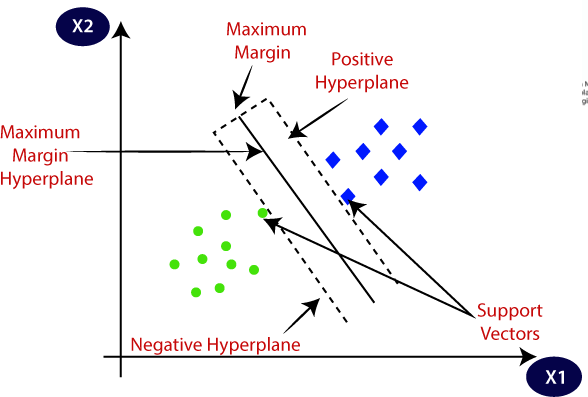
# Support Vector Machine (SVM)

(Not affected by outliers)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a **hyperplane**.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as **Support Vector Machine**. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



SVM is suitable for high dimensional data (no of variables are high compared to number of observations)

**Hyperplane:**

* There can be multiple lines/decision boundaries to segregate the classes in n- dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.
* The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features, then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.
* We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

**Support Vectors:**

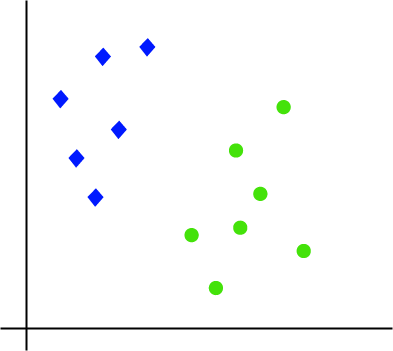
* + Support vectors are those data points which represent the entire class and are nearest to the opposite class

**SVM can be of two types:**

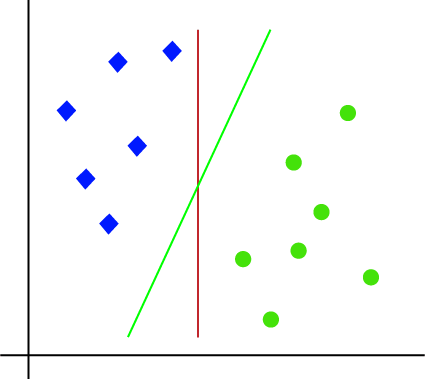
* **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
* **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

**Linear SVM:**

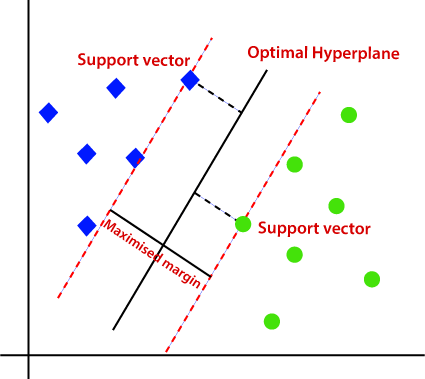
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



So, as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

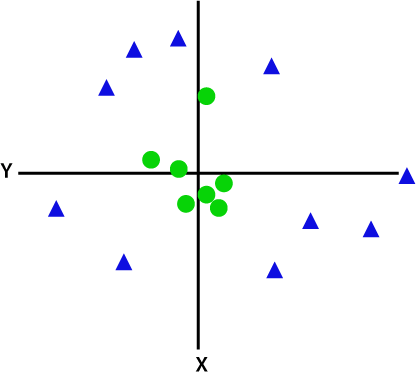


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.

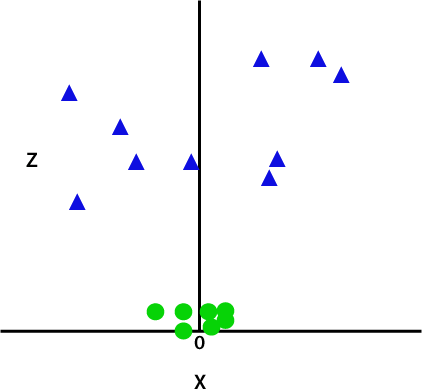


# Non-Linear SVM:

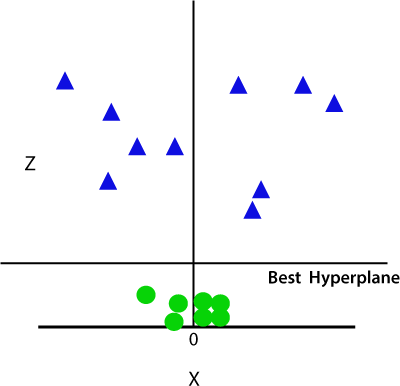
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



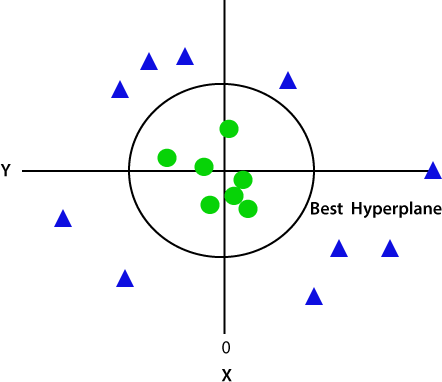
So, to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third-dimension z. By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



Hence, we get a circumference of radius 1 in case of non-linear data.

**Training:**

Input  x, y

Output  best fit hyperplane equation

Wx +b =0  hyperplane equation Were,

b = bias

W = Weight

**Testing:**

Input  x

Output  y on the basis of equation

If the value of y < 0 towards -1, left class towards the hyperplane is the predicted class If the value of y > 0 towards 1, right class towards the hyperplane is the predicted class

* + In case of regression after substituting x values of the test data point into the equation, whatever value we get becomes the predicted value

**Tuning:**

1. Feature Selection
2. Hyperparameter Tuning (Kernel, Gamma, C)

* **Kernel**
  1. Linear  one of the least used kernel and it is only used when we have high dimensional data because studies have proved that very high dimensional data is usually linearly separable.
  2. Sigmoid  one of the least used kernel mostly suitable for binary classification
  3. Poly  It stands for Polynomial kernel used to fit non-linear data
  4. rbf  radial basis function kernel which if required will perform the kernel trick transformation and fit an almost linear hyperplane into data, rbf is a **default kernel** used by sklearn because it can handle both linear and non-linear data.

What is kernel trick transformation?

* It allows to convert non – linearly separable data into a linearly separable data, It will try to map the non – linear data by adding a new dimension where it usually squares the original value such that it becomes linearly separable data and we can fit a linear hyperplane into it.

**Gamma:**

* Gamma value indicates how many farther support vectors should be considered while fitting the best fit hyperplane
* It is not applicable when kernel = Linear
* We can use it only when kernels are poly, rbf and sigmoid
* For the base model gamma = 0.1
* If we try to increase the gamma value i.e., 1,10, It will go very close to data points and fit the hyperplane which might lead to overfitting
* So, the ideal way to tune the gamma value is to reduce it toward 0 i.e., 0.01, 0.001 such that we end up getting good amount of boundary towards each of the class

**C**  **Cast Parameter (regularization)**

* It is a cost penalty implemented to penalize the model for the misclassification
* By default, C = 1, which is a low penalty which indicates we are ok with misclassification
* We can tune the c value we can increase it to 10, 20, 30 to indicate we are not ok with misclassifications
* But a very high value of c could shift the hyperplane a lot and may lead to an over –

fitted model

* So ideally we should gradually increase the value of c only till the time we notice that model improves