Machine Learning

(BE Computer 2019 PAT)

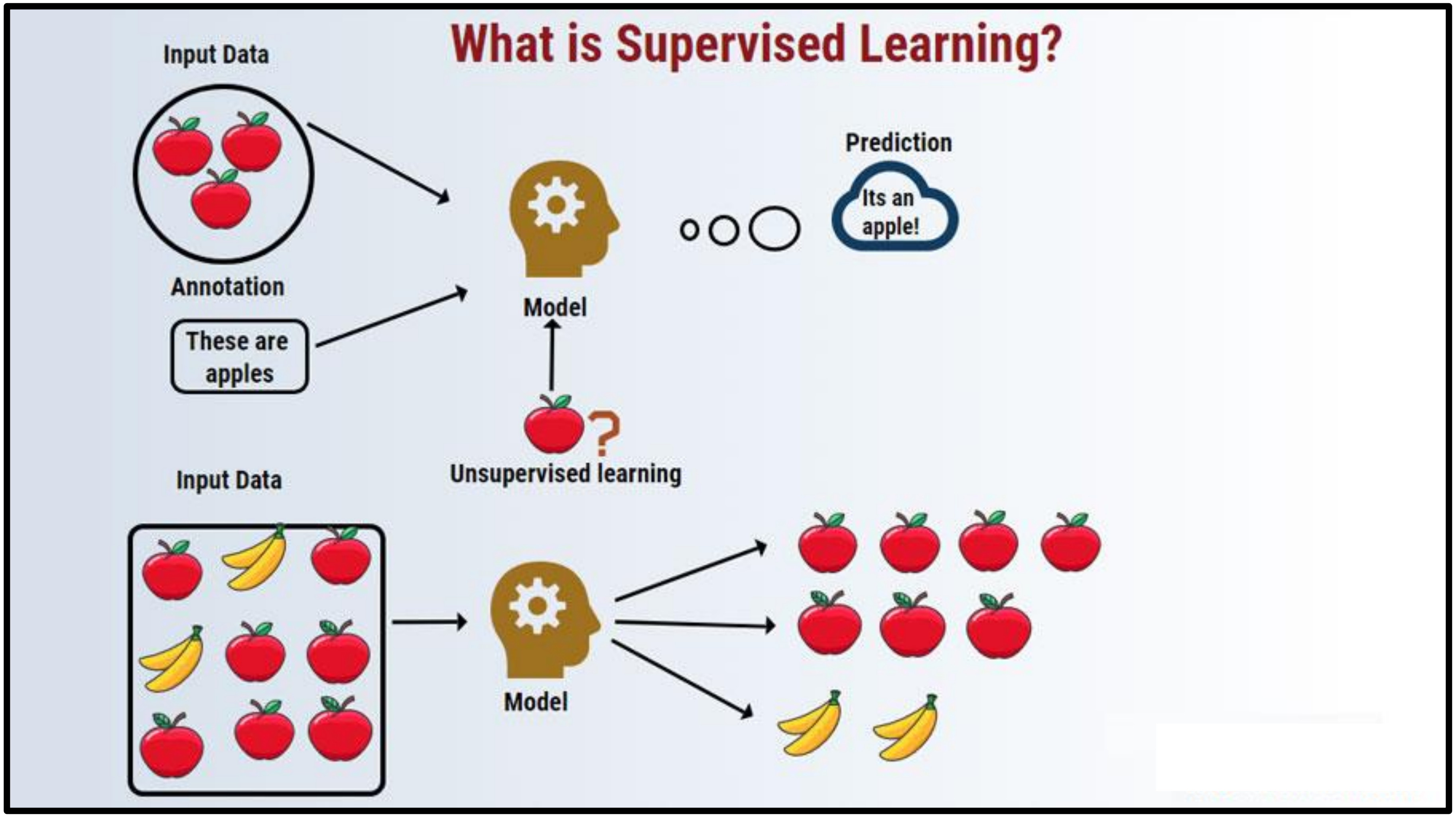
SYLLABUS

Classification: K-nearest neighbour, Support vector machine.
Ensemble Learning: Bagging, Boosting, Random Forest,
Adaboost.
Binary-vs-Multiclass Classification, Balanced and Imbalanced
Multiclass Classification
Problems,Variants of Multiclass Classification: One-vs-One
and One-vs-All

Evaluation Metrics and Score: Accuracy, Precision, Recall,
Fscore, Cross-validation, Micro-
Average Precision and Recall, Micro-Average F-score, Macr
Average Precision and Recall,

Macro-Average F-score.

SUPERVISED LEARNING



K-nearest neighbour

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

- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

K-nearest neighbour

What is KNN?

K Nearest Neighbour is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure. It is mostly used to classifies a data point based on how its neighbours are classified.

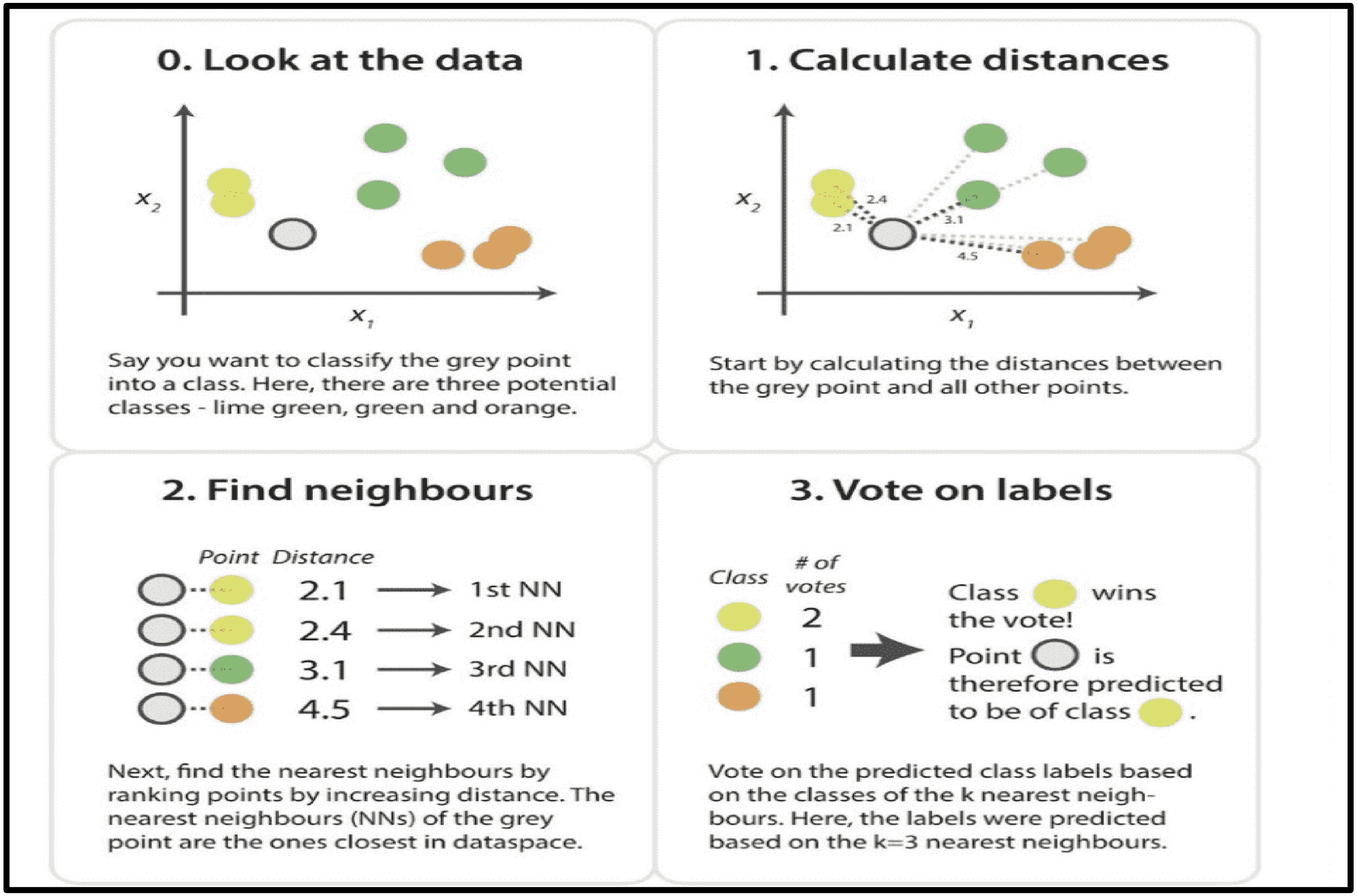
$$d(x,y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

WORKING OF K-nearest neighbour

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

K-nearest neighbour Example



0. Look at the data

Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

1. Calculate distances

Start by calculating the distances between the grey point and all other points.

2. Find neighbours

Point	*Distance*		
○··●	2.1	⟶	1st NN
○··●	2.4	⟶	2nd NN
○··●	3.1	⟶	3rd NN
○··●	4.5	⟶	4th NN

Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

3. Vote on labels

Class	*# of votes*
●	2
●	1
●	1

Class ● wins the vote!

Point ○ is therefore predicted to be of class ● .

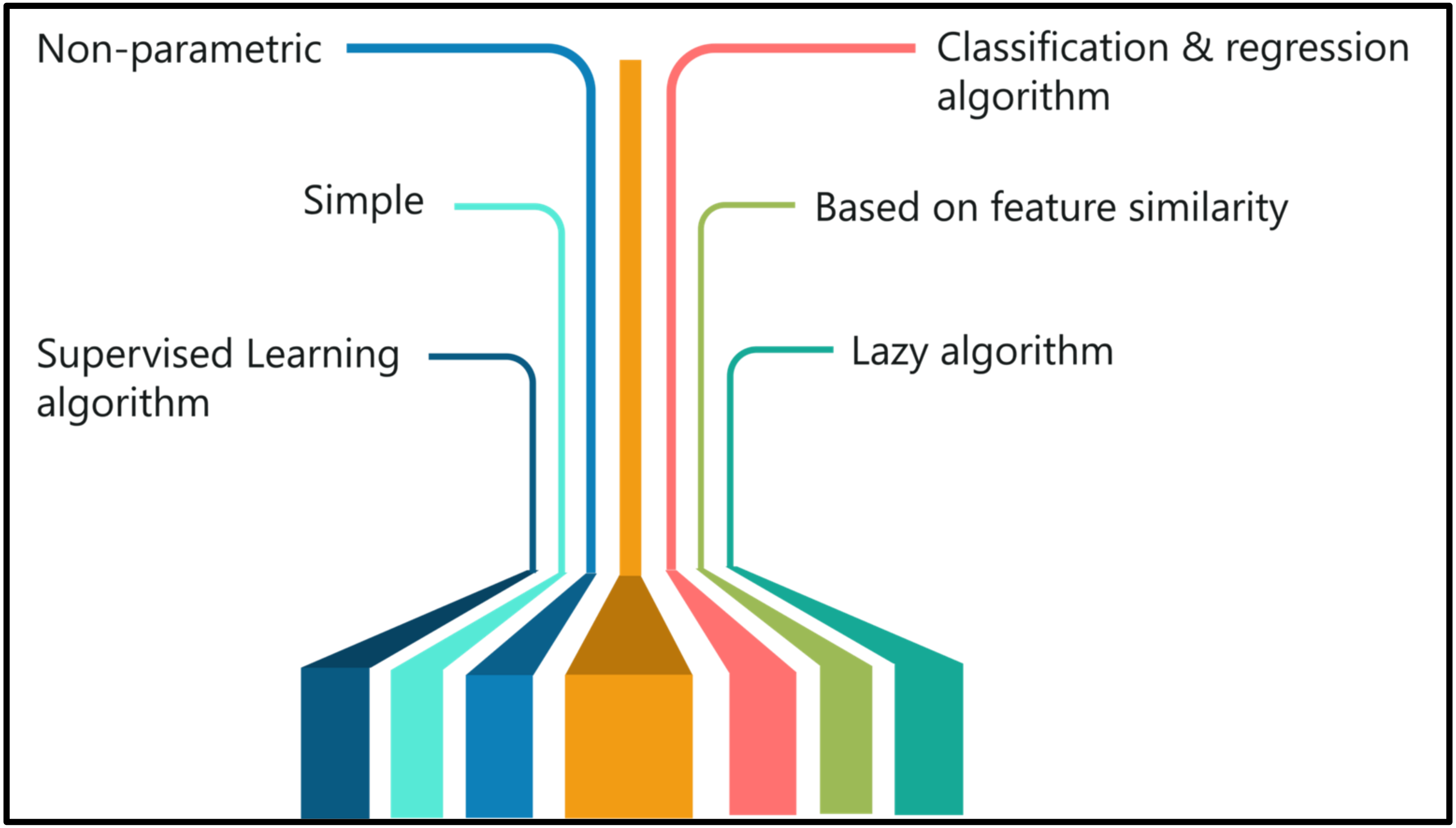
Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

How to select the value of K in the K-NN Algorithm?

- Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
    - A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
    - Large values for K are good, but it may find some difficulties.

K-nearest neighbour Features


Non-parametric
Simple
Supervised Learning algorithm
Classification & regression algorithm
Based on feature similarity
Lazy algorithm

ADVANTAGES

1. **No Training Period**- KNN modeling does not include training period as the data itself is a model which will be the reference for future prediction and because of this it is very time efficient in term of improvising for a random modeling on the available data.

2. **Easy Implementation**- KNN is very easy to implement as the only thing to be calculated is the distance between different points on the basis of data of different features and this distance can easily be calculated using distance formula such as- Euclidian or Manhattan

3. As there is **no training period** thus new data can be added at any time since it wont affect the model.

DISADVANTAGES

1. **Does not work well with large dataset** as calculating distances between each data instance would be very costly.

2. **Does not work well with high dimensionality** as this will complicate the distance calculating process to calculate distance for each dimension.

3. **Sensitive to noisy and missing data**

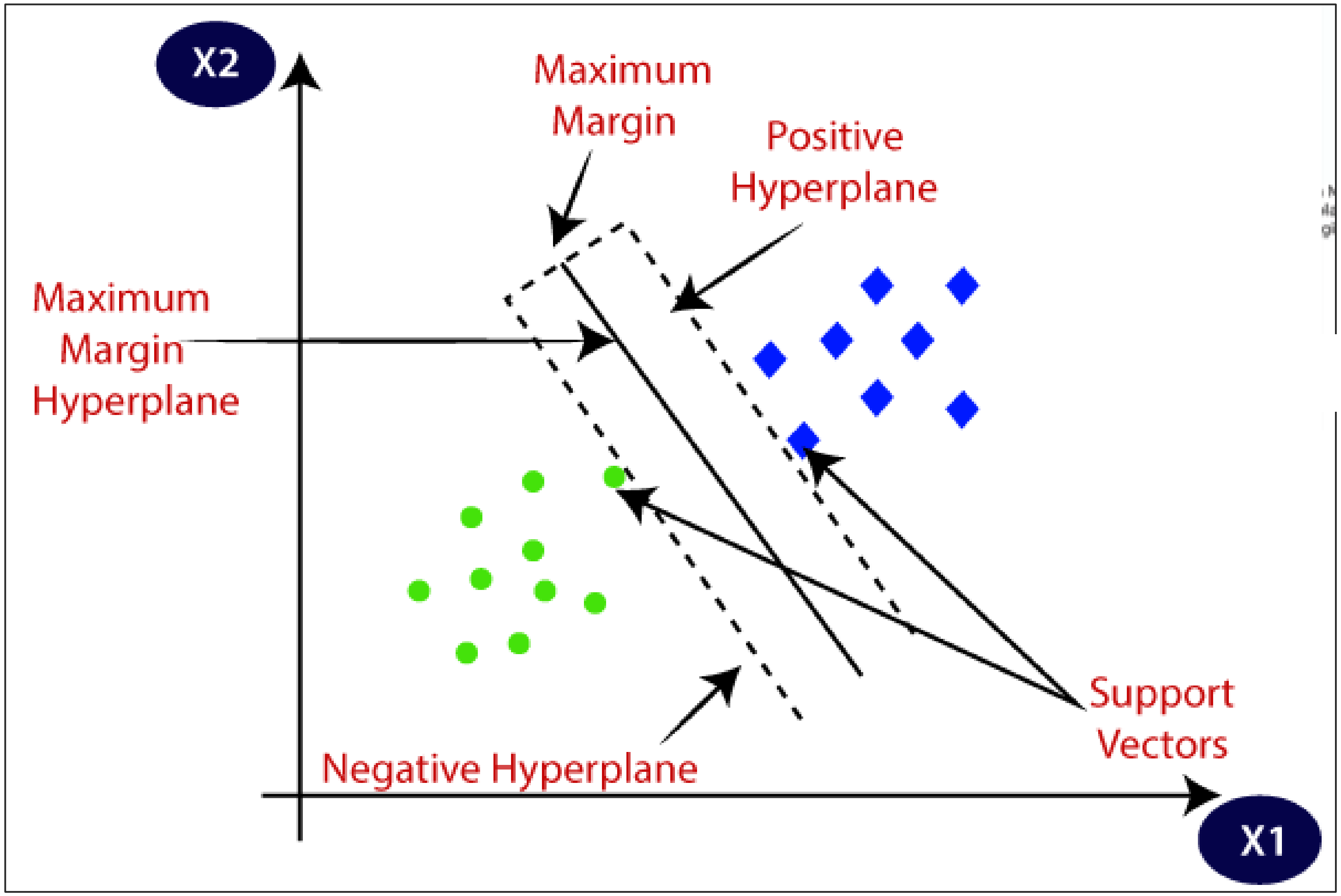
4. **Feature Scaling-** Data in all the dimension should be scaled (normalized and standardized) properly .

Support vector machine

Support vector machine

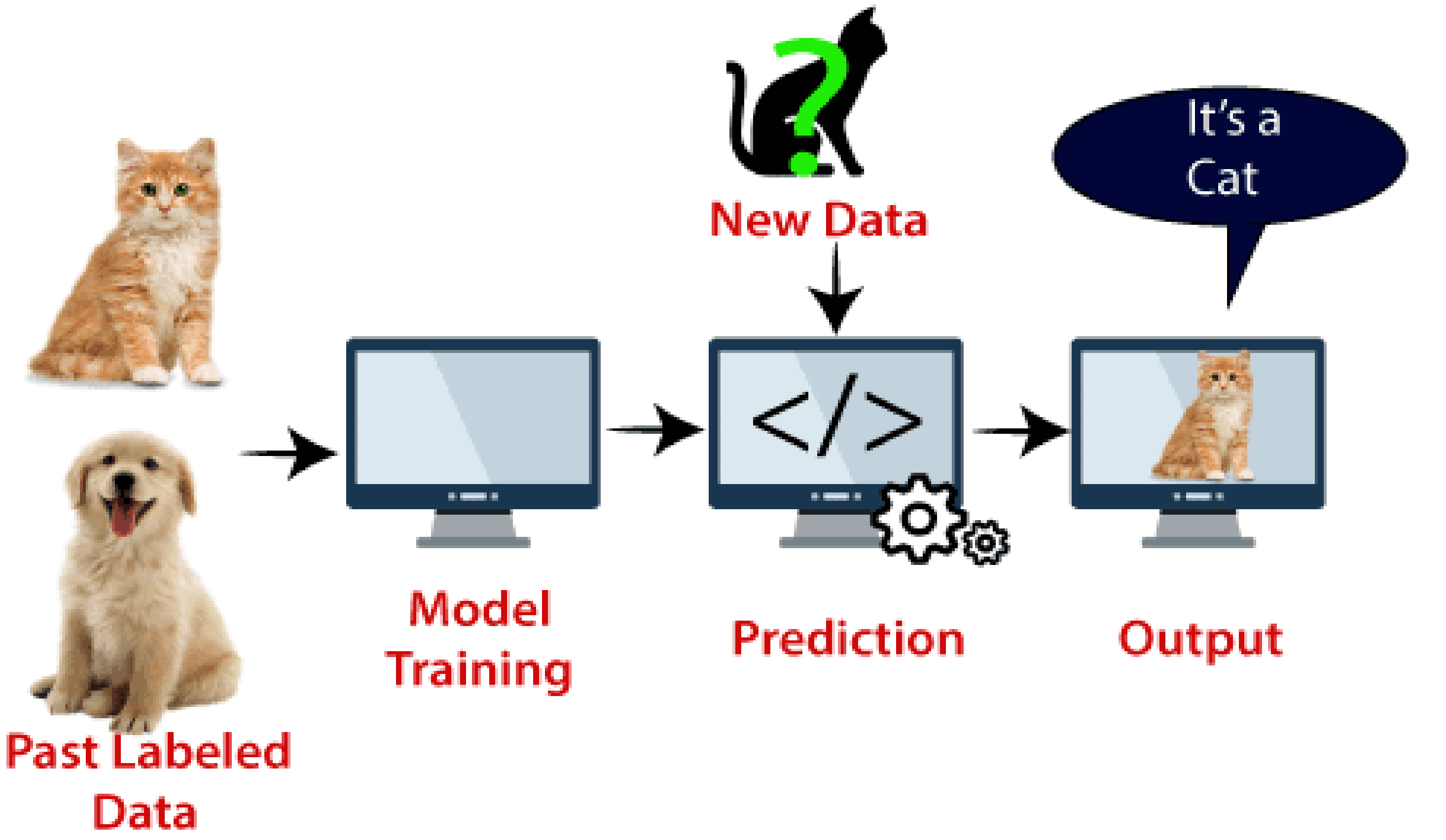
- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.
- However, primarily, it is used for Classification problems in Machine Learning.
- The goal of the SVM algorithm is to create the best line or decision 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:



Example

- SVM can be understood with the example that we have used in the KNN classifier.
- Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm.
- We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature.
- So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog.
- On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of Support vector machine

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.

Ensemble Learning

- Ensemble learning is a machine learning technique that aggregates two or more learners (e.g. regression models, neural networks) in order to produce better predictions.

- In other words, an ensemble model combines several individual models to produce more accurate predictions than a single model alone.

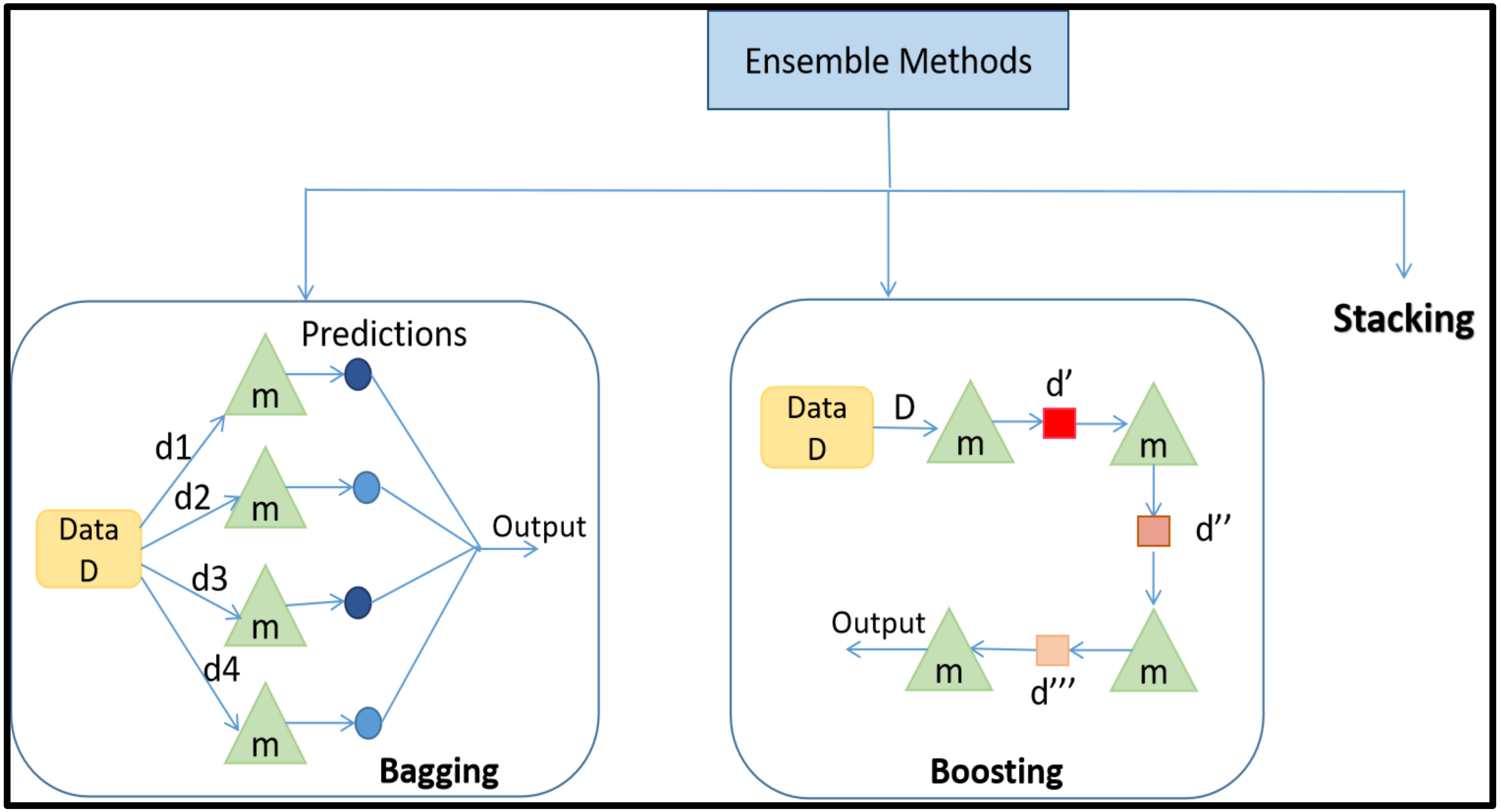
- At times, sources may refer to this technique as committee-based learning. Ensemble learning rests on the principle that a collectivity of learners yields greater overall accuracy than an individual learner.

- Ensemble learning is a widely-used and preferred machine learning technique in which multiple individual models, often called base models, are combined to produce an effective optimal prediction model.

- The Random Forest algorithm is an example of ensemble learning.

- Bagging and Boosting are two types of Ensemble Learning. These two decrease the variance of a single estimate as they combine several estimates from different models. So the result may be a model with higher stability.

Ensemble Learning



- Bagging: It is a homogeneous weak learners’ model that learns from each other independently in parallel and combines them for determining the model average.

- Boosting: It is also a homogeneous weak learners’ model but works differently from Bagging. In this model, learners learn sequentially and adaptively to improve model predictions of a learning algorithm.

Bagging

- It is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression.

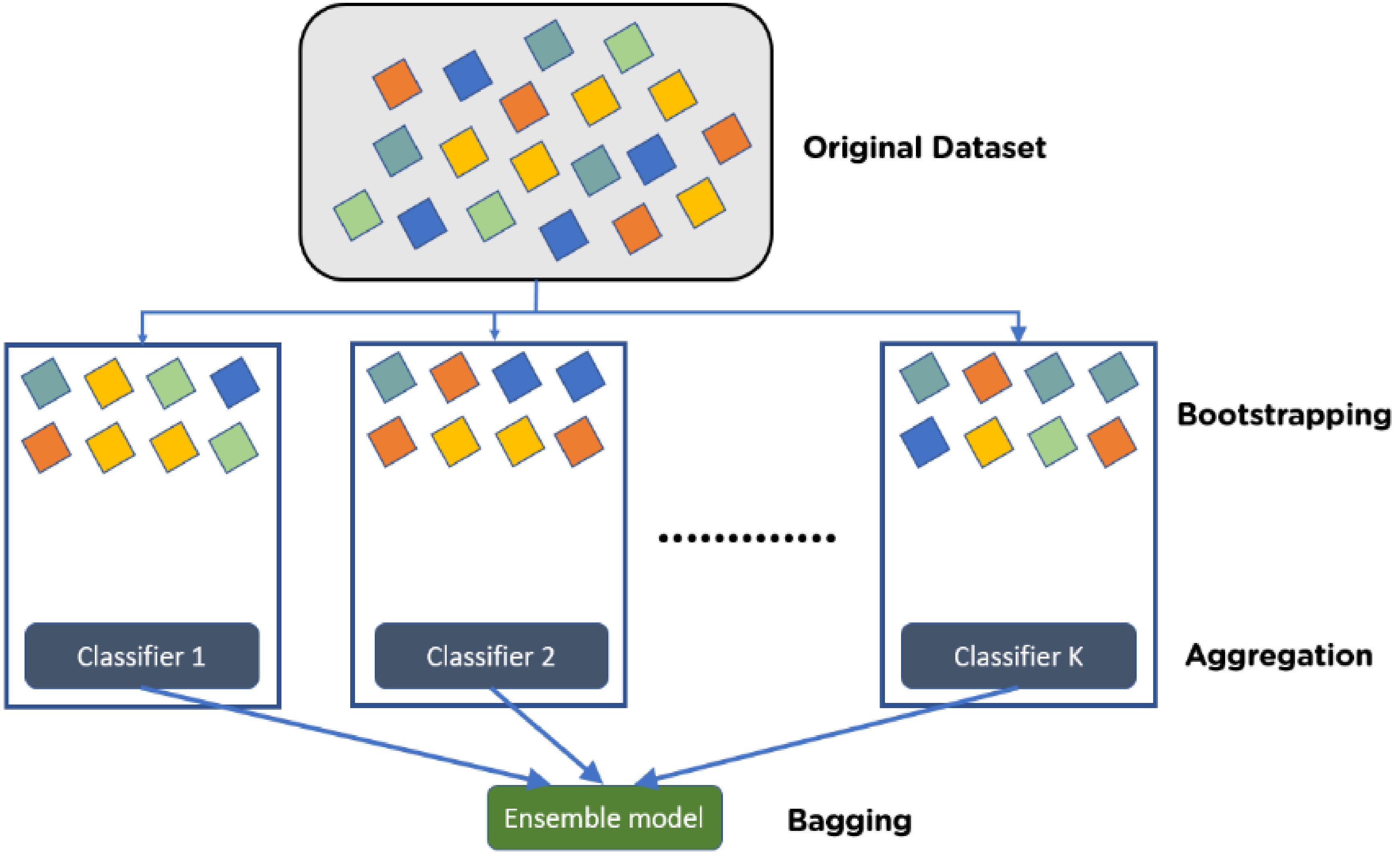
- It is used to deal with bias-variance trade-offs and reduces the variance of a prediction model.

- It decreases the variance and helps to avoid overfitting.

- It is usually applied to decision tree methods.

- Bagging is a special case of the model averaging approach.

Bagging


Original Dataset
Bootstrapping
Aggregation
Classifier 1
Classifier 2
Classifier K
Ensemble model
Bagging

Steps to Perform Bagging

Step 1: Multiple subsets are created from the original data set with equal tuples, selecting observations with replacement.

Step 2: A base model is created on each of these subsets.

Step 3: Each model is learned in parallel with each training set and independent of each other.

Step 4: The final predictions are determined by combining the predictions from all the models.

Advantages of Bagging

- Bagging minimizes the overfitting of data
- It improves the model’s accuracy
- It deals with higher dimensional data efficiently

Example of Bagging

- The Random Forest model uses Bagging, where decision tree models with higher variance are present. It makes random feature selection to grow trees. Several random trees make a Random Forest.

Boosting

- Boosting is an ensemble modelling technique designed to create a strong classifier by combining multiple weak classifiers. The process involves building models sequentially, where each new model aims to correct the errors made by the previous ones.

- Initially, a model is built using the training data.
- Subsequent models are then trained to address the mistakes of their predecessors.
- boosting assigns weights to the data points in the original dataset.

- Higher weights: Instances that were misclassified by the previous model receive higher weights.
- Lower weights: Instances that were correctly classified receive lower weights.

Boosting

- Training on weighted data: The subsequent model learns from the weighted dataset, focusing its attention on harder-to-learn examples (those with higher weights).

- This iterative process continues until:
- The entire training dataset is accurately predicted, or
- A predefined maximum number of models is reached.

Boosting Algorithms

- There are several boosting algorithms.
- The original ones, proposed by Robert Schapire and Yoav Freund were not adaptive and could not take full advantage of the weak learners.
- Schapire and Freund then developed AdaBoost, an adaptive boosting algorithm that won the prestigious Gödel Prize.
- AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification.
- AdaBoost is short for Adaptive Boosting and is a very popular boosting technique that combines multiple “weak classifiers” into a single “strong classifier”.

Algorithm AdaBoost

```
1. Initialize the dataset and assign equal weight to each of
   the data point.
2. Provide this as input to the model and identify the
   wrongly classified data points.
3. Increase the weight of the wrongly classified data points.
4. if (got required results)
     Goto step 5
   else
     Goto step 2

5. End
```

Similarities Between Bagging and Boosting

- Bagging and Boosting, both being the commonly used methods, have a universal similarity of being classified as ensemble methods. Here we will explain the similarities between them.

- Both are ensemble methods to get N learners from 1 learner.
- Both generate several training data sets by random sampling.
- Both make the final decision by averaging the N learners (or taking the majority of them i.e Majority Voting).
- Both are good at reducing variance and provide higher stability.

Differences Between Bagging and Boosting

S.NO	Bagging	Boosting
1.	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built independently.	New models are influenced by the performance of previously built models.
5.	Different training data subsets are selected using row sampling with replacement and random sampling methods from the entire training dataset.	Iteratively train models, with each new model focusing on correcting the errors (misclassifications or high residuals) of the previous models
6.	Bagging tries to solve the over-fitting problem.	Boosting tries to reduce bias.
7.	If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
8.	In this base classifiers are trained parallelly.	In this base classifiers are trained sequentially.
9	Example: The Random forest model uses Bagging.	Example: The AdaBoost uses Boosting techniques

Random Forest

- Random Forest algorithm is a powerful tree learning technique in Machine Learning.

- It works by creating a number of Decision Trees during the training phase.

- Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition.

- This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance.

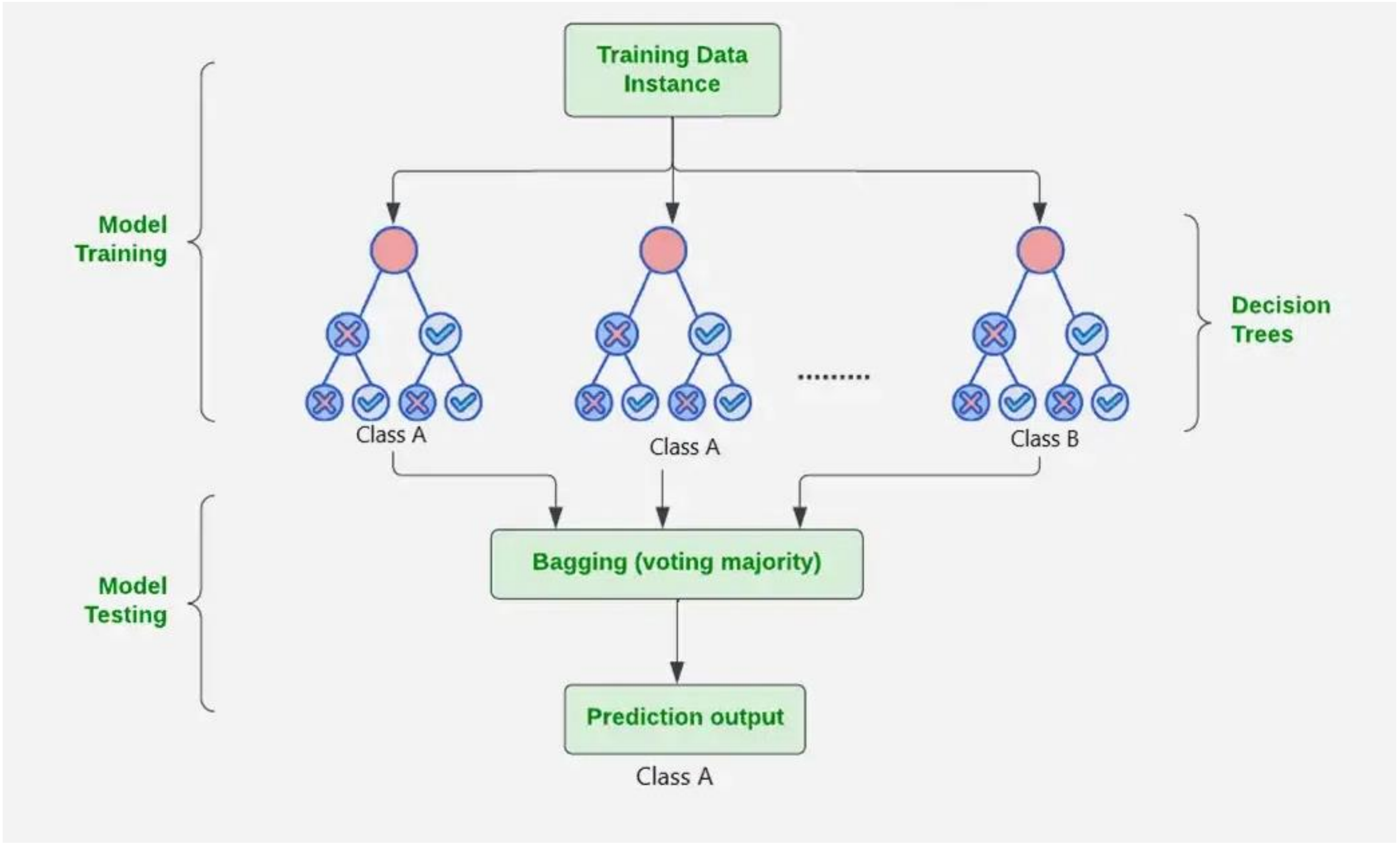
Random Forest

- In prediction, the algorithm aggregates the results of all trees, either by voting (for classification tasks) or by averaging (for regression tasks)

- This collaborative decision-making process, supported by multiple trees with their insights, provides an example stable and precise results.

- Random forests are widely used for classification and regression functions, which are known for their ability to handle complex data, reduce overfitting, and provide reliable forecasts in different environments.

Random Forest

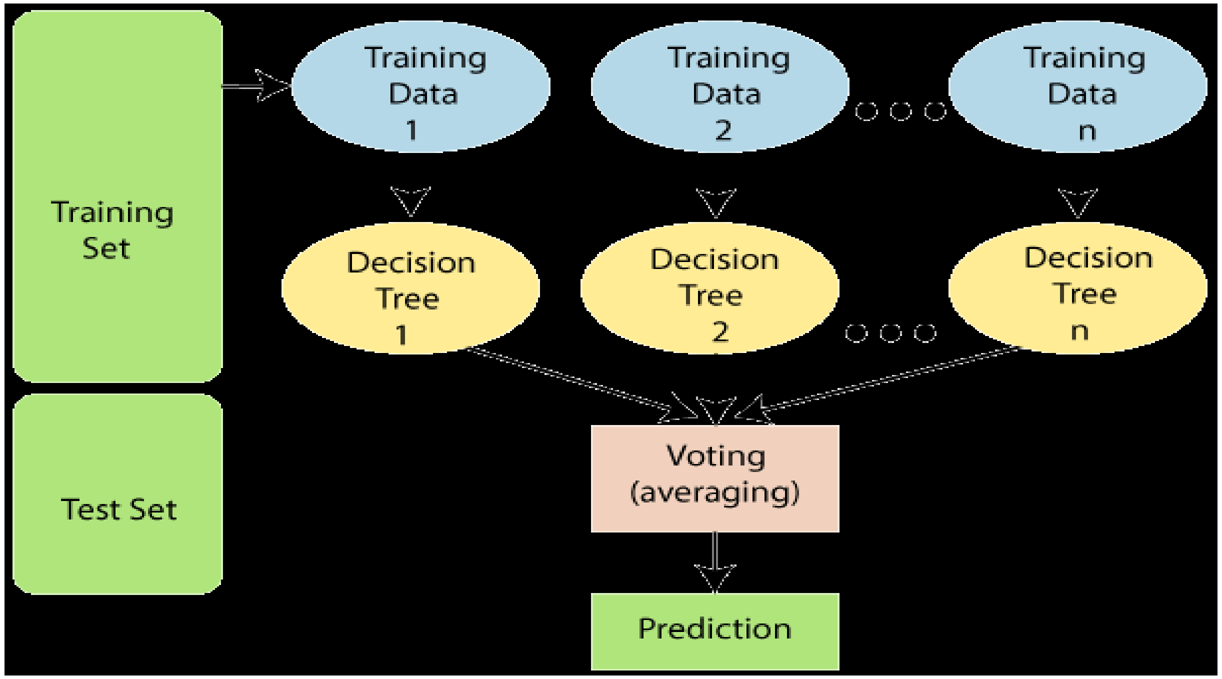


Why to use Random Forest

- 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.

Working of Random Forest

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.



Application of Random Forest

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

Random Forest

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.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

Classification

In machine learning, Classification, as the name suggests, classifies data into different parts/classes/groups. It is used to predict from which dataset the input data belongs to.

There are two types of classifications;

- **Binary classification**
- **Multi-class classification**

Binary Classification

It is a process or task of classification, in which a given data is being classified into two classes. It's basically a kind of prediction about which of two groups the thing belongs to.

the most common algorithms used by binary classification are .

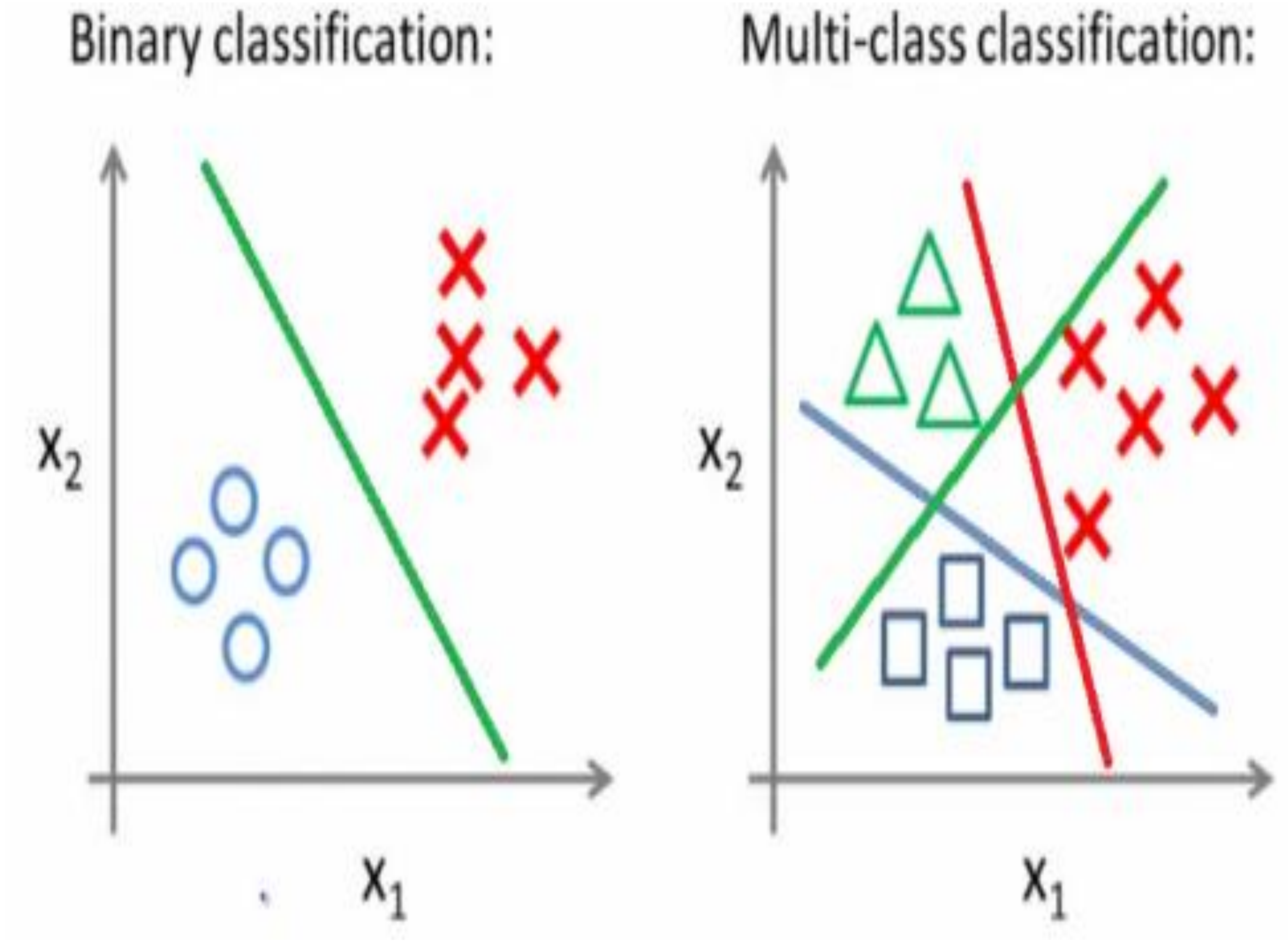
- **Logistic Regression**
- **k-Nearest Neighbors**
- **Decision Trees**
- **Support Vector Machine**
- **Naive Bayes**

Multi-class Classification

Multi-class classification is the task of classifying elements into different classes. Unlike binary, it doesn't restrict itself to any number of classes.

Examples of multi-class classification are

- classification of news in different categories,
- classifying books according to the subject,
- classifying students according to their streams etc.


Binary classification:
Multi-class classification:
x_2
x_1
x_2
x_1

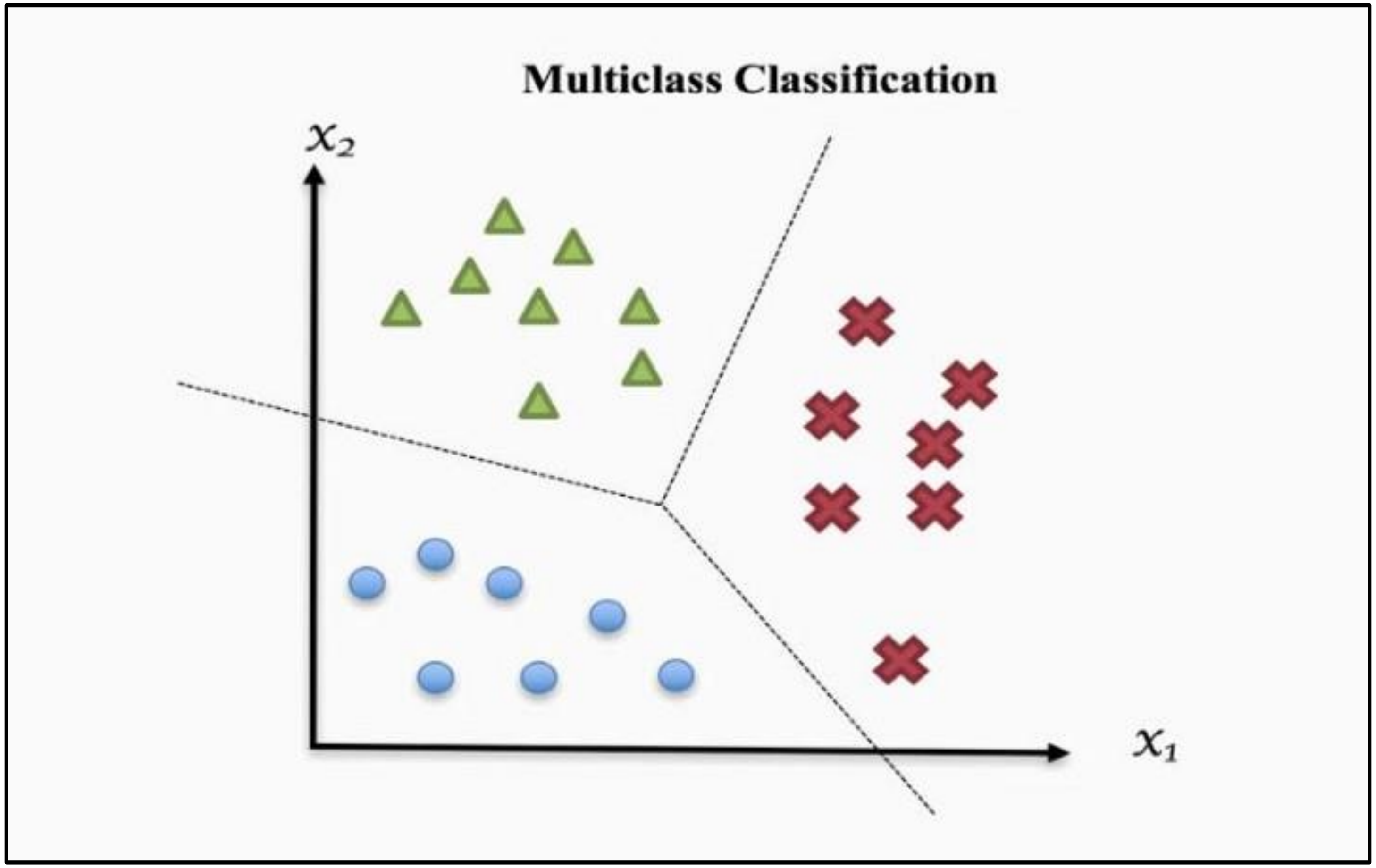
Binary vs Multi-class Classification

Parameters	Binary classification	Multi-class classification
No. of classes	It is a classification of two groups, i.e. classifies objects in at most two classes.	There can be any number of classes in it, i.e., classifies the object into more than two classes.
Algorithms used	The most popular algorithms used by the binary classification are- • Logistic Regression • k-Nearest Neighbors • Decision Trees • Support Vector Machine • Naive Bayes	Popular algorithms that can be used for multi-class classification include: • k-Nearest Neighbors • Decision Trees • Naive Bayes • Random Forest • Gradient Boosting

Binary vs Multi-class Classification

Parameters	Binary classification	Multi-class classification
Examples	Examples of binary classification include- • Email spam detection (spam or not) • Churn prediction (churn or not). • Conversion prediction (buy or not).	Examples of multi-class classification include: • Face classification. • Plant species classification. • Optical character recognition.

Variants of Multiclass Classification:



Multi-class classification is the classification technique that allows us to categorize the test data into multiple class labels present in trained data as a model prediction.

Variants of Multiclass Classification: One-vs-One

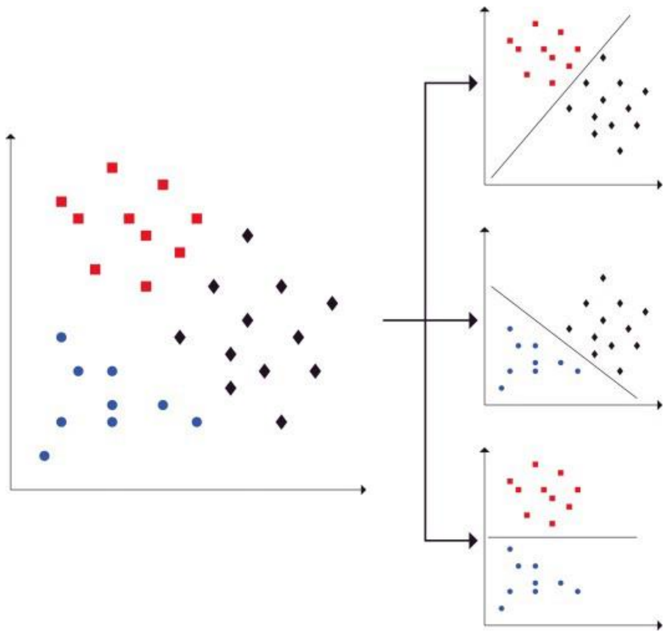
In One-vs-One classification, for the **N-class** instances dataset, we have to generate the **N* (N-1)/2** binary classifier models. Using this classification approach, we split the primary dataset into one dataset for each class opposite to every other class.

Taking the above example, we have a classification problem having three types: **Green**, **Blue**, and **Red (N=3).**

We divide this problem into **N* (N-1)/2 = 3** binary classifier problems:

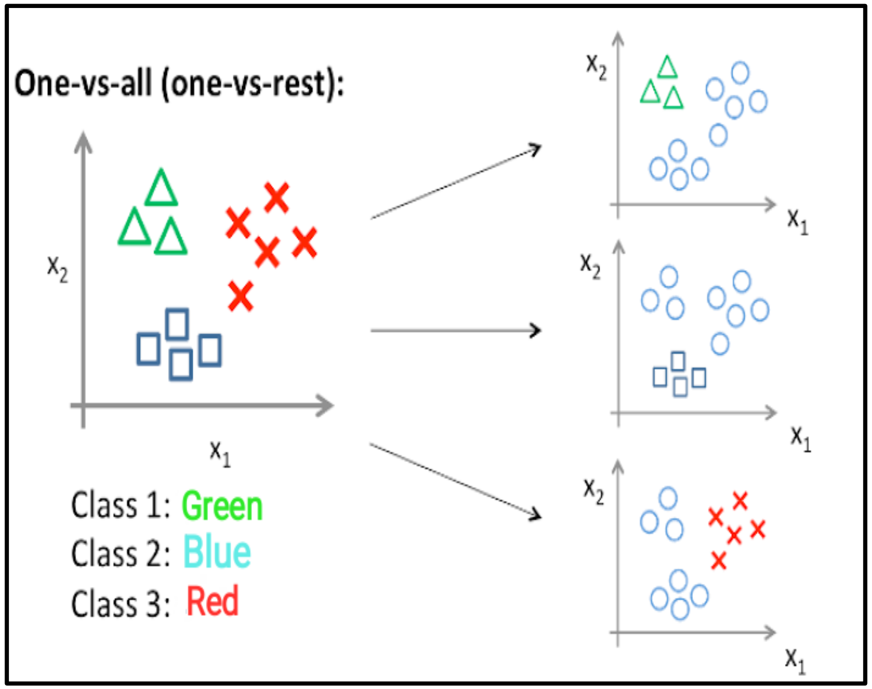
- Classifier 1: Green vs. Blue
- Classifier 2: Green vs. Red
- Classifier 3: Blue vs. Red

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 Multiclass Classification: One-vs-All

In one-vs-All classification, for the N-class instances dataset, we have to generate the N-binary classifier models. The number of class labels present in the dataset and the number of generated binary classifiers must be the same.



As shown in the image, consider we have three classes, for example, type 1 for Green, type 2 for Blue, and type 3 for Red.

Now, as I told you earlier that we have to generate the same number of classifiers as the class labels are present in the dataset, So we have to 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]**

Evaluation Metrics

What are Evaluation Metrics?

Evaluation metrics are used to measure the quality of the statistical or machine learning model. Evaluating machine learning models or algorithms is essential for any project. There are many different types of evaluation metrics available to test a model.

Why is this Useful?

It is very important to use multiple evaluation metrics to evaluate your model. This is because a model may perform well using one measurement from one evaluation metric, but may perform poorly using another measurement from another evaluation metric. Using evaluation metrics are critical in ensuring that your model is operating correctly and optimally.

Applications of Evaluation Metrics

Statistical Analysis
Machine Learning

Evaluation Metrics

What are Evaluation Metrics?

Evaluation metrics are used to measure the quality of the statistical or machine learning model. Evaluating machine learning models or algorithms is essential for any project. There are many different types of evaluation metrics available to test a model.

Why is this Useful?

It is very important to use multiple evaluation metrics to evaluate your model. This is because a model may perform well using one measurement from one evaluation metric, but may perform poorly using another measurement from another evaluation metric. Using evaluation metrics are critical in ensuring that your model is operating correctly and optimally.

Applications of Evaluation Metrics

Statistical Analysis
Machine Learning

Evaluation Metrics and Score: Accuracy

Accuracy simply measures how often the classifier correctly predicts. We can define accuracy as the ratio of the number of correct predictions and the total number of predictions.

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

When any model gives an accuracy rate of 99%, you might think that model is performing very good but this is not always true and can be misleading in some situations.

Evaluation Metrics and Score: Precision and Recall

Precision:

Precision indicates out of all positive predictions, how many are actually positive. It is defined as a ratio of correct positive predictions to overall positive predictions.

Precision = Predictions actually positive/Total predicted positive.

Precision = TP/TP+FP

Recall:

Recall indicates out of all actually positive values, how many are predicted positive. It is a ratio of correct positive predictions to the overall number of positive instances in the dataset.

Recall = Predictions actually positive/Actual positive values in the dataset.

Recall = TP/TP+FN

Evaluation Metrics and Score: F1score

Consider a scenario where your model needs to predict if a particular employee has to be promoted or not and promotion is the positive outcome. In this case, promoting an incompetent employee(false positive) and not promoting a deserving candidate(false negative) can both be equally risky for the company.

When avoiding both false positives and false negatives are equally important for our problem, we need a trade-off between precision and recall. This is when we use the f1 score as a metric. **An f1 score is defined as the harmonic mean of precision and recall.**

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

Evaluation Metrics and Score

Micro Precision

Micro-precision **measures the precision of the aggregated contributions of all classes**. It's short for micro-averaged precision. Precision = 1 means the model's predictions are perfect, all samples classified as the positive class are truly positive.

Micro Recall

Micro-recall **measures the recall of the aggregated contributions of all classes**. It's short for micro-averaged recall. Micro-recall = 1 means the model's predictions are perfect, all truly positive samples was predicted as the positive class. Emphasis on common classes.

Micro F1-score

Micro F1-score (short for micro-averaged F1 score) is **used to assess the quality of multi-label binary problems**. It measures the F1-score of the aggregated contributions of all classes. If you are looking to select a model based on a balance between precision and recall, don't miss out on assessing your F1-scores!

$$Micro - Precision = \frac{TruePositives1 + TruePositives2}{TruePositives1 + FalsePositives1 + TruePositives2 + FalsePositives2}$$

$$Micro - Recall = \frac{TruePositives1 + TruePositives2}{TruePositives1 + FalseNegatives1 + TruePositives2 + FalseNegatives2}$$

$$Micro - F - Score = 2.\frac{Micro - Precision\,.\,Micro - Recall}{Micro - Precision + Micro - Recall}$$

Evaluation Metrics and Score

Micro averaged precision

Micro-precision **measures the precision of the aggregated contributions of all classes**. It's short for micro-averaged precision. Precision = 1 means the model's predictions are perfect, all samples classified as the positive class are truly positive.

Micro average recall

The micro average recall is the sum of true positives for all classes divided by actual positives (rather than predicted positives).

Micro-averaged F1 score

Micro F1-score (short for micro-averaged F1 score) is **used to assess the quality of multi-label binary problems.** It measures the F1-score of the aggregated contributions of all classes. If you are looking to select a model based on a balance between precision and recall, don't miss out on assessing your F1-scores!