Multiclass & Multilabel

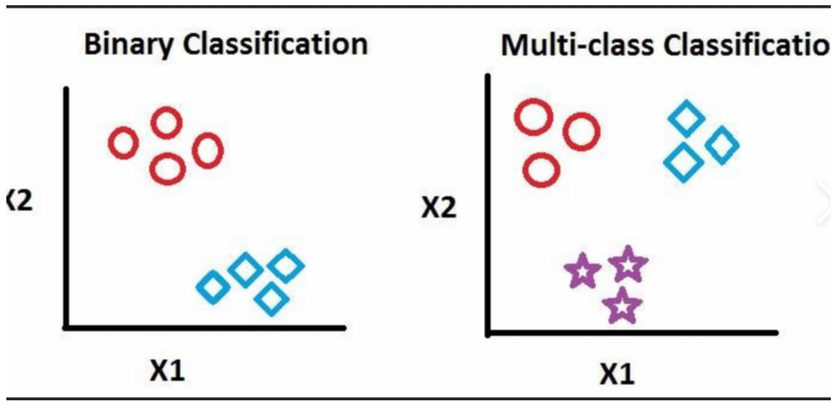
Classification

Single Label:

It is the process of dividing the data into different categories or groups by adding labels to it, based on some conditions

- Grouping the data points to different classes

- Logistic Regression
- Naive-Bayes
- KNN
- Decision Tree
- Support Vector Machines



What is Multi-class Classification?

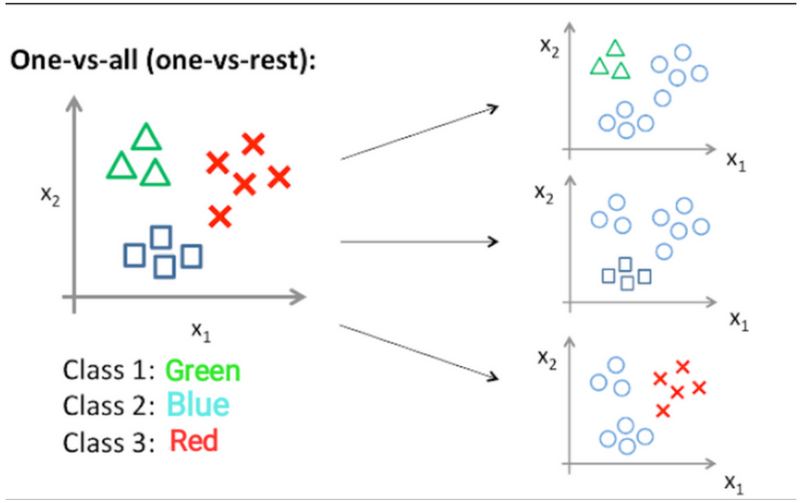
Refers to a type of machine learning task where we aim to classify instances into one of three or more classes. Unlike binary classification, which involves classifying instances into just two classes, multiclass classification deals with scenarios where there are multiple categories in the target variable.

Methods that can be used for multi-class classification include:

1. One-vs.-rest
2. One-vs.-one

1.One VS Rest

- It is used for creating a multiclass classifier
- Training an One VS Rest for our model actually involves ‘N’number of binary trees



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]**

2. 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.

• In One vs. One multi-class classification, we split the primary dataset into one binary classification dataset for each pair of classes.

- Taking the example, we have a classification problem having three types:

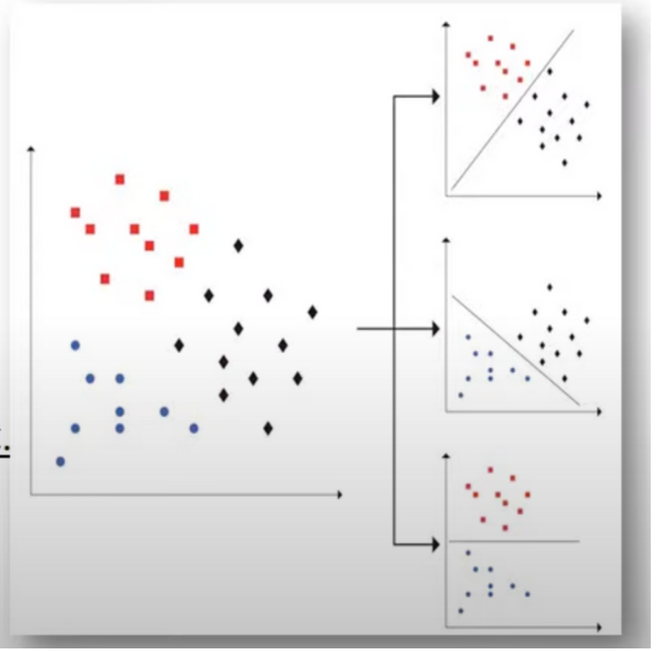
  Green, **Blue**, and **Red (N=3).**

- 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.



Multilabel Classification

1 Problem Transformation Method

- Transform your problem to single label problem
- Then solve using binary or multiclass classification algorithms

2 Algorithm Adaptation Method

- Take a single label classification problem
- Try to adapt this algorithm to your multilabel problem

Sl No	Attributes	Label Set
1	A1	{P, Q, R S }
2	A2	{P}
3	A3	{R S }
4	A4	{P, Q, S }
5	A5	{Q, R S }

Problem Transformation Methods

- copy
- copy-wt
- select-min
- select-max
- select-random
- ignore
- Binary Relevance
- Labeled Powerset (LP)
- Pruned set (PS, EPS)
- Calibrated Label Ranking (CLR)
- Pairwise Ranking via comparison (RPC)
- Classifier Chain (CC, ECC)
- Random K Label Sets (RAKEL)

Algorithm Adaptation Methods

- ML DT
- ML KNN
- ML NB
- SVM HF
- SVM-RANK
- Adboost MR
- Adaboot MH

Classifier Chains:

In this, the first classifier is trained just on the input data and then each next classifier is trained on the input space and all the previous classifiers in the chain.

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

In classifier chains, this problem would be transformed into 4 different single label problems, just like shown below. Here yellow colored is the input space and the white part represent the target variable.

X	y1
x1	0
x2	1
x3	0

Classifier 1

X	y1	y2
x1	0	1
x2	1	0
x3	0	1

Classifier 2

X	y1	y2	y3
x1	0	1	1
x2	1	0	0
x3	0	1	0

Classifier 3

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

Classifier 4

ADABOOST Regressor

Here’s how AdaBoost Regressor works:

1. AdaBoost combines several weak learners into a strong learner. In the context of regression, a weak learner is a model that is only slightly correlated with the true regression.
2. Each weak learner makes a prediction for each data point. The AdaBoost algorithm assigns a “weight” to each prediction, favoring the ones that accurately predict the data.
3. The weights are then updated, giving more importance to the incorrectly predicted data points. This means that the next weak learner will focus more on the “hard” cases that were previously mispredicted.
4. This process is repeated for a number of iterations defined by the user, or until the algorithm can perfectly predict the training data.
5. The final model is a weighted sum of the weak learners.

AdaBoost is to set the weights of classifiers and training data points in a careful, iterative manner that reduces the overall error of the final model. This makes AdaBoost a powerful and popular boosting technique in machine learning.


Original Data
Weighted Data
Weighted Data
Classifer
Classifer
Classifer