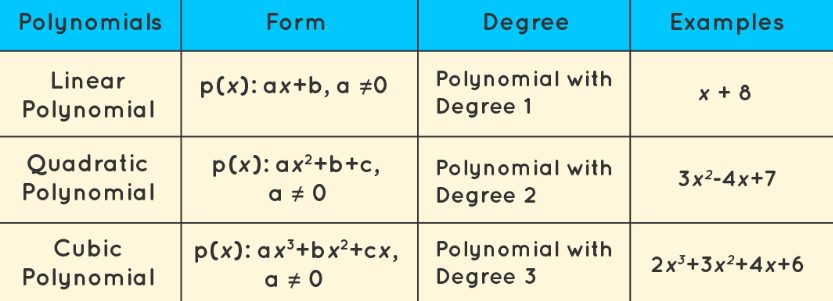
**Polynomial Regression**

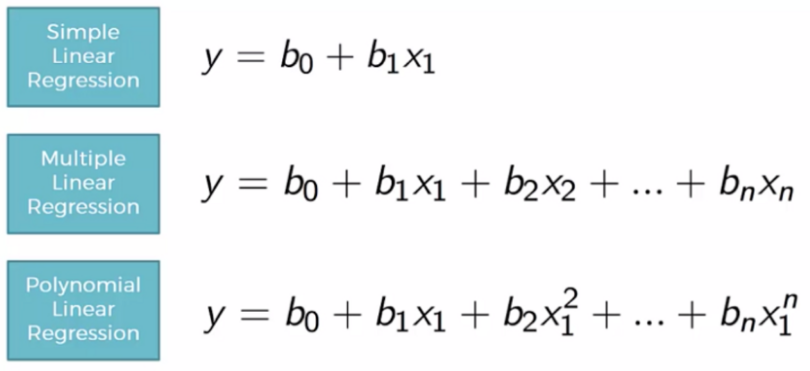
In polynomial regression, it describes the relationship between the independent variable x and the dependent variable y using an nth-degree polynomial in x. Polynomial regression, denoted as E(y | x), characterizes fitting a nonlinear relationship between the x value and the conditional mean of y. Typically, this corresponds to the least-squares method. This represents a type of Linear Regression where the dependent and independent variables exhibit a curvilinear relationship and the polynomial equation is fitted to the data.

## **Types of Polynomial Regression**

A quadratic equation is a general term for a second-degree polynomial equation. This degree, on the other hand, can go up to nth values. Here is the categorization of Polynomial Regression:

* Linear – if degree as 1
* Quadratic – if degree as 2
* Cubic – if degree as 3 and goes on, on the basis of degree.

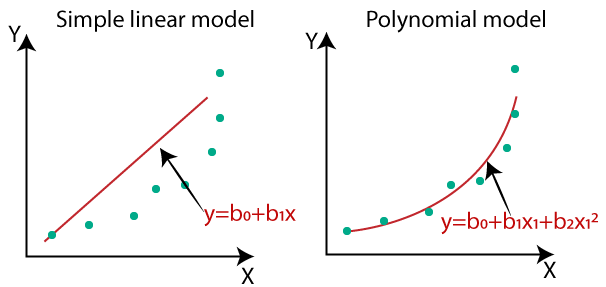




## **Need for Polynomial Regression**

The need of Polynomial Regression in Machine Learning:

* If we apply a linear model on a linear dataset, then it provides us a good result as we have seen in Simple Linear Regression, but if we apply the same model without any modification on a non-linear dataset, then it will produce a drastic output. Due to which loss function will increase, the error rate will be high, and accuracy will be decreased.
* So for such cases, where data points are arranged in a non-linear fashion, we need the Polynomial Regression model. We can understand it in a better way using the below comparison diagram of the linear dataset and non-linear dataset.



* In the above image, we have taken a dataset which is arranged non-linearly. So if we try to cover it with a linear model, then we can clearly see that it hardly covers any data point. On the other hand, a curve is suitable to cover most of the data points, which is of the Polynomial model.
* Hence, if the datasets are arranged in a non-linear fashion, then we should use the Polynomial Regression model instead of Simple Linear Regression.

When we compare the above three equations, we can clearly see that all three equations are Polynomial equations but differ by the degree of variables. The Simple and Multiple Linear equations are also Polynomial equations with a single degree, and the Polynomial regression equation is Linear equation with the nth degree. So if we add a degree to our linear equations, then it will be converted into Polynomial Linear equations.

**Building the Polynomial regression model**

Now we will build the Polynomial Regression model, but it will be a little different from the Simple Linear model. Because here we will use PolynomialFeatures class of preprocessing library. We are using this class to add some extra features to our dataset.

#Fitting the Polynomial regression to the dataset

from sklearn.preprocessing import PolynomialFeatures

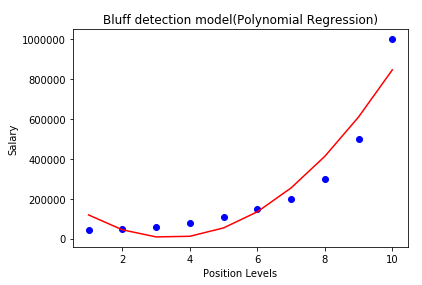
poly\_regs= PolynomialFeatures(degree= 2)

x\_poly= poly\_regs.fit\_transform(x)

lin\_reg\_2 =LinearRegression()

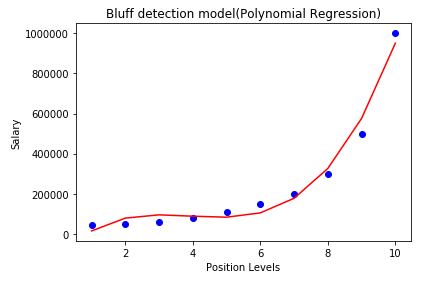
lin\_reg\_2.fit(x\_poly, y)

**Output**



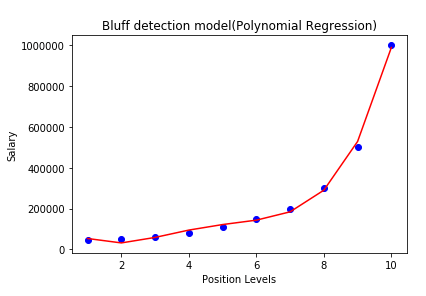
**For degree= 3:**

If we change the degree=3, then we will give a more accurate plot, as shown in the below image.



**Degree= 4:**

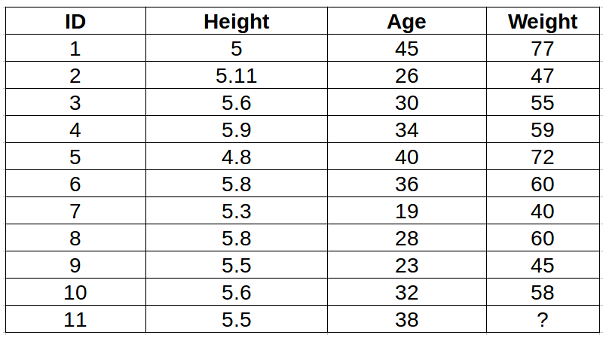
Let's again change the degree to 4, and now will get the most accurate plot. Hence we can get more accurate results by increasing the degree of Polynomial.

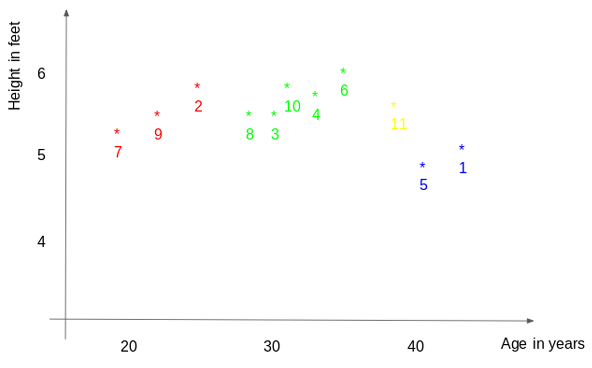


# **KNN (K-Nearest Neighbors) Algorithm for Regression**

* K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
* K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
* K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
* K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

Let us start with a simple example. In the following table, it consists of the height, age, and weight (target) values for 10 people. As we can see, the weight value of ID11 is missing. We need to predict the weight of this person based on their height and age.

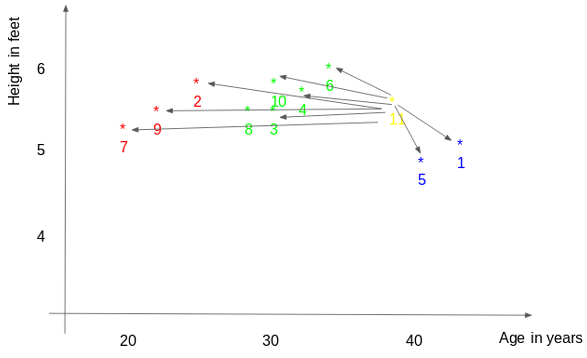




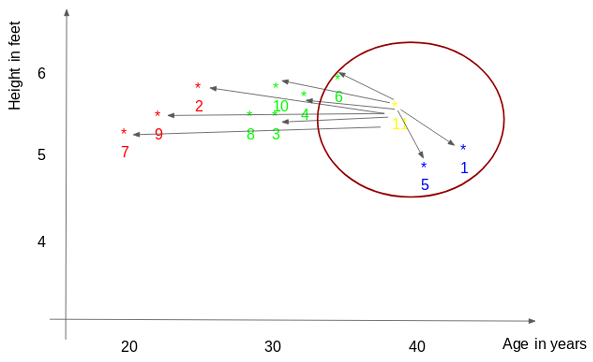
In the above graph, the y-axis represents the height of a person (in feet) and the x-axis represents the age (in years). The points are numbered according to the ID values. The yellow point (ID 11) is our test point. Since ID11 is closer to points 5 and 1, it must have a weight similar to these IDs, probably between 72-77 kgs (weights of ID1 and ID5 from the table).

**Below is a stepwise explanation of the algorithm:**

1. First, the distance between the new point and each training point is calculated.



2. The closest k data points are selected (based on the distance). In this example, points 1, 5, and 6 will be selected if the value of k is 3.

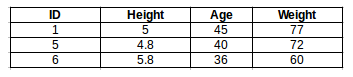


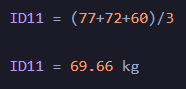
3. The average of these data points is the final prediction for the new point. Here, we have the weight of ID11 = (77+72+60)/3 = 69.66 kg.

## **K-factor**

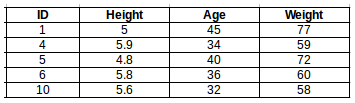
The second step is to select the k value. This determines the number of neighbors we look at when we assign a value to any new observation.

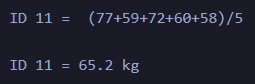
In our example, for a value k = 3, the closest points are ID1, ID5, and ID6.



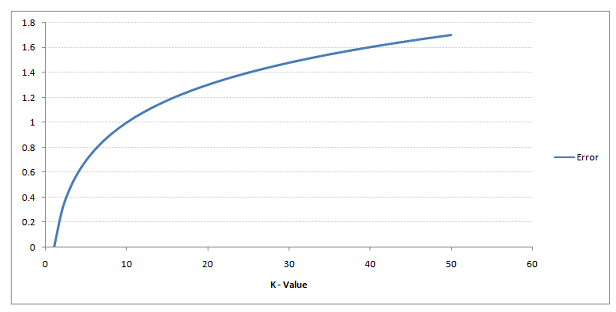


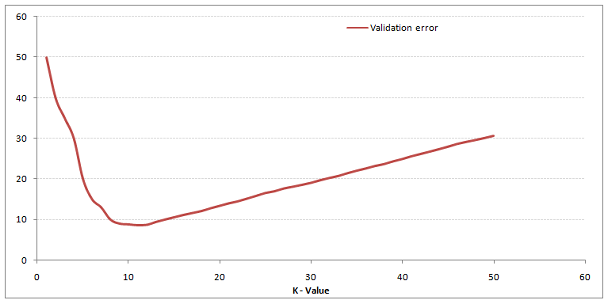
For the value of k=5, the closest point will be ID1, ID4, ID5, ID6, and ID10.





For a very low value of k (suppose k=1), the model is overfitting the training data, which leads to a high error rate on the validation set. On the other hand, for a high value of k, the model performs poorly on both the train and validation sets. If you observe closely, the validation error curve reaches a minimum at a value of k = 9. This value of k is the optimum value of the model (it will vary for different datasets). This curve is known as an ‘elbow curve‘ (because it has a shape like an elbow) and is usually used to determine the k value.





Here, when we take k=1, we get a very high RMSE value. The RMSE value decreases as we increase the k value. At k= 7, the RMSE is approximately 1219.06 and shoots upon further increasing the k value. We can safely say that k=7 will give us the best result in this case.