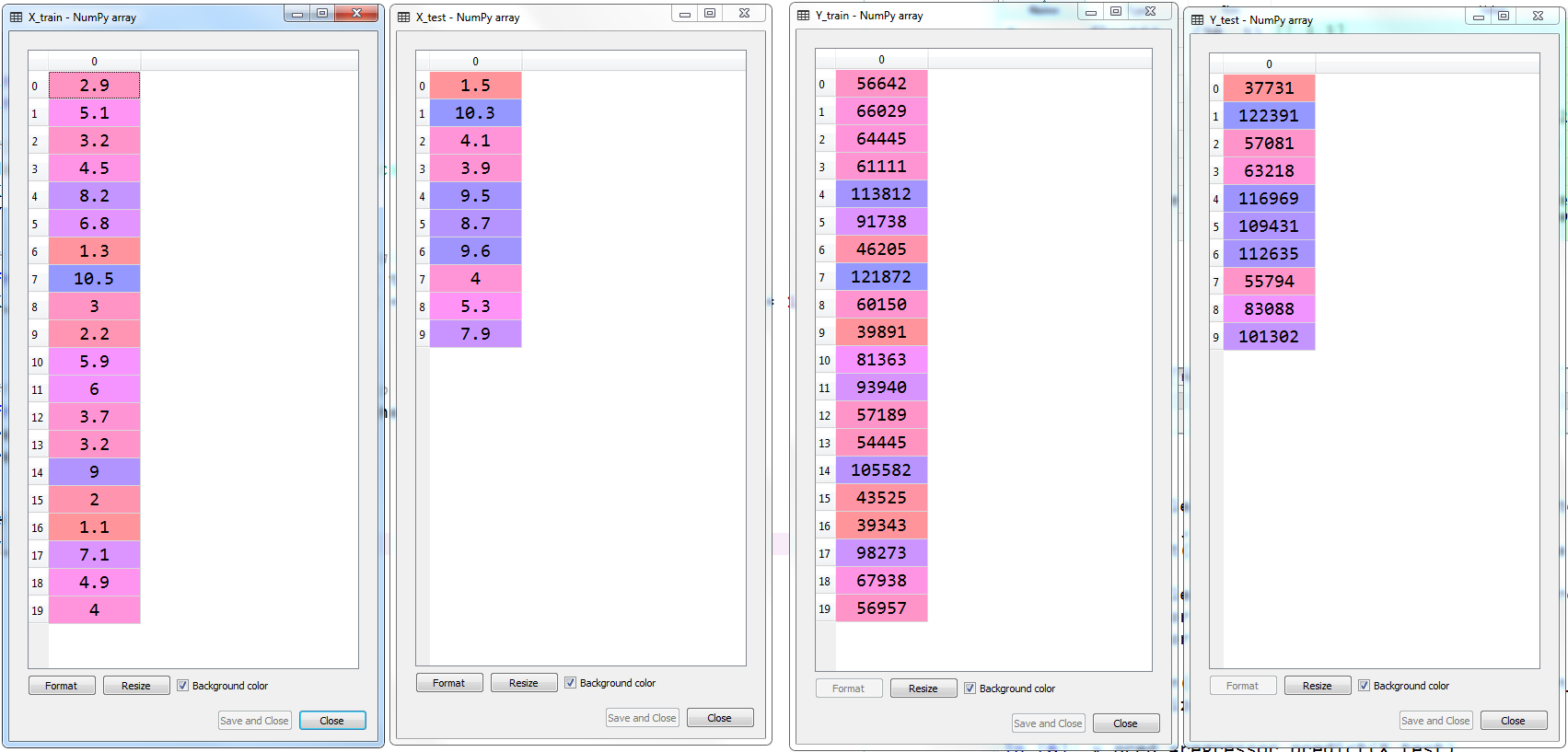


Fig: 3.1.3

## 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)

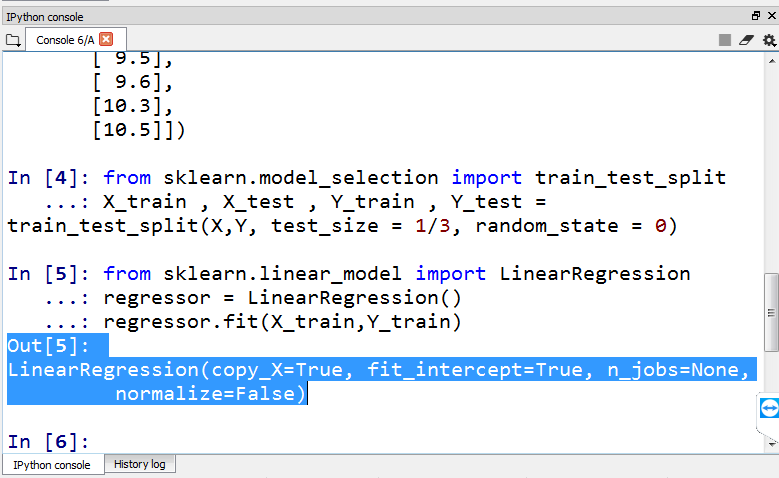
Importing the training set and splitting the given data into training set and test sets in some ratio here (1/3) to test set and 2/3 to training set



## 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)

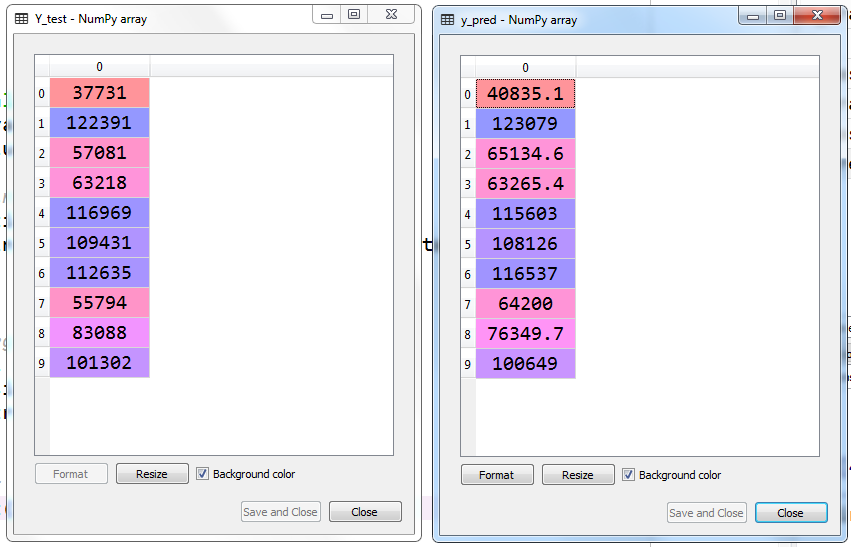
Linear Regress or is set to the training set



## 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.

****

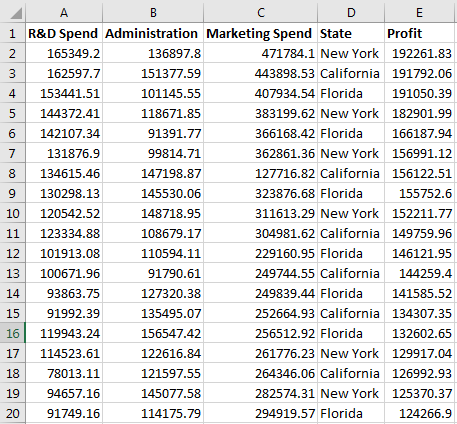
# 4.Multiple linear Regression

## 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.

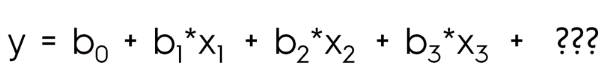
***Assumptions Of a Linear Regression***:

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

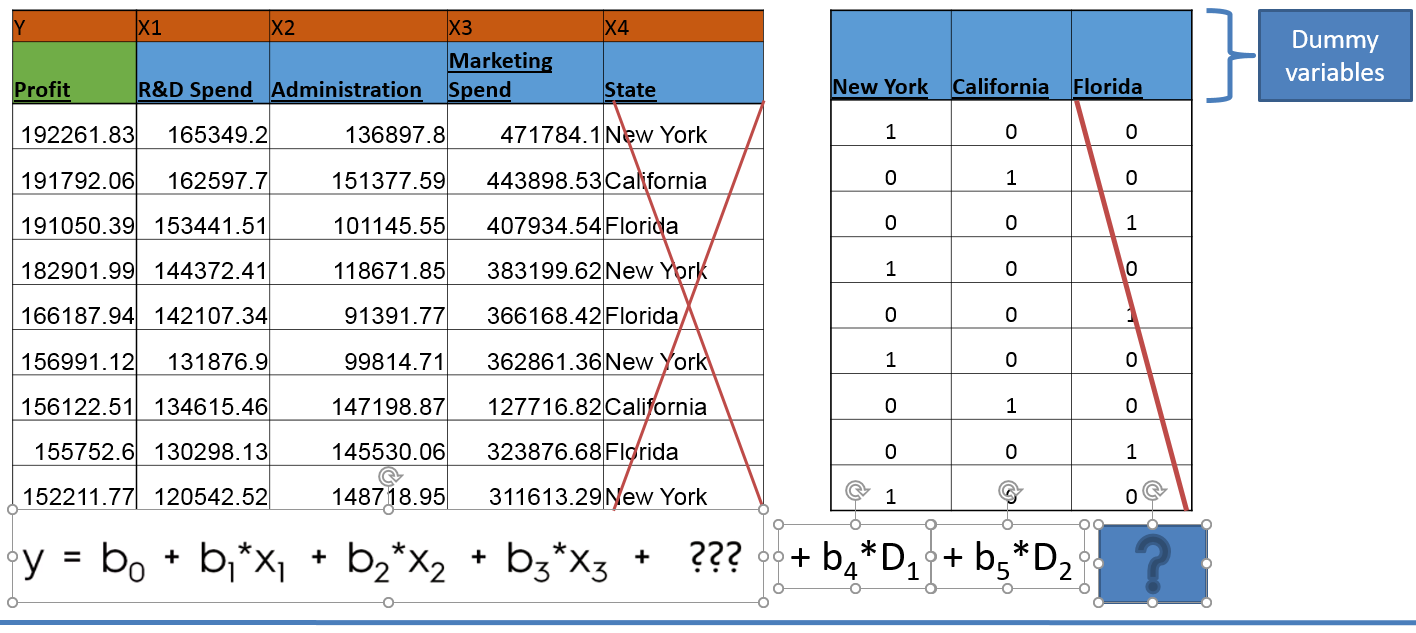
## 4.2 Dummy Variables

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 |

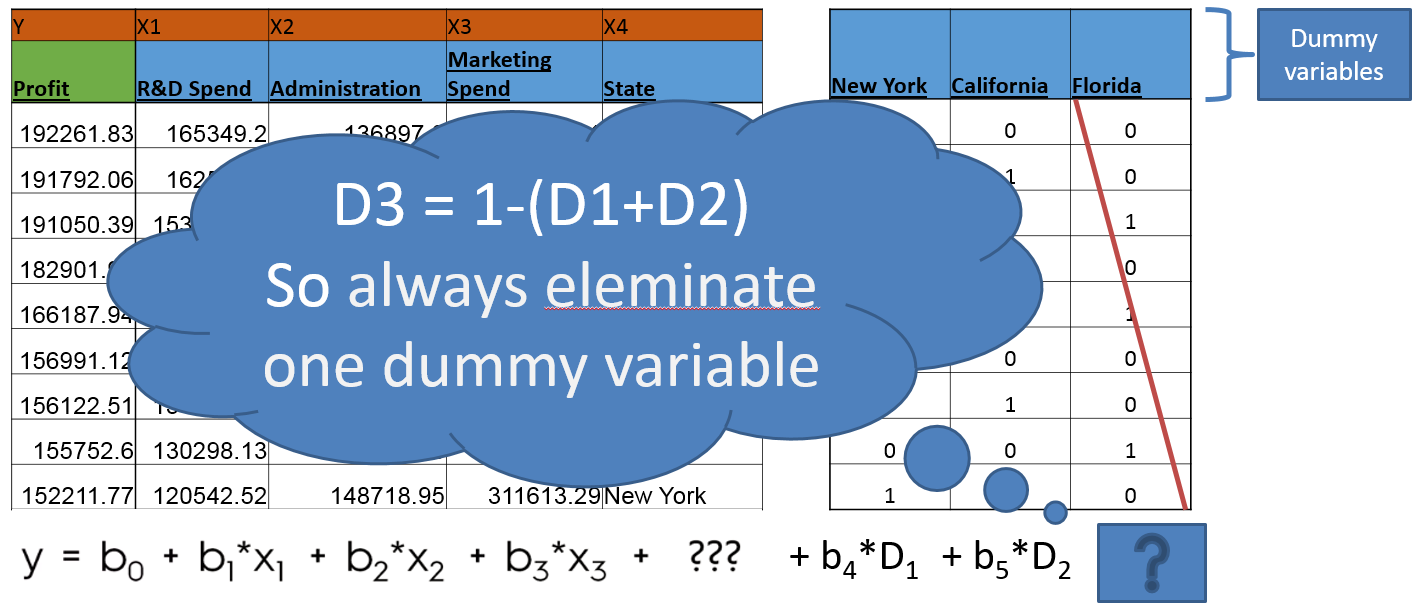


* State: Categorical variable
* Dummy variables to replace the categorical variables.



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



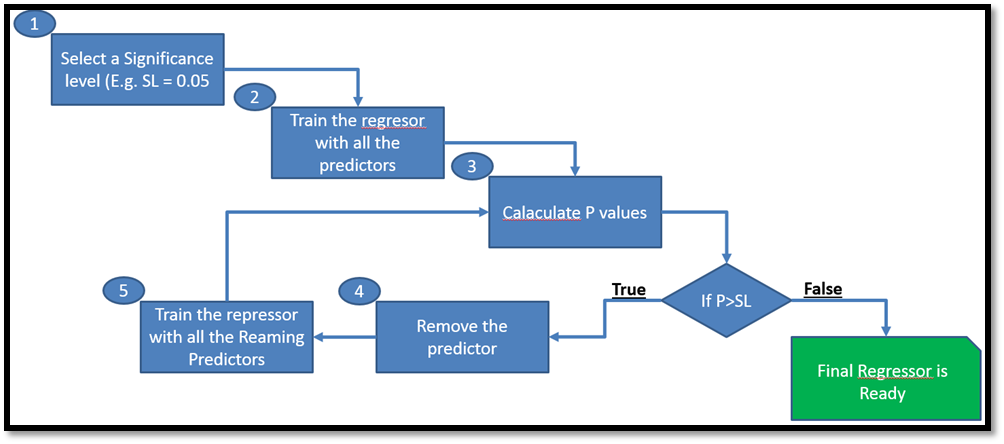
This is because the we divide the weight of the variables into n categories and the last category is added in the other variable parameters.

## 4.3 Building A Model

Methods to Build the models

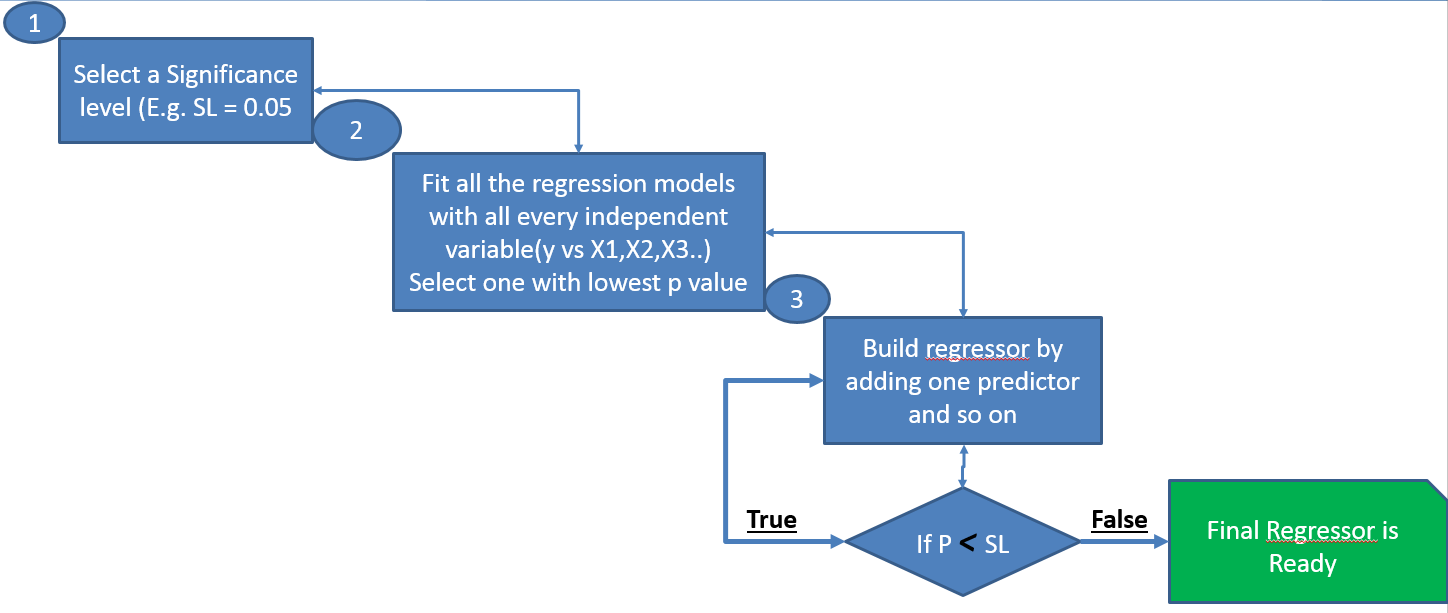
* All –in
* Step wise Regression Models
* Backward Elimination
* Forward Selection
* Bidirectional Elimination
* Score Comparison

4.4 Backward Elimination



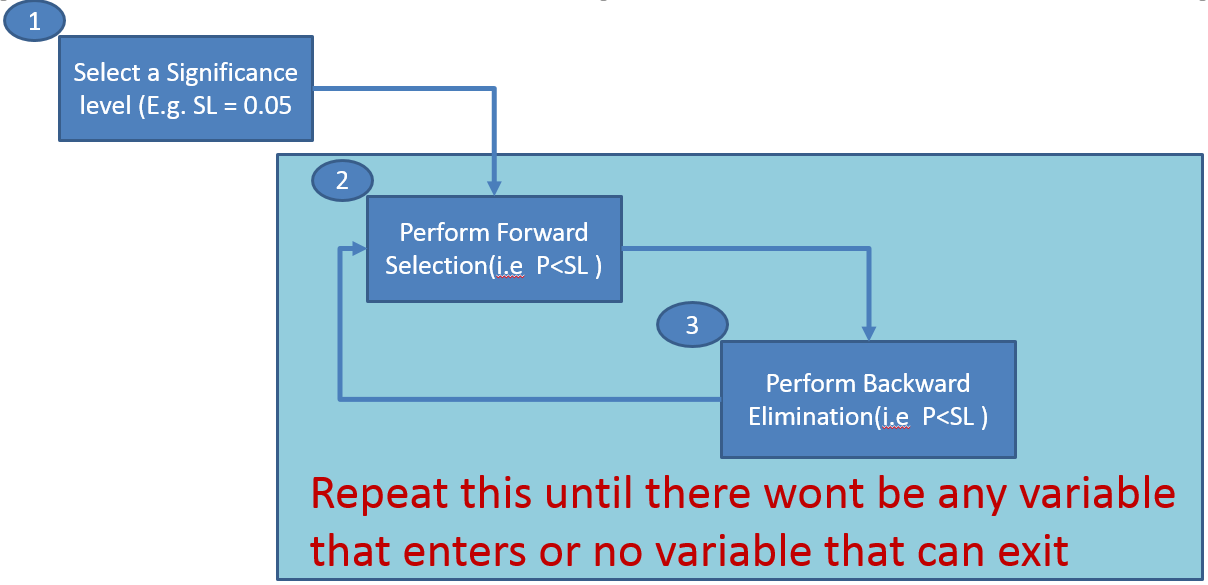
Select a significance level and then training the repressor followed by calculation of p values for all corresponding sets. After calculating the P values if P>SL remove the predictor or the independent variables and then train the regressor with reaming independent variables and calculate the P values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

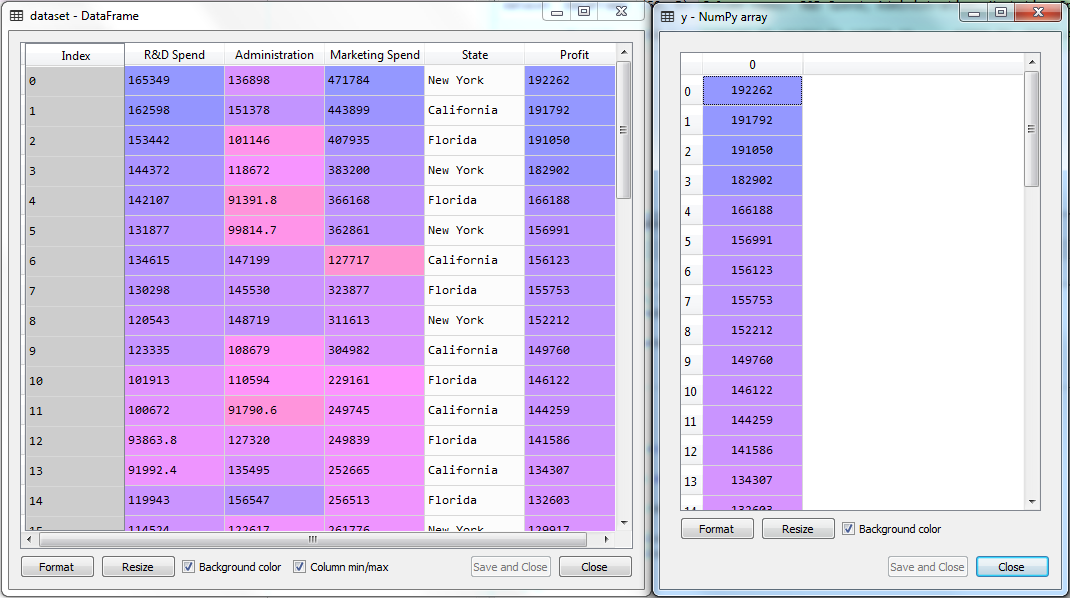
4.6 Bi-Directional Elimination



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

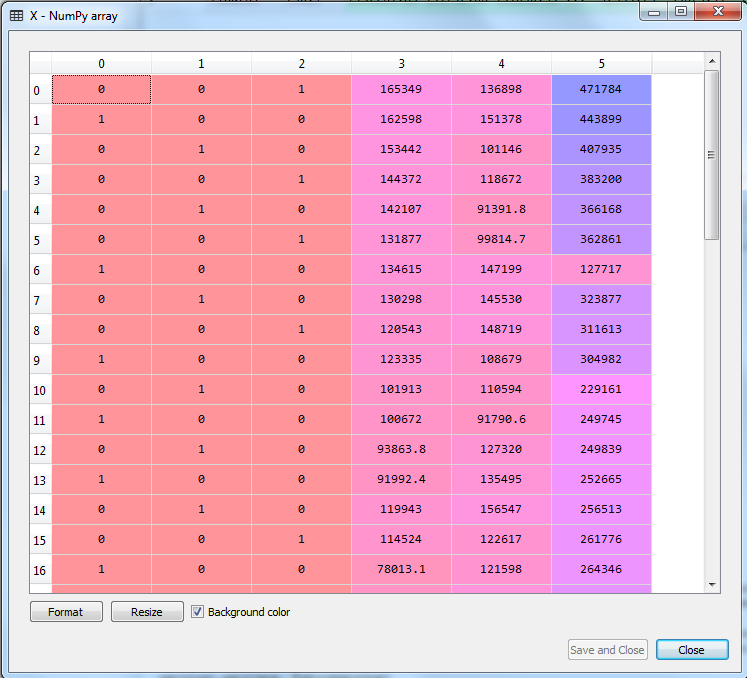
1. # Importing the libraries
2. **import** numpy as np
4. **import** pandas as pd
6. # Importing the dataset
7. dataset = pd.read\_csv('50\_Startups.csv')
8. X = dataset.iloc[:, :-1].values
9. y = dataset.iloc[:, 4].values



4.7 Encoding categorical data

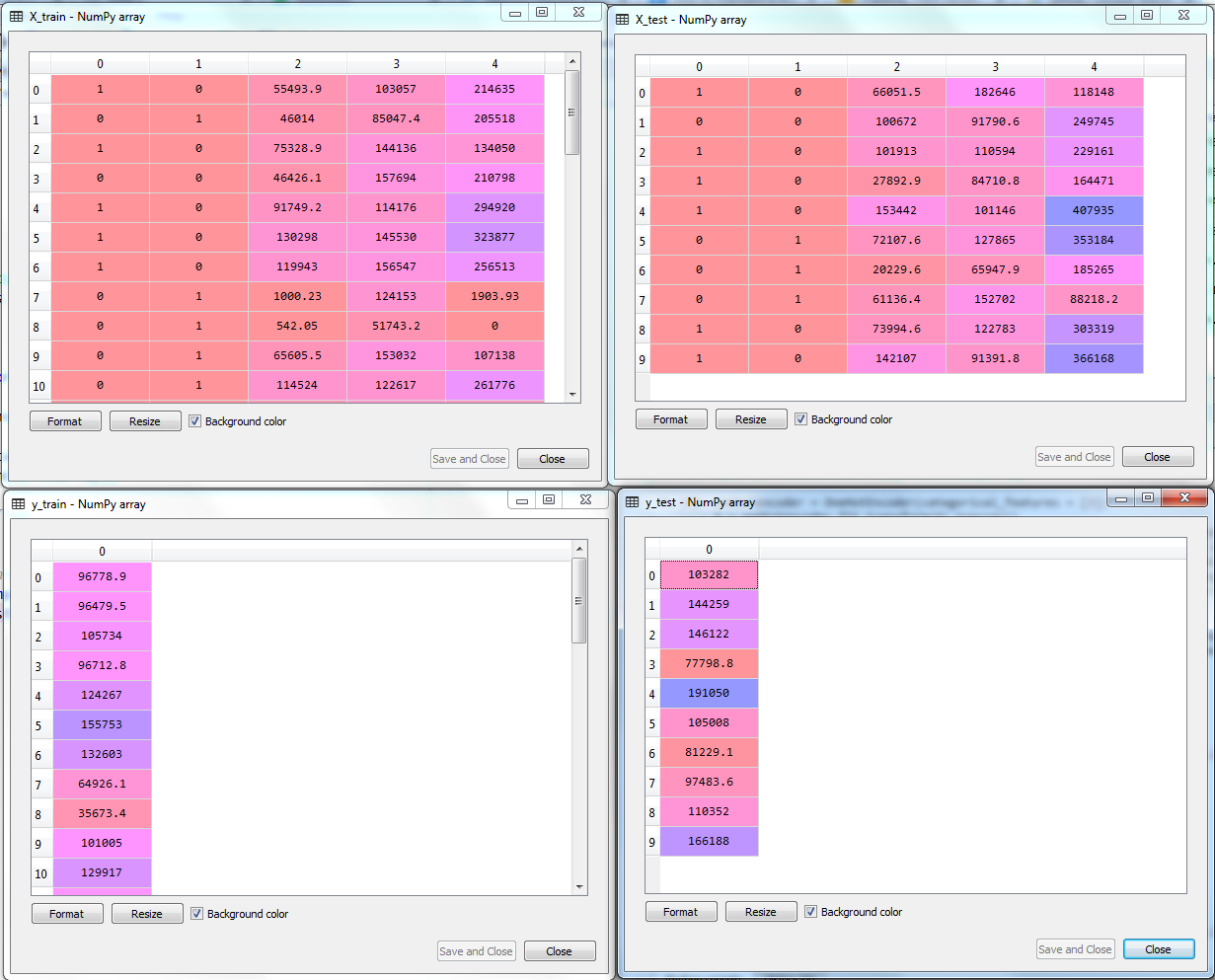
1. # Encoding categorical data
2. **from** sklearn.preprocessing **import** LabelEncoder, OneHotEncoder
3. labelencoder = LabelEncoder()
4. X[:, 3] = labelencoder.fit\_transform(X[:, 3])
5. onehotencoder = OneHotEncoder(categorical\_features = [3])
6. X = onehotencoder.fit\_transform(X).toarray()

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)



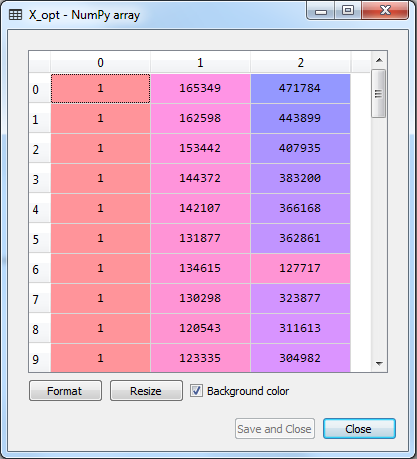
4.9 Fitting Multiple Linear Regression to the Training set

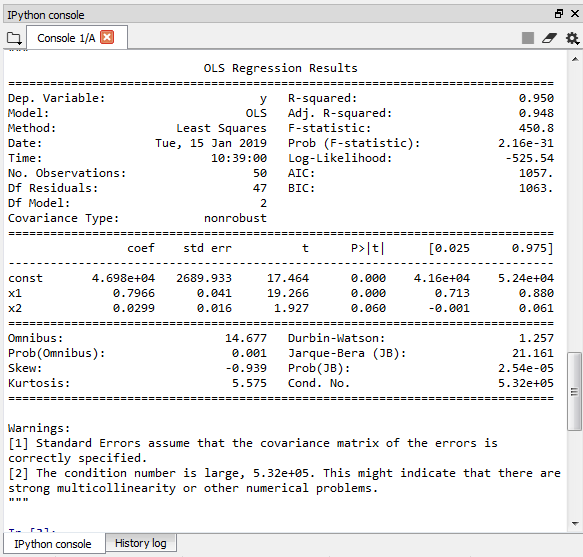
1. # Fitting Multiple Linear Regression to the Training set
2. **from** sklearn.linear\_model **import** LinearRegression
3. regressor = LinearRegression()
4. regressor.fit(X\_train, y\_train)

## 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()

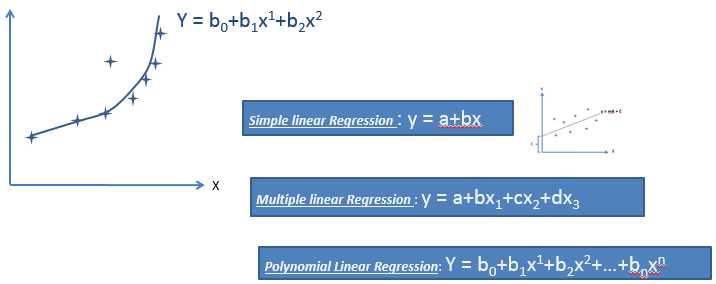
Step by step elimination of all the variables with P values greater than SL will lead us to the following result with no P>SL. That would be our final regressor.



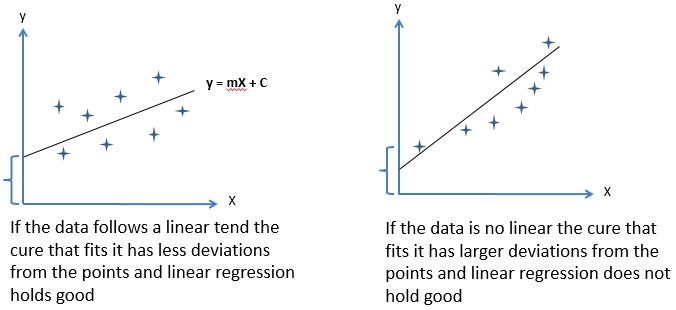


# 5.0 Polynomial Regression

## 5.1 Difference and need of Polynomial Regression :



If the data follows a linear tend the cure that fits it has less deviations from the points and linear regression holds good. If the data is no linear the cure that fits it has larger deviations from the points and linear regression does not hold good.



The cure fits the data almost with negligible deviations. The type of data is the factor for use to decide which regression analysis to be used.

E.g. Spread of urban Population, Epidemic etc..

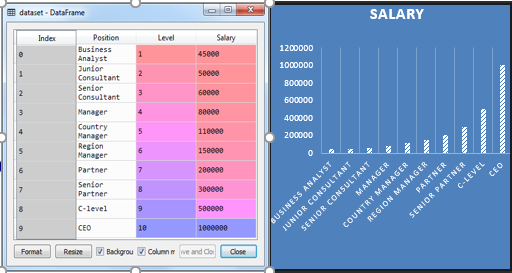
## Analysis of the Problem

**Problem Statement**: Identifying weather if the salary declared by a new employee in his previous company is true or false

**Given**: data Set of Positions experience level and salaries of 10 different positions at different expertise level.

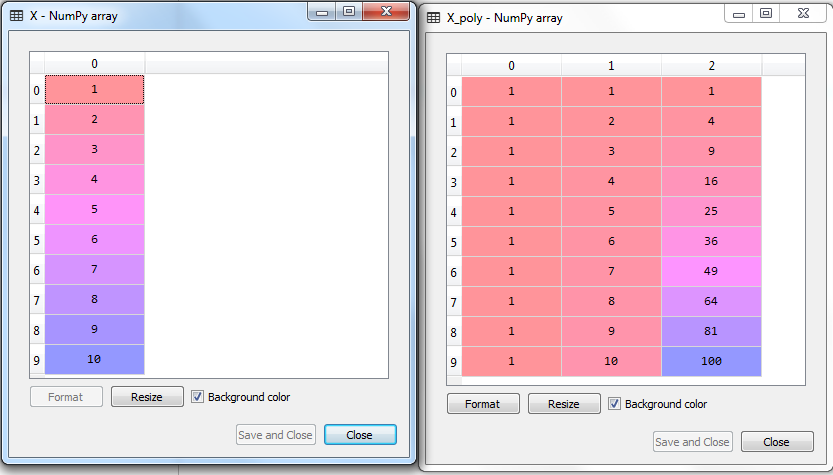
**Expectation of the regression**: To develop a Regression model which would help the Employer to estimate the salary for his level of experience & decide on the honesty of his new employee about his salary.

1. # Importing the libraries
2. **import** numpy as np
3. **import** pandas as pd
4. # Importing the dataset
5. dataset = pd.read\_csv('Position\_Salaries.csv')



## Fitting Polynomial Regression to the dataset

1. # Polynomial Regression
2. """
3. Created on Tue Jan 15 15:01:59 2019
4. @author: sashila
5. """
6. # Importing the libraries
7. **import** numpy as np
8. **import** pandas as pd
10. # Importing the dataset
11. dataset = pd.read\_csv('Position\_Salaries.csv')
12. X = dataset.iloc[:, 1].values
13. y = dataset.iloc[:, 2].values
15. # Fitting Linear Regression to the dataset
16. **from** sklearn.linear\_model **import** LinearRegression
17. lin\_reg = LinearRegression()
18. lin\_reg.fit(X, y)
20. # Fitting Polynomial Regression to the dataset
21. **from** sklearn.preprocessing **import** PolynomialFeatures
22. poly\_reg = PolynomialFeatures(degree = 2)
23. X\_poly = poly\_reg.fit\_transform(X)
24. poly\_reg.fit(X\_poly, y)
25. lin\_reg\_2 = LinearRegression()
26. lin\_reg\_2.fit(X\_poly, y)

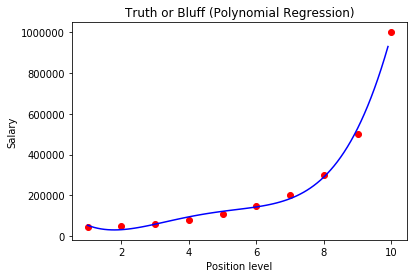


Polynomial regression with degree 2 is fixed and the X\_poly variables is created with the corresponding values and its square terms.

## Visualising the Polynomial Regression results

1. # Visualising the Polynomial Regression results
2. plt.scatter(X, y, color = 'red')
3. plt.plot(X, lin\_reg\_2.predict(poly\_reg.fit\_transform(X)), color = 'blue')
4. plt.title('Truth or Bluff (Polynomial Regression)')
5. plt.xlabel('Position level')
6. plt.ylabel('Salary')
7. plt.show()

The regressor which follows this can be visualized by the above code and looks as follows:plot, title, Xlable, y label , show are the methods of the matplot .



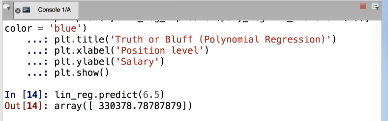
X\_grid **=** np**.**arange**(**min**(**X**),** max**(**X**),** 0.1**)**

X\_grid **=** X\_grid**.**reshape**((**len**(**X\_grid**),** 1**))**

Using grid make the cure smoother.

## Predicting a new result with Linear Regression

1. # Predicting a new result with Linear Regression
2. lin\_reg.predict(6).reshape(1,-1)

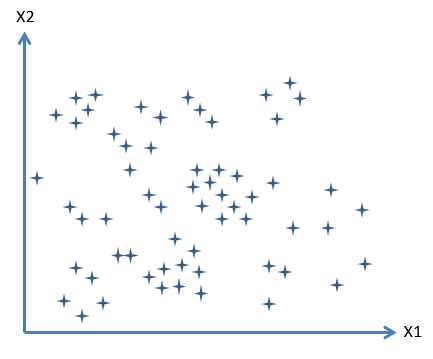


Predict method is called on poly lin\_regressor object to find the predicted value of the given input.

# 6.0 Decision Tree Regression

**Introduction**

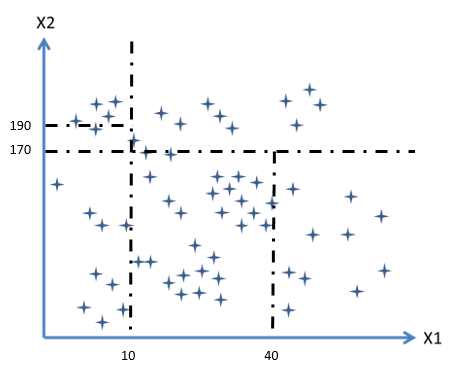
Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

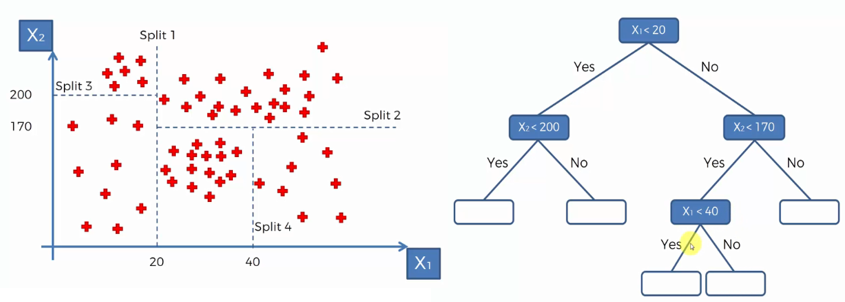


X1,X2 are the independent variables, and Y the dependent variable is on the third dimension. The work should be done on these independent variables to build the decision tree. Then actual prediction of the independent variables is made.

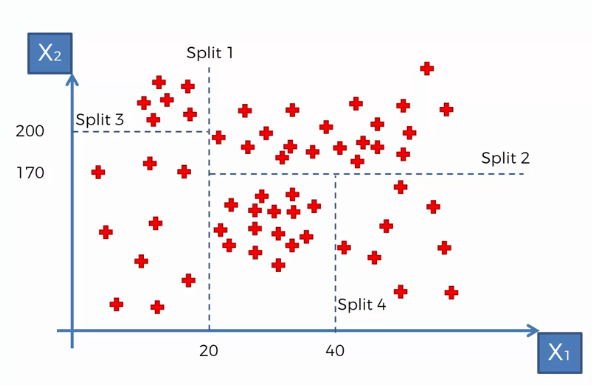
## Splitting the scatter plot to segments

Scatterplot will be split up into segments and -how an algorithm could go about doing that.? How and where these splits are conducted is determined by the algorithm. Is it actually adding some value to the way that we want to group our points and the algorithm knows when to stop is when there is a certain minimum for the information that needs to be added.





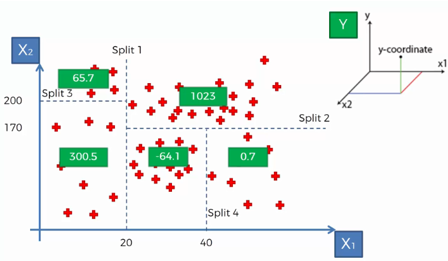
If the value falls in the categories, (i.e. if x1 <20 then it checks for the X2 values and so on) the average of the dependent variable in that leaf will be the final predictor.



For any value that falls in the ***green block*** (i.e. the independent variables X1 value ranging between 20 &40 and X2 values between 0&170. will have the average value of the all the Y values corresponding to the pints in that split group.

## Working of a decision Tree Algorithm

The regular method was to take the average of all data points and that would be our answer. But now here we split the data and we add additional value to the prediction.

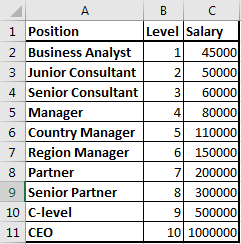


## Analysis of the Problem

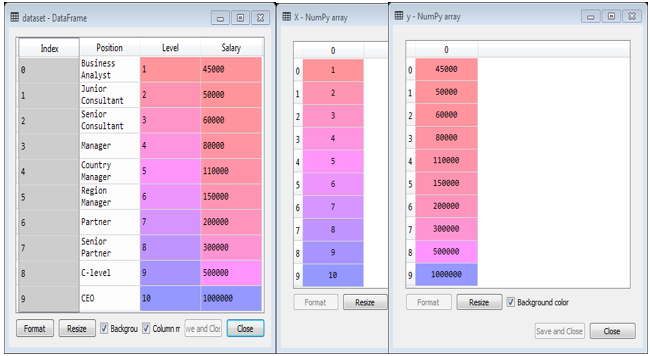
**Problem Statement**: Identifying weather if the salary declared by a new employee in his previous company is true or false

**Given**: data Set of Positions experience level and salaries of 10 different positions at different expertise level.

**Expectation of the regression**: To develop a Regression model which would help the Employer to estimate the salary for his level of experience & decide on the honesty of his new employee about his salary.

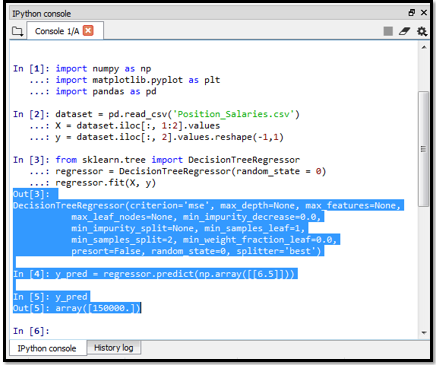


1. **import** numpy as np
2. **import** matplotlib.pyplot as plt
3. **import** pandas as pd
5. # Importing the dataset
6. dataset = pd.read\_csv('Position\_Salaries.csv')
7. X = dataset.iloc[:, 1:2].values
8. y = dataset.iloc[:, 2].values.reshape(-1,1)



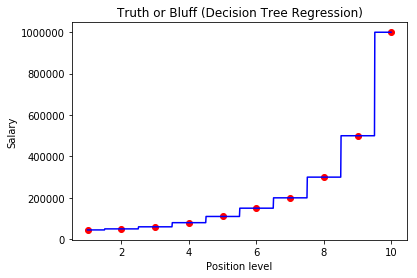
## Fitting Decision Tree Regression to the dataset

1. # Fitting Decision Tree Regression to the dataset
2. **from** sklearn.tree **import** DecisionTreeRegressor
3. regressor = DecisionTreeRegressor(random\_state = 0)
4. regressor.fit(X, y)
5. # Predicting a new result
6. y\_pred = regressor.predict(np.array([[6.5]]))

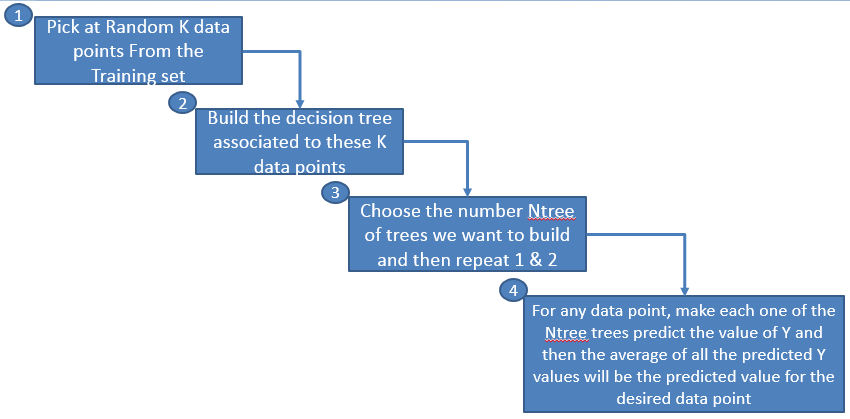


## Visualizing the Decision Tree Regression results

1. # Visualizing the Decision Tree Regression results (higher resolution)
2. X\_grid = np.arange(min(X), max(X), 0.01)
3. X\_grid = X\_grid.reshape((len(X\_grid), 1))
4. plt.scatter(X, y, color = 'red')
5. plt.plot(X\_grid, regressor.predict(X\_grid), color = 'blue')
6. plt.title('Truth or Bluff (Decision Tree Regression)')
7. plt.xlabel('Position level')
8. plt.ylabel('Salary')
9. plt.show()



# 7.0 Random Forest Regression



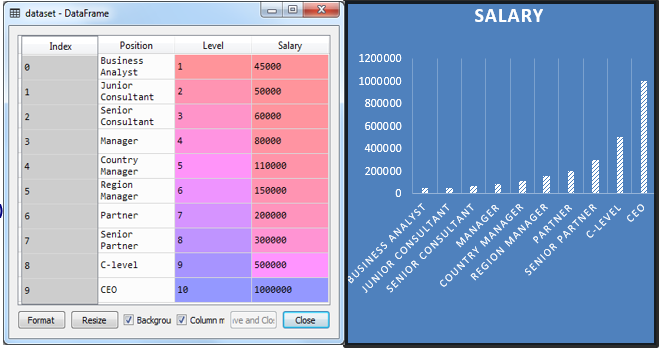
## 7.1 Analysis of the Problem

**Problem Statement:** Identifying weather if the salary declared by a new employee in his previous company is true or false

**Given:** data Set of Positions experience level and salaries of 10 different positions at different expertise level.

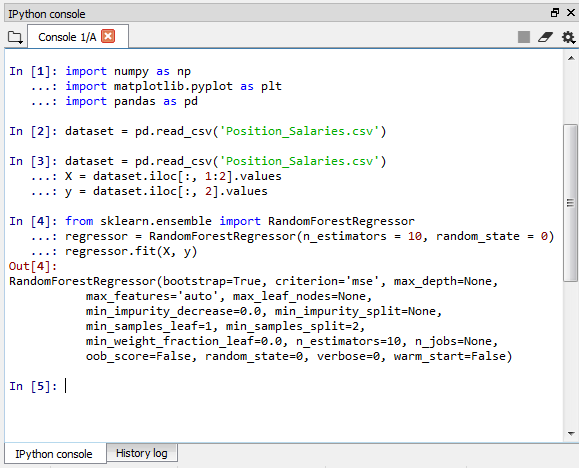
**Expectation of the regression:** To develop a Regression model which would help the Employer to estimate the salary for his level of experience & decide on the honesty of his new employee about his salary.

1. **import** numpy as np
2. **import** pandas as pd
3. # Importing the dataset
4. dataset = pd.read\_csv('Position\_Salaries.csv')



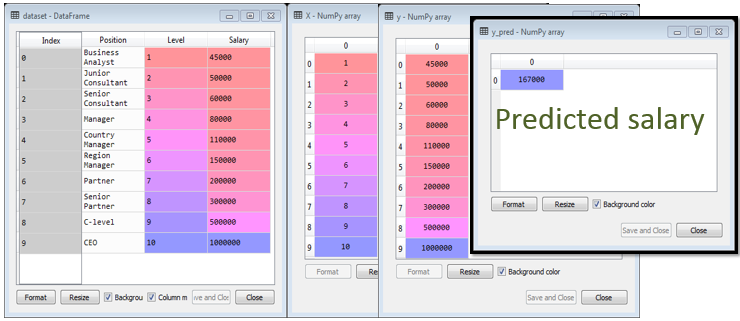
## 7.2 Fitting Random Forest Regression to the dataset

1. # Fitting Random Forest Regression to the dataset
2. **from** sklearn.ensemble **import** RandomForestRegressor
3. regressor = RandomForestRegressor(n\_estimators = 10, random\_state = 0)
4. regressor.fit(X, y)



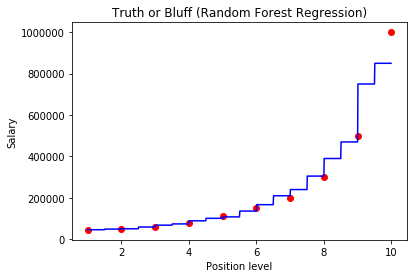
## 7.3 Predicting the value

1. y\_pred = regressor.predict([[6.5]])



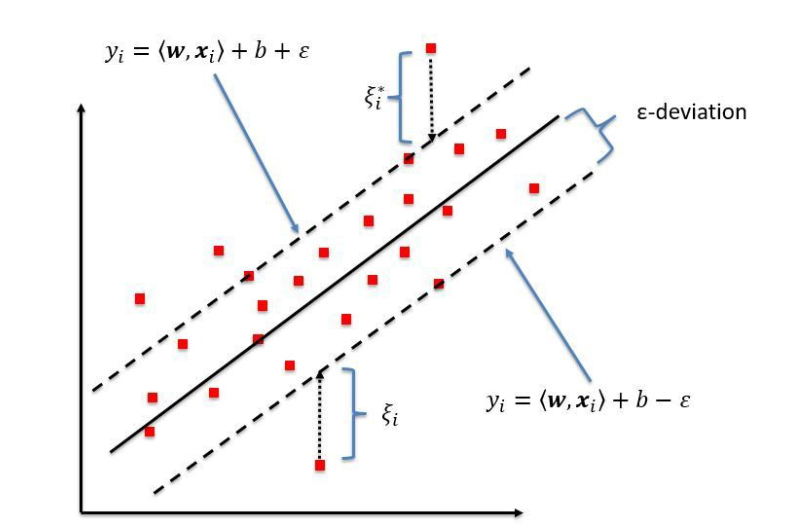
## 7.4 Visualizing the Random Forest Regression results

1. # Visualizing the Random Forest Regression results (higher resolution)
2. X\_grid = np.arange(min(X), max(X), 0.01)
3. X\_grid = X\_grid.reshape((len(X\_grid), 1))
4. plt.scatter(X, y, color = 'red')
5. plt.plot(X\_grid, regressor.predict(X\_grid), color = 'blue')
6. plt.title('Truth or Bluff (Random Forest Regression)')
7. plt.xlabel('Position level')
8. plt.ylabel('Salary')
9. plt.show()



# 8.0 Support Vector Regression

In simple regression we try to minimize the error rate. While in SVR we try to fit the error within a certain threshold. This might be a bit confusing but let me explain.



## 8.1 Introduction

Support Vector Machines support linear and nonlinear regression that we can refer to as SVR. Instead of trying to fit the largest possible street between two classes while limiting margin violations, SVR tries to fit as many instances as possible on the street while limiting margin violations. The width of the street is controlled by a hyper parameter Epsilon.

SVR performs linear regression in a higher (dimensional space). We can think of SVR as if each data point in the training represents it's own dimension. When you evaluate your kernel between a test point and a point in the training set the resulting value gives you the coordinate of your test point in that dimension. The vector we get when we evaluate the test point for all points in the training set, k'' is the representation of the test point in the higher dimensional space. Once you have that vector you the use it to perform a linear regression.

In a classification problem, the vectors X are used to define a hyperplane that separates the two different classes in your solution. These vectors are used to perform linear regression. The vectors closest to the test point are referred to as support vectors. We can evaluate our function anywhere so any vectors could be closest to our test evaluation location.

## 8.2 Building a SVR

1. Collect a training set T = {X, Y}
2. Choose a kernel and it's parameters as well as any regularization needed.
3. Form the correlation matrix, K
4. Train your machine, exactly or approximately, to get contraction coefficients a = {a*i*}
5. Use those coefficients, create your estimator f (I, a, x\*) = y\*

## 8.3 Feature Scaling

Feature scaling is a method used to standardize the range of independent variables or features of data. In [data processing](https://en.wikipedia.org/wiki/Data_processing), it is also known as data normalization and is generally performed during the data preprocessing step.

1. # Importing the libraries
2. **import** numpy as np
3. **import** pandas as pd
5. # Importing the dataset
6. dataset = pd.read\_csv('Position\_Salaries.csv')
7. X = dataset.iloc[:, 1:2].values
8. y = dataset.iloc[:, 2].values.reshape(-1,1)
10. # Feature Scaling
11. **from** sklearn.preprocessing **import** StandardScaler
12. sc\_X = StandardScaler()
13. sc\_y = StandardScaler()
14. X = sc\_X.fit\_transform(X)
15. y = sc\_y.fit\_transform(y)

Since the range of values of raw data varies widely, in some [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms, objective functions will not work properly without [normalization](https://en.wikipedia.org/wiki/Normalization_(statistics)). For example, the majority of [classifiers](https://en.wikipedia.org/wiki/Statistical_classification) calculate the distance between two points by the [Euclidean distance](https://en.wikipedia.org/wiki/Euclidean_distance). If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance.

Another reason why feature scaling is applied is that [gradient descent](https://en.wikipedia.org/wiki/Gradient_descent) converges much faster with feature scaling than without it.

## 8.4 Visualizing the SVR results

1. # Visualizing the SVR results
2. plt.scatter(X, y, color = 'red')
3. plt.plot(X, regressor.predict(X), color = 'blue')
4. plt.title('Truth or Bluff (SVR)')
5. plt.xlabel('Position level')
6. plt.ylabel('Salary')
7. plt.show()

