## Logistic Regression

Model in which logistic function or sigmoid function is used to map the values to 0 to 1

The sigmoid function is:

$$F(x) = \frac{1}{1 + e^{-x}} = \frac{e^{x}}{e^{x} + 1}$$

After applying the sigmoid function we try to squeeze the data into a line or a linear function
We start by assuming a linear function p(x) but p is the probability that should vary from 0 to 1 p(x) is an unbounded linear equation so we assume p(x) to be a linear function of x and further bound it between the 0 to 1 range

$$\log \frac{p(x)}{1 - p(x)} = \alpha_0 + \alpha . x$$

And then solving for p(x):;

$$p(x) = \frac{e^{\alpha_0 + \alpha}}{e^{\alpha_0 + \alpha} + 1}$$

We can also make the logistic regression model a linear classifier by setting certain thresholds

## Kmeans-Clustering

Samples that can be put on a linear graph, an X-Y plane, or a heat map can be clustered using this method. This method is used to categorize data.

The method to do Kmeans Clustering:

1. First, we need to identify the number of clusters(K) in K-means clustering
2. Then we need to select K number of points(clusters) on the data which randomly will eventually be the centroids of the clusters after multiple iterations
3. We then measure the distance from each cluster we have chosen to all the data points
4. We then assign the points to the nearest clusters respectively
5. We then calculate the mean of the squares of distances for each cluster which is the position of the centroids.
6. We then make this centroid the new cluster center and repeat the process until the centroid doesn't change.
7. We do this for multiple different initial cluster centers to avoid errors

How to find the value of K:

1. We can do it by checking all values of K and finding the best K
2. Elbow method:
    a. We can plot the value of K with the relative change in variance of data for each K and in the graph the place where it changes the least is the best amount of clusters to use(K)

Mathematical formulae used:

Euclidean distance in 2D or 3D geometry to find out the centroids of the clusters

## Neural Network

A neural network is a computational model inspired by the human brain's structure and functions. It consists of interconnected nodes, also known as neurons, organized into layers. The neurons in one layer are connected to those in the adjacent layers. Weights represent the connections between neurons.

Let's break down how a simple feedforward neural network works using mathematical statements:

Input Layer:

The input layer receives the raw data as input. For a neural network with n input features, the input layer will consist of n neurons.
Let's represent the input features as $x_1$, $x_2$, ..., $x_i$, where i ranges from 1 to n.
The input layer simply passes the input features to the next layer.
Hidden Layers:

The hidden layers perform the computation and transformation of the input data to learn patterns and representations.
Each neuron in a hidden layer takes inputs from all neurons in the previous layer, multiplies those inputs by corresponding weights, and applies an activation function to produce the output.
Let's represent the output of the j-th neuron in the k-th hidden layer as $h_{\square\square}$.
The output $h_{\square\square}$ of a neuron in a hidden layer is computed as follows:
$h_{\square\square}$ = activation_function($\sum(w_{\square i} * x_i) + b_{\square}$)

where:
$w_{\square i}$ represents the weight of the connection between the j-th neuron in the k-th layer and the i-th neuron in the (k-1)-th layer.
$x_i$ is the output of the i-th neuron in the (k-1)-th layer.
$b_{\square}$ is the bias term for the j-th neuron in the k-th layer.
activation_function is a non-linear function that introduces non-linearity into the neural network. Common activation functions include ReLU, sigmoid, and tanh.
Output Layer:

The output layer produces the final predictions or outputs of the neural network.
Similar to the hidden layers, each neuron in the output layer takes inputs from all neurons in the last hidden layer, multiplies those inputs by corresponding weights, and applies an activation function to produce the output.

The number of neurons in the output layer depends on the type of task. For binary classification, there is one neuron with a sigmoid activation function. For multi-class classification, there are multiple neurons, each representing a class, and a softmax activation function is used to convert the outputs into probability distributions.
The output $\hat{y}_j$ of the j-th neuron in the output layer is computed as follows:
$\hat{y}_j = \text{activation\_function}(\sum(w_{ji} * h_{ji}) + b_j)$

Training:

During training, the neural network iteratively adjusts its weights and biases using optimization algorithms (e.g., gradient descent) to minimize a defined loss function that quantifies the difference between the predicted outputs and the actual labels (ground truth).
The backpropagation algorithm is used to compute the gradients of the loss function with respect to the weights and biases, enabling the optimization algorithm to update the parameters in the right direction.
By adjusting the weights and biases in the hidden layers, a neural network learns to map the input data to the correct output labels, enabling it to make accurate predictions on new, unseen data. The process of learning from data and updating the parameters is what allows a neural network to perform various tasks, such as image recognition, natural language processing, and more.