# Machine Learning in Numeric Analytics

Alejandro Gil Maya

Ricardo Ocampo Castro

Daniel Hernández Stalhuth

Investigation about Machine Learning

Francisco Jose Correa Zabala

EAFIT University

Department of Software Engineer

Medellín

2018

## Table of Content

**Graphics List**

## 1 Abstract

In this report we are trying to explain in the best way possible an introduction to machine learning, we are covering many areas of machine learning like regression, classification, clustering and neuronal networks, the main objective is to obtain that anyone who read this report knows how to start in the world of artificial intelligence but most of all in the world of machine learning. The first step in this is regression, with linear, quadratic and polynomial regression, the second step is Fast Fourier transformations and Genetic algorithms, the last step is Neuronal networks with deep learning algorithms, to fully understand machine learning we need to cover, the basics of what is for? What does it do? And how do I do it? with that questions we must explain the equations in the core of the algorithms of machine learning, the implementation method in code and the right way to use it with data, is not the same using some algorithms with few data than with many data or as we like to call it Big Data. For the implementation in code in this report we are going to use python as programing language because in machine learning algorithms is the most versatile for implementation.

## 2 Introduction

What is Machine learning? Machine learning is a subgroup of Artificial intelligence, is often confused with it but artificial intelligence is a more complex area, machine learning is the core of artificial intelligence, without machine learning is not possible to advance in the future of AI. So basically what is ML, if we decompose the two words learning is the ability to get knowledge and skills through study, experience or being taught, and machine is a device powered by mechanical energy to perform different tasks, so we get the two words together we have machine learning that means a device powered by mechanical energy with the capacity to get knowledge and skills through study, experience or being taught, also like the methods to get knowledge or skills, machine learning has his own methods to give knowledge to the machine, like supervised learning, unsupervised learning and reinforced learning. Machine learning has different techniques to apply to different situations in learning, like decisions trees, regression, classification, clustering and neuronal networks, in the investigation we are trying to get every one of them.

The first steps into machine learning are regression, regression is a series of algorithms and process to determine the best fit line into a series of data, regression starts with the most basic of all that is linear regression, linear regression creates a straight line in the data with the best fit to every point, in linear data this methods could be the best, but not every data is linear, so we must learn to apply the other methods, like quadratic regression, polynomial regression, Fourier series and more.

The next steps are a bit difficult but are the most important steps in today society, neural network and deep learning algorithms with Big Data and Blockchain are the main topics of the present time, and Big data have many relations with machine learning, every day we can see better the importance of data and the vast quantity of data that is on the internet and we also can see how it grows, that why we like to add to this report as many information about neural networks as we can.

# 3 Regression

As we explain before in the introduction regression find the best fit line to series of data, the first one is linear regression, then quadratic and finally polynomial regression.

## 3.1 Linear Regression

### 3.1.1 Theory

First of all we must talk about the equation in linear regression, in the school we learn about this equation: Y = mX + b, Y is the coordinates y for the point and x is the coordinates x for the same point, m is the slope coefficient, and b is the intercept f the line with the y axis, the main purpose for this equation was to find a line that pass for 2 points and give the slope and the intercept, with this line we can find further points, linear regression is almost the same as this equations, it tries to find the best straight line that pass for not just two points, but with many points as we want, in the case where we have many points the line is not going to touch every point, if that would happen it would be interpolation of every point and not regression, the equation in linear regression varies to find the slope and the intercept, to find the slope we use this equation:

$$\mathrm{m} = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{n(\Sigma x^2) - (\Sigma x)^2}$$

*Equation 1*

This is the equation we get to obtain the slope of the best fit line in a large number of points, the only new valor is n who is the number of points in the data, we use a summation to interact with every point in the series of data, the summation in everyone we can see in the equation is between 0 and the n points of the data, the other main equation we are going to use is the equation to find the intercept with the y axis, which is:

$$\mathrm{b} = \frac{(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma xy)}{n(\Sigma x^2) - (\Sigma x)^2}$$

*Equation 2*

This equation is pretty similar to the slope equation but with this we find the interception, the summation is the same, between 0 and n points, once we got the result of both equation we can add it to the main equation which is: Y = mX + b. with the result of this equation we find the best fit line to any series of data between 0 and n.

There is one more equation in regression that tell us if that I in fact the best fit line, and that is r_squared, this is the coefficient of determination, what does this mean, r^2 is used for get a number who is like a percentage of error, it calculates the line vs the data in the graph to see how good of a fit have the line that we have just created, to calculate this we must apply this equation:

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

*Equation 3*

So first, what is this equation? We can see the equation is 1 minus something, the 1 in there because like we said before this is used for calculate something like a percentage, if the result of that "something" is 0 the percentage would be 100%, that means the line pass through all the points in the graph, the line is not linear, is interpolated or we have all the points in a similar linear factor, next we can explain what is this "something" we are telling about, this is the division between the total summation of squares and the regression summation of squares, the first we get it from the points in coordinate Y and the second we get it form the best fit line, respectively the equations of this ones and the results we get it from here:

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

*Equation 5*

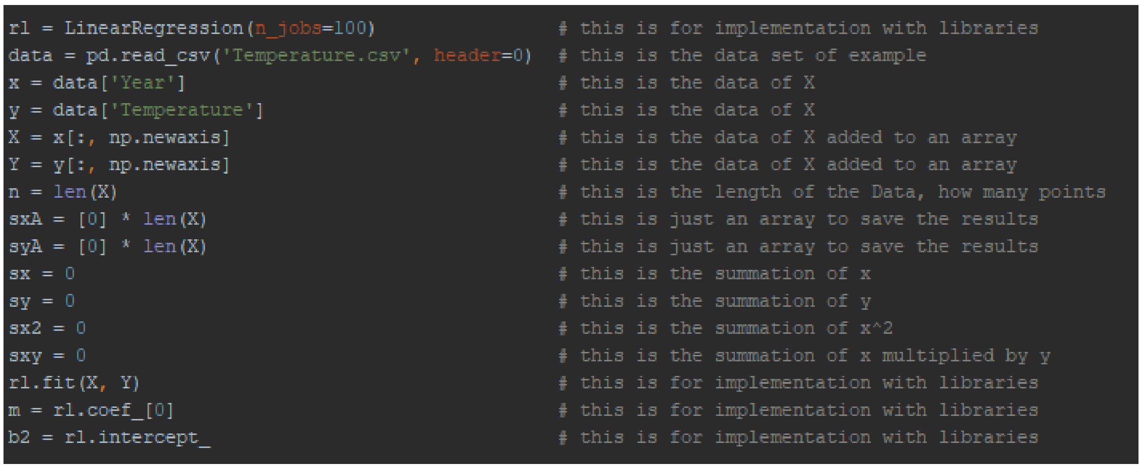
$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

*Equation 4*

As we can see these are just summations of the variables we obtain calculating the best fit line and adding the set of data, **IMPORTANT NOTE,** we are going to use r_squared in every kind of regression, this help us see if the best fit line for a set of data is linear, quadratic or polynomial.

### 3.1.2 Implementation in code

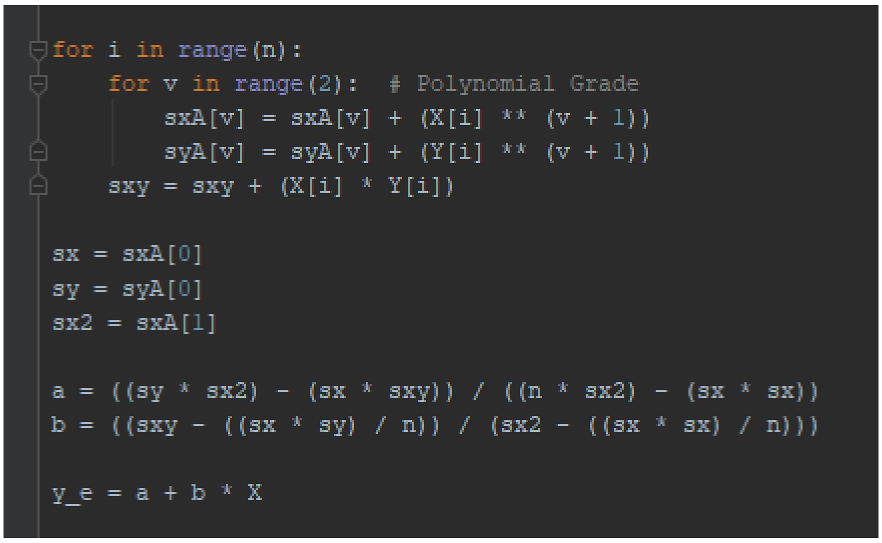
So the first we must know is that python has a library to implement linear regression that makes it easy to implement linear regression, but in this case we want to learn to implement our own code to know the source of linear regression in coding, we are doing this for every other method in regression and we are doing the comparison with the libraries in python, first of all we must call the variables that we are going to use:



```
rl = LinearRegression(n_jobs=100)                # this is for implementation with libraries
data = pd.read_csv('Temperature.csv', header=0)  # this is the data set of example
x = data['Year']                                 # this is the data of X
y = data['Temperature']                          # this is the data of X
X = x[:, np.newaxis]                             # this is the data of X added to an array
Y = y[:, np.newaxis]                             # this is the data of X added to an array
n = len(X)                                       # this is the length of the Data, how many points
sxA = [0] * len(X)                               # this is just an array to save the results
syA = [0] * len(X)                               # this is just an array to save the results
sx = 0                                           # this is the summation of x
sy = 0                                           # this is the summation of y
sx2 = 0                                          # this is the summation of x^2
sxy = 0                                          # this is the summation of x multiplied by y
rl.fit(X, Y)                                     # this is for implementation with libraries
m = rl.coef_[0]                                  # this is for implementation with libraries
b2 = rl.intercept_                               # this is for implementation with libraries
```

*Code 1*

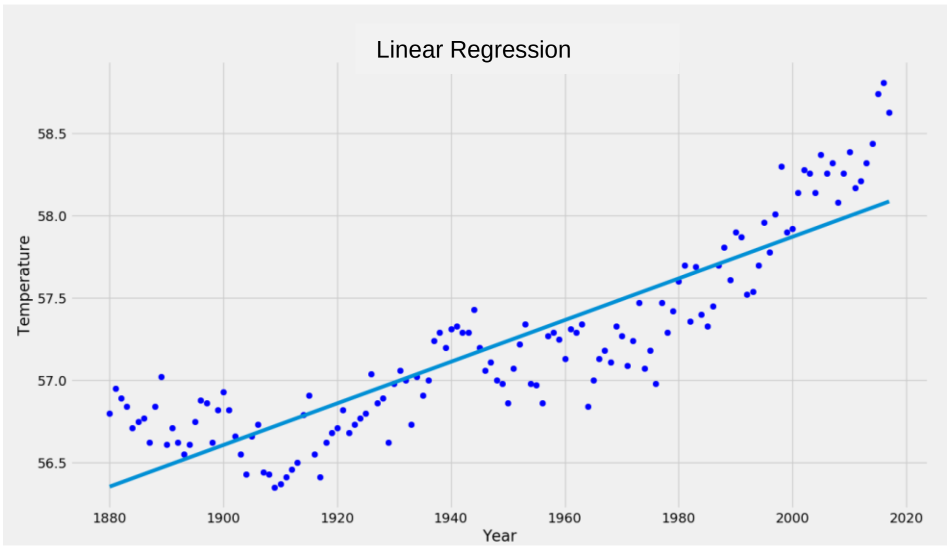
The next step is to implement the Equations in code, its simple, it just need a *for* to make the summation, in a future step we don't even need the for, we can use a simple sum command in python to make the summation, in the next image we can see the implementation of the code:



```
for i in range(n):
    for v in range(2):  # Polynomial Grade
        sxA[v] = sxA[v] + (X[i] ** (v + 1))
        syA[v] = syA[v] + (Y[i] ** (v + 1))
    sxy = sxy + (X[i] * Y[i])

sx = sxA[0]
sy = syA[0]
sx2 = sxA[1]

a = ((sy * sx2) - (sx * sxy)) / ((n * sx2) - (sx * sx))
b = ((sxy - ((sx * sy) / n)) / (sx2 - ((sx * sx) / n)))

y_e = a + b * X
```

*Code 2*

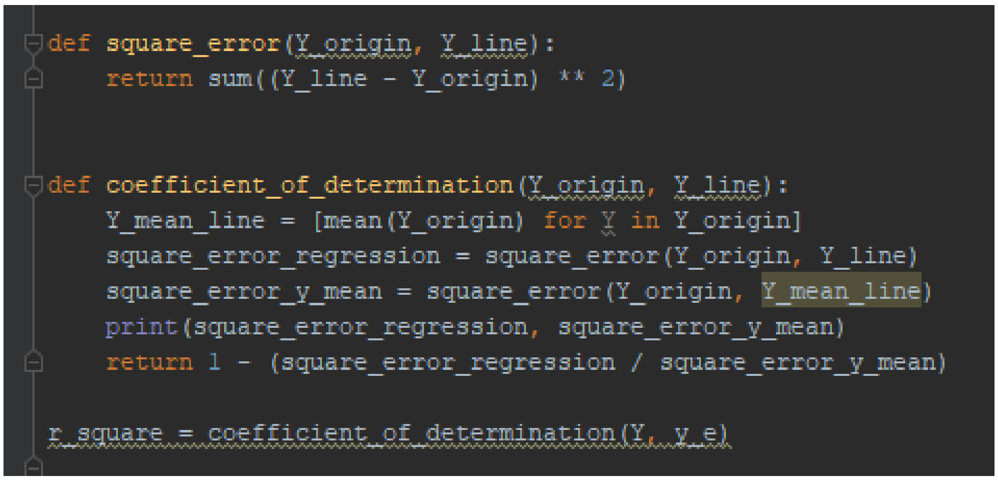
Here we use the for to make the summation, the we implement simple arithmetic get the result of Y who is represented as y_e, linear regression is simple to make but is important to understand it we want to advance furthermore in regression. If we use a graphics tool or library to represent the results we get, we have this:



*Graph 1*

If we apply the algorithm via libraries of python, we get the same exact result, we can see in *variables 1* that we have some variables to implement with libraries like m and b2, this are the slope and the intercept respectively, we just add those to the equation and we get the same exact result.

When we get this result we can see that the line is centered in the set of data but how do we know how good is that line, for this we use what we see in the theory that is the r_squared, to implement that equation in python we can use a library to simple calculate the r square with the data, but as we want to learn how it work we can code it pretty simple like this:


```
def square_error(Y_origin, Y_line):
    return sum((Y_line - Y_origin) ** 2)


def coefficient_of_determination(Y_origin, Y_line):
    Y_mean_line = [mean(Y_origin) for Y in Y_origin]
    square_error_regression = square_error(Y_origin, Y_line)
    square_error_y_mean = square_error(Y_origin, Y_mean_line)
    print(square_error_regression, square_error_y_mean)
    return 1 - (square_error_regression / square_error_y_mean)

r_square = coefficient_of_determination(Y, y_e)
```

*Code 3*

In here we can see a summation but without a **for**, jut using a commando of python to apply the summation, in the method coefficient of determination we past it a Y_origin who I the set of data of Y and a Y_line who is the result of the best fit line, the result of this operation give us this: **R^2:** 0.7730494175750082

We can see is close to 1 and that the line we have is a really good fit for that set of data, if we analyze result of r_squared applying the methods in the libraries of python we get that: **R^2:** 0.7730494175750079

Is almost like the one we get, that means we coded right the methods of linear regression, we can apply the method here to any set of data with n points, and it will work as good as it works with 3 points or 3000 points, but sometimes we get set of data which the best method is not linear regression because the points are very scattered or the points have an exponential growth, in this cases we use quadratic regression.

## 3.2 Quadratic Regression

To do quadratic regression we need to see the equation of slope in a different way, we always knew the equation was Y = mX + b and we know what it means in every aspect with the last point of linear regression, to do quadratic regression we are going to change the equation to Y = B0 + B1X, is basically the similar equation but with different names for the variables, B0 is equal to the intercept of Y, and B1 is equals to the slope in X, with this different equation we can understand quadratic regression that is: Y = B0 + B1X +B2X^2, the change here is that we add two more variables that is B2(slope in X2) and

X^2 that is the X with exponent 2 to form a quadratic equation, this give us a curve line instead a straight line like in linear regression.

### 3.2.1 Theory

So how do we find the value of all the Bs to get Y, this is not the same as we did in linear regression, we must understand matrixes, because we get a system of equations, to find the values of Bs we apply this formula for quadratic regression:

$$\begin{bmatrix} n & \sum x_i & \sum {x_i}^2 \\ \sum x_i & \sum {x_i}^2 & \sum {x_i}^3 \\ \sum {x_i}^2 & \sum {x_i}^3 & \sum {x_i}^4 \end{bmatrix} \begin{bmatrix} \mathrm{B0} \\ \mathrm{B1} \\ \mathrm{B2} \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum {x_i}^2 y_i \end{bmatrix}$$

*Equation 6*

In here we can see a system of equations, we have all the variables of X and Y, we need to find the result for the Bs, the way we can do this is with is a simple matrix arithmetic and Gaussian elimination to find the triangular top matrix, we create a matrix 3*4 in which the 3 by 3 is the first part of the equation and the vector we add is the result we find in this equation, the following matrix

$$\begin{matrix} \mathrm{B0}\,n & \mathrm{B1}\sum x_i & \mathrm{B2}\sum {x_i}^2 & \sum y_i \\ \mathrm{B0}\sum x_i & \mathrm{B1}\sum {x_i}^2 & \mathrm{B2}\sum {x_i}^3 & \sum x_i y_i \\ \mathrm{B0}\sum {x_i}^2 & \mathrm{B1}\sum {x_i}^3 & \mathrm{B2}\sum {x_i}^4 & \sum {x_i}^2 y_i \end{matrix}$$

*Equation 7*

We just need to transform this matrix to a triangular top to find the values of Bs, with this we can find first B2 and next B1 one and so on, in the next images we show an example of how to do it,

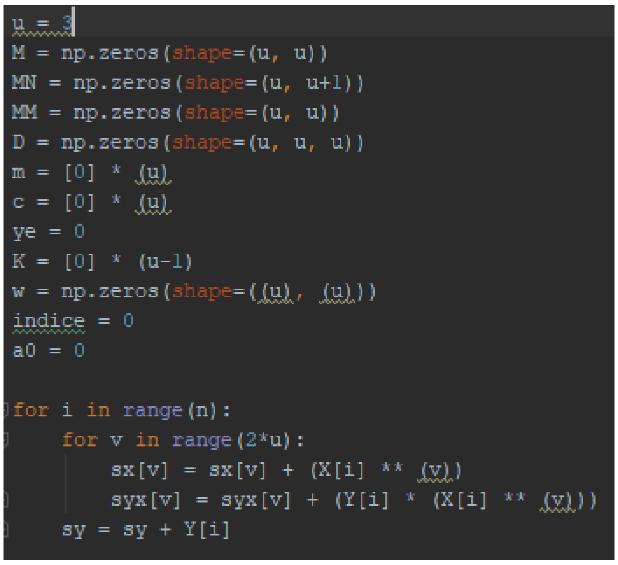
$$\begin{matrix} \mathrm{B0}\,n^{\mathrm{R}} & \mathrm{B1}\overset{\mathrm{R}}{\sum} x_i & \mathrm{B2}\overset{\mathrm{R}}{\sum} {x_i}^2 & \overset{\mathrm{R}}{\sum} y_i \\ 0 & \mathrm{B1}\overset{\mathrm{R}}{\sum} {x_i}^2 & \mathrm{B2}\overset{\mathrm{R}}{\sum} {x_i}^3 & \overset{\mathrm{R}}{\sum} x_i y_i \\ 0 & 0 & \mathrm{B2}\overset{\mathrm{R}}{\sum} {x_i}^4 & \overset{\mathrm{R}}{\sum} {x_i}^2 y_i \end{matrix}$$

*Equation 8*

We apply gauss elimination to create the triangular top, with this we can just pass the changed summation of X4 to divide the summation of X^2*Y and with that we get the value of B2, next we multiply B2 in the second file to find B1 and so on, that is how we get the Bs to calculate the equation to find Y = B0 + B1X1 + B2X^2, now with that equation we find the best fit line for quadratic regression.

### 3.2.2 Implementation in code

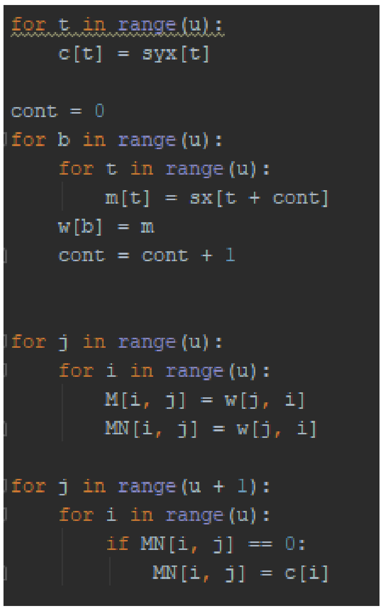
To implement quadratic regression in code we must first assign the variables and apply the summations, we have many similar variables that in linear regression, but we add some more to apply the functions, for example the u is for the grade of the polynomial function, in this case is 3 because its quadratic, in polynomial we are going to see how this affect the equation if we increase the number, but for now it stays in 3, another variable we must add are the matrixes M and MN, M is the matrix of summation in X, and MN is the matrix 3*4 that we add the vector of the summation with Y*X^n:



```
u = 3
M = np.zeros(shape=(u, u))
MN = np.zeros(shape=(u, u+1))
MM = np.zeros(shape=(u, u))
D = np.zeros(shape=(u, u, u))
m = [0] * (u)
c = [0] * (u)
ye = 0
K = [0] * (u-1)
w = np.zeros(shape=((u), (u)))
indice = 0
a0 = 0

for i in range(n):
    for v in range(2*u):
        sx[v] = sx[v] + (X[i] ** (v))
        syx[v] = syx[v] + (Y[i] * (X[i] ** (v)))
    sy = sy + Y[i]
```

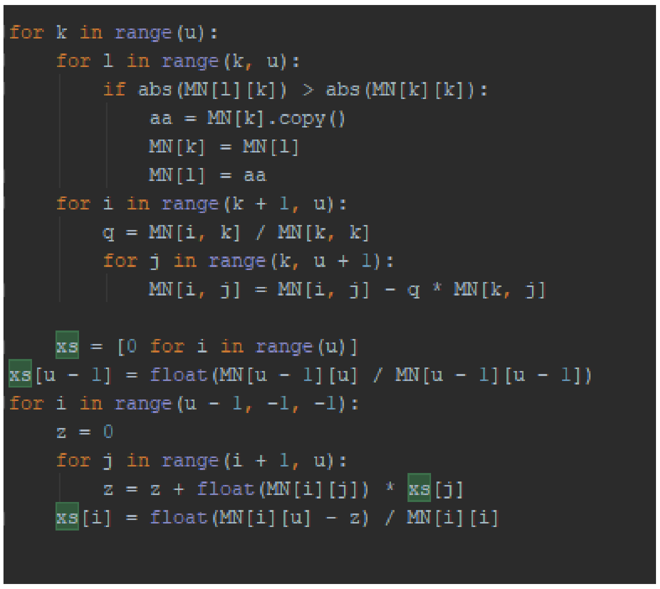
Code 4

In the last code we can see that we in the for apply a similar summation than in linear regression to get the summations of X, Y and Y*X, next we add the results of the summations of Y and Y*X^n to an empty vector c of 3 elements, after that we create a temporal matrix w to assign the n points and the summations of X, after that we add the values to the Matrix M and MN, the matrix MN have now all the values needed, after that we apply gaussian elimination with pivoting in MN and we get the result matrix *code 4* we this matrix triangular top we just apply the method of passing variables and get all the Bs.



```
for t in range(u):
    c[t] = syx[t]

cont = 0
for b in range(u):
    for t in range(u):
        m[t] = sx[t + cont]
    w[b] = m
    cont = cont + 1


for j in range(u):
    for i in range(u):
        M[i, j] = w[j, i]
        MN[i, j] = w[j, i]

for j in range(u + 1):
    for i in range(u):
        if MN[i, j] == 0:
            MN[i, j] = c[i]
```

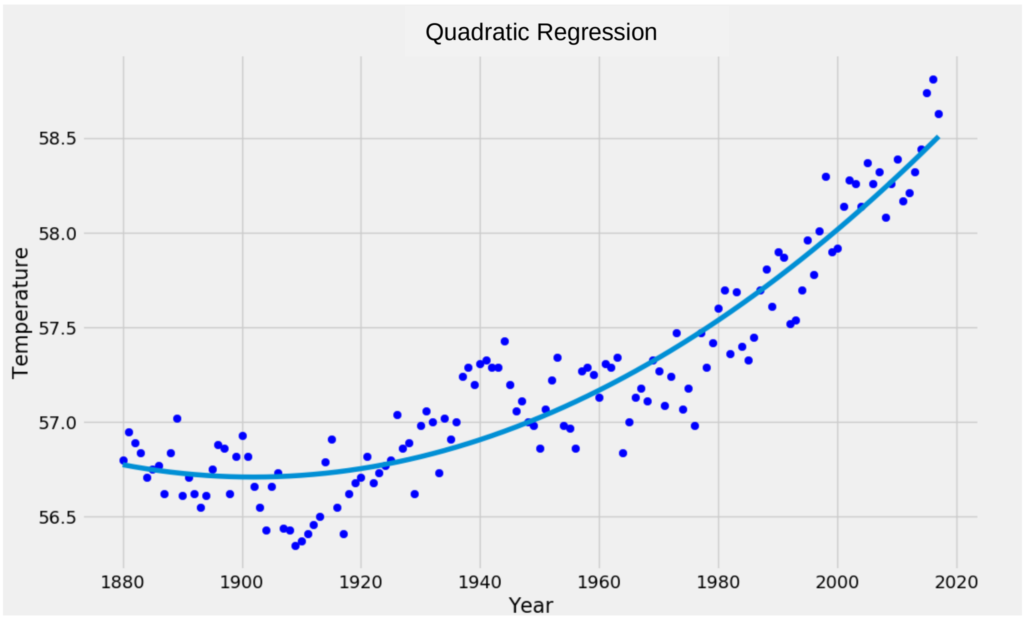
*Code 6*



```
for k in range(u):
    for l in range(k, u):
        if abs(MN[l][k]) > abs(MN[k][k]):
            aa = MN[k].copy()
            MN[k] = MN[l]
            MN[l] = aa
    for i in range(k + 1, u):
        q = MN[i, k] / MN[k, k]
        for j in range(k, u + 1):
            MN[i, j] = MN[i, j] - q * MN[k, j]

    xs = [0 for i in range(u)]
xs[u - 1] = float(MN[u - 1][u] / MN[u - 1][u - 1])
for i in range(u - 1, -1, -1):
    z = 0
    for j in range(i + 1, u):
        z = z + float(MN[i][j]) * xs[j]
    xs[i] = float(MN[i][u] - z) / MN[i][i]
```

*Code 5*

After this the final step is to add the Bs to the equation Y = B0 + B1X1 + B2X^2 and we get the Y to the best fit line in quadratic form:



*Graph 2*

In *Graph 2* we can see the best fit line in quadratic form, if we calculate the R^2 same as we did with linear regression, we get that,

R^2: 0.8859750225449559

With this value of R_squared we can se clear that the value of R in quadratic is much better for this set of data that in linear regression.

## 3.3 Polynomial regression

### 3.3.1 Theory

Polynomial regression is basically the same as quadratic regression, the only change is that we work with bigger matrixes and vectors, in quadratic regression we use the variable u which in that case was 3 because is was quadratic, we can associate the 3 in the u with the variables we need, B0, B1 and B2, in polynomial regression the things change in that we don't use u: 3, we use u: 3 <= u <=n-1, n is the number of points that we have, is minus 1 because the polynomial equation can't have the same grade as the number of point, for example if we use u: 4 is a cubic regression, if we use 5 is a polynomial grade 4, and so, the equation to get a polynomial regression is for example a polynomial grade

4:

$$\begin{pmatrix} n & \sum x_i & \sum {x_i}^2 & \sum {x_i}^3 & \sum {x_i}^4 \\ \sum x_i & \sum {x_i}^2 & \sum {x_i}^3 & \sum {x_i}^4 & \sum {x_i}^5 \\ \sum {x_i}^2 & \sum {x_i}^3 & \sum {x_i}^4 & \sum {x_i}^5 & \sum {x_i}^6 \\ \sum {x_i}^3 & \sum {x_i}^4 & \sum {x_i}^5 & \sum {x_i}^6 & \sum {x_i}^7 \\ \sum {x_i}^4 & \sum {x_i}^5 & \sum {x_i}^6 & \sum {x_i}^7 & \sum {x_i}^8 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum x_i y_i \\ \sum {x_i}^2 y_i \\ \sum {x_i}^3 y_i \\ \sum {x_i}^4 y_i \end{pmatrix}$$

*Equation 9*

We can see that the matrix in comparison with quadratic or in other names polynomial grade 2, is the same, the major change is the order of the matrix, in grade 2 it was 3*3 and with the vector it was 3*4, in this case grade 4 is 5*5 and adding the vector with the results in Y is 5*6, in cubic regression it would be 4*5 and in grade 5 it would be 6*7 and so on, we can define a function in this case, for a polynomial grade n we can use a matrix order n+1*n+2, when we get the matrix doesn't matter the grade we use the same theory and code as quadratic, create the matrix, and apply gaussian elimination to find the Bs. The main equation for polynomial regression is:

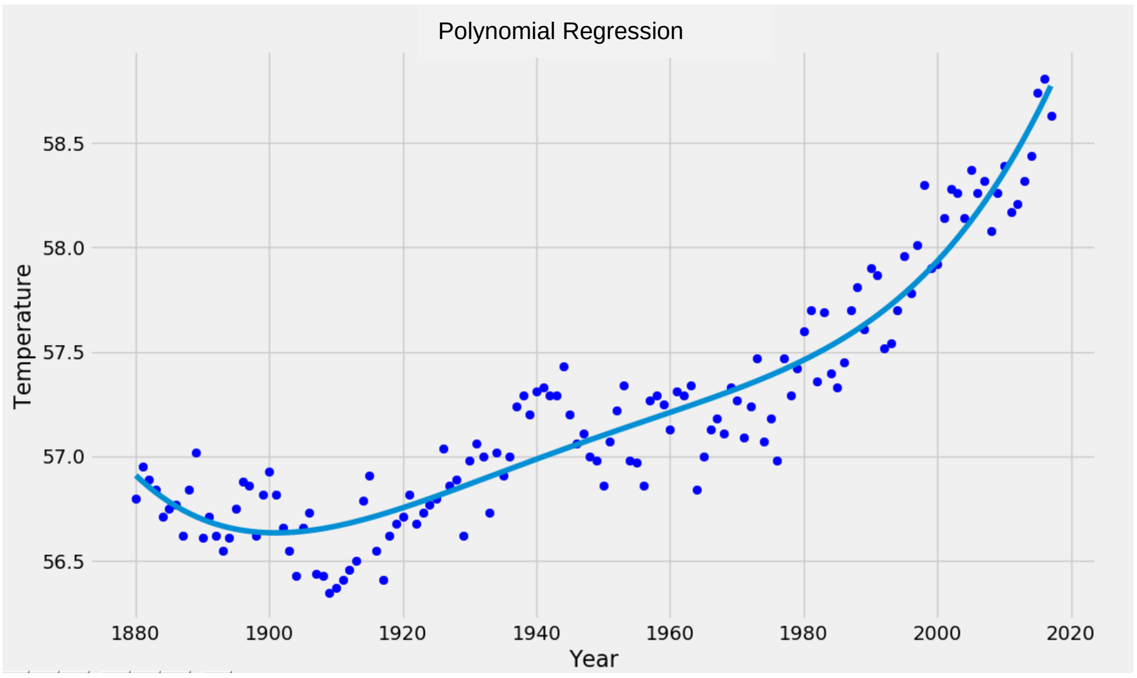
$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 ... a_p x^p, p < n$$

*Equation 10*

Let's change the a in the matrix and in the equation for Bs, if we do that, we can see that the equation is the same as quadratic regression and linear, the only variation is that we add B3X^3 + B4X^4 + … + BnX^n, with this we comprehend regression at its fullest.

### 3.3.2 Implementation in code

The implementation in code is exactly the same as quadratic regression, the code in quadratic regression works exactly the same as in quadratic regression, we just adjust the u, and the result of a bigger u in the same data set, for example a variable u of 4 for cubic regression, the matrix would be as we say before 4*5 and the result line would be:



*Graph 3*

We can see a little variation in the best fit line in this graph, but the biggest change we can see it in the R_squared value, for example for quadratic regression it was:

**R^2**: 0.8859750225449559

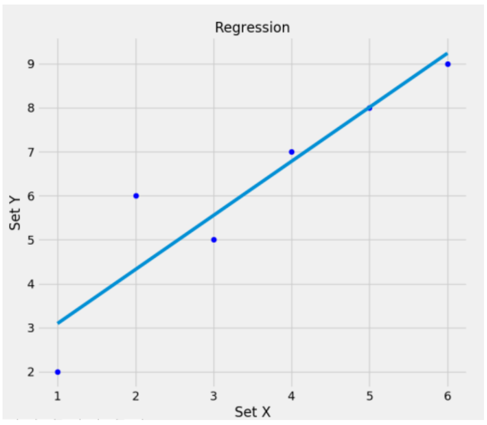
And in this, in a polynomial equation of grade 4 is:

**R^2:** 0.9035002141528568

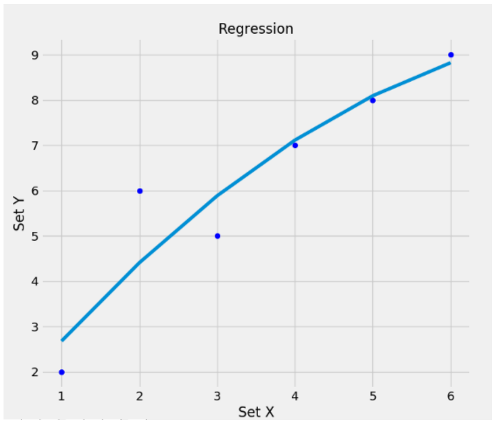
As we can see here the value of R is best for the grade 4, we can use polynomial regression for better precision in a set of data with many variations.

## 4 Interpolation

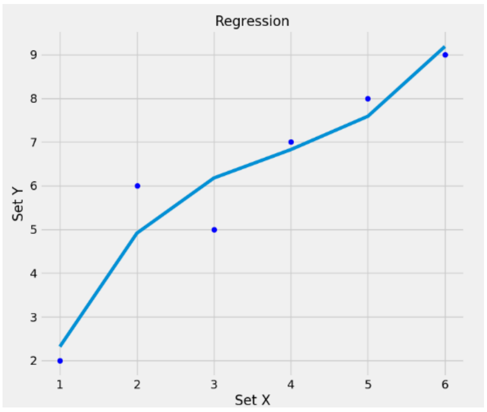
We are covering interpolation here because interpolation can be taken as a fundamental part in regression, it is not used much by machine learning but have a connection with regression, so let's take por example a dataset with few points (it also works well with larger datasets but for better understanding we are going to use a dataset of 6 points),
So, let's apply the methods last seen in this data set, the result we got is this:



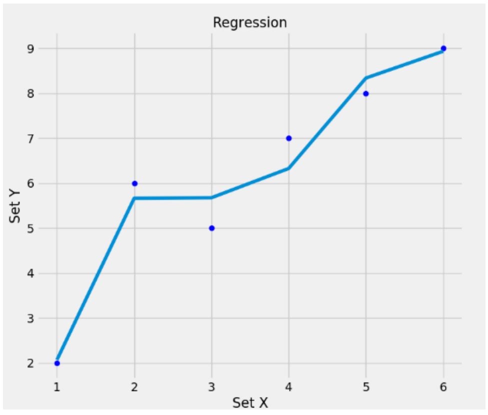
Graph 4 – Linear Regression



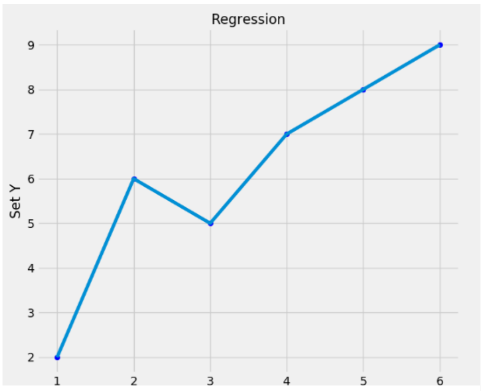
Graph 5 – Quadratic Regression



Graph 7 – Cubic Regression



Graph 6 – Polynomial Regression, Grade 4

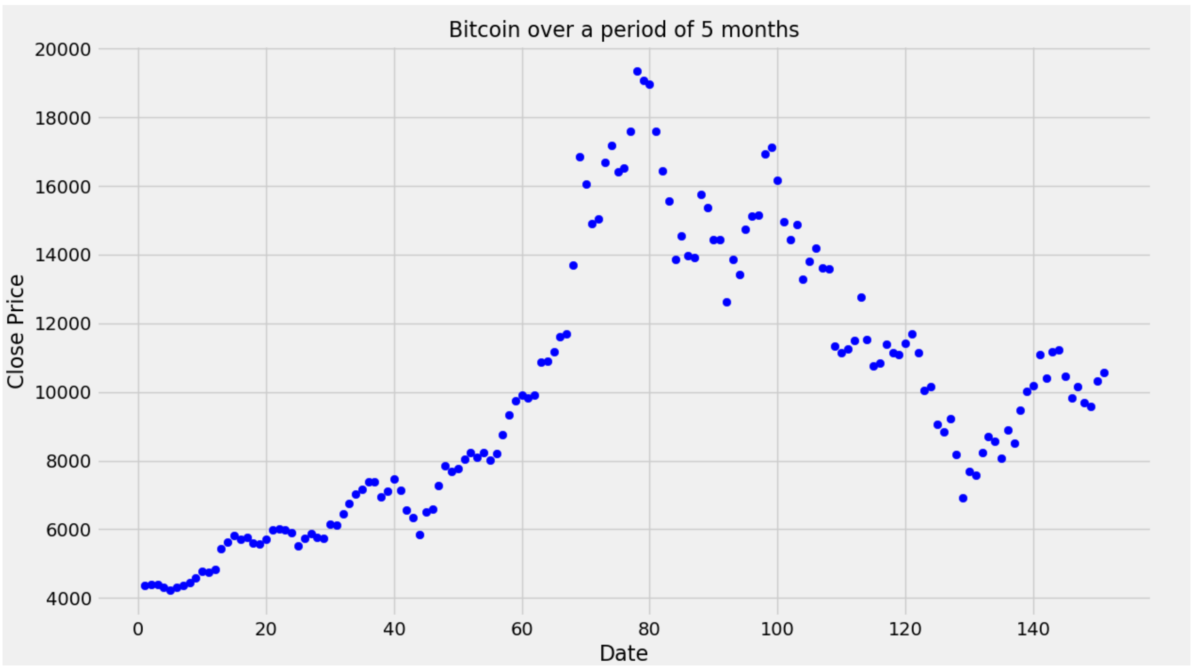


Graph 8 – Polynomial Regression, Grade 5

As we can see if we increase the grade the line gets closer and closer to all the points, in the graph 8 we can see that the line touches every point, we can call this interpolation, this is the line that interpolates every point and the equation of this line is:

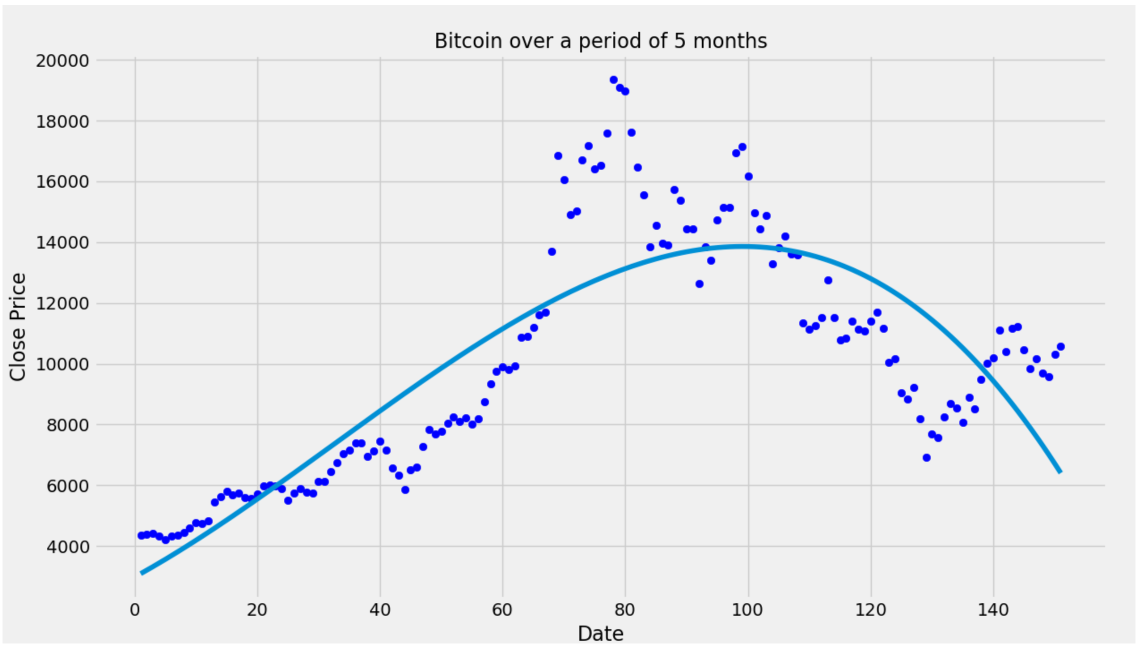
Y = B0 + B1X + B2X^2 + B3X^3 + B4X^4 + B5X^5

Here we can see the grade for an interpolation line is equals the number of points – 1, or better say n-1, we can do this with larger data set and get the same result, but the error increase and it would work for machine learning well, because if we have an interpolated line we can predict his future behavior and can do anything with this, its much harder and imprecise to extrapolate an interpolated line than a linear regression line, we can apply this to the grade of the line, linear regression it's the better for extrapolate, second is quadratic regression, third is cubic regression and so on, the more grades the line has or the equation has the harder it gets to predict future behavior, that's why linear, quadratic and cubic are the more used in regression models of Machine learning, why don't we always use linear?, because as we say before not every dataset acts the same, some data set change more in time, like this one:



*Graph 9*

We can see here the increase and decrease of the Bitcoin price over a period of 5 months between 9/30/2017 and 2/27/2018, the changing price varies much in this period of time, a linear model would be efficient in this model, a quadratic would be better, but if we use a cubic model it would be the better representation of regression for this kind of model,



*Graph 10*

The best fit line here is a cubic, and we can see it tries to get the regression model as best as he can, we can evaluate the R_squared which is,

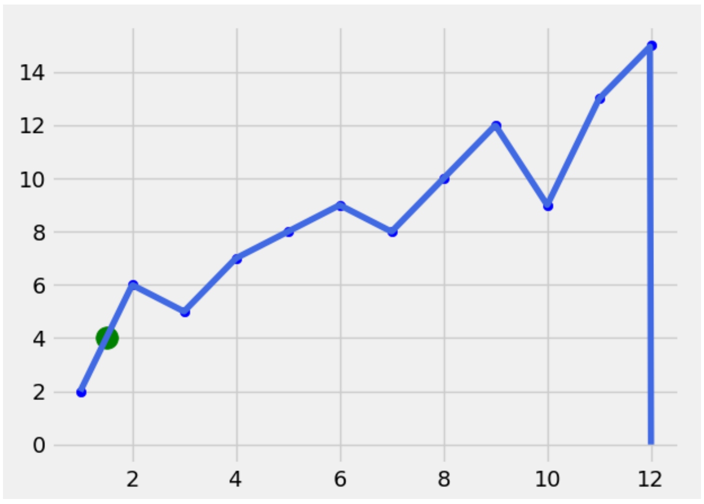
R^2: 0.6923458831556241

It gets a pretty good fit for this set of data that changes much over time, if we apply linear regression we get that the R_squared would be less than 0.5, who is like saying less than 50%, that's not even a good fit line.

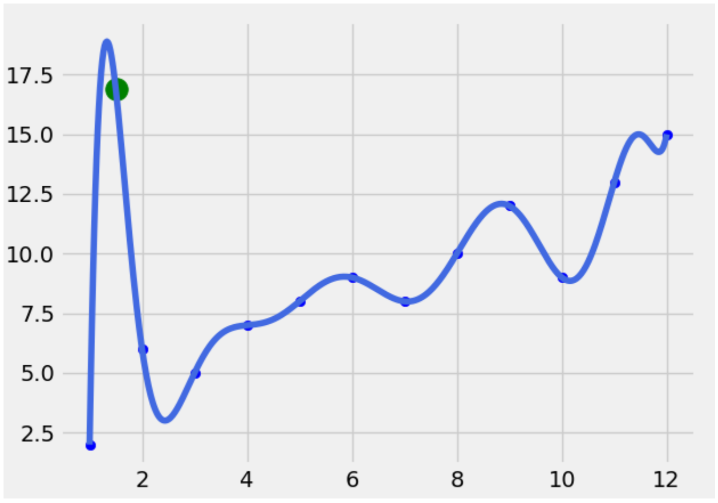
Interpolation is also used to predict future point added in X, if we add a new point in X between the range of X, with the regression line and the R_squared we can see where it fits best in the model, this is another meaning for interpolation, where you put a new point between the range of X. If it is outside X, that is extrapolation.

## 4.1 Newton, Lagrange and Regression

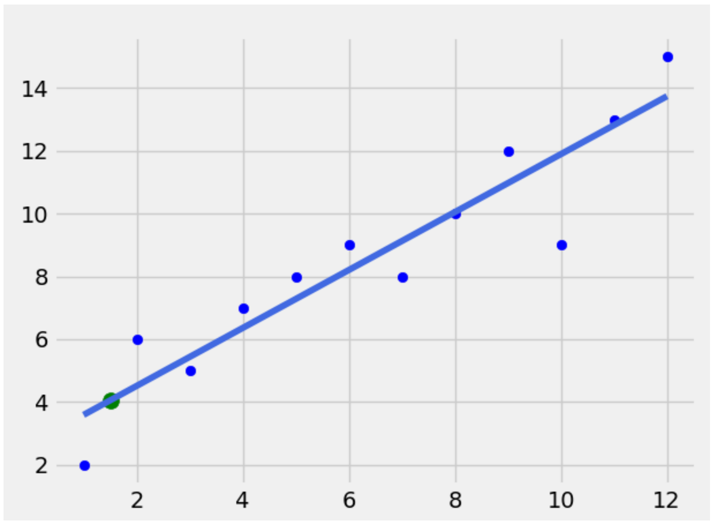
We can compare the two methods of interpolation with regression, Newton with divided differences, Lagrange and linear regression, to show us a new point in space interpolated in all the methods, the results are:



Graph 11 – Newton with divided differences



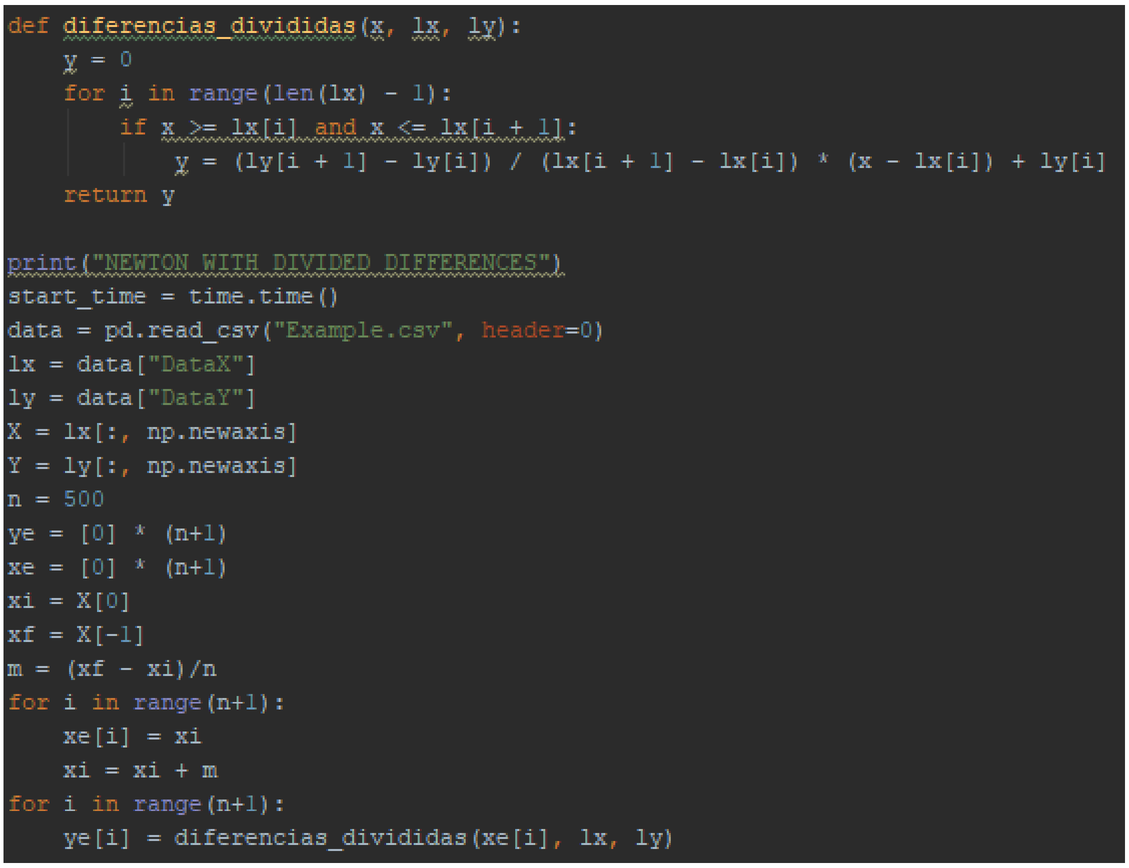
Graph 12 - Lagrange



Graph 13 – Linear Regression

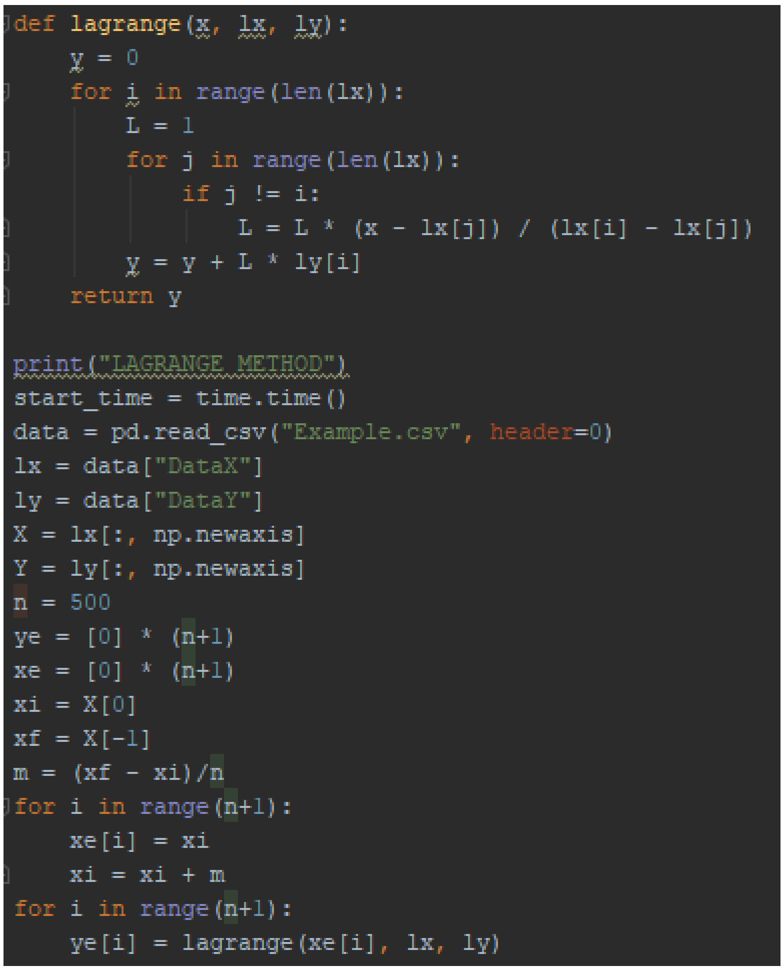
As we can see in the three graphs, the function that creates the line is different for all three methods, even for the two methods that interpolates the points. If we can see the r_squared for all 3, for the 2 interpolating methods is 1 or 100% better say, and for linear regression is 0.86 or 86%, that means if we interpolate a new point that is exactly like the one in X value it would be in the interpolating methods right in the line, and in regression it would be closer but not in the line, but lets add a new point that is not in the set of data X, the point would have different values for every method, we can see the new point that is 1.5 in the graphs as a green point, with Newton it would be 4.0, with Lagrange it would be 16.87 and with linear regression would be 4.05, for a new point between the range of the set of data X, the two methods of interpolation are better than regression and if we could calculate polynomial regression of n-1(n is the number of points) it would be exactly like Lagrange, but for extrapolation the two methods would be completely inefficient, that's why for predictions we use regression for better extrapolation results.

### 4.1.1 Code of Newton and Lagrange



```python
def diferencias_divididas(x, lx, ly):
    y = 0
    for i in range(len(lx) - 1):
        if x >= lx[i] and x <= lx[i + 1]:
            y = (ly[i + 1] - ly[i]) / (lx[i + 1] - lx[i]) * (x - lx[i]) + ly[i]
    return y

print("NEWTON WITH DIVIDED DIFFERENCES")
start_time = time.time()
data = pd.read_csv("Example.csv", header=0)
lx = data["DataX"]
ly = data["DataY"]
X = lx[:, np.newaxis]
Y = ly[:, np.newaxis]
n = 500
ye = [0] * (n+1)
xe = [0] * (n+1)
xi = X[0]
xf = X[-1]
m = (xf - xi)/n
for i in range(n+1):
    xe[i] = xi
    xi = xi + m
for i in range(n+1):
    ye[i] = diferencias_divididas(xe[i], lx, ly)
```

*Code 7 - Newton*



```
def lagrange(x, lx, ly):
    y = 0
    for i in range(len(lx)):
        L = 1
        for j in range(len(lx)):
            if j != i:
                L = L * (x - lx[j]) / (lx[i] - lx[j])
        y = y + L * ly[i]
    return y

print("LAGRANGE METHOD")
start_time = time.time()
data = pd.read_csv("Example.csv", header=0)
lx = data["DataX"]
ly = data["DataY"]
X = lx[:, np.newaxis]
Y = ly[:, np.newaxis]
n = 500
ye = [0] * (n+1)
xe = [0] * (n+1)
xi = X[0]
xf = X[-1]
m = (xf - xi)/n
for i in range(n+1):
    xe[i] = xi
    xi = xi + m
for i in range(n+1):
    ye[i] = lagrange(xe[i], lx, ly)
```

*Code 8 - Lagrange*

## 5 Classification: Fourier

### 5.1 Theory

Before starting to talk about the use we can give the Fourier methods inside machine learning, it is necessary to define the concepts and Fourier investigations. There are many methods that has emerged with the investigations realized by Jean-Baptise Joseph Fourier, a French mathematician and physicist. This person introduced the concepts we know by this day as Fourier Analysis, more specific, Fourier Series. He made an investigation in 1807 by the name of *Mémoire sur la propagation de la chaleur dans les corps solides* (Treatise on the propagation of heat in solid bodies), appointing for the first time the concept Fourier Series, later in 1822 he continued his investigations publishing *Théorie analytique de la chaleur* (Analytical theory of heat), the objective of this investigation consisted in solving the heat equation in a metal plate. With this research, the interesting fact is that an arbitrary function can be represented by trigonometric series. It is known that the first studies and researches found in the entire history about decomposing a periodic function in a sum of oscillated functions goes back to the 3rd century B.C. where mane astronomers proposed an empiric model for the investigation and study of the planetary rotations.

Fourier research has been considered a bit informal, due to the lack of precision we could have with the calculations of functions and integrals by the time. Later, the mathematicians Bernhard Riemann and Peter Lejeune Dirichlet where the one that gave Fourier results more formality and precision. Although in a beginning the investigation consisted about solving the heat equation, with the course of time many professionals realized that these techniques could be used in a variety of math and physics problems, especially for the linear differential equation with constant coefficients, where the eigensolutions are the sinusoids. An eigensolution is any of the results of the calculation of eigenvalues. Which in linear algebra, an eigenvector or characteristic vector of a linear transformation is a non-zero vector that changes by only a scalar factor when that linear transformation is applied to it.

After many researches carried out by other mathematicians and physicist, and the technological evolution that was achieved over time, there emerged plenty of methods and algorithms based on Fourier's studies, which have a great applicability in machine learning.

The most common is Fast Fourier Transform (FFT), but first it is necessary to clarify what a Fourier Transform (FT) is. The FT method decompose a function of time (signals) in frequencies, the FT of a function of time is itself a frequency function of complex values in which the absolute value represents the amount of frequency present in the original function, and the complex values are a corresponding phase of the sinusoid in the frequency. The Fourier Transform of a function is denoted by adding a circumflex $\hat{f}$, and we will use the following equation of an integrable function:

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i x \xi}\, dx,$$

*Equation 11*

for any real number ξ. Where x represents time, and ξ represents frequency.

The FFT algorithm, created by Cooley and Tukey, consists that it gets a sampling of signal in a period of time and it divides it into its frequency components, these components are single sinusoidal oscillations at different frequencies each with their own amplitude. This algorithm is considered by the Professor Gilbert Strang from MIT as "the most important numerical algorithm of our lifetime", it is used daily with signal processing, audio and video processing, medical imaging, AND sequences, GPS and many more. An FFT computes the Discrete Fourier Transform (DFT) exactly in the same way, but much faster. The DFT uses 'x' as complex numbers and is defined by the formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \qquad k = 0, \ldots, N-1.$$

*Equation 12*

Evaluating the DFT requires O ($N^2$) operations. An FFT is any method computing the same results in O ($N$ log $N$) operations and this is what it consists the FFT algorithm.

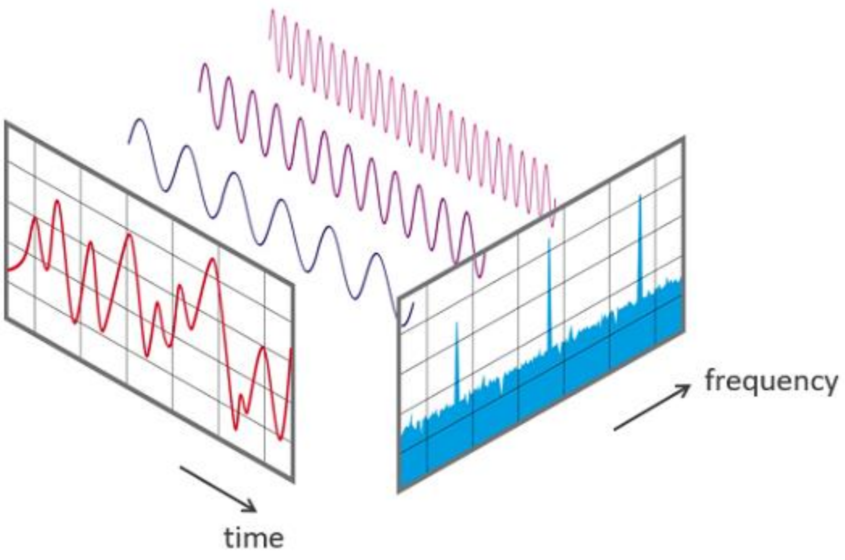


*Figure 1 - View of a signal in the time and frequency domain. (FFT)*

Having the knowledge of this, we will analyze how we can use this for our benefit in machine learning. Basically, the function that adapts better with this definition to the machine learning is the signal processing, and with this algorithm it is possible to achieve this objective. To use better a portion of signal data, oscillations such as a sound or vibration, it is advisable to use FFT and consider the peaks in those frequency coefficients as inputs for a neural network, or some related algorithm to use this data. There are more complex methods and algorithms related to Fourier that can be used in machine learning, as Discrete-Time Fourier Transform (DFTF), Fractional Fourier Transform (FRFT) and the most recent, Sparse Fourier Transform (SPT).

The FFT is the easiest to understand, explain and implement, since the results are easily related physically, although it rarely provides an optimal set of values according to the problem to be covered, and often it does give a blur image of the information of time that could be very useful in the future with other data. To provide a better description of its functionality, it compares the signals made by the FFT and the data known in a project, and how it can collect data, make comparisons and increase the accuracy with the results combining these factors together. One of the most important bases in machine learning, is the self-learning a project can perform with the amount of data it receives, even more knowing the most exhaustive and important part of a project like this is about collecting data.

Though it may be difficult to work with directly, raw, fully-detailed, time-domain data; input collected is extremely valuable to improve the collection of data, since we have to make the program learn by itself and collect more data as time progress. We need to start collecting of what it is called as “Rich Data”, this means what it is a time waveform and a high resolution FFT. It contains a big amount of analytic information to detect anomalies which brings a big help with detailed information, meaning a signal may not be vibrating more, but it will be vibrating different. The more information contains a signal, more variations or anomalies will be found in them, more accurate will it be and can be useful in the future. Successful real-world machine learning is an exercise in overcoming variation with data. This variation can be related both of what it is trying to detect and the background line the noise, different environments and conditions, as well as to the collection equipment (sensors). It is better to minimize any unnecessary variation, usually is the easiest to control or eliminate to make sure the data captured gets as much of the likely real-world target variation in as many different backgrounds as possible because it will be able to make accurate predictions.

Machine learning works better as an iterative process, so collecting enough data is important to build models that will be effective on the technique, even though for the full range of variation expected from the real world, and use these results to fine tune the approach, then take the next data and test it to get an accuracy benchmark, and do this over and over again to improve the contained data to get the best possible performance.

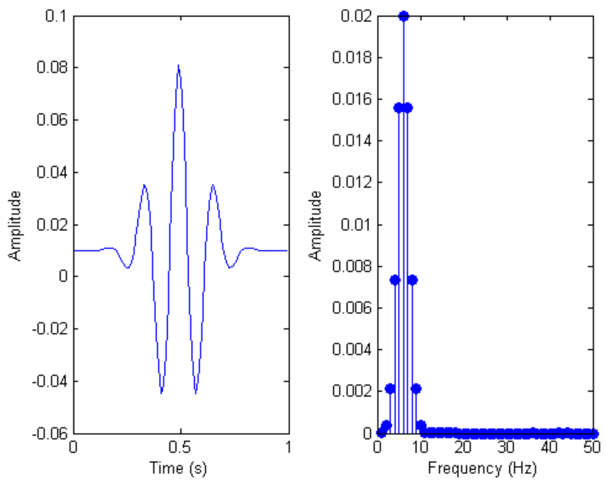


*Figure 2 - A signal before and after FFT analysis*

The perfect example of FFT analysis in machine learning is the use it has in the application Shazam, this works as a recording device that mimics the process a human has when hearing a sound, a vibration comes to our ears, it moves small bones that transforms the vibration further to little hair cells deep in our ear and then produce electrical impulses, which are transmitted to our brain. So, this application uses the pressure of a sound wave to convert it into electrical signal. This signal is not so useful in the digital world, so we need to find something that process it to translate it, so it can be stored digitally, there is where an analog to digital converter is needed, it performs many conversions on small pieces of the signal, this process is known as sampling.

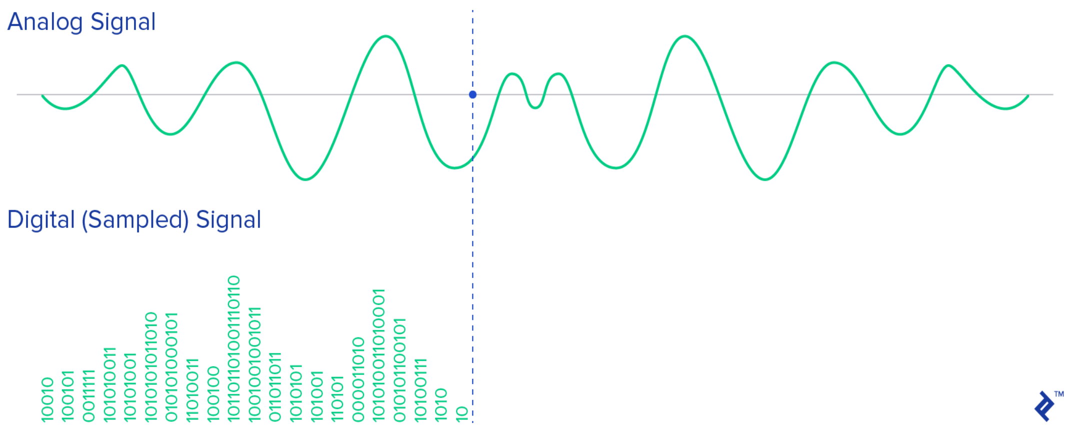


*Figure 3 – Sampling*

Then, it captures the sound or signal, but it is recorded in the time domain signal, which represents the amplitude change of the signal over time, so there is where we use Fourier series to represent any time domain signal by giving a set of frequencies, amplitudes, and phases corresponding to each sinusoid that makes the signal. This representation is the frequency domain, that it is known as a type of audio or signal fingerprint for the time domain signal. After analyzing a signal, we can study the spectrum and determine which frequencies are missing or not. So here is where we use FFT to convert the signal from the time domain to the frequency domain.

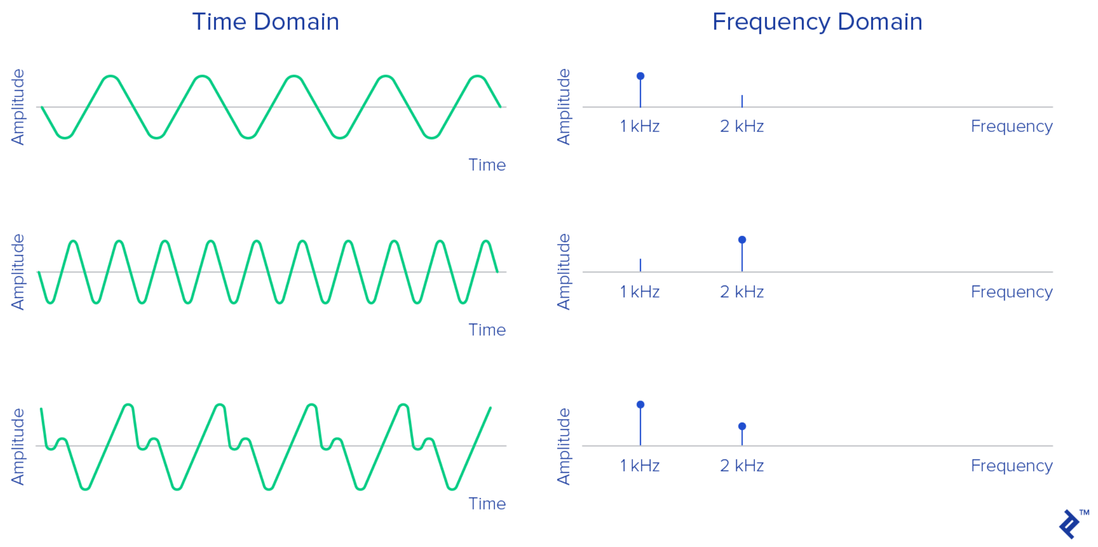


*Figure 4 - Time domain – Frequency domain comparison*

Even though it is a three-minute song we are recognizing, it only needs a small number of spaced samples of sinusoids to determine which song is. It begins to search on the database for matching samples. As it happens many of the songs are alike, but the application is constantly learning about the variations a signal has with the specified song that it became accurate giving a precise result. But it exists the possibility that it will not find any match, either because there is a lot of noise that will give some errors, or because it’s not recognizing clearly the signals.

## 6 Genetic Algorithms

### 6.1 Theory

To define what does consists a genetic algorithm we will begin by taking the definition of “algorithm” from The American Heritage®, as it says: “A step-by-step problem-solving procedure, especially an established, recursive computational procedure for solving a problem in a finite number of steps." Thus, we can summarize this definition in no more than a set of steps to do a task. With this, we can move on to the topic that makes this type of algorithm magical and is its evolution. Let's take as an example a living being, which has as basic characteristics: being born, evolving and reproducing, so a genetic algorithm is a heuristic searching algorithm that is based on the ideas of Darwin's evolution of the survival of the fittest.

Continuing with this it is worth mentioning that this genetic algorithm is not a new topic in the modern world. It has been studied since 1970, initially by John Holland, a psychology and electrical energy Professor of the University of Michigan, he found great potential in this related to the resource optimization issues. One of the first examples of its application was the solution given to famous Knapsack problem* but as time has passed humanity has realized that you can go beyond basic problems. One of the best examples of its application in today's life is the planning of the most efficient route for delivery routes, based on the resolution of optimization problems, and in this case as an example is the problem of the traveling agent (Traveling Salesman problem). So, a company that distributes merchandise can find the best route dynamically while making their deliveries. This just mentioning one of the many applications that it has. In addition, involves in general everything that has to do with designing, as these algorithms help reduce the conceptualization time for the construction of any piece (Vehicles, hardware, chemicals, etc.) because with the use of genetic algorithms you can define which materials are better (more durable, cheaper, etc.) for the construction of any product.

Now, we realize that there may be an intersection between this type of algorithms, neural networks and artificial intelligence in general, but they should not be confused since it is worth remembering that AI seeks to learn from data that were collected previously and that what is intended, to find a pattern to prepare and avoid a failure or find the right way to do it. On the other hand, genetic algorithm seeks the more efficient and effective way of doing a certain task; having a series of variables, it seeks for the best way to combine them to give the best and most appropriate solution. Although the world is just beginning to apply artificial intelligence, one day we could combine these two issues to find the best route within a neural network.

# 7 Neuronal Networks

## 7.1 Theory

To understand neuronal networks we must comprehend first that neuronal networks are not just a method to solve a problem, it is a series of algorithms that works as a system to be able to solve a problem without the necessity of human interaction, is a vast world full of variables and functions with great significant everyone of them, first lets start breaking down the 2 words, neuronal or neuron we can explain it as a biological neuron in which we pass several inputs to the neuron and it give us several outpost with a variation function inside the neuron, we can draw it like this:

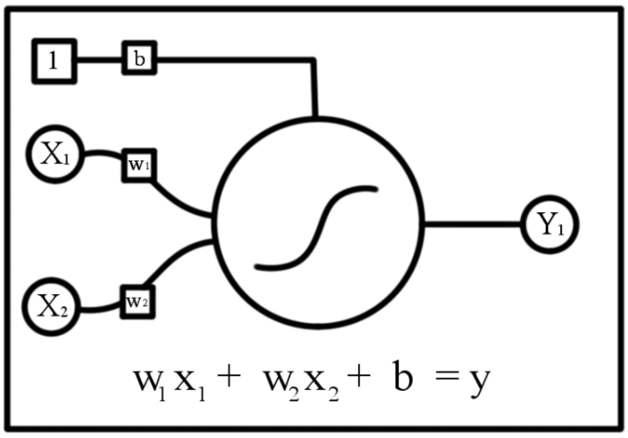


*Figure 5*

In here we can see different many variables and symbols, but let's take the familiars part first, the equation we can see under the representation of a neuron is similar to something we see before, linear regression, we can see that the equation of linear regression is similar to the one we use in neuronal networks, but why is this? Ok so lets start explaining what every aspect of that Figure is, we can see the Big circle that represents the neuron, with a big "S" like symbol inside, we are going to talk later about that symbol, first lets see the variables, we have normal input variables X and output variables Y, we know those variables, but what is the **w** and the **b**, the w is the weight we gave every variable X, like a percentage we give to ever specific X, and the b is the coefficient of bais, this coefficient help us determine if a weight and a variable are good, we can change the weight and the bais to adjust a better output, but here is the catch, we don't change the weight and the bais, the neuronal networks does this.

We know what this variables are but what output can we get from just one neuron, if we just got one neuron we have a limited set of results just or working for AND or OR gates, we need a system that can resolve a bigger problem, then we add more neurons, with one neuron we can sort out with the linear equation he gave us, but with more neurons we can classify more complex sets of information.

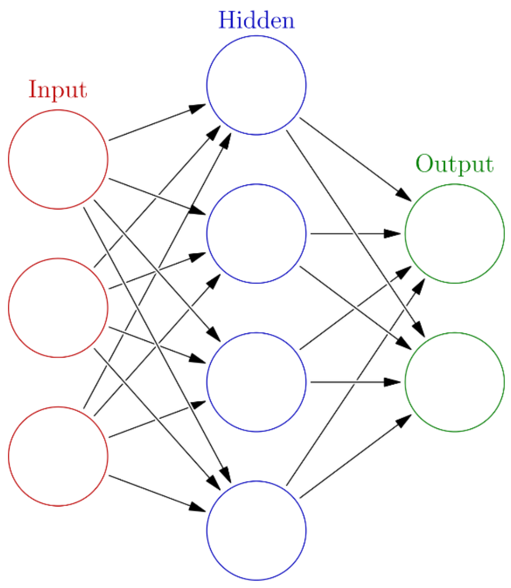


*Figure 6*

Here we can see a neuronal network, with 4 neurons, in this system we just have one layer, the layers are like the name said a layer that's keeps the neurons in an specified position, the input layer are just Xs, the output layer are the Ys, and the hidden in the figure are what we call the hidden layer, there can be many hidden layers, if we have 2 or more we call this deep neuronal networks, the hidden layers jobs is to generalize the variables in the last layer, as we can see here the Xs touch every neuron in the hidden layer and the hidden layer touches every output, this is the process to analyze the every input in every neuron to get the best result, but we have a problem, lets take lake this with just the equation we see before in the *figure 1,* as we can see in this linear equation, if we do the add every equation of every neuron that is similar to this we get a weighted valor with just one equation to solve, we can see this better as if we have the summations of many linear equations we don't get an equation of many linear equations, we get just one balanced equation, to solve this we need to add one more variable to every equation in every neuron, this variable would be sigmoid or as we call it later the "S" inside the neuron, we can work with many other variables besides this, like tanh and relu, but the better is sigma.

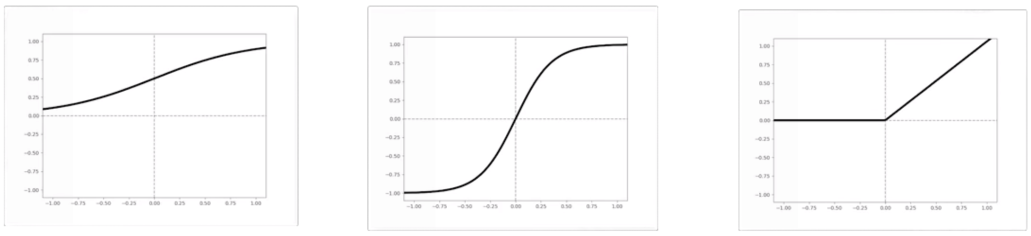


*Figure 7*

In this figure we can se sigma(the first one), tanh(the second) and relu(the third one), sigma is a simple line that works between 0 and 1, this is because the every output given by every neuron is binary, is either a 1 or a 0, we can say like its turn on or off, this variable is the one that modifies every function inside every neuron to make it unique, the output we also get from all of this is a surface with an specified form to calculate in the best way possible the best result, this surface is also showing us the best way possible to calculate the best weight and bais for every function, in other words, we train the system to with some training data, with that the system can start to learn the best way to get a good result, we can call this steps, every time the system runs and gives a result, we evaluate the result with our own result and tells the system if he is doing it good or bad, if its doing bad he does another step and reevaluate the weight and bais, but the system doesn’t do it starting from the beginning again, it implements an algorithm called Backpropagation, what this algorithm does is that we start from the output when the system runs the first time and we go backwards to know where the most significant error is, we also get the error from every other neuron and give it a significant level, wit the surface we get we can calculate the slope, to know where is the general minimal point of this surface, we get an input variable somewhere int the surface and we calculate the gradient of that point to know the slope, and we use the slope to move to the other point, remember we need to find the minimal not the maximal, with every step the programs make we get closer to the minimal point, to calculate to pass of the movement of this we use another variable multiplied by this temporary pint at w, this variable is the longitude of the step, if we set this big we will have problems to find the minimal point, and if we set this value small, we will have to make too many steps to get to the minimal point, and in computing that’s not very efficient, we can see in the next figure that we have a random pint in the surface and with every step calculating the gradient we get closer to minimal point, when the system gets to the minimal point it means that the model already learned to recognize and identify the data to give a correct output.

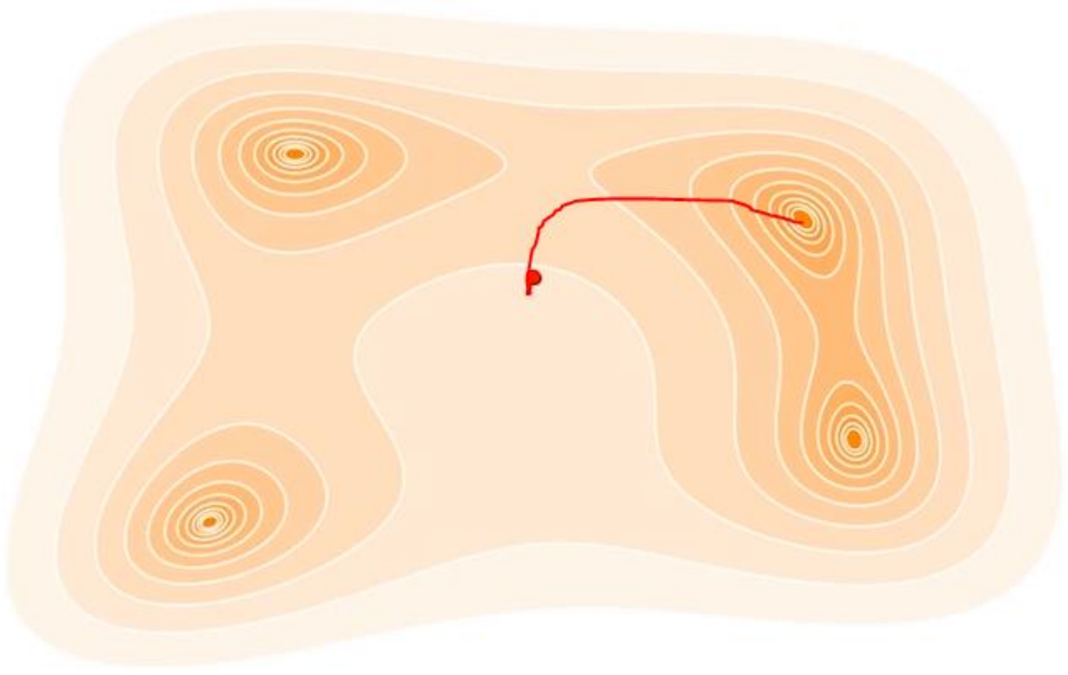


*Figure 8*

A representation of that could be like this:

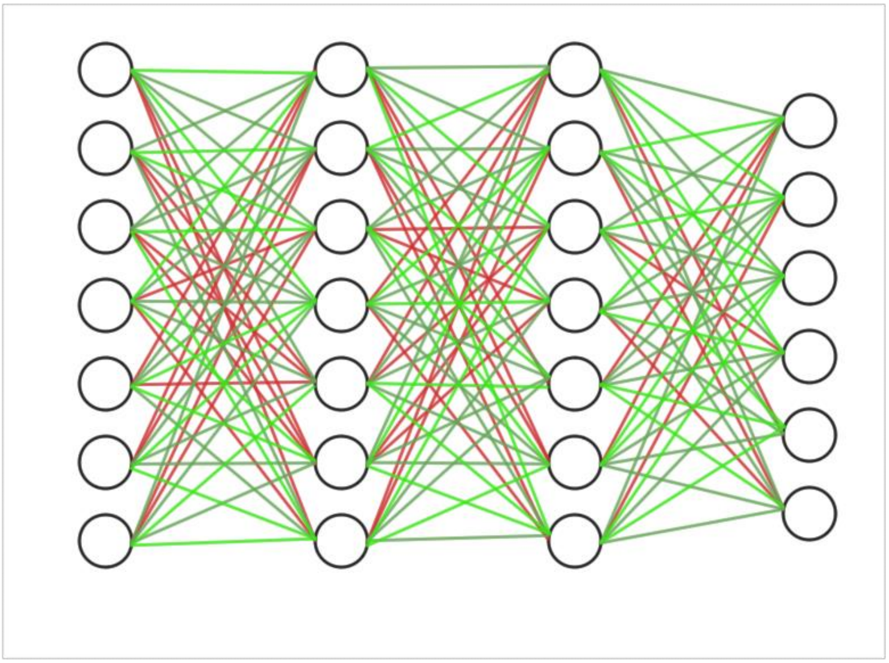


*Figure 9 – Step 1*

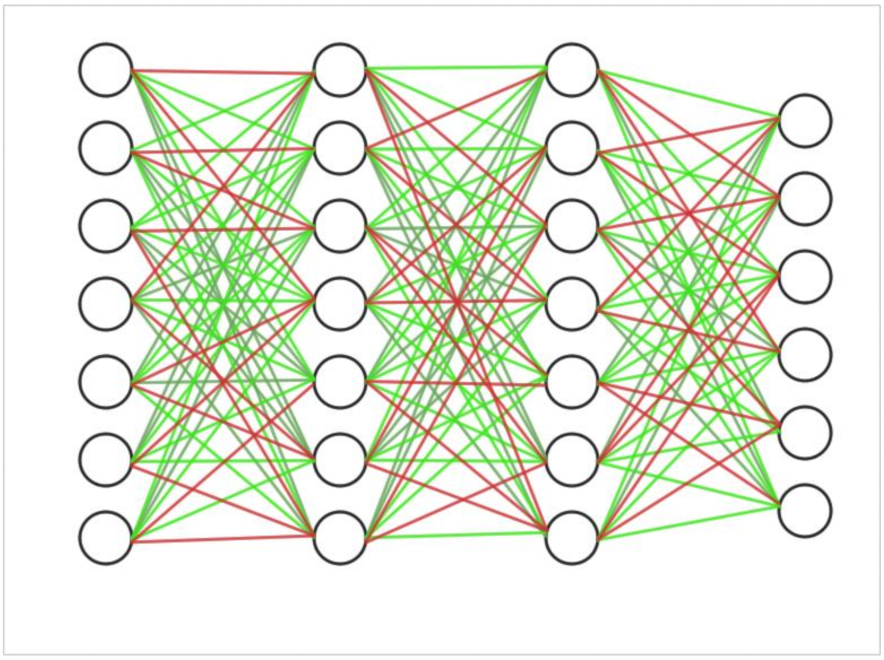


*Figure 10 – Step 2*

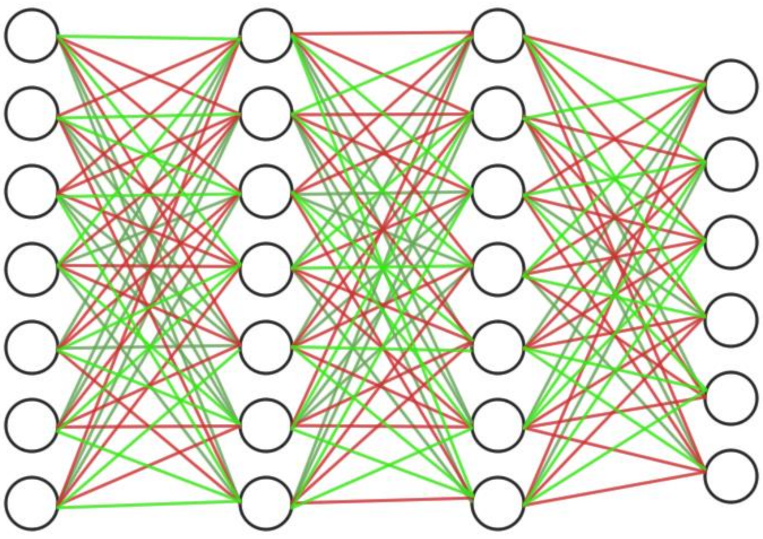


*Figure 11 – Step 3*

We can see here what it happens after 3 steps, the line represented in bright green has more weight, the ones in plain green has less weight and the ones in red has the least weight, we can see that the lines changes with every step because the system is trying to find the best combination of weights to create the best model to give the best output. That's the basics of neuronal networks, the mathematics behind this are complicated but the results given are pretty precise as we refer to machine learning.

## 8 Conclusions

In conclusion to start learning about machine learning we must start with regressions, comprehend everything about linear, quadratic and polynomial regression, this is the beginning of machine learning not just in theory but in mathematics and in code too, with regression we can visualize the equations that makes the best fit line, also the best representations of data sets and changes to the equations if the grade of the polynomial equations change, we can even compare for better comprehension the linear equation to the neuronal networks single neuron equation, and with the theory of regression we can visualize better that complex system that is neuronal networks, we also can see the correlation between regression and interpolation, regression is most used to know the trend data of new data in a graph and interpolation is used to join all the data in a graph, but they were not very different in the mathematical aspect when we apply bigger polynomial regressions. As to Fourier, we can see there are plenty derivative methods from it, but it is the Fast Fourier Transform (FFT) being one of the most important algorithms nowadays as it is used to identify signals and its patterns, in spite of the algorithm is not purely for machine learning, it is well referenced with the use that can be applied to it, such as signal processing and constantly compare the digital data it collects to give more precise and accurate predictions over time. At last, the genetic algorithms are going to be very important in a lot of fields, we must be prepared for a future where the machines are becoming self-thinkers, where they will be the ones creating new things every day.

## 9 Bibliography

[Dot CSV]. (2017, Nov 01). ¿Qué es el Machine Learning? ¿Y Deep Learning? Un mapa conceptual [Video File]. Retrieved from https://www.youtube.com/watch?v=KytW151dpqU

[Dot CSV]. (2018, Mar 19). ¿Qué es una Red Neuronal? Parte 1-3: La Neurona [Video File]. Retrieved from https://www.youtube.com/watch?v=MRIv2IwFTPg

[sentdex]. (2016, Apr 10). Tutorial práctico de aprendizaje de maquinas con Python Introducción [Video File]. Retrieved from https://www.youtube.com/watch?v=OGxgnH8y2NM&index=1&list=PLQVvvaa0QuDfKTOs3Keq_kaG2P55YRn5v

IBM. Fourier Transform in action - Digital Signal Processing in Machine Learning | Coursera. Retrieved from https://www.coursera.org/lecture/advanced-machine-learning-signal-processing/fourier-transform-in-action-w2ztK

Cooley–Tukey FFT algorithm. (2018). Retrieved from https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm
It's all about the features - Machine Learning and Signal Processing tips. (2018). Retrieved from https://reality.ai/it-is-all-about-the-features/

Jovanovic, J. (2014). How does Shazam work? Music Recognition Algorithms, Fingerprinting, and Processing. Retrieved from https://www.toptal.com/algorithms/shazam-it-music-processing-fingerprinting-and-recognition

How does Shazam work | Coding Geek. (2015). Retrieved from http://coding-geek.com/how-shazam-works/

Cook, J. (2017). Unified Theory of Fourier Transforms - DZone AI. Retrieved from https://dzone.com/articles/unified-theory-of-fourier-transforms

Ghobrial, S. (2017). Is the Fourier Transform used in machine learning? | Quora. Retrieved from https://www.quora.com/Is-the-Fourier-Transform-used-in-machine-learning

Transformada de Fourier. (2018). Retrieved from https://es.wikipedia.org/wiki/Transformada_de_Fourier

[ZettaBytes, EPFL]. (2017, Mar 30). 3 Applications of the (Fast) Fourier Transform (ft. Michael Kapralov) [Video File]. Retrieved from https://youtu.be/aqa6vyGSdos

Puig Adam P. Curso teórico-práctico de cálculo integral aplicado a la Física y a la Técnica. Biblioteca matemática (1972), págs 151-161.

Michael Nielsen. (2017). Neural Networks and Deep Learning, http://neuralnetworksanddeeplearning.com/

P.Lotus. (2013). Polynomial Regression Data Fit, https://arachnoid.com/polysolve/

[ZettaBytes, EPFL]. (2012). 3 Proof (part 4) minimizing squared error to regression line [Video File]. Retrieved from https://www.khanacademy.org/math/statistics-probability/describing-relationships-quantitative-data/more-on-regression/v/proof-part-4-minimizing-squared-error-to-regression-line

Louis Nicolle. (2017, Aug 29). Was Darwin a Great Computer Scientist?, https://blog.sicara.com/getting-started-genetic-algorithms-python-tutorial-81ffa1dd72f9

Clinton Sheppard. (2016, Aug 06). Introduction to Genetic Algorithms with Python - Hello World!, https://www.codeproject.com/articles/1104747/introduction-to-genetic-algorithms-with-python-hel

Vijini Mallawaarachch. (2017, Jul 07). Introduction to Genetic Algorithms—Including Example Code, https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3