SPPU

MACHINE LEARNING

UNIT 4

Supervised Learning : Classification

Contents

- 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, MicroAverage Precision and Recall, Macro-Average F-score.

K-NN

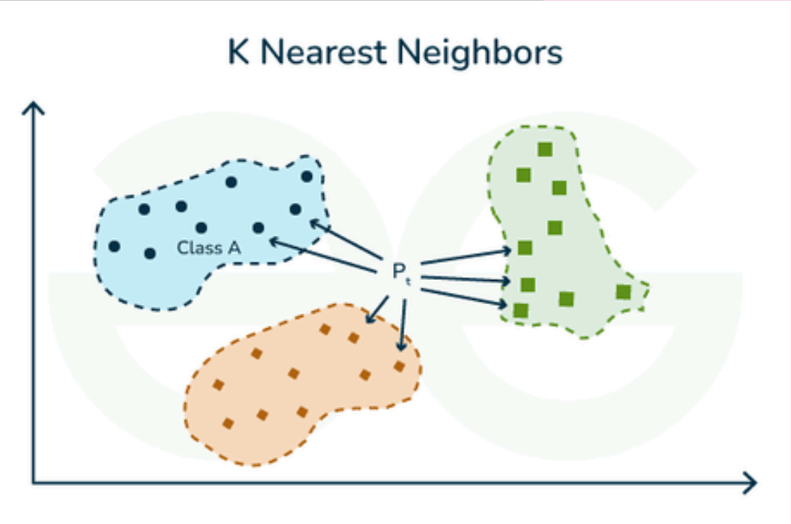
K-Nearest Neighbour (K-NN) is a supervised machine learning algorithm used for classification and regression.

It works on a very simple idea:

“To predict something for a new point, look at its K nearest neighbours and choose the class that most neighbours belong to.”

<u>How K-NN Works</u>

1. Choose a value of K
2. (Example: K = 3 or 5)
3. For a new data point, calculate its distance from all other training points.
4. Find the K closest points (neighbours).
5. Majority voting
   - If it’s classification → select the class most neighbours belong to
   - If it’s regression → take average of neighbours’ values



Example (Very Simple):

You want to classify whether a fruit is an apple or orange based on color and size.

For a new fruit → K-NN checks nearby stored examples.

If out of 5 neighbors,

- 3 are apples
- 2 are oranges

Then new fruit = apple (majority class).

Advantages of K-NN

- Simple and easy to understand : No complex model; works on direct distance.
- No training phase : Model creation is instant; computation happens only during prediction.
- Works well with small datasets : Accurate when data size is limited.
- Non-parametric method : No assumptions about data distribution (flexible).
- Naturally supports multi-class classification : Can handle 2, 3 or more class types easily.

Disadvantages of K-NN

- Slow during prediction : Must calculate distance from all training points.
- High memory usage : Entire dataset must be stored.
- Sensitive to noise and outliers : Wrong or noisy points can mislead the result.
- Performance decreases with high dimensions : Suffers from curse of dimensionality.
- Requires proper feature scaling : Since it uses distance, unscaled features give incorrect results.

Distance Metrics Used in K-NN

1. Euclidean Distance (Most common)

Formula:

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

when

- Data is continuous
- Points are in real number space

2. Manhattan Distance (City Block Distance)

Formula:

$$d = |x_1 - y_1| + |x_2 - y_2| + \dots \quad {}_1| + |x_2 - y_2| + \dots$$

when:

- Movement like grid or streets
- High-dimensional data

SVM

Support Vector Machine (SVM) is a supervised machine learning algorithm used for:

- Classification (main use)
- Regression (called SVR)

SVM tries to separate data points into classes using the best possible boundary called a hyperplane.

SVM finds a hyperplane (line in 2D, plane in 3D) that best separates the classes.

- "Best" hyperplane = One that has maximum margin.

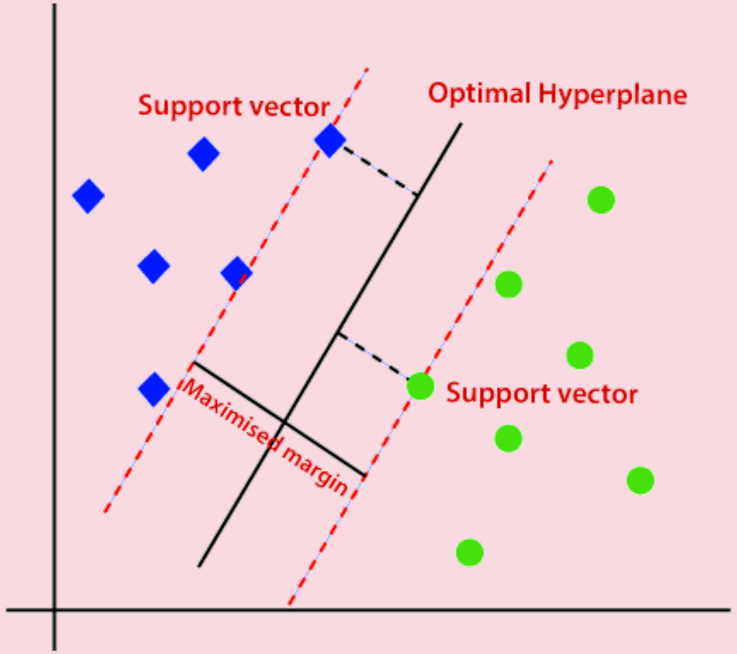
✔ Margin:

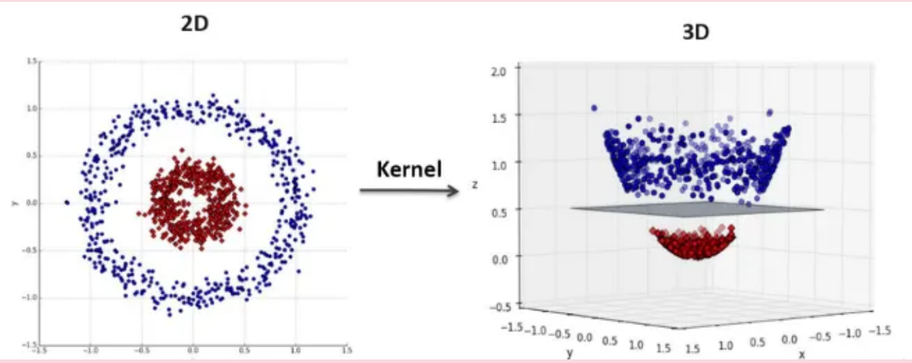
- The distance between the hyperplane and the nearest data points from each class.

✔ Support Vectors:

- These nearest points (closest to the hyperplane) are called support vectors.

They are very important — if you move them, the hyperplane changes.



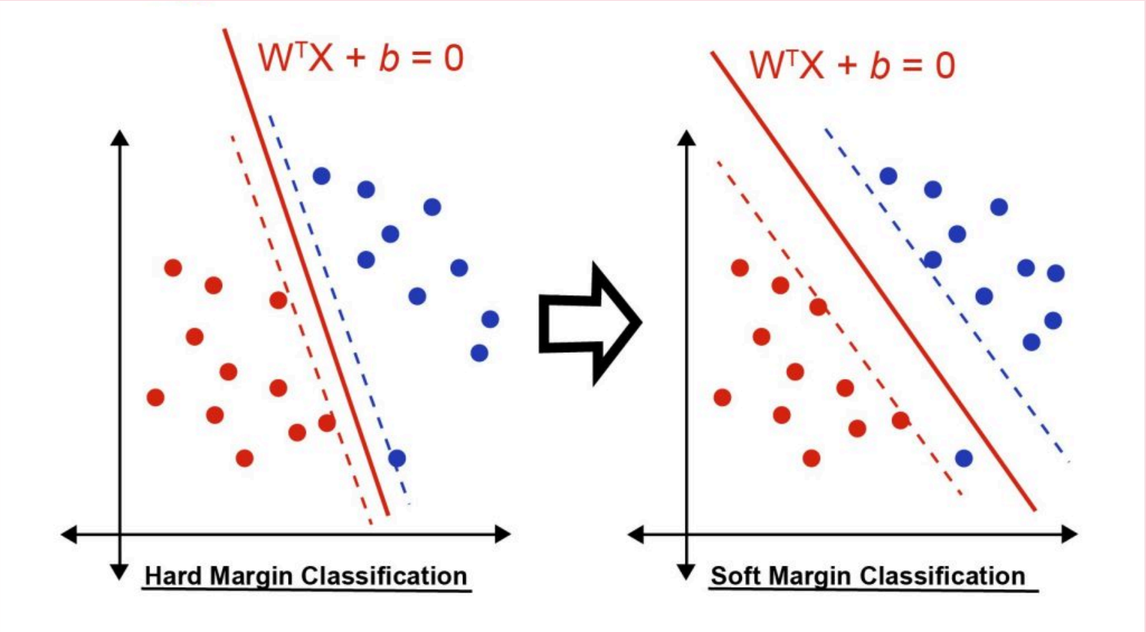


✓ Hard Margin SVM

- Strictly separates classes
- No misclassifications allowed
- Works only when data is perfectly separated
- Rarely used

✓ Soft Margin SVM

- Allows some misclassification
- More practical and realistic
- Helps handle noise
- Used in real-world problems



Advantages of SVM

1. Works well with high-dimensional data (Text, images, gene expression, etc.)
2. Effective when data has clear margin
3. Uses kernel trick
4. Memory-efficient
5. Good generalization

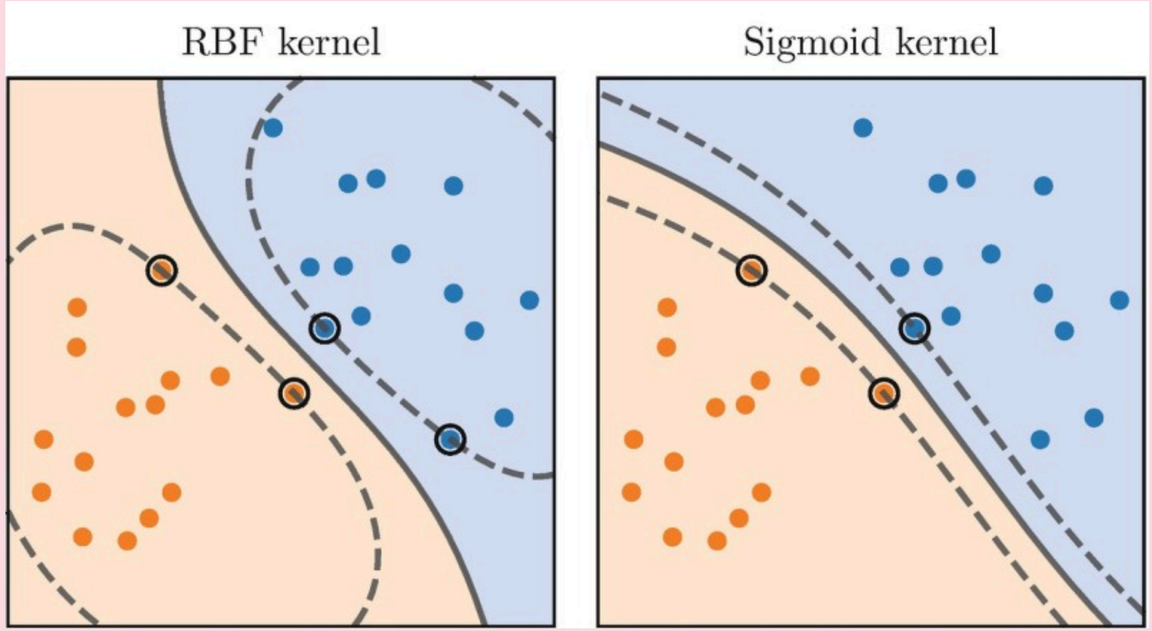
Disadvantages of SVM

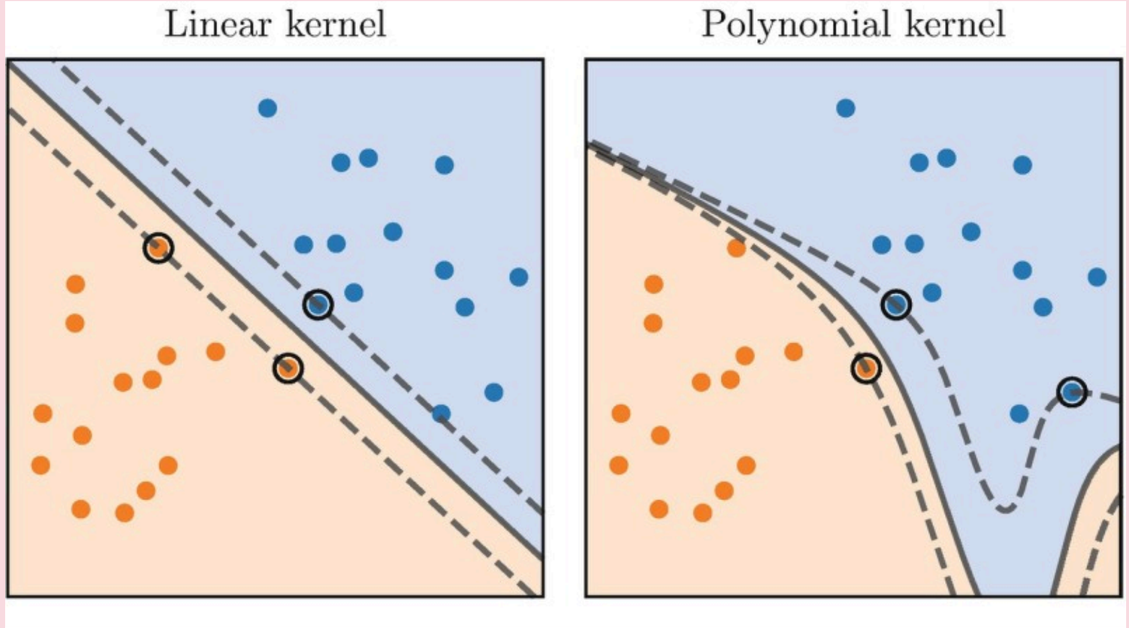
1. Slow for large datasets
2. Choosing the right kernel is difficult
3. Not suitable for noisy datasets
4. Difficult to interpret
5. Training takes more time than KNN or Logistic Regression.

Common Kernels in SVM

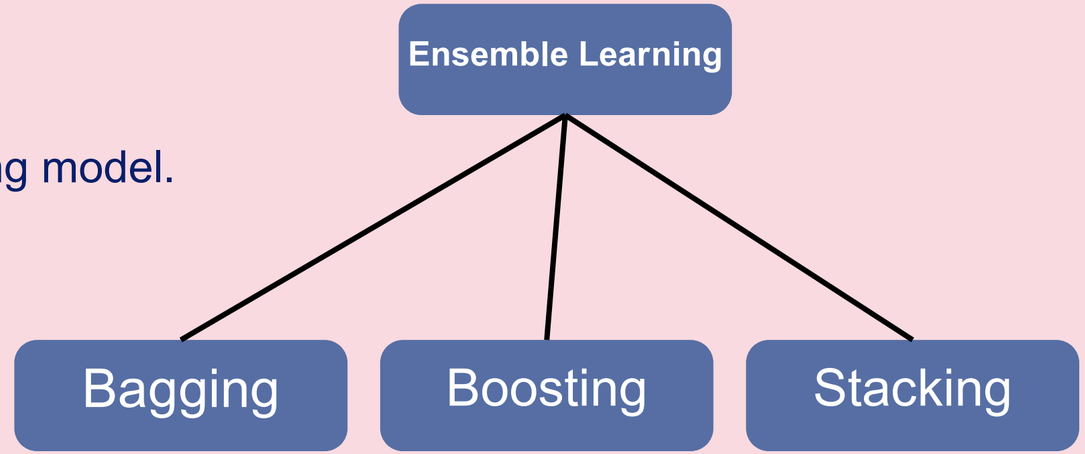
- **Linear Kernel :** Used when data is linearly separable ,Fastest
- **Polynomial Kernel :** Creates curved boundaries ,Good for medium complexity data
- **RBF (Radial Basis Function) Kernel :** Most popular ,Works well when data has complex shapes

  Creates circular/irregular decision boundaries
- **Sigmoid Kernel :** Similar to neural networks





Ensemble Learning



Ensemble learning combines multiple weak models to create one strong model.

Goal = improve accuracy, reduce variance, reduce overfitting.

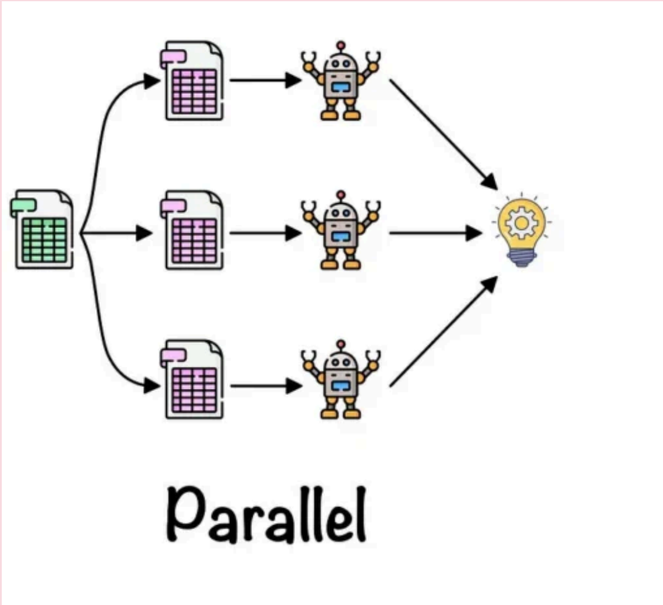
Bagging (Bootstrap Aggregating)

Idea:

- Train multiple independent models on random subsets of data, and then combine their predictions (by majority vote/average).

How it works:

- Create many random samples from dataset (with replacement → bootstrap samples).
- Train a separate model on each sample (usually decision trees).
- Combine the predictions.



Purpose:

- Reduces variance
- Makes model more stable and robust
- Helps avoid overfitting

Types of Bagging:

- Random Forest (most popular)
- Bagged Decision Trees
- Extra Trees (Extremely Randomized Trees)

Random Forest (A Bagging Algorithm)

An ensemble of many decision trees, each trained on a different sample AND different set of features.

✓ Special features:

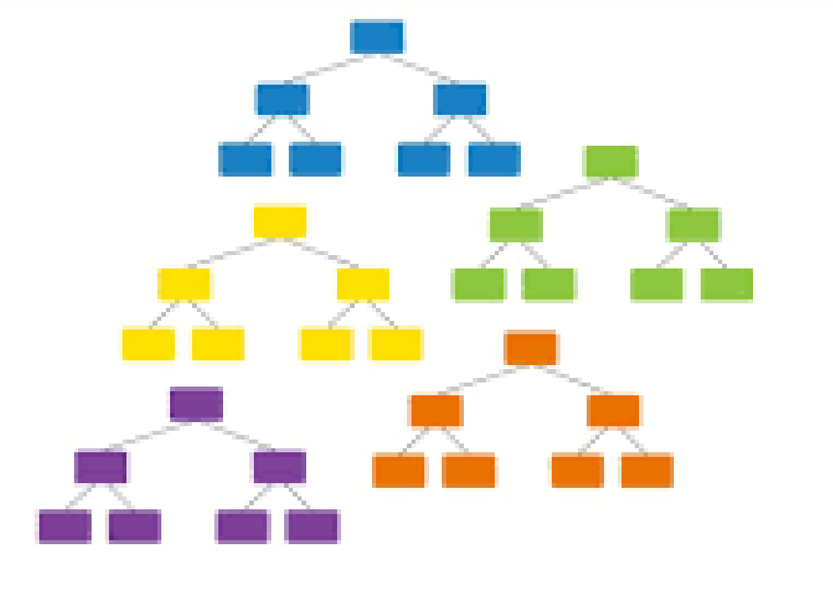
- Random rows (like bagging)
- Random columns (features)
- Trees are grown fully → high diversity

✓ Advantages:

- High accuracy
- Works well for classification & regression
- Handles missing values & mixed data types

✓ Why it works:

Combining many uncorrelated trees reduces variance and increases stability.

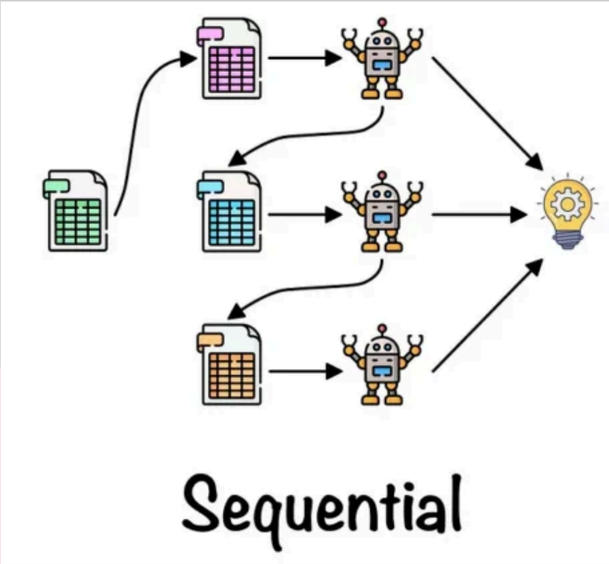


Boosting

Models are trained sequentially, not independently.
Each next model focuses on fixing the errors made by the previous model.

✓ Purpose:

- Reduces bias
- Makes model more accurate
- Learns difficult patterns



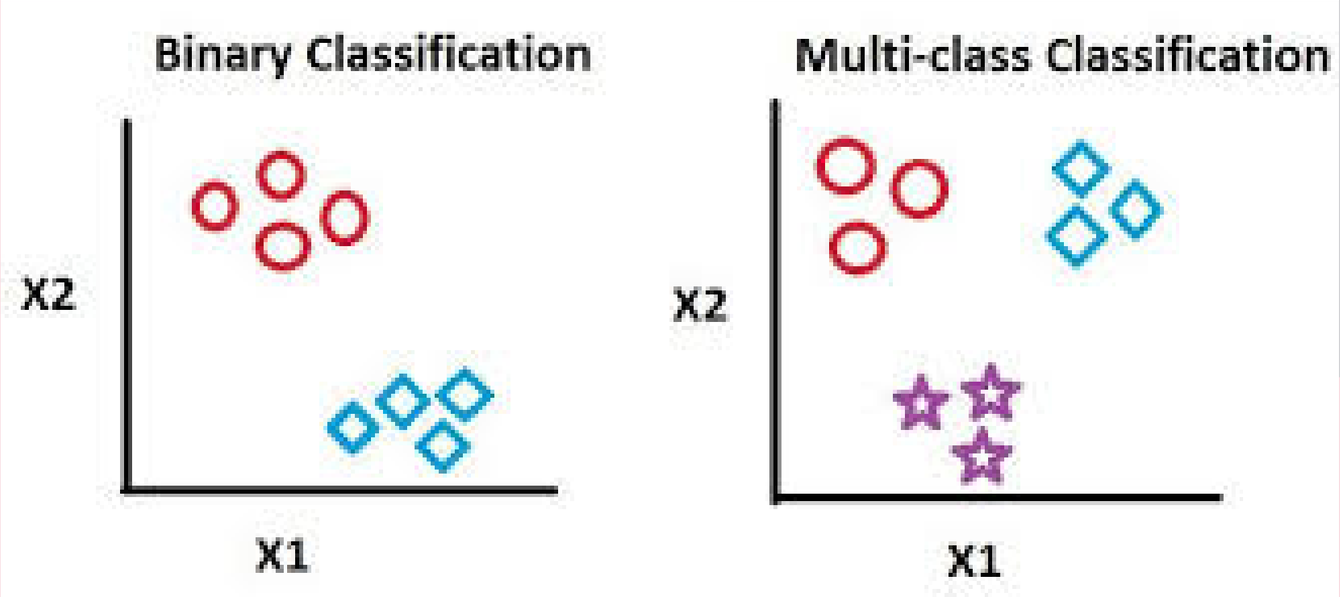
AdaBoost (Adaptive Boosting)

A boosting algorithm that assigns weights to samples and increases weights of misclassified samples.

✓ How it works:

1. Train a weak learner (usually a small decision tree = stump)
2. Increase weight of incorrectly predicted samples
3. Train next learner to focus on them
4. Combine all learners with weighted majority vote

Binary-vs-Multiclass Classification



Binary-vs-Multiclass Classification

Aspect	Binary Classification	Multiclass Classification
Definition	Task of classifying data into **two classes** (e.g., spam or not spam, yes/no).	Task of classifying data into **more than two classes** (e.g., cat/dog/rabbit, sentiment: positive/neutral/negative).
Number of Labels	2 (often labeled as 0 and 1).	≥3 (can be labeled 0, 1, 2, ..., N-1).
Algorithms	Logistic Regression, SVM, Decision Trees, etc.	Many binary algorithms can be extended using strategies like **one-vs-rest (OvR)** or **one-vs-one (OvO)**; also works with Decision Trees, Random Forest, Neural Networks.
Output	Usually a single probability (p) of belonging to one class.	Multiple probabilities (one per class) or a single label from N classes.
Evaluation Metrics	Accuracy, Precision, Recall, F1-score, ROC-AUC.	Accuracy, macro/micro-averaged Precision, Recall, F1-score, confusion matrix.
Complexity	Simpler; decision boundary separates two classes.	More complex; requires separating multiple classes, sometimes with overlapping regions.
Example	Email classification: Spam vs. Not Spam.	Handwritten digit recognition: 0–9 digits.

Balanced and Imbalanced Multiclass Classification Problems

Aspect	Balanced Multiclass Classification	Imbalanced Multiclass Classification
Definition	Each class has **roughly the same number of samples**.	Some classes have **many more samples than others**.
Example	Dataset with 1000 images of cats, 1000 of dogs, 1000 of rabbits.	Dataset with 5000 images of cats, 300 of dogs, 200 of rabbits.
Problem with Model Training	Standard algorithms usually perform well; no class dominates.	Model may **bias toward majority classes**; minority classes are often ignored.
Evaluation Challenges	Standard metrics like accuracy work fine.	Accuracy becomes misleading (high accuracy even if minority classes are misclassified). Metrics like **F1-score, Precision, Recall, and Confusion Matrix** are better.
Techniques to Handle	No special handling needed; standard training works.	- **Resampling**: Oversample minority / undersample majority classes. - **Class weighting**: Assign higher weight to minority classes in loss function. - **Synthetic data generation**: e.g., SMOTE for minority classes. - **Ensemble methods**: Boosting, bagging with focus on minority classes.
Impact on Decision Boundary	Clear separation for all classes; equal representation.	Decision boundary tends to **favor majority classes**, making minority classes prone to misclassification.

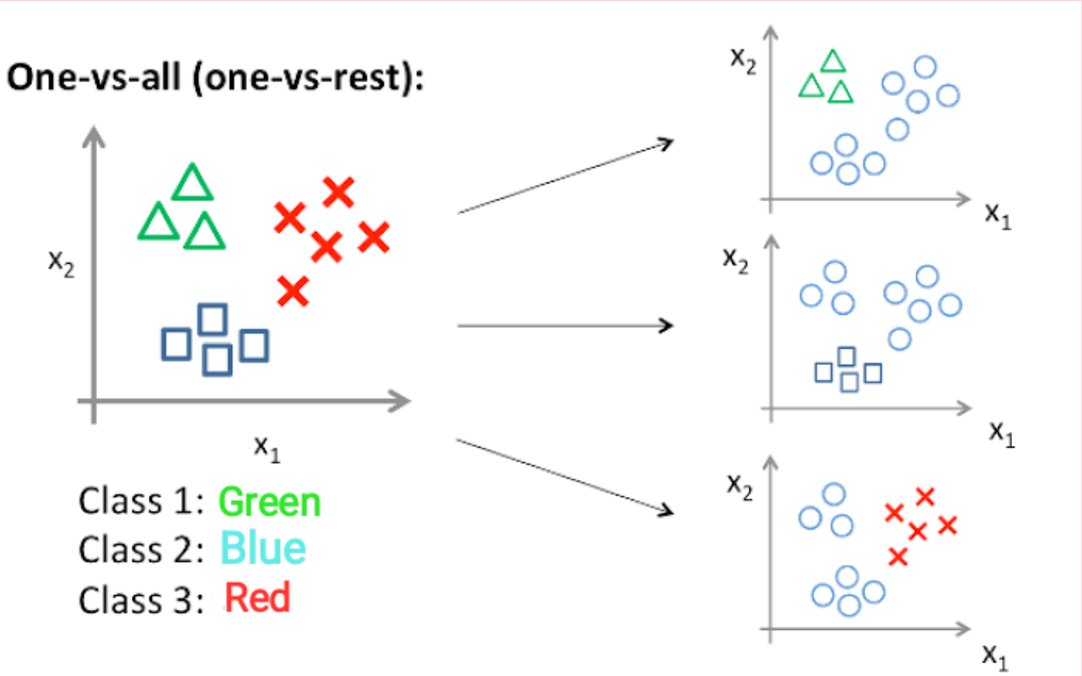
Variants of Multiclass Classification

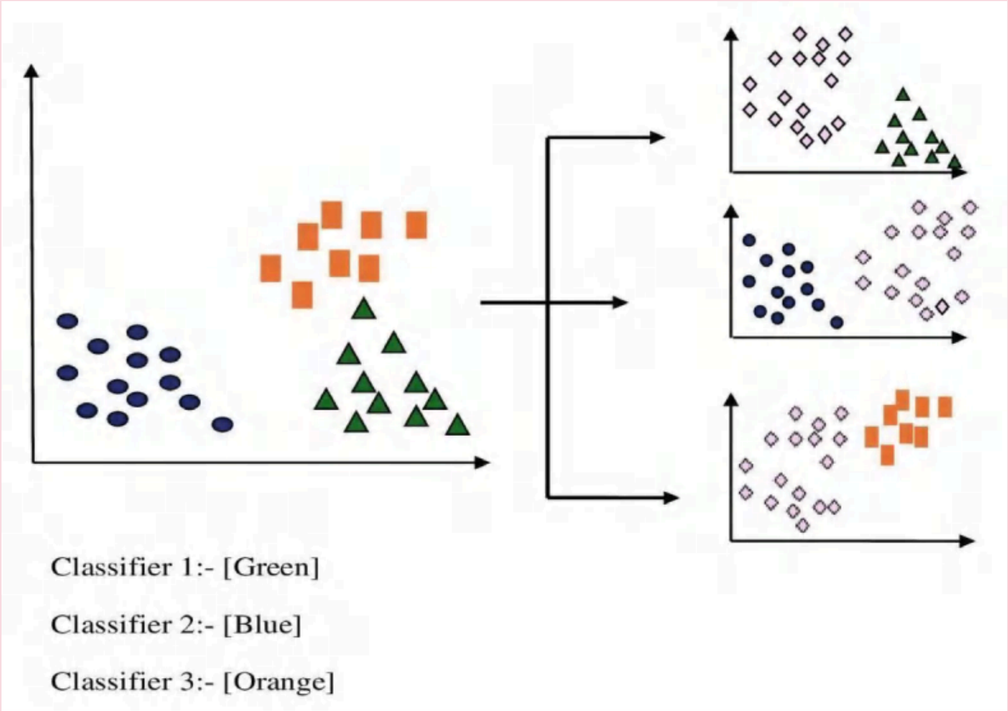
One-vs-All (OvA) / One-vs-Rest

- Build K binary classifiers for K classes
- Each classifier: Class i vs All other classes
- Final prediction: classifier with highest score
- Pros: Simple, fast, fewer models
- Cons: “All other classes” group is large → can reduce accuracy

One-vs-One (OvO)

- Build K(K−1)/2 classifiers
- Each classifier: Class i vs Class j (pairwise)
- Final prediction: majority voting
- Pros: Better performance, simpler binary decisions, good for SVM
- Cons: More models → slower for large K





Evaluation Metrics and Scorce

1. Accuracy
The ratio of correct predictions to total predictions.

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

Usefulness:

- Good when classes are balanced.
- Misleading when data is imbalanced.

2. Precision
Out of all samples predicted as positive, how many were actually positive?

$$Precision = \frac{TP}{TP + FP}$$

Low precision = many false positives → model is “careless”.

3. Recall
Out of all actual positives, how many did the model correctly detect?

$$Recall = \frac{TP}{TP + FN}$$

Low recall = many false negatives → model is “missing cases”.

4. F1-Score
Harmonic mean of precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- High F1 = balanced precision and recall
- Useful for imbalanced datasets

Cross-Validation (CV)
A resampling method to test model performance on different data splits.
Most common: K-Fold Cross Validation

- Dataset split into K parts (say 5 or 10).
- Train on (K–1) folds → test on 1 fold.
- Repeat for all folds.
- Final score = average performance over all folds.

Benefits:

- Reduces overfitting
- Gives stable, reliable performance estimate
- Uses data efficiently

Multiclass Metrics: Micro & Macro Averaging

When you have more than 2 classes, you compute TP/FP/FN separately for each class, then combine them using:

Micro-Average Precision & Recall
Micro-Averaging sums up all TP, FP, FN across all classes first, then computes metrics.

$$Micro\ Precision = \frac{\sum TP}{\sum TP + \sum FP}$$

$$Micro\ Recall = \frac{\sum TP}{\sum TP + \sum FN}$$

- Treats every sample equally
- Good when:
  - Classes are imbalanced
  - Focus on overall performance

Micro-Average F1-Score

$$Micro\ F1 = \frac{2 \times Micro\ Precision \times Micro\ Recall}{Micro\ Precision + Micro\ Recall}$$

- Very useful for imbalanced datasets
- Gives a single performance number

When to use MICRO?
When your dataset has class imbalance and you want a global performance measure.

Macro-Average Precision & Recall:
Compute Precision & Recall for each class separately, then take the average.

$$Macro\ Precision = \frac{1}{N}\sum Precision_{class\ i}$$

$$Macro\ Recall = \frac{1}{N}\sum Recall_{class\ i}$$

Meaning:
Treats all classes equally regardless of size
Gives equal importance to majority and minority classes

When to use MACRO?
When you want to check how your model performs on all classes equally, especially minority classes.

Macro-Average F1-Score

$$Macro\ F1 = \frac{1}{N}\sum F1_{class\ i}$$

- Average of all per-class F1 scores
- Shows how well model handles each class individually

GYAAN SANDHI

YOUR BRIDGE TO BRILLIANCE

THANK YOU

LIKE

Share

SUBSCRIBE