UNIT-4

Learning with Support Vector Machines:

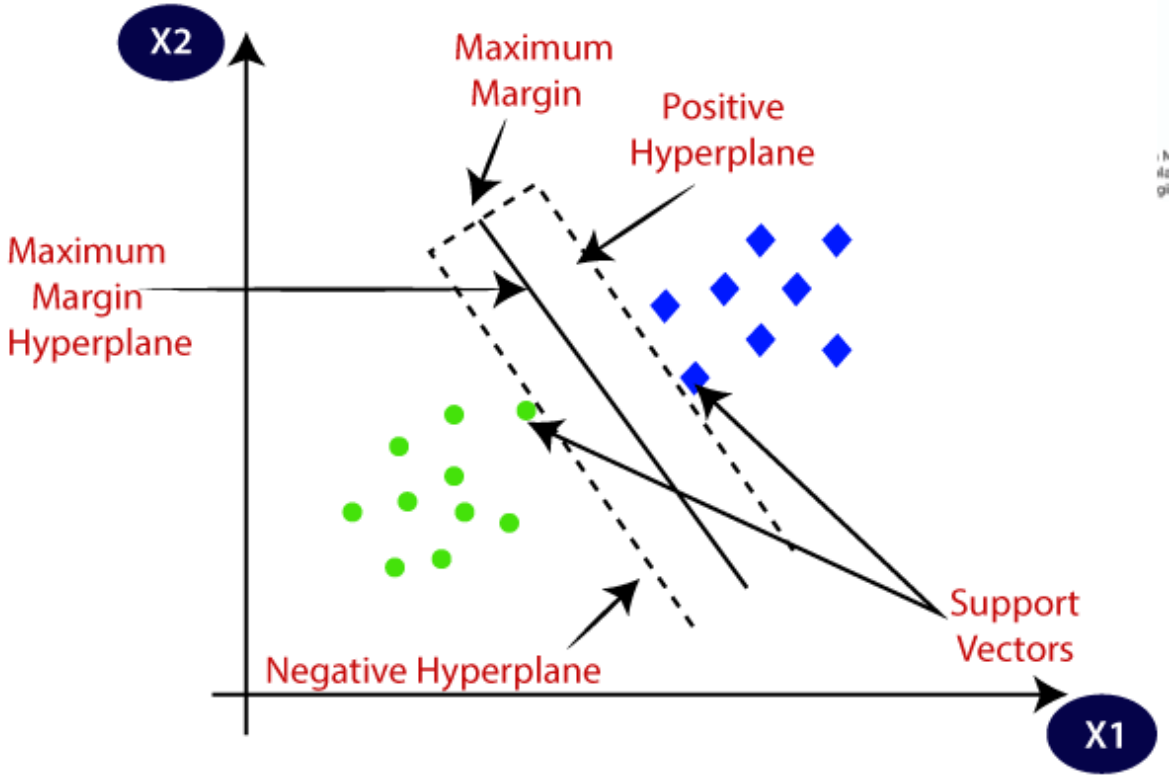
Introduction, Linear Discriminant Functions for binary classification, Linear maximal margin classifier for linearly separable data, Linear soft margin classifier for overlapping classes, kernel induced feature spaces, nonlinear classifiers, Regression by Support Vector Machines, Decomposing Multiclass Classification Problem into Binary Classification Tasks, Variants of Basic SVM Techniques

ENSEMBLE METHODS:

Bagging, Committee Machines & Stacking, Boosting, Gradient Boosting, Random Forest

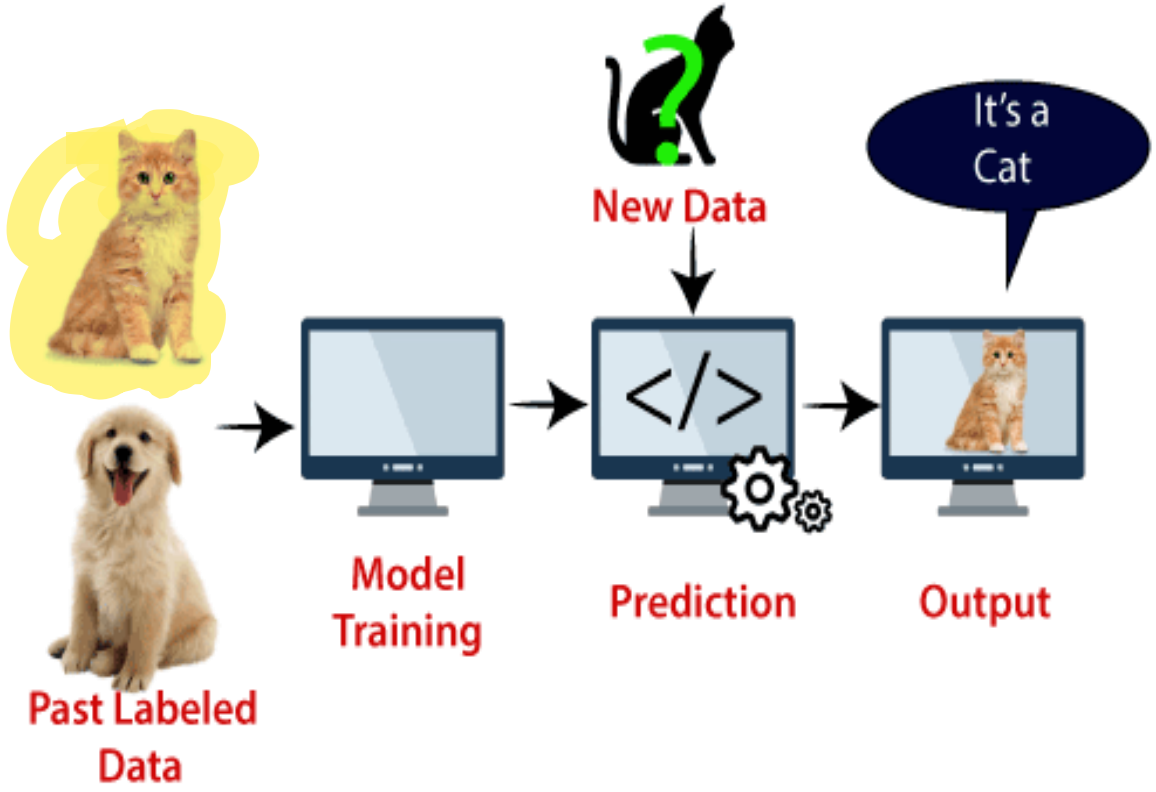
Introduction to Support Vector Machine

- SVM 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.



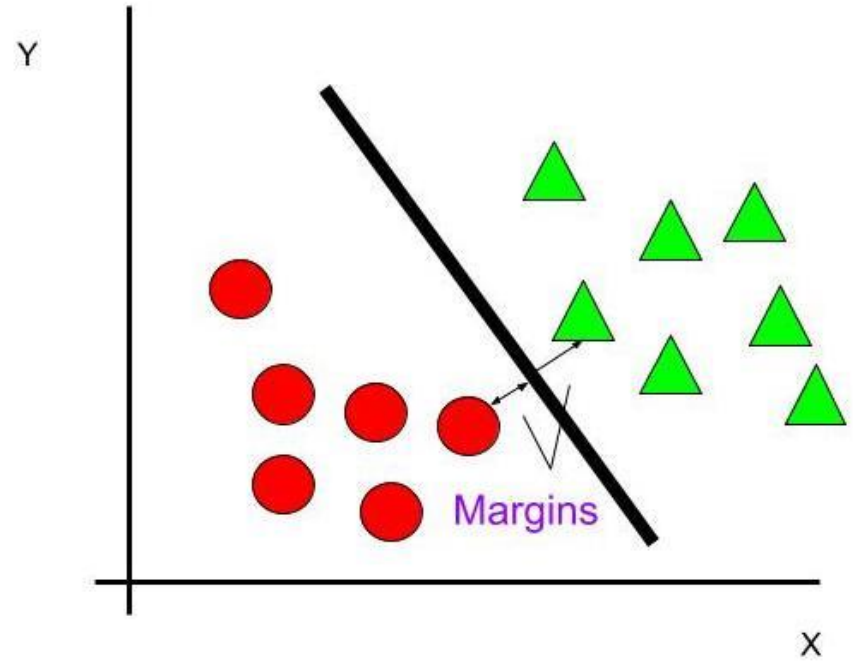
Support vector machine

Example

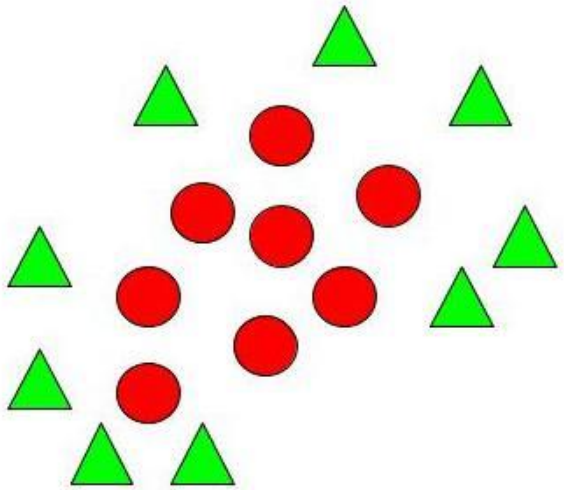


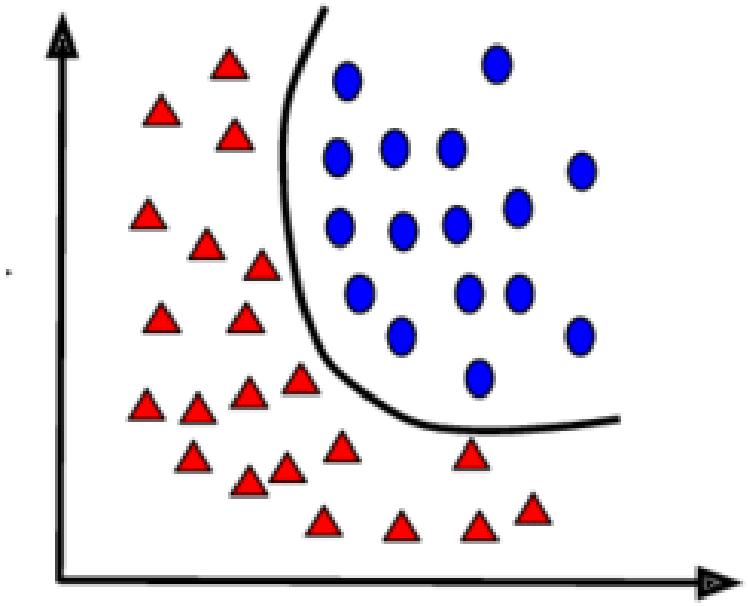
Types of SVM

1. 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



2. 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.



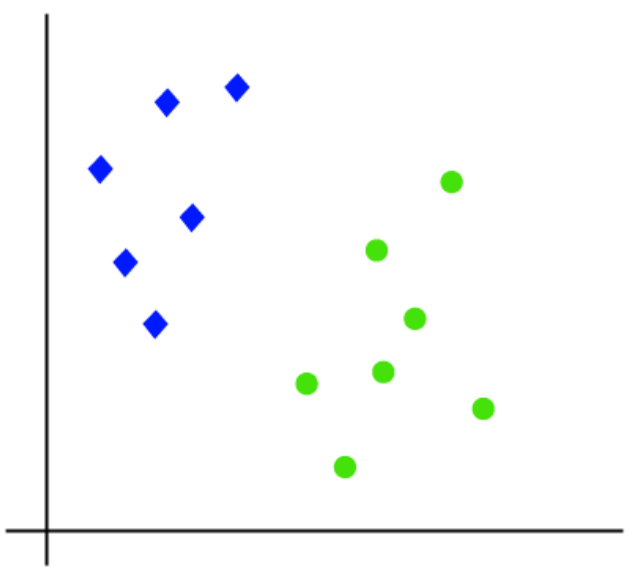


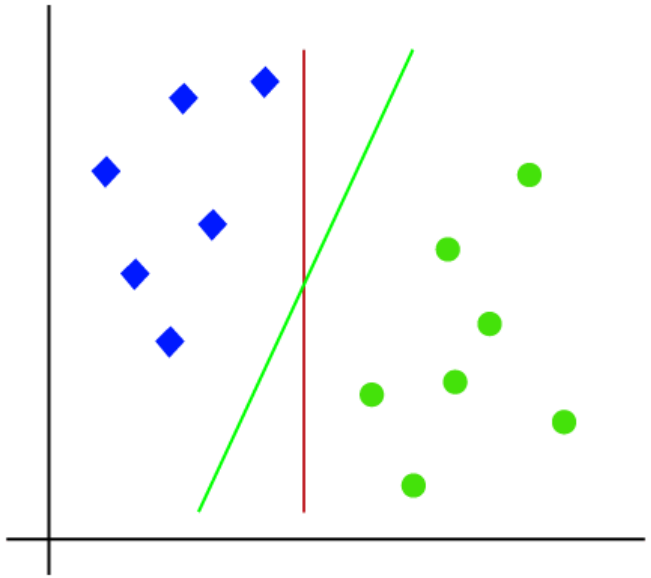
Hyperplane and Support Vectors

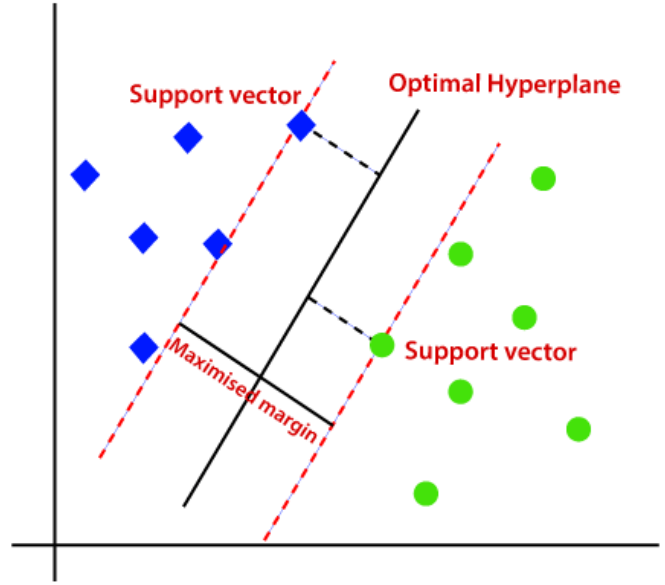
- **Hyperplane:**
  - There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.
  - The dimensions of the hyperplane 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
- **Support Vectors:**
  - The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

How does SVM works?

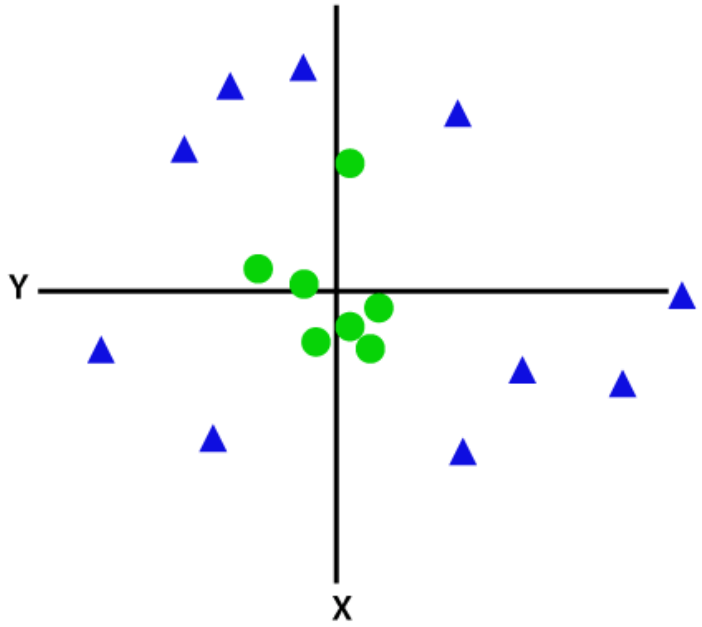
- **Linear SVM:**



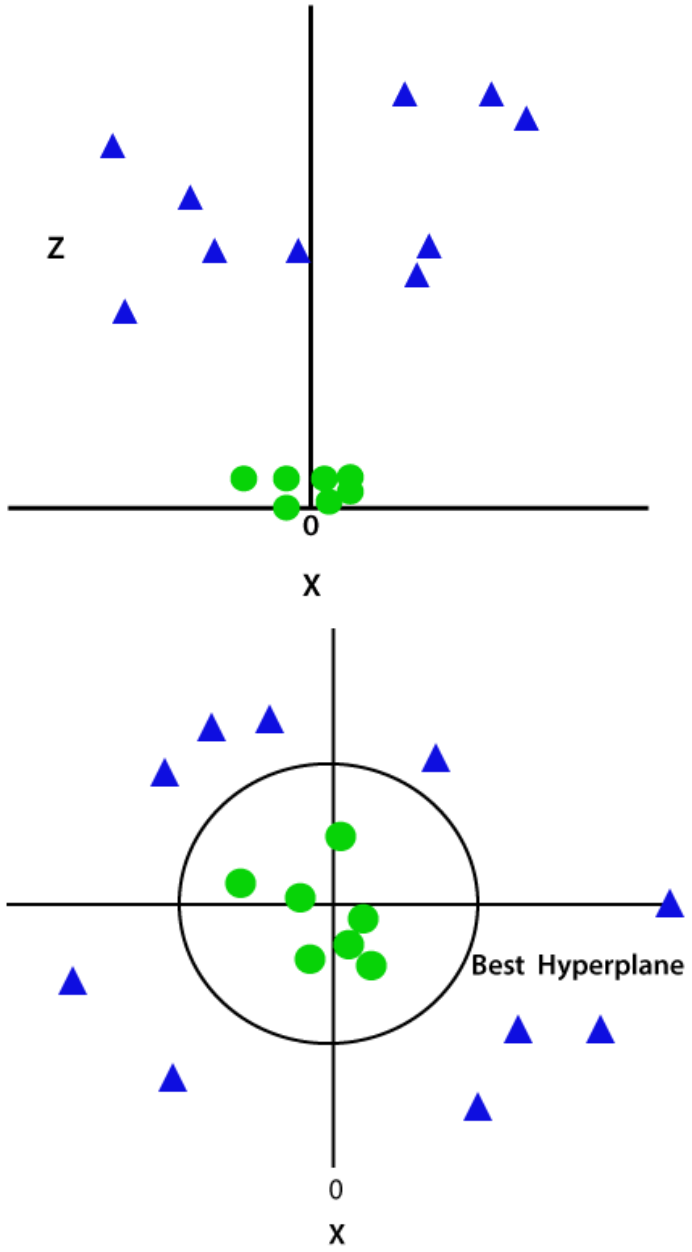




- **Non-Linear SVM:**



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



Linear Discriminant Functions for binary classification

- Linear discriminant analysis is an extremely popular dimensionality reduction technique.
- Linear Discriminant Analysis was developed as early as 1936 by Ronald A. Fisher. The original Linear discriminant applied to only a 2-class problem. It was only in 1948 that C.R. Rao generalized it to apply to multi-class problems.
- The goal is to do this while having a decent separation between classes and reducing resources and costs of computing.

Dimensionality Reduction:

- Multi-dimensional data is data that has multiple features which have a correlation with one another. Dimensionality reduction simply means plotting multi-dimensional data in just 2 or 3 dimensions.

Linear Discriminant Analysis

(i) Calculate the separability between different classes. This is also known as between-class variance and is defined as the distance between the mean of different classes.

$$S_b = \sum_{i=1}^{g} N_i (\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T$$

(ii) Calculate the within-class variance. This is the distance between the mean and the sample of every class.

$$S_w = \sum_{i=1}^{g} (N_i - 1) S_i = \sum_{i=1}^{g} \sum_{j=1}^{N_i} (x_{i,j} - \bar{x}_i)(x_{i,j} - \bar{x}_i)^T$$

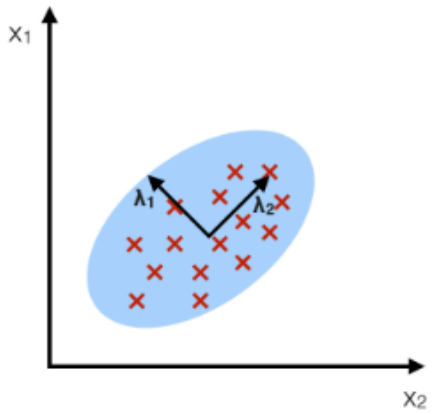
(iii) Construct the lower-dimensional space that maximizes Step1 (between-class variance) and minimizes Step 2(within-class variance). In the equation below P is the lower-dimensional space projection. This is also known as Fisher's criterion.

$$P_{lda} = \arg\max_{P} \frac{\left| P^T S_b P \right|}{\left| P^T S_w P \right|}$$

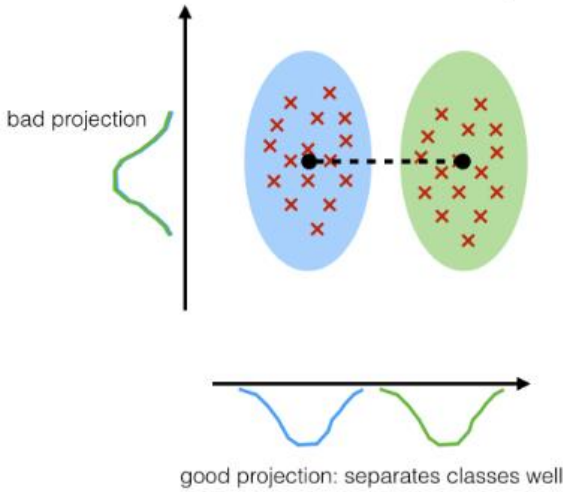
LDA vs PCA

PCA	LDA
1. PCA is an unsupervised algorithm. It ignores class labels altogether and aims to find the principal components that maximize variance in a given set of data	1. Linear Discriminant Analysis, on the other hand, is a supervised algorithm that finds the linear discriminants that will represent those axes which maximize separation between different classes.

PCA:
component axes that maximize the variance



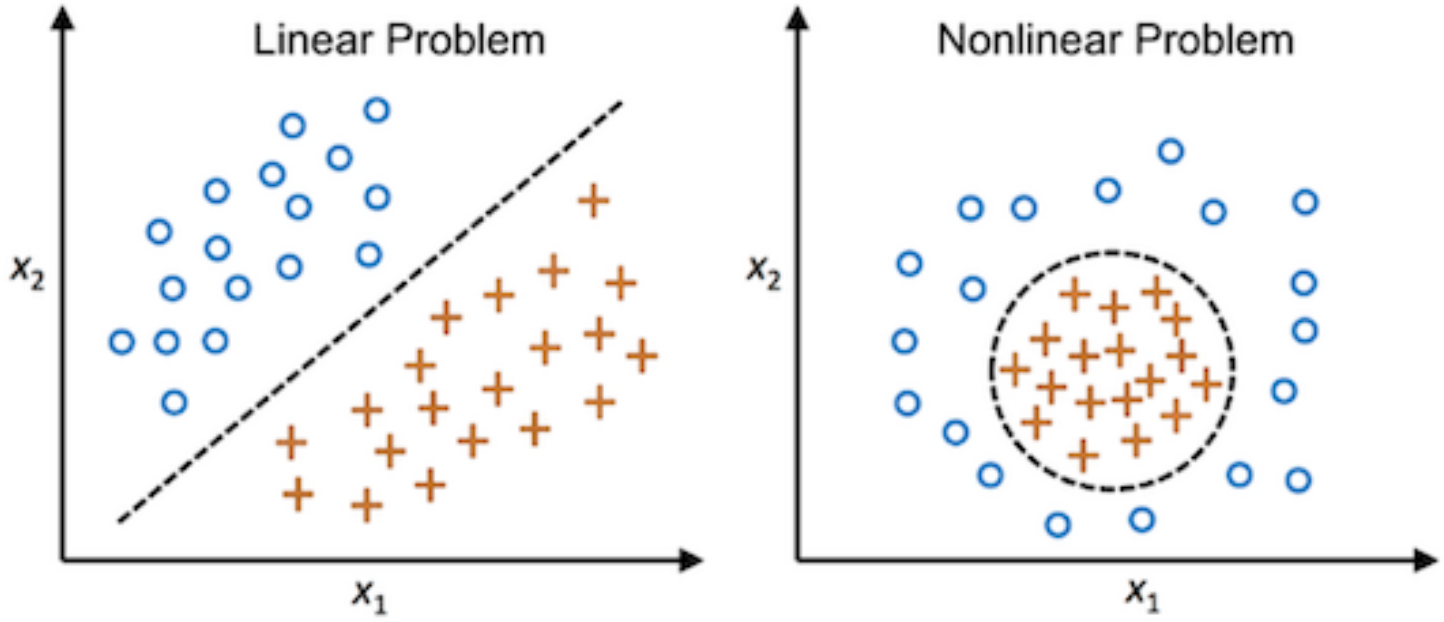
LDA:
maximizing the component axes for class-separation



Linear maximal margin classifier for linearly separable data

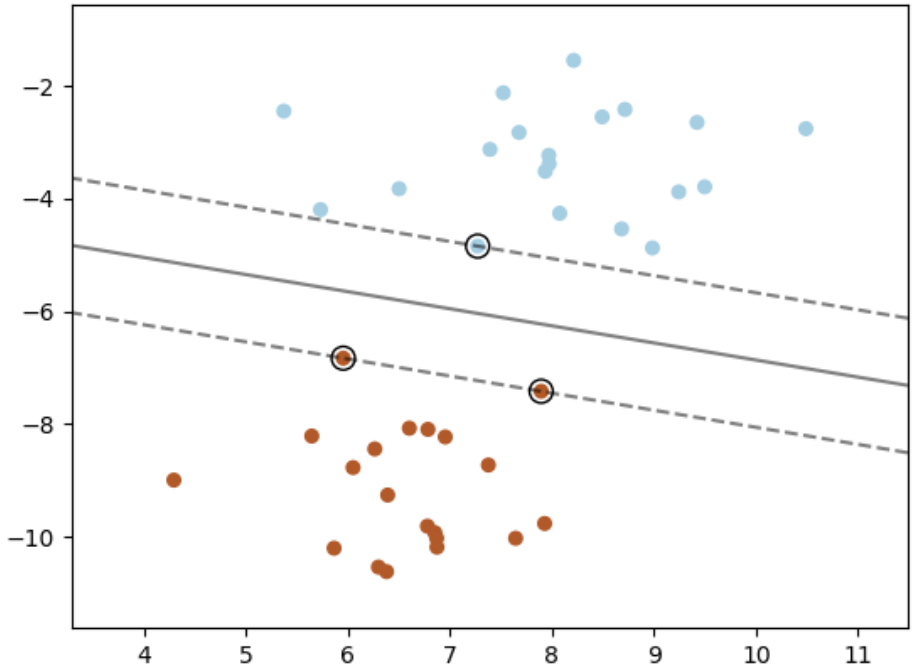
- This classifier is designed specifically for linearly separable data, refers to the condition in which data can be separated linearly using a hyperplane.

Linearly separable and non-linearly separable data:



How do we choose the hyperplane that we really need?

- Based on the maximum margin, the Maximal-Margin Classifier chooses the optimal hyperplane. The dotted lines, parallel to the hyperplane in the following diagram are the **margins** and the distance between both these dotted lines (Margins) is the Maximum Margin



Maximal Margin classifier :-

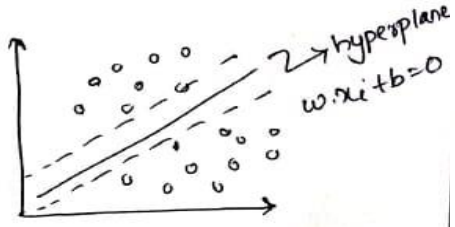
For linearly separable data we have the equation like $\boxed{y = mx + c}$ in replacement of this equation $w.x+b$ (hyper plane).

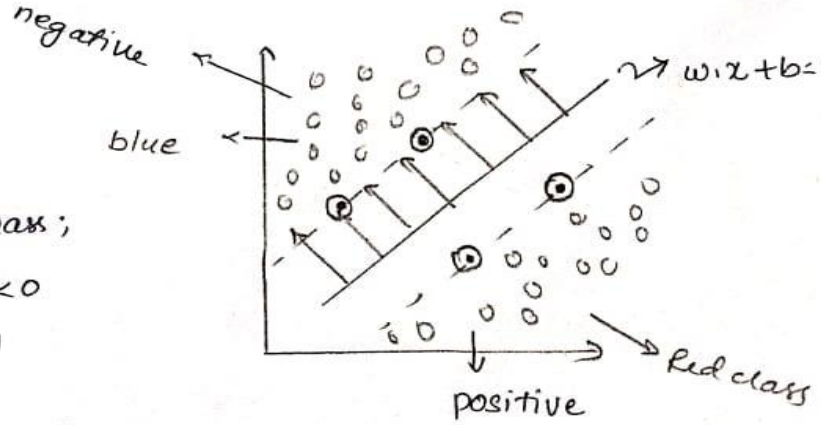
$$\underset{w,b}{\arg\max} \; \min_{i \in \{1,2,3\ldots n\}} |w.x_i + b|$$
$$\|w\| = 1$$

Such that

$$y_i (w.x_i + b) \geq 0$$

for all i's.





for blue class;

$w.x_i + b < 0$

$y_i = -1$

so,

$y_i (w.x_i + b) < 0$

$(-1)(-1) < 0$

$1 < 0$

for red class

$w.x_i + b \geq 0$

$y_i = +1$

$y_i (w.x_i + b) > 0 \Rightarrow 2 > 0$

So both $y_i = -1$ & $y_i = +1$ will show

$y_i (w.x_i + b) > 0$ for whole the data points belonging to the both classes.

wrong side

$\Rightarrow$ if any of the data point is lying on the hyperplane then the value of the expression is $y_i (w.x_i + b) < 0$. It will violate our constraint.

Drawbacks

- This classifier is heavily reliant on the support vector and changes as support vectors change. As a result, they tend to overfit.
- They can’t be used for data that isn’t linearly separable. Since the majority of real-world data is non-linear. As a result, this classifier is inefficient.
- The maximum margin classifier is also known as a “**Hard Margin Classifier”** because it prevents misclassification and ensures that no point crosses the margin.
- It tends to **overfit** due to the hard margin. An extension of the Maximal Margin Classifier, “**Support Vector Classifier**” was introduced to address the problem associated with it.

Linear soft margin classifier for overlapping classes

- Support Vector Classifier is an **extension** of the Maximal Margin Classifier. It is less sensitive to individual data. Since it allows certain data to be misclassified, it's also known as the "**Soft Margin Classifier".**
- To tackle this problem what we do is modify that equation in such a way that it allows few misclassifications that means it allows few points to be wrongly classified.

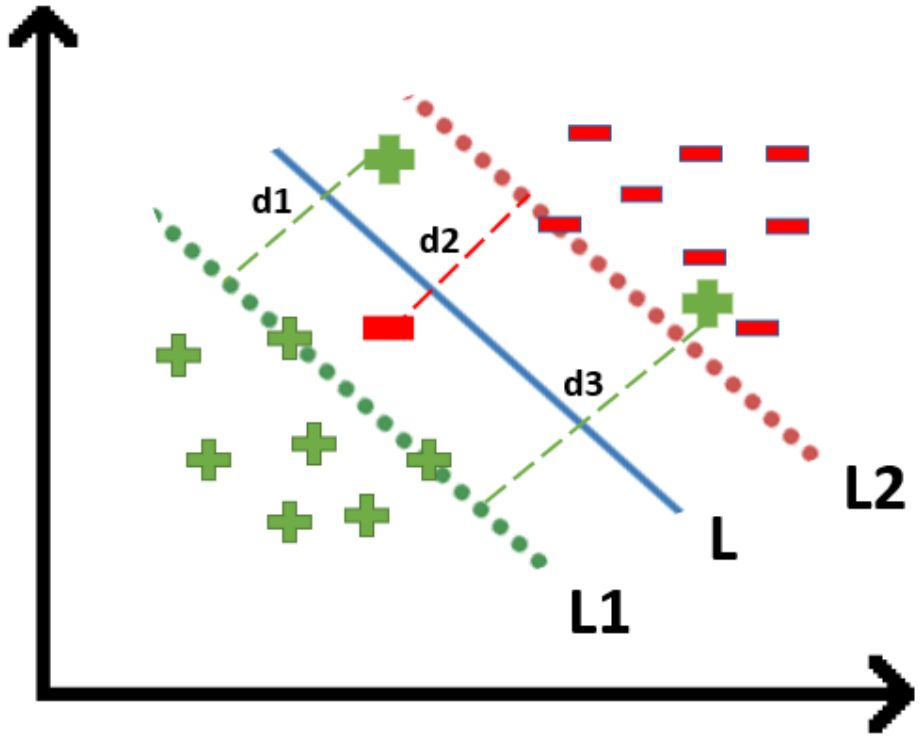
We know that *max[f(x)]* can also be written as *min[1/f(x)]*, it is common practice to minimize a cost function for optimization problems; therefore, we can invert the function.

$$\text{argmin}(\text{w}^*, \text{b}^*) \ \frac{\|\text{w}\|}{2} \text{ such that } \text{y}_\text{i}\left(\vec{w}.\vec{X} + b\right) \geq 1$$

- To make a soft margin equation we add 2 more terms to this equation which is **zeta** and multiply that by a **hyperparameter 'c'**

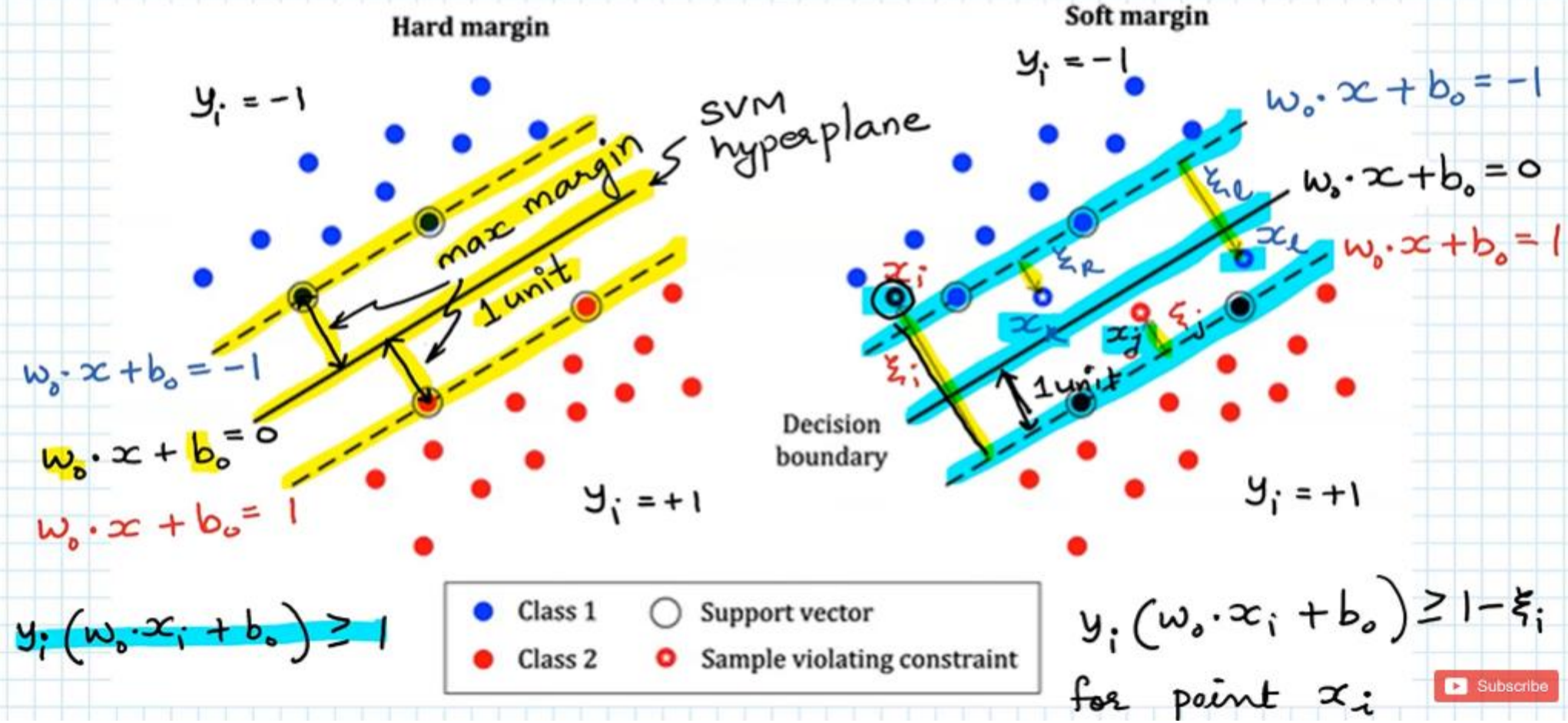
$$\text{argmin}\left(\text{w}^*, \text{b}^*\right) \ \frac{\|\text{w}\|}{2} \ + c\sum_{i=1}^{n} \zeta_i$$

For all the ***correctly classified*** points our **zeta** will be equal to 0 and for all the ***incorrectly classified*** points the **zeta** is simply the distance of that particular point from its correct hyperplane that means if we see the wrongly classified green points the value of **zeta** will be the distance of these points from L1 hyperplane and for wrongly classified redpoint **zeta** will be the distance of that point from L2 hyperplane.



Soft Support Vector Machine (Soft-SVM)

Soft SVM relaxes the linear separability constraint by introducing slack variables.



$y_i (w_0 \cdot x_i + b_0) \geq 1$

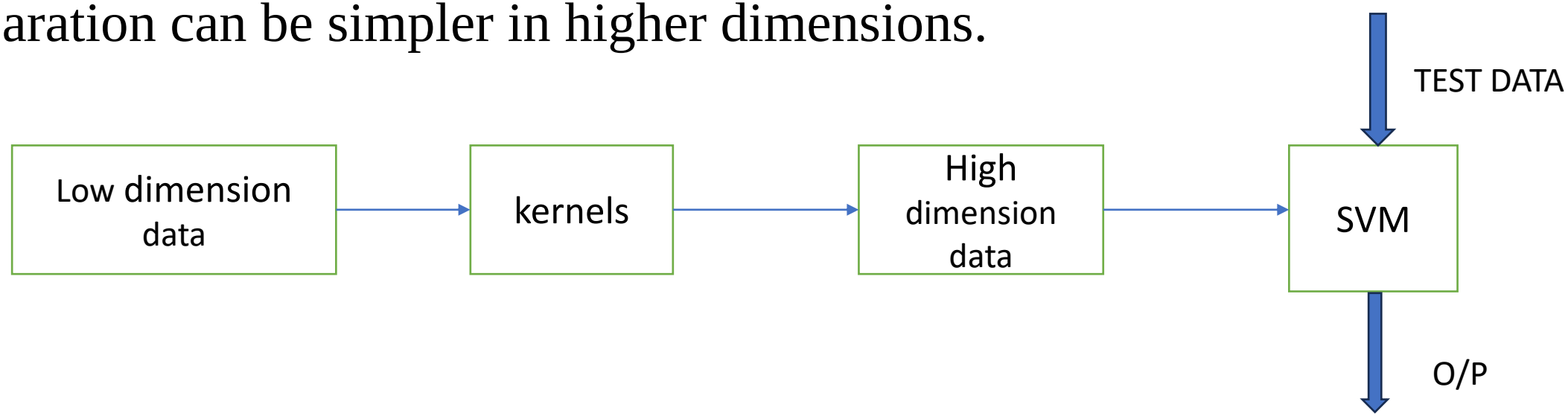
$y_i (w_0 \cdot x_i + b_0) \geq 1 - \xi_i$ for point x_i

Kernal induced functions

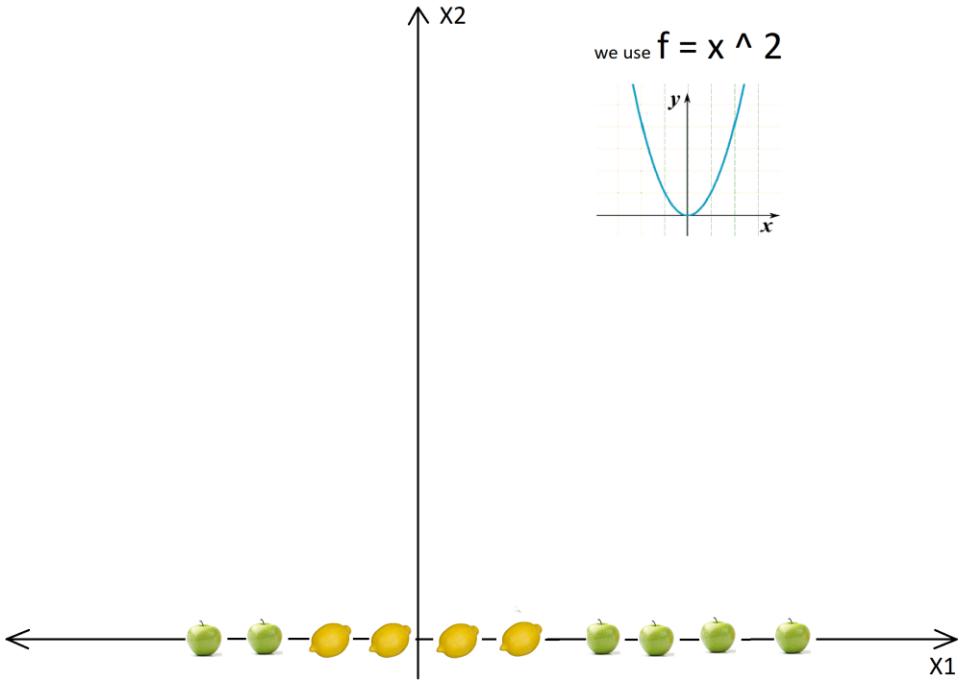
- Support vector machines are an extension of soft margin classifier. It can also be used for nonlinear classification by using the kernel.

Kernel:

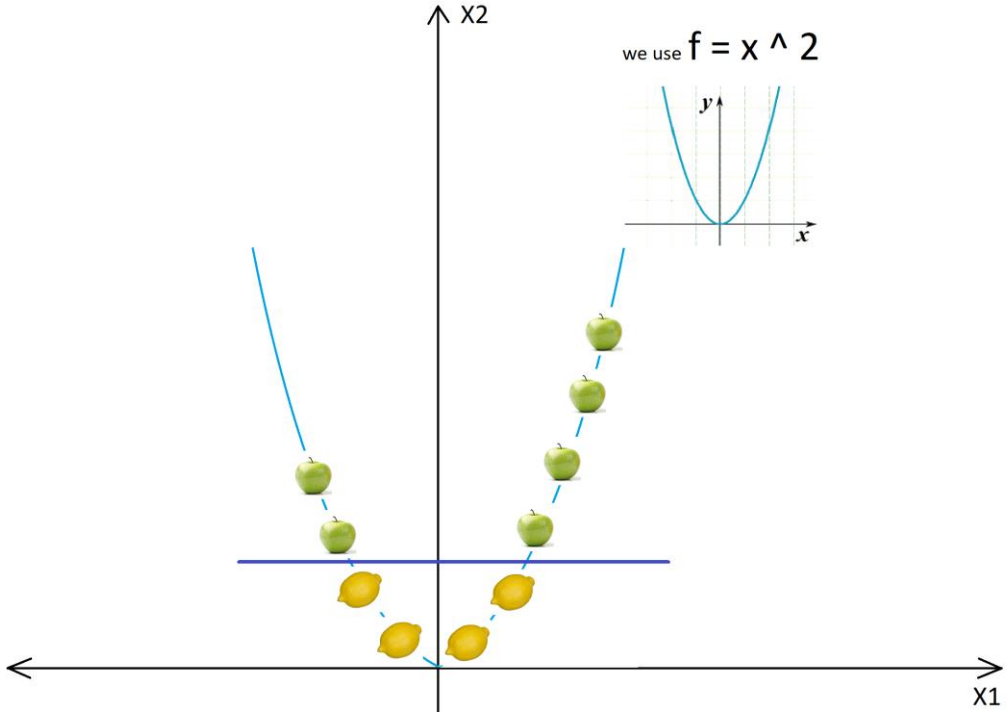
- It transforms non linear separable data from lower to higher dimensions to facilitate linear classification using mathematical formulas (dot product -> mapping not converting).
- We use the kernel based technique to separate non linear data because separation can be simpler in higher dimensions.



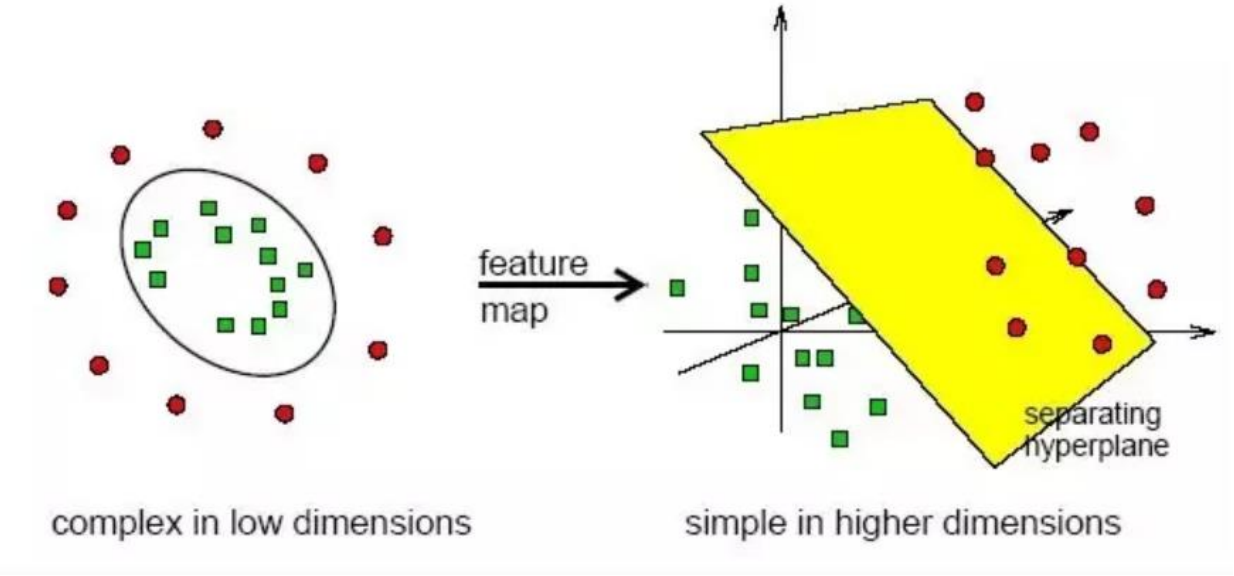
Understand the working of Kernel using another example



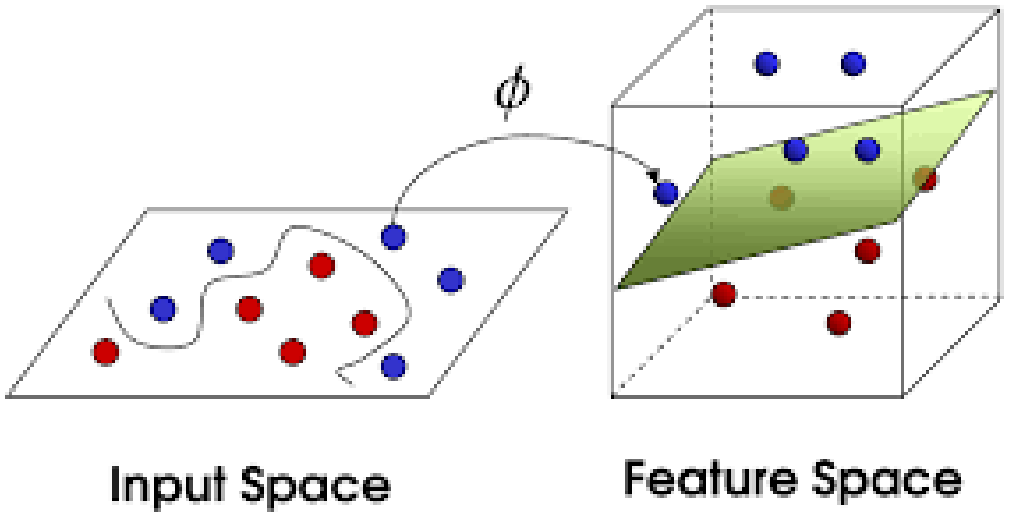
1 dimensional data



2 dimensional data



2 D data is changed to 3D



Why do we need kernel trick

- Kernel trick means replacing the dot product in mapping functions with a kernel function.

$$k(x, y) = \varphi(x) \cdot \varphi(y) \quad [\varphi \text{ is the mapping function}]$$

- Similar to the mapping functions, kernel helps in mapping data from input space to higher dimensional feature space with least computations.
- Performing the kernel operation is much easier compared to mapping functions

Example

Kernel in SVM :-

Illustrate with numerical example.

→ Consider two data points (1, 2) and (3, 4)

→ Apply polynomial Kernel or Quadratic Kernel $K(x, y) = (x^T y)^2$ and show that it is equivalent to mapping function $\phi = (x^2, y^2, \sqrt{2}xy)$

→ The mapping function is given as $\phi = (x^2, y^2, \sqrt{2}xy)$

Let us apply mapping function first for the first data point (1,2)

$\therefore \phi = (1^2, 2^2, 2\sqrt{2})$

$\phi = (1, 4, 2\sqrt{2})$

Now consider Second data point (3,4)

$\phi = (9, 16, 12\sqrt{2})$

$K(x, y) = \phi(x) \cdot \phi(y)$

$= (1, 4, 2\sqrt{2}) \cdot (9, 16, 12\sqrt{2})$

$= (1 \times 9 + 4 \times 16 + 24(2)) = 121$

START WRITING HERE

estion No.

Now Take $(x^T y)^2$

$K(x, y) = \begin{matrix} K(1,2) \\ K(3,4) \end{matrix}$ $\left.\begin{matrix} x = (1,2) \\ y = (3,4) \end{matrix}\right\}$

$= \left[\left(\frac{1}{2}\right) \cdot (3,4)\right]^2$

$[(1 \times 3) + (2 \times 4)]^2$

$= [3 + 8]^2 = [11]^2 = 121$

Types of kernels

1. Linear Kernel
2. Polynomial or quadratic kernel
3. Homogeneous kernel
4. Inhomogeneous kernel
5. Gaussian Kernel or RBF kernel
6. Sigmoid Kernel

Types of Kernels:

1. Linear Kernel :-

→ Linear Kernel are of the type

$$K(x, y) = x^T \cdot y$$

→ where x and y are two vectors

→ Therefore $K(x, y) = \phi(x) \cdot \phi(y) = x^T \cdot y$

equivalent to dot product.

2. Polynomial Kernel :-

→ Polynomial Kernel are of the type;

$$K(x, y) = (x^T \cdot y)^q$$

→ This is called homogeneous Kernel

→ Here, q is the degree of the polynomial.

→ If $q = 2$ then it is called "quadratic Kernel".

→ If we add constant term to this quadratic Kernel then it is "inhomogeneous Kernel".

ie. $K(x, y) = (c + x^T \cdot y)^q$

here c is a constant and q is degree of the polynomial.

→ If c is zero and degree is 1, the polynomial Kernel is reduced to linear Kernel

$$\therefore K(x, y) = (0 + x^T, y)^1$$

→ The value of degree q should be optimal otherwise leads to "overfitting".

Example :-

Consider two data points

$x = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ and $y = (2,3)$ with $c = 1$

Apply Linear, homogeneous & inhomogeneous kernels.

Sol :-

The kernel is given by $K(x,y) = (x^T y)^q$

* if $q = 1$ then it is called "Linear kernel".

$$K(x,y) = \left(\begin{pmatrix} 1 \\ 2 \end{pmatrix}^T \cdot (2,3)\right)^1 = 8.$$

$$\Downarrow$$

$$(1\ 2) \cdot (2,3) = 2(2+6) = 8$$

* If $q = 2$ then "homogeneous or quadratic kernel"

$$K(x,y) = \left(\begin{pmatrix} 1 \\ 2 \end{pmatrix}^T \cdot (2\ 3)\right)^2 = 8^2 = 64.$$

* If $q = 2$ and $c = 1$ "inhomogeneous kernel"

$$K(x,y) = (c + x^T \cdot y)^q$$

$$= \left(1 + \begin{pmatrix} 1 \\ 2 \end{pmatrix}^T \cdot (2\ 3)\right)^2 = (1+8)^2 = 81$$

Gaussian Kernel :-

→ Radial Basis Functions (RBFs) or Gaussian Kernels are extremely useful in SVM

→ The RBF function is shown as below

$$K(x,y) = e^{\frac{-(x-y)^2}{2\sigma^2}}$$

Here, γ is an important parameter.

If γ is small, then the RBF is similar to Linear SVM. and if γ is large, then the Kernel is influenced by more support vectors.

example :-

Consider two data points $x = (1,2)$; $y = (2,3)$ with $\sigma = 1$

→ The squared distance between the points (1,2) and (2,3) is given as:-

$$(1-2)^2 + (2-3)^2 = 2$$

$$K(x,y) = e^{\left[\frac{-2}{2}\right]} = e^{-1} \Rightarrow 0.3679.$$

Sigmoid Kernel :-

The sigmoid kernel is given as

$$K(x_i, y_i) = \tanh(k \cdot x_i y_i - \sigma).$$

Non linear Classifiers

- To deal with nonlinear case, the formulation and solution methods employed for the linear case are still applicable.
- Only input data is transformed from its original space into another space (generally, a much higher dimensional space) so that a linear decision boundary can separate Class 1 examples from Class 2 in the transformed space, called the feature space.
- The original data space is known as the input space. Let the set of training (data) examples be

$$\mathcal{D} = \{\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$$

where $\mathbf{x} = [x_1\ x_2\ \ldots\ x_n]^T$.

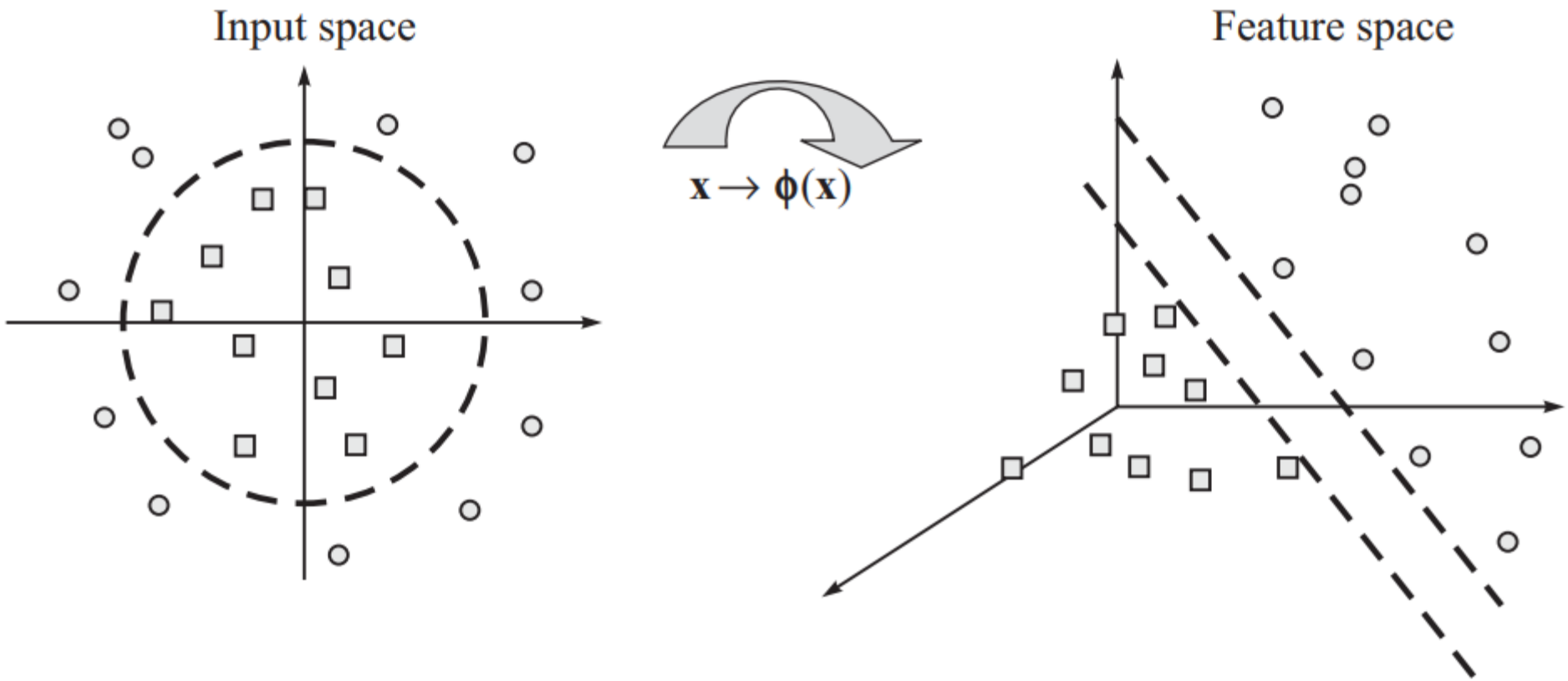
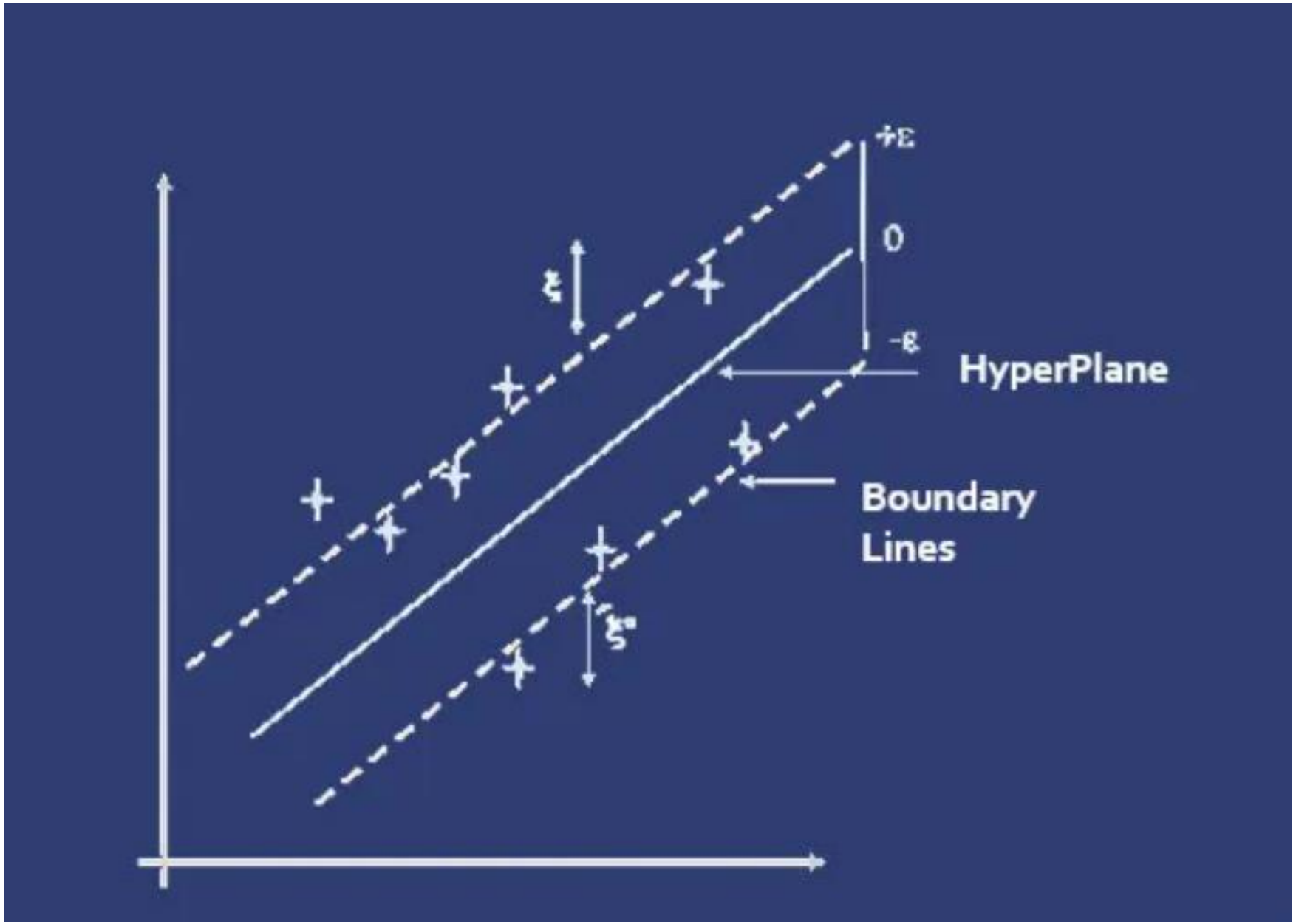


Figure 4.13 Transformation from the input space to feature space

Regression by Support Vectors

- Support vector regression as the name suggests is a regression algorithm that supports both linear and Non linear regressions.
- This method works on the principle of the support vector machine.
- SVR differs from SVM in the way that SVM is a classifier that is used for predicting discrete categorical labels while SVR is a regressor that is used for predicting continuous ordered variables
- In simple regression, the idea is to minimize the error rate while in SVR the idea is to fit the error inside a certain threshold which means, work of SVR is to approximate the best value within a given margin is called ϵ - tube.

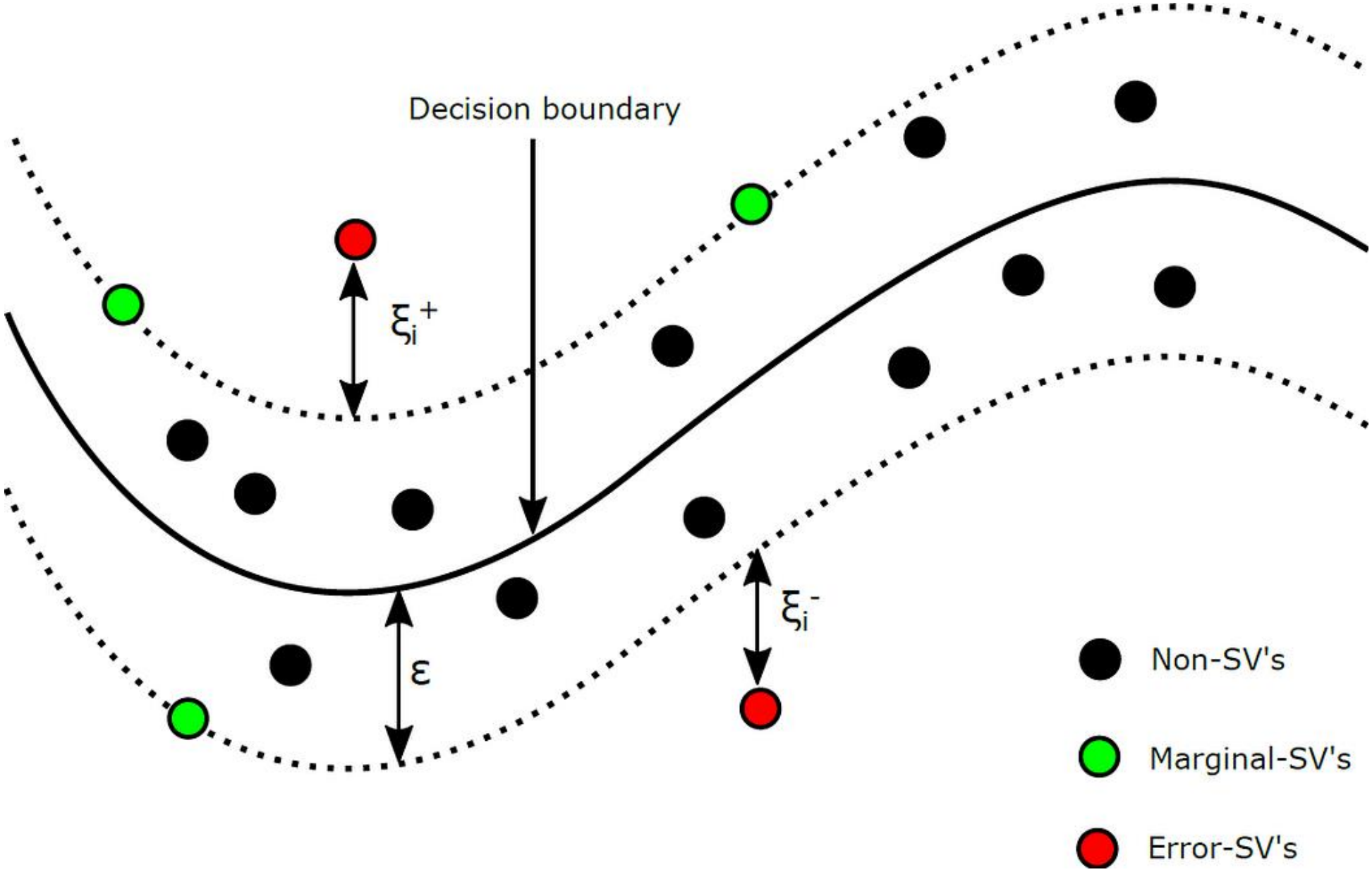

+ε
0
-ε
HyperPlane
Boundary
Lines

Working of SVR

- Given data points, it tries to find the curve. But since it is a regression algorithm instead of using the curve as a decision boundary it uses the curve to find the match between the vector and position of the curve.
- Support vectors helps in determining the closest match between the data points and the function which is used to represent them.

Following are the steps needed in the working of SVR:

- Collection of the training set
- Selection of Kernel along with its parameters and any regularization if required.
- Creation of Correlation matrix.
- Train machine to get the contraction coefficients .
- Create an estimator using the coefficients.

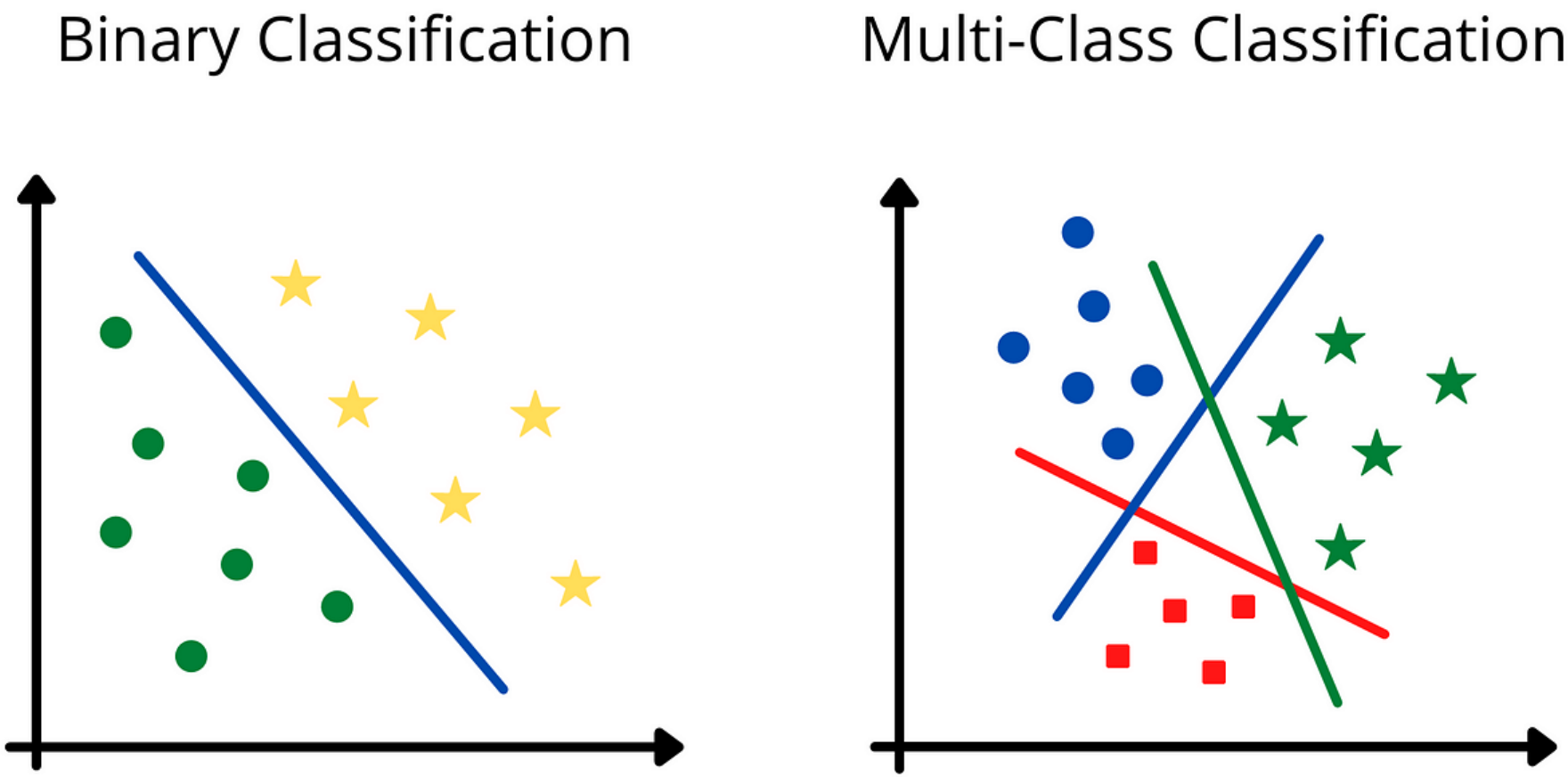

Decision boundary
ξ_i^+
ξ_i^-
ε
Non-SV's
Marginal-SV's
Error-SV's

Decomposing Multiclass Classification Problem into Binary Classification Tasks

What is multiclass classification?

- When we solve a classification problem having only two class labels, then it becomes easy to filter the data, apply any classification algorithm, train the model with filtered data and predict the outcomes.
- But when we have more than two class instances in input train data, then it might get complex to analyse the data, train the model, and predict relatively accurate results.
- To handle these multiple class instances , we use multiclass classification
- Multiclass classification is the classification technique that allows us to categorize the test data into multiple class labels present in trained data as model prediction.

Binary vs Multi class classification



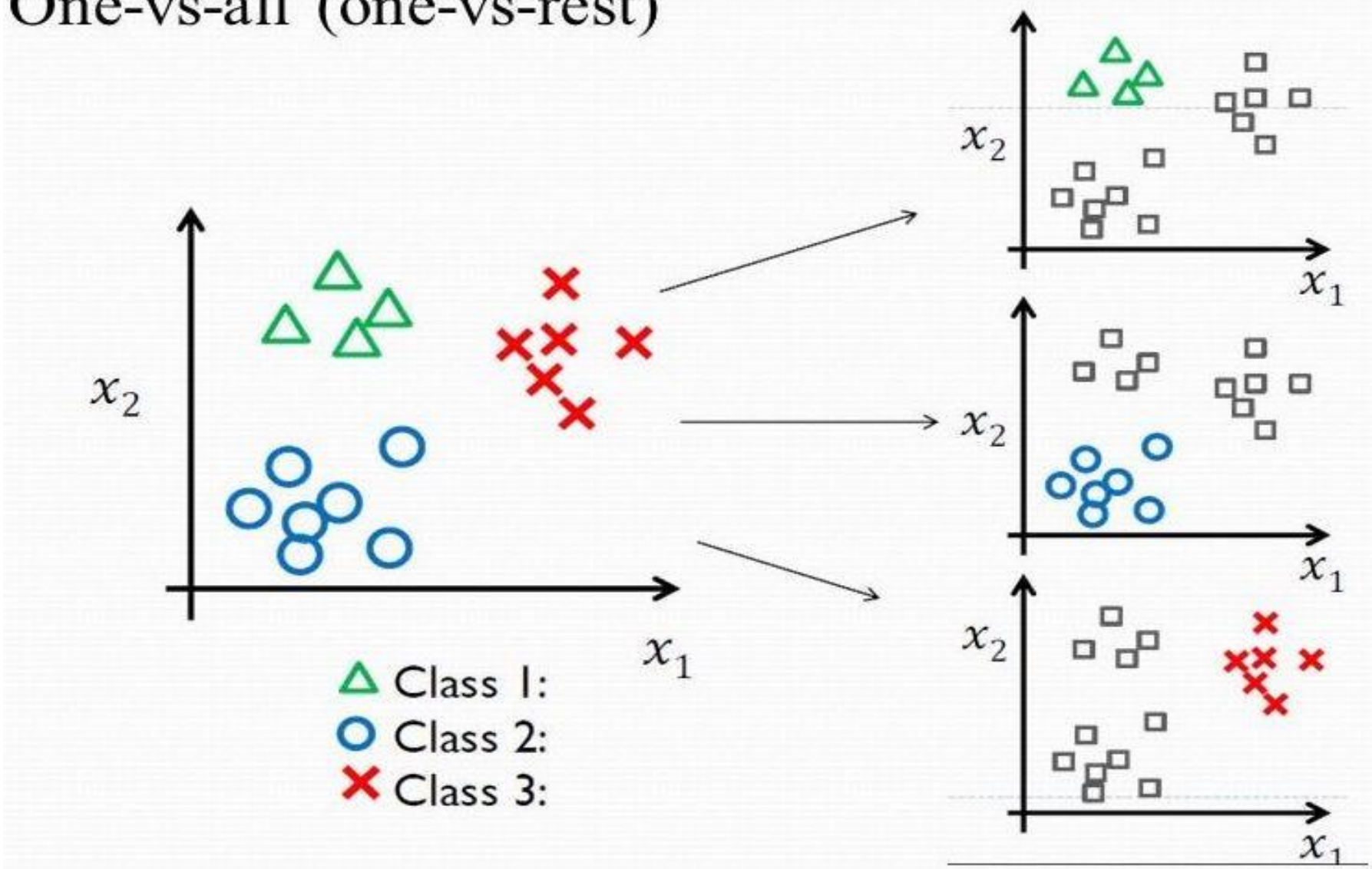
Multiclass Classification

- Multiple class labels are present in the dataset
- The number of classifier models depends on the classification technique we are applying to.
- There are two popular methods in the category of indirect methods:
  - One-Against-All (OAA) / One vs All – N class instances then N binary classifiers
  - One-Against-One (OAO) /One vs One – N class instances then N*(N-1)/2 binary classifier models.
- In the general case, both OAA and OAO are special cases of error correcting-output codes that decompose a multiclass problem to a set of two class problems.
- Example: check whether the fruit is apple, orange or banana.

One-Against-All (OAA)

- In One vs All classification, for the N – class instances dataset, we must generate the N-binary classifier models
- The number of class labels present in the dataset and the number of generated binary classifiers must be same.
- Consider we have three classes for example Green, Blue and Red
- Now, we create three classifiers here for three respective classes
  - classifier 1: [Green] vs [Red, Blue]
  - classifier 2: [Blue] vs [Green, Red]
  - classifier 3: [Red] vs [Blue, Green]

One-vs-all (one-vs-rest)



One vs. All (One-vs-Rest)

- You can see that there are three class labels **Green**, **Blue,** and **Red** present in the dataset. Now we must create a training dataset for each class.

Features			Classes
x1	x2	x3	G
x4	x5	x6	B
x7	x8	x9	R
x10	x11	x12	G
x13	x14	x15	B
x16	x17	x18	R

Class 1 :- Green

Class 2 :- Blue

Class 3 :- Red

One vs. All (One-vs-Rest)

Main Dataset

Features			Classes
x1	x2	x3	G
x4	x5	x6	B
x7	x8	x9	R
x10	x11	x12	G
x13	x14	x15	B
x16	x17	x18	R

Class 1 :- Green Class 2 :- Blue Class 3 :- Red

Training Dataset 1
Class :- Green

Features			Green
x1	x2	x3	**+1**
x4	x5	x6	-1
x7	x8	x9	-1
x10	x11	x12	**+1**
x13	x14	x15	-1
x16	x17	x18	-1

One vs. All (One-vs-Rest)

Training Dataset 2
Class :- Blue

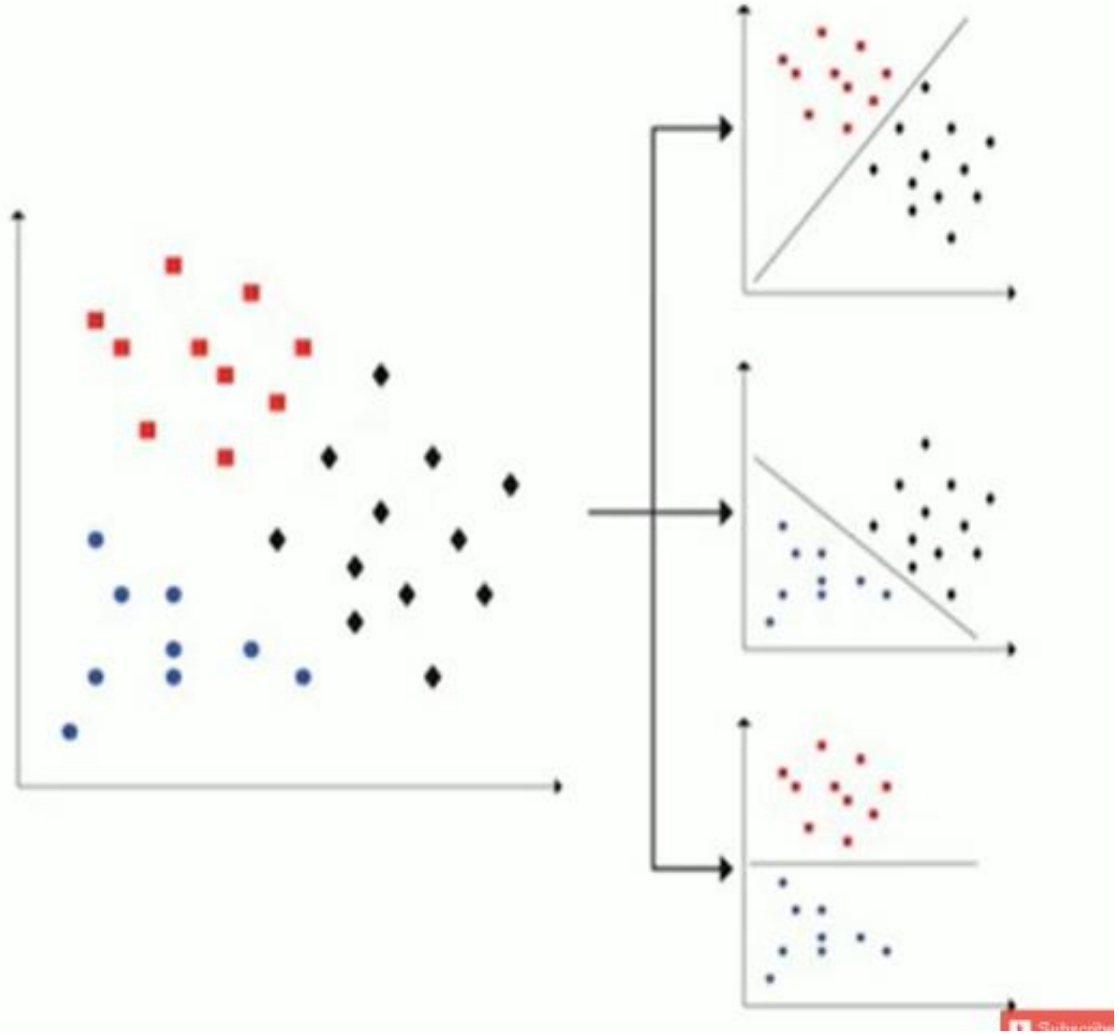
Features			Blue
x1	x2	x3	-1
x4	x5	x6	**+1**
x7	x8	x9	-1
x10	x11	x12	-1
x13	x14	x15	**+1**
x16	x17	x18	-1

Training Dataset 3
Class :- Red

Features			Red
x1	x2	x3	-1
x4	x5	x6	-1
x7	x8	x9	**+1**
x10	x11	x12	-1
x13	x14	x15	-1
x16	x17	x18	**+1**

One vs. One (OvO)

- In One-vs-One classification, for the **N-class** instances dataset, we must generate the **N* (N-1)/2** binary classifier models.
- Using this classification approach, we split the primary dataset into one dataset for each class opposite to every other class.
- Taking the above example, we have a classification problem having three types: **Green**, **Blue**, and **Red (N=3)**



One vs. One (OvO)

- We divide this problem into **N* (N-1)/2 = 3** binary classifier problems:
  - Classifier 1: Green vs. Blue
  - Classifier 2: Green vs. Red
  - Classifier 3: Blue vs. Red
- Each binary classifier predicts one class label.
- When we input the test data to the classifier, then the model with the majority counts is concluded as a result.

VARIANTS OF BASIC SVM TECHNIQUES

1. Linear SVM(LSVM):
   - Which finds a linear hyperplane that best separates data points of different classes.
2. Non Linear SVM (NLSVM)
   - It uses Kernel functions to map data into a higher dimensional space, where a linear hyperplane can separate non linearly separable data.
3. Multiclass SVM
   - The basic SVM is binary by design. Multiclass SVM extends this to handle multiple classes using techniques such as One vs all or One vs One classification

VARIANTS OF BASIC SVM TECHNIQUES (Cont..)

4. Weighted SVM

- It assigns different weights to different classes, making it more tolerant to class imbalance. It is useful when some classes have significantly fewer samples than others.

5. Soft Margin SVM

- In a basic SVM, the goal is to find a hard margin that perfectly separates data. Soft margin SVM allows for some misclassification, introducing a trade off between maximizing margin and minimizing misclassification errors. This is particularly useful when the data is not perfectly separable.

6. SVM for Anomaly Detection

- It is used for Anomaly Detection by treating the majority class as normal and the minority classes as anomalies. This is useful in situations where the focus is on detecting rare events or outliers.

ENSEMBLE METHODS

- Bagging
- Committee Machines & Stacking
- Boosting
- Gradient Boosting
- Random Forest

Introduction to Ensemble methods

- Ensemble methods are techniques that aim at improving the accuracy of results in models by combining multiple models instead of using a single model. The combined models increase the accuracy of the results significantly.

Suppose you are a movie director and you have created a short movie on a very important and interesting topic. Now, you want to take preliminary feedback (ratings) on the movie before making it public. What are the possible ways by which you can do that?

A: You may ask one of your friends to rate the movie for you.

B: Another way could be by asking 5 colleagues of yours to rate the movie.

C: How about asking 50 people to rate the movie?

Simple Ensemble Techniques

1.Max Voting:

In this technique, multiple models are used to make predictions for each data point. The predictions by each model are considered as a ‘vote’. The predictions which we get from the majority of the models are used as the final prediction.

The result of max voting would be something like this:

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4

2. **Averaging:**

Multiple predictions are made for each data point in averaging. In this method, we take an average of predictions from all the models and use it to make the final prediction.

For example, in the below case, the averaging method would take the average of all the values.

i.e. (5+4+5+4+4)/5 = 4.4

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4.4

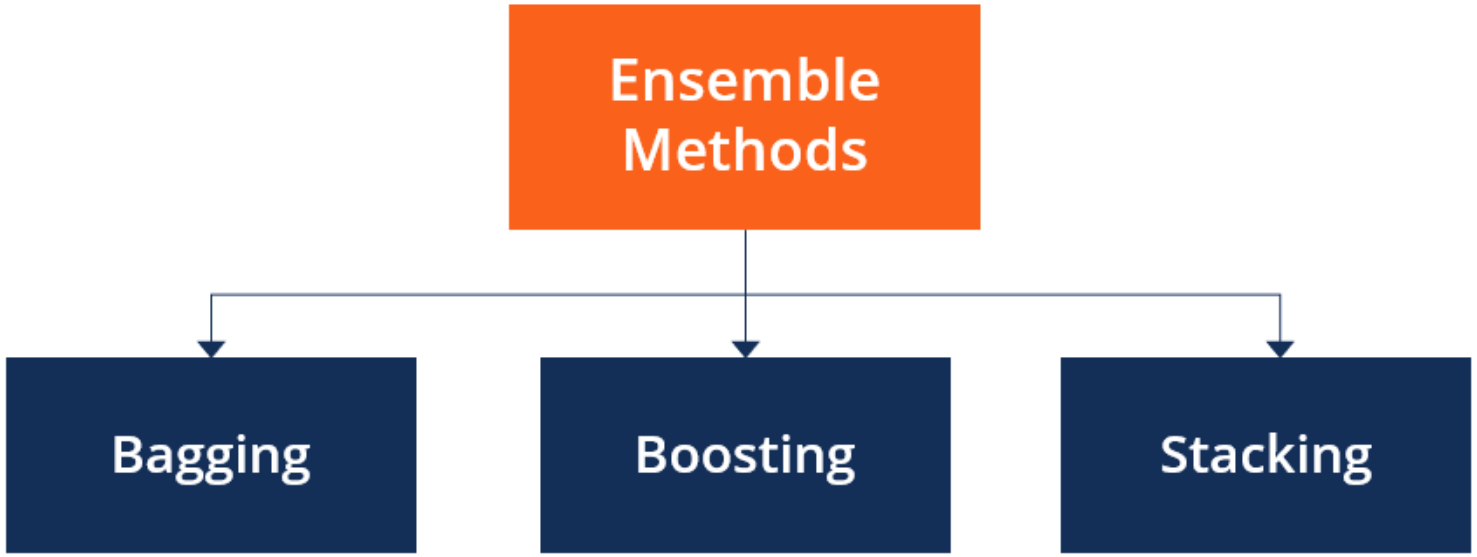
3. Weighted Average:

All models are assigned different weights defining the importance of each model for prediction

The result is calculated as [(5*0.23) + (4*0.23) + (5*0.18) + (4*0.18) + (4*0.18)] = 4.41.

	Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
weight	0.23	0.23	0.18	0.18	0.18	
rating	5	4	5	4	4	4.41

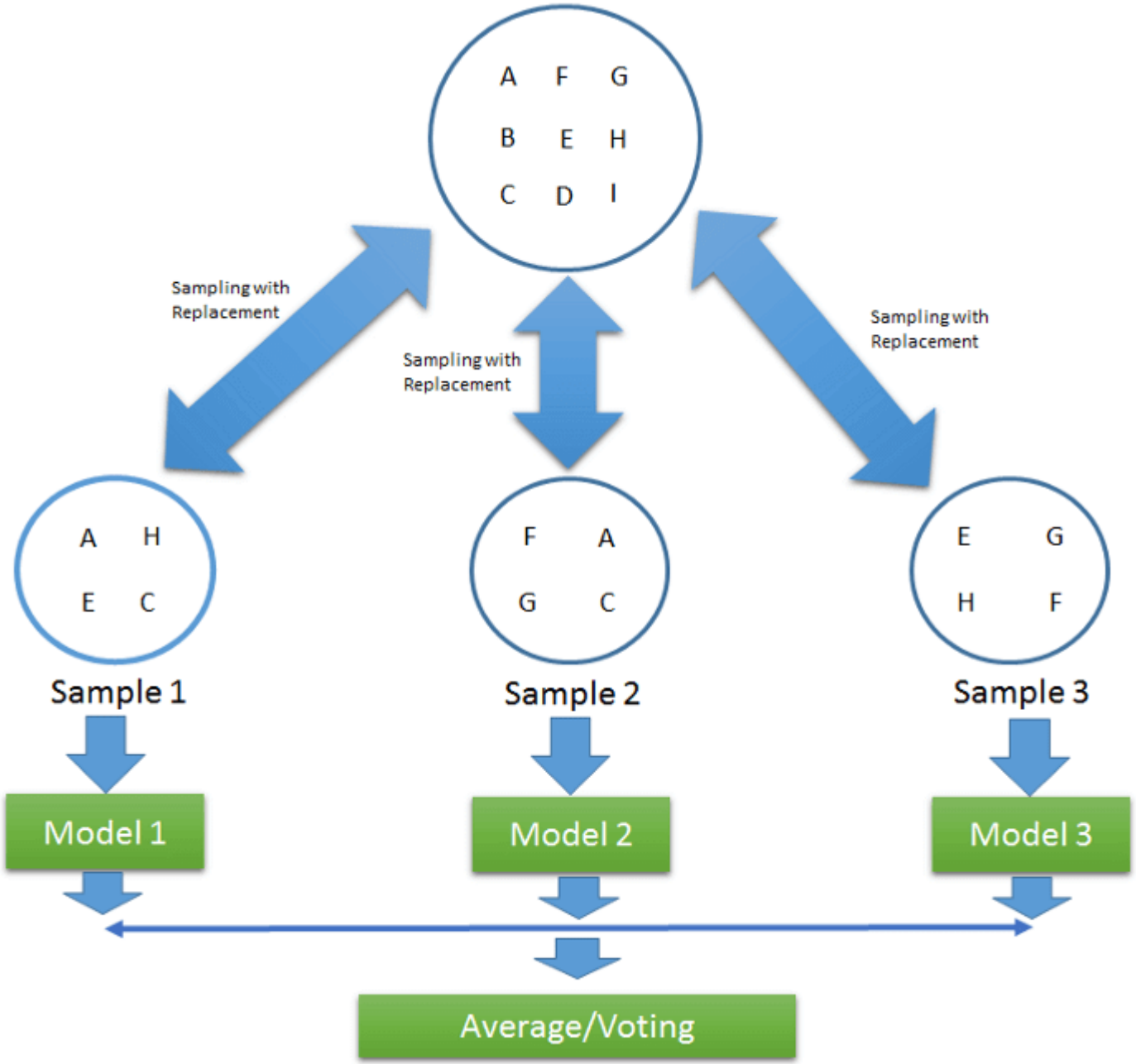
Advanced ensemble learning technique:



1. Bagging:

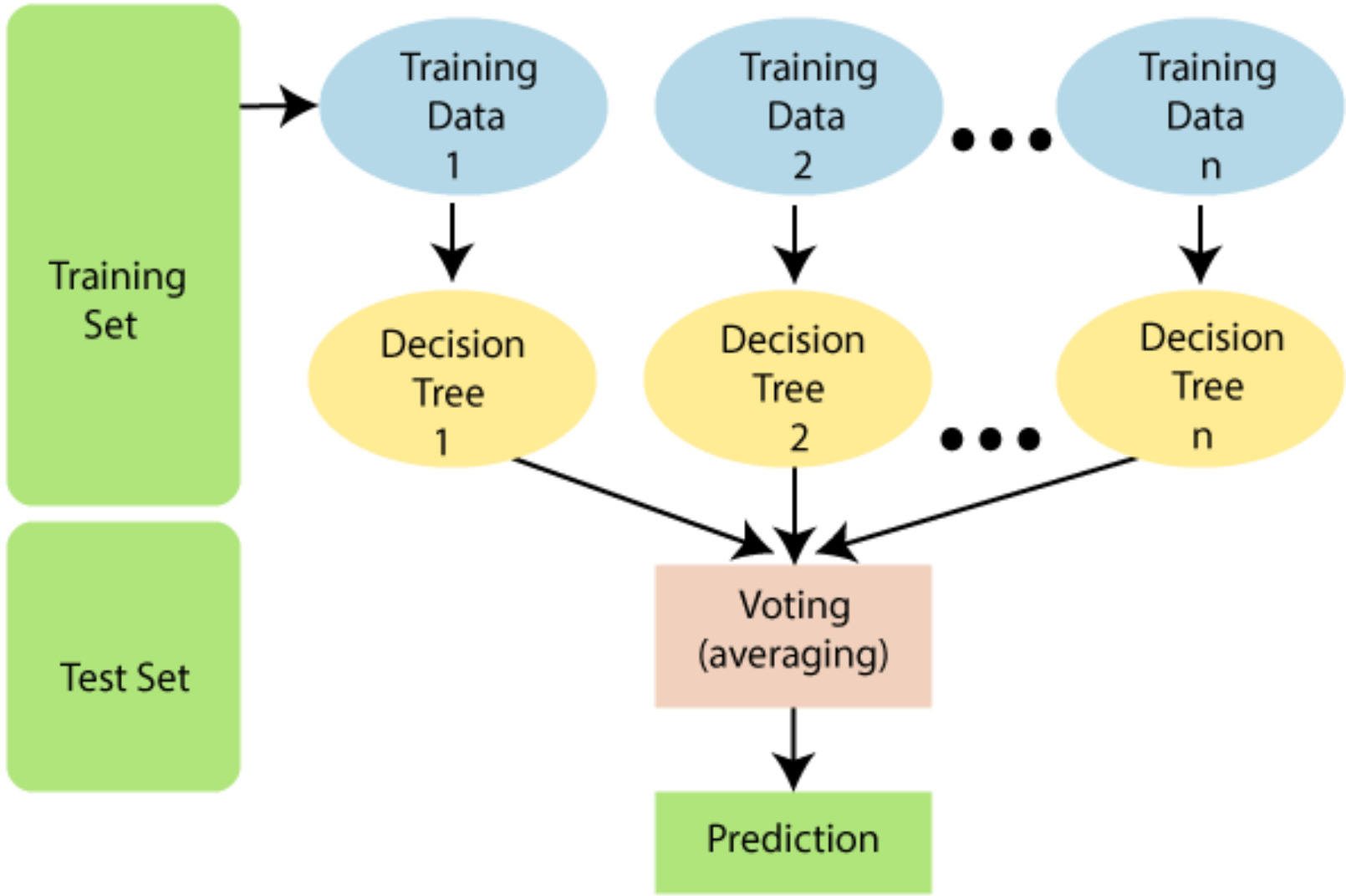
- The primary goal of "bagging" or "bootstrap aggregating" ensemble method is to minimize variance errors in decision trees.
- The objective here is to randomly create samples of training datasets with replacement (subsets of the training data).
- The subsets are then used for training decision trees or models. Consequently, there is a combination of multiple models, which reduces variance, as the average prediction generated from different models is much more reliable and robust than a single model or a decision tree.

Example: Random Forest


A F G
B E H
C D I
Sampling with Replacement
Sampling with Replacement
Sampling with Replacement
A H
E C
F A
G C
E G
H F
Sample 1
Sample 2
Sample 3
Model 1
Model 2
Model 3
Average/Voting

Random Forest

- ***Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.***
- Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
- **The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**


Training Set
Test Set
Training Data 1
Training Data 2
Training Data n
Decision Tree 1
Decision Tree 2
Decision Tree n
Voting (averaging)
Prediction

Assumptions of Random Forest:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations

Why use Random Forest?

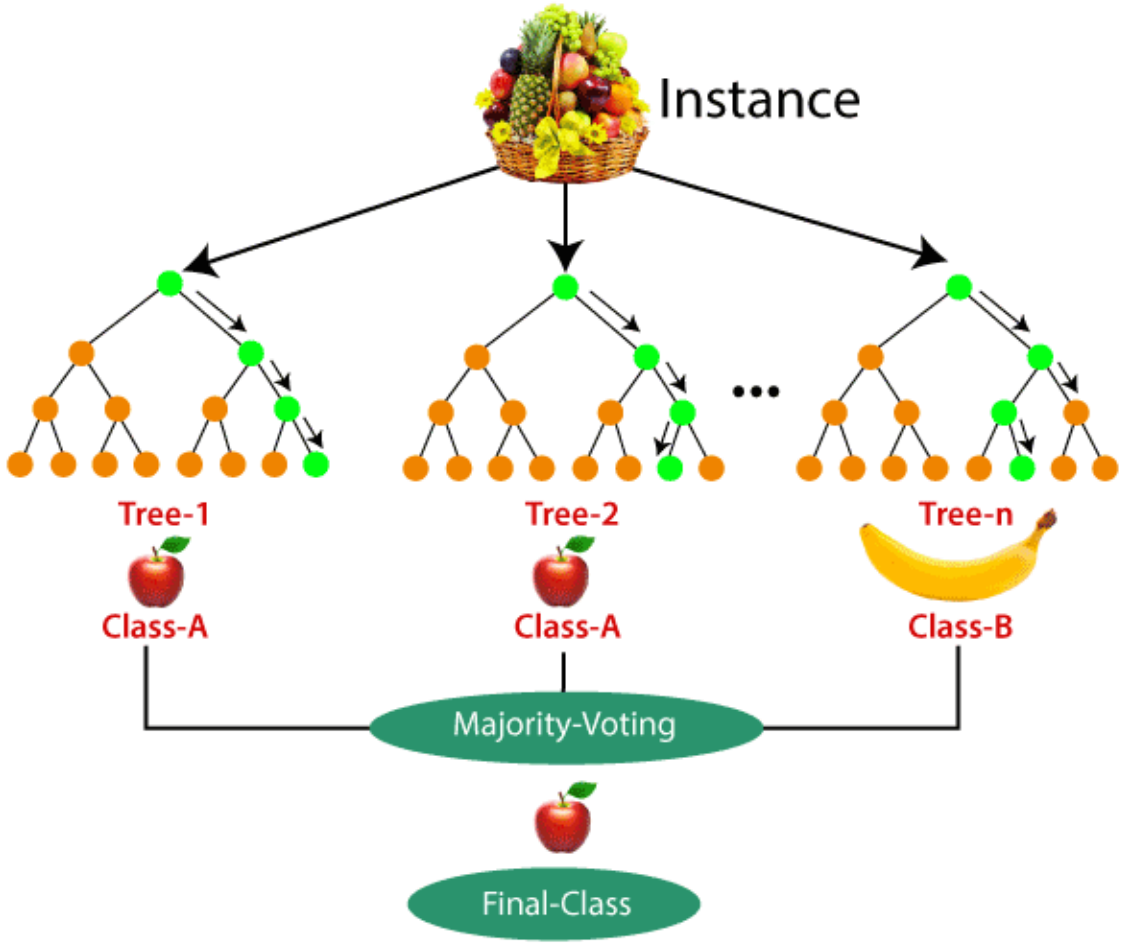
- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

- The Working process can be explained in the below steps and diagram:
- **Step-1:** Select random K data points from the training set.
- **Step-2:** Build the decision trees associated with the selected data points (Subsets).
- **Step-3:** Choose the number N for decision trees that you want to build.
- **Step-4:** Repeat Step 1 & 2.
- **Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision



Applications of Random Forest:

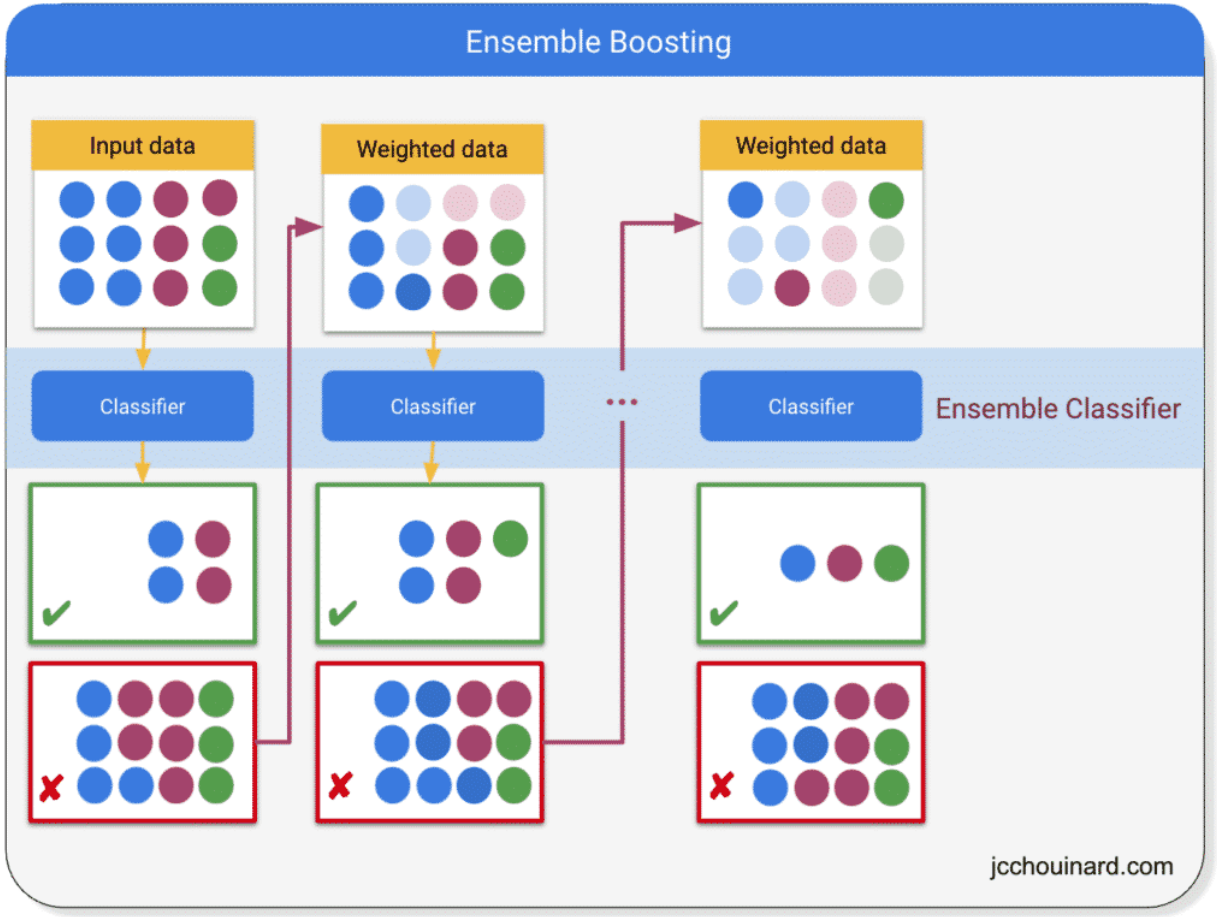
1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine**: With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use**: We can identify the areas of similar land use by this algorithm.
4. **Marketing**: Marketing trends can be identified using this algorithm.

Advantages of Random Forest:

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

2. Boosting:

- Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model. The succeeding models are dependent on the previous model



Boosting Algorithms:

- AdaBoost
- GBM
- XGBM
- Light GBM
- CatBoost

How boosting works?

1. A subset is created from the original dataset. Initially, all data points are given equal weights.
2. A base model is created on this subset. This model is used to make predictions on the whole dataset
3. Errors are calculated using the actual values and predicted values.
4. The observations which are incorrectly predicted, are given higher weights.
5. Another model is created and predictions are made on the dataset.
6. Similarly, multiple models are created, each correcting the errors of the previous model.
7. The final model (strong learner) is the weighted mean of all the models (weak learners).

Bagging vs Boosting

Bagging	Boosting
Various training data subsets are randomly drawn with replacement from the whole training dataset.	Each new subset contains the components that were misclassified by previous models.
Bagging attempts to tackle the over-fitting issue.	Boosting tries to reduce bias.
If the classifier is unstable (high variance), then we need to apply bagging.	If the classifier is steady and straightforward (high bias), then we need to apply boosting.
Every model receives an equal weight.	Models are weighted by their performance.
Objective to decrease variance, not bias.	Objective to decrease bias, not variance.
It is the easiest way of connecting predictions that belong to the same type.	It is a way of connecting predictions that belong to the different types.
Every model is constructed independently.	New models are affected by the performance of the previously developed model.

Gradient Boosting Machine

- Gradient Boosting or GBM is another ensemble machine learning algorithm that works for both regression and classification problems.
- GBM uses the boosting technique, combining a number of weak learners to form a strong learner.
- Regression trees used as a base learner, each subsequent tree in series is built on the errors calculated by the previous tree.

We will use a simple example to understand the GBM algorithm. We have to predict the age of a group of people using the below data:

ID	Married	Gender	Current City	Monthly Income	Age (target)
1	Y	M	A	51,000	35
2	N	F	B	25,000	24
3	Y	M	A	74,000	38
4	N	F	A	29,000	30
5	N	F	B	37,000	33

1. The mean age is assumed to be the predicted value for all observations in the dataset.
2. The errors are calculated using this mean prediction and actual values of age.

ID	Married	Gender	Current City	Monthly Income	Age (target)	Mean Age (prediction 1)	Residual 1
1	Y	M	A	51,000	35	32	3
2	N	F	B	25,000	24	32	-8
3	Y	M	A	74,000	38	32	6
4	N	F	A	29,000	30	32	-2
5	N	F	B	37,000	33	32	1

3. A tree model is created using the errors calculated above as target variable. Our objective is to find the best split to minimize the error.

4. The predictions by this model are combined with the predictions 1.

ID	Age (target)	Mean Age (prediction 1)	Residual 1 (new target)	Prediction 2	Combine (mean+pred2)
1	35	32	3	3	35
2	24	32	-8	-5	27
3	38	32	6	3	35
4	30	32	-2	-5	27
5	33	32	1	3	35

5. This value calculated above is the new prediction.

6. New errors are calculated using this predicted value and actual value.

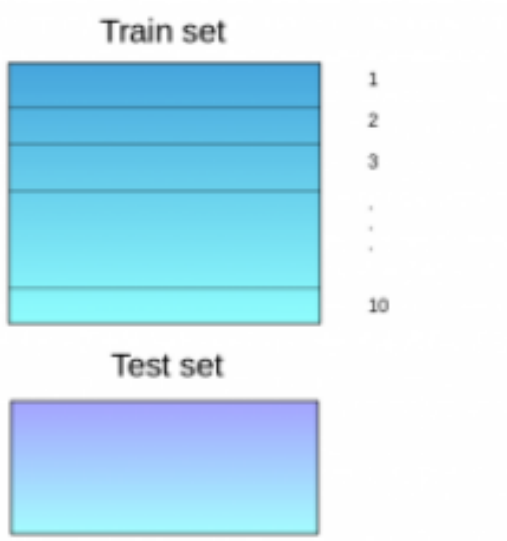
ID	Age (target)	Mean Age (prediction 1)	Residual 1 (new target)	Prediction 2	Combine (mean+pred2)	Residual 2 (latest target)
1	35	32	3	3	35	0
2	24	32	-8	-5	27	-3
3	38	32	6	3	35	-3
4	30	32	-2	-5	27	3
5	33	32	1	3	35	-2

7. Steps 2 to 6 are repeated till the maximum number of iterations is reached (or error function does not change).

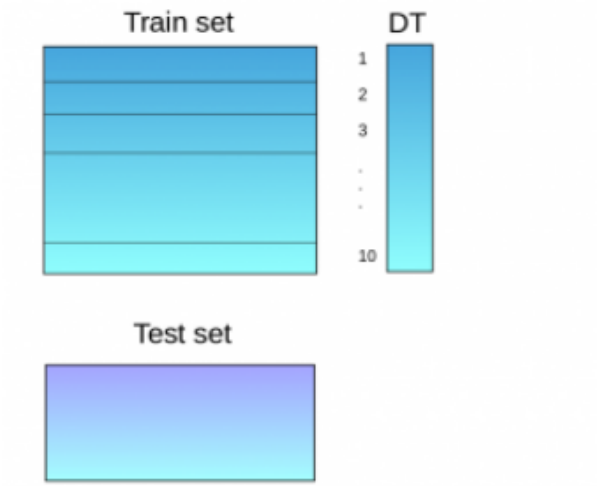
3. Stacking

- Stacking is an ensemble learning technique that uses predictions from multiple models (for example decision tree, KNN or SVM) to build a new model. This model is used for making predictions on the test set.
- Below is a step-wise explanation for a simple stacked ensemble:

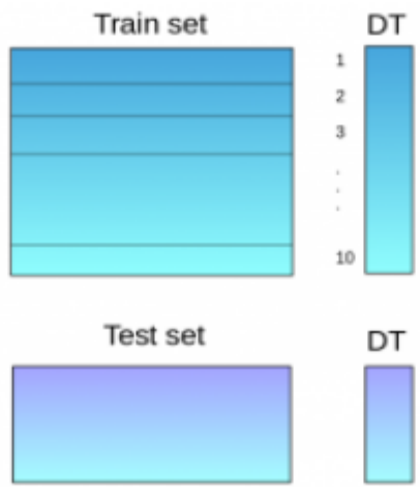
1. The train set is split into 10 parts.



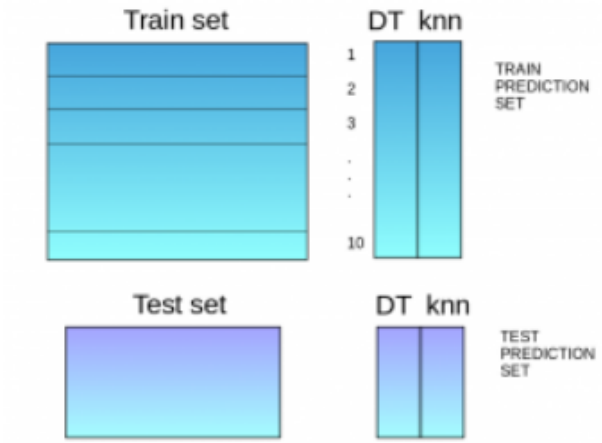
2. A base model (suppose a decision tree) is fitted on 9 parts and predictions are made for the 10th part. This is done for each part of the train set.



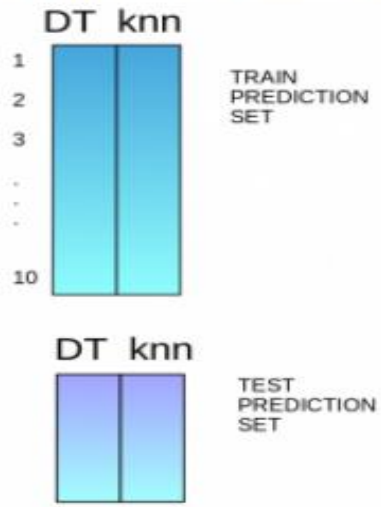
3. The base model (in this case, decision tree) is then fitted on the whole train dataset.
4. Using this model, predictions are made on the test set.



5. Steps 2 to 4 are repeated for another base model (say knn) resulting in another set of predictions for the train set and test set.



6. The predictions from the train set are used as features to build a new model.



7. This model is used to make final predictions on the test prediction set.