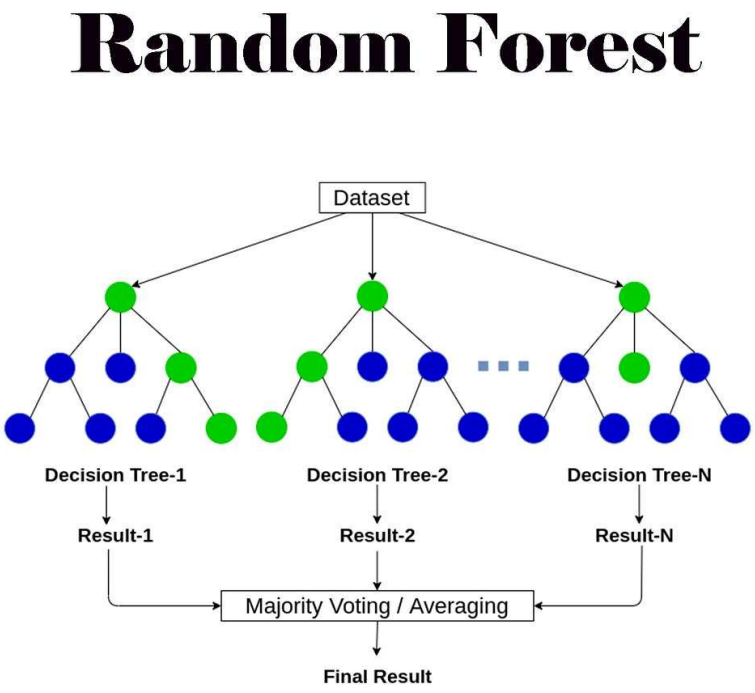
Task-09

Random Forest and **Support Vector Machine (SVM)**

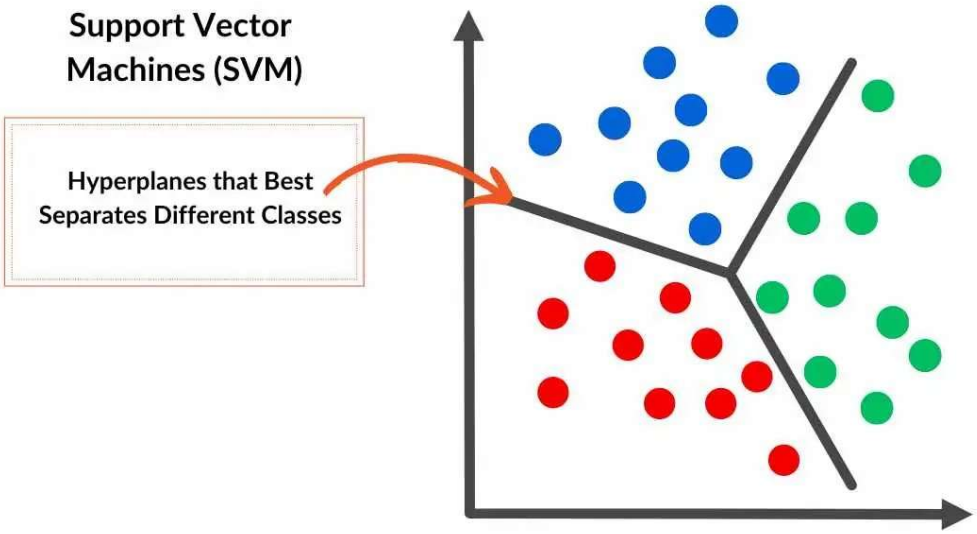
1. Introduction:

In the field of Machine Learning (ML), two of the most widely used and powerful algorithms for classification and regression are Random Forest (RF) and Support Vector Machine (SVM).
Both algorithms are supervised learning methods, meaning they learn from labelled data to predict outcomes for unseen data.

- **Random Forest** is an ensemble learning algorithm based on the concept of combining multiple decision trees to improve accuracy and reduce overfitting.



- **SVM**, on the other hand, is a margin-based classifier that seeks to find the optimal boundary (hyperplane) that best separates data points of different classes in a high-dimensional space.



These algorithms are widely used in finance, healthcare, agriculture, cybersecurity, bioinformatics, and many other domains due to their robustness and strong theoretical foundations.

2. Random Forest Algorithm:

a) Concept and Working Principle:

Random forest is a supervised learning algorithm. The “forest” it builds is an ensemble of decision trees, usually trained with the bagging method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Put simply: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

It is based on two key ideas:

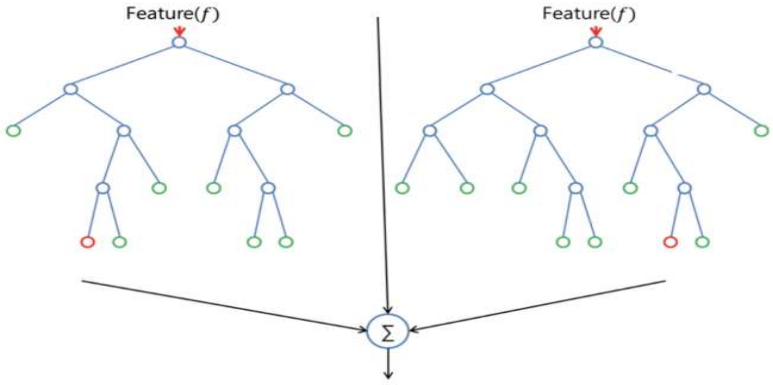
i. **Bootstrap Aggregation (Bagging):**
Multiple subsets of the original dataset are created by sampling

with replacement. Each subset trains a different decision tree.
→ This introduces **variance reduction** by averaging across trees.

ii. **Random Feature Selection:**
At each node of a tree, only a **random subset of features** is considered for splitting, ensuring diversity among trees.
→ This reduces **correlation between trees** and improves generalization.

b) Step-by-Step Working:

- Select *k* random samples from the dataset (with replacement).
- Train a **decision tree** on each sample.
- At each node, choose the best split only among a **random subset of features**.
- Repeat steps 1–3 to grow multiple trees (typically 100–1000).
- For prediction:
- **Classification:** Take the **majority vote** from all trees.
- **Regression:** Take the **average** of all tree outputs.



c) Mathematical Representation:

For a dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$:

Each tree $h_i(x)$ produces an output.
The Random Forest prediction is:

$$\hat{y} = \begin{cases} \text{mode}(h_1(x), h_2(x), ..., h_m(x)) & \text{for classification} \\ \frac{1}{m}\sum_{i=1}^{m} h_i(x) & \text{for regression} \end{cases}$$

where m is the number of trees.

d) Advantages:

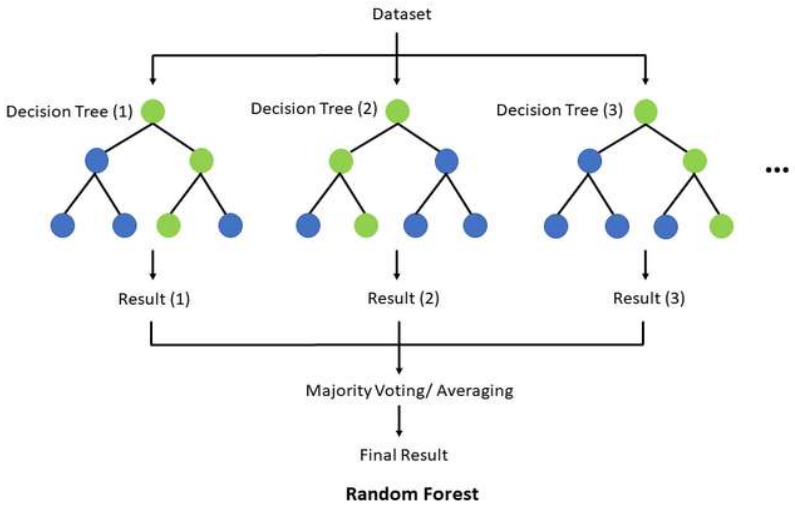
- **High accuracy** due to ensemble averaging.
- **Robust to noise** and **outliers**.
- **Handles missing data** well.
- Works with both **categorical and numerical** data.
- Automatically estimates **feature importance**.

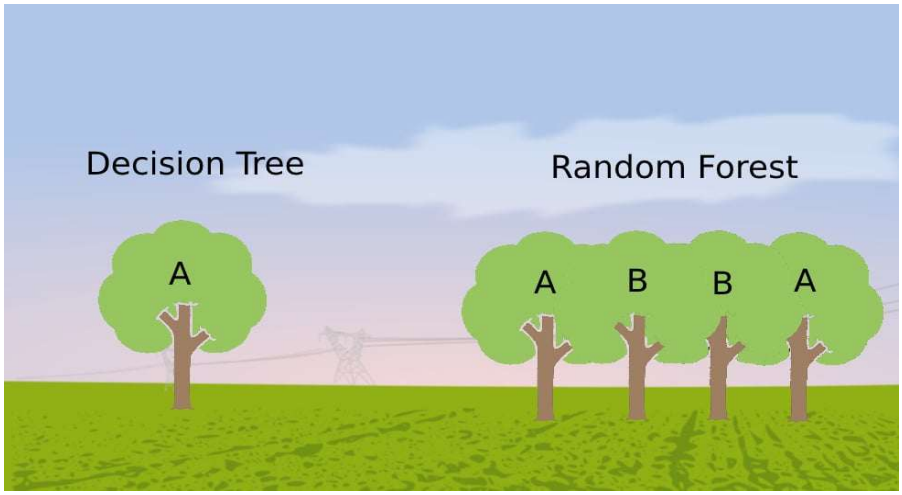
e) Limitations:

- **Computationally expensive** for large datasets.
- **Less interpretable** compared to single decision trees.
- May **overfit** on noisy data if too many trees are built.

f) Applications:

- **Healthcare:** Disease prediction (e.g., diabetes, heart disease).
- **Finance:** Credit risk modelling, stock market predictions.
- **Agriculture:** Crop yield prediction and soil classification.
- **Cybersecurity:** Intrusion detection systems.
- **Environmental Science:** Predicting forest fires and air quality.





3. Support Vector Machine (SVM):

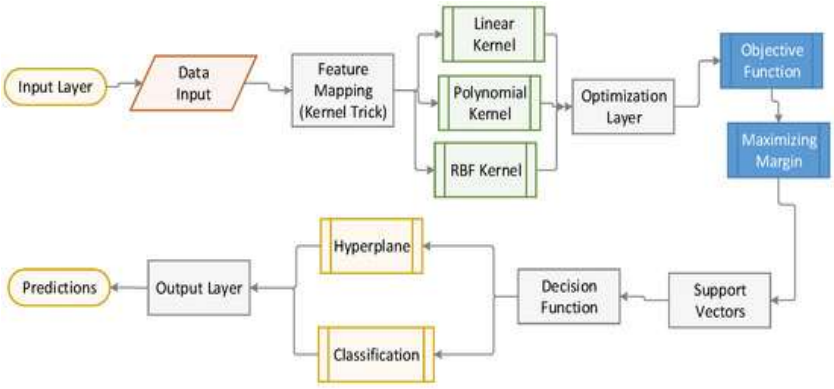
a) Concept and Working Principle:

Support Vector Machine is a **supervised classification algorithm** that aims to find the **optimal separating hyperplane** between classes such that the **margin (distance)** between the hyperplane and the nearest data points (support vectors) is **maximized**.

It can also handle **non-linear data** using **kernel functions** that project data into higher dimensions where a linear separator can exist.
Given data points from two classes, SVM tries to find a line (in 2D) or hyperplane (in higher dimensions) that separates them with the **maximum possible margin**.
The **margin** is the distance between the hyperplane and the nearest data points from either class (called **support vectors**).

b) Step-by-Step Working:

- Find the **best separating hyperplane** between two classes.
- "Best" = **maximum margin** (largest distance between the classes).
- Equation of hyperplane: $\mathbf{w \cdot x + b = 0}$
- Constraints: $\mathbf{y_i (w \cdot x_i + b) \geq 1}$
- Objective: **Minimize** $\mathbf{(1/2) \|w\|^2}$ (to maximize margin)
- The **closest points** to the hyperplane.
- They **define** the margin and the decision boundary.
- Allow some errors using **slack variables** (ξ_i).
- Add penalty term $\mathbf{C \Sigma \xi_i}$ to control trade-off between margin size and misclassification.
- Problem depends only on **dot products** $(x_i \cdot x_j)$.
- Introduce multipliers $\alpha_i \rightarrow$ only $\alpha_i > 0$ (support vectors) affect decision.
- Replace dot product with **kernel** $K(x_i, x_j)$.
- Decision Function: $f(x) = \text{sign}\left(\sum_i \alpha_i y_i K(x_i, x) + b\right)$
- Use **SMO (Sequential Minimal Optimization)** or **gradient methods** to find optimal α_i.
- Tune **C** (penalty) and **kernel parameters** using cross-validation.
- Plug new data into **f(x)**.
- Classify as +1 or -1 depending on the sign.



c) <u>Mathematical Formulation:</u>

For a dataset (x_i, y_i), where $y_i \in \{-1, +1\}$:

We want to find w and b such that:

$$y_i(w \cdot x_i + b) \geq 1$$

The optimization problem becomes:

$$\min_{w,b} \frac{1}{2}||w||^2$$

subject to the above constraint.

For **non-linearly separable data**, a **slack variable (ξ)** and **regularization parameter (C)** are introduced:

$$\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_i \xi_i$$

subject to $y_i(w \cdot x_i + b) \geq 1 - \xi_i$.

d) Kernel Trick:

To handle non-linear data, SVM uses **kernel functions** to map data into higher-dimensional spaces:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

Common kernels:

- **Linear Kernel:** $K(x_i, x_j) = x_i \cdot x_j$
- **Polynomial Kernel:** $K(x_i, x_j) = (x_i \cdot x_j + c)^d$
- **RBF (Gaussian) Kernel:** $K(x_i, x_j) = e^{-\gamma ||x_i - x_j||^2}$

e) Advantages:

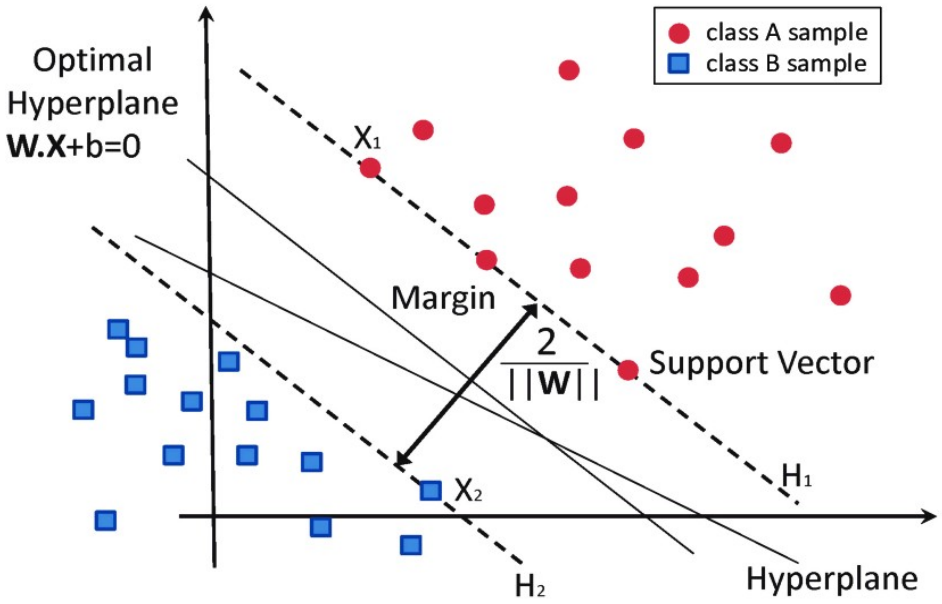
- Effective in **high-dimensional spaces**.
- **Robust to overfitting**, especially with proper kernel choice.
- Works well for **non-linear classification** using kernels.
- Only depends on **support vectors**, making it memory efficient.

f) Limitations:

- **Training time** increases with dataset size (not ideal for large data).
- **Choice of kernel** and parameters (C, γ) requires tuning.
- **Less interpretable** compared to simpler models.
- Performance drops with **overlapping classes** or **noisy data**.

g) Applications:

- **Text classification** and **spam filtering**.
- **Face and handwriting recognition**.
- **Bioinformatics:** Protein classification, cancer detection.
- **Finance:** Fraud detection and credit scoring.
- **Industrial quality control** using image-based inspection.



4. Comparison: Random Forest vs SVM:

Feature	Random Forest	Support Vector Machine (SVM)
Type	Ensemble (Decision Tree-based)	Margin-based (Linear/Non-linear classifier)
Working Principle	Combines predictions from multiple decision trees (Bagging)	Finds optimal hyperplane maximizing margin between classes
Complexity	Moderate; scalable for large datasets	High; computationally expensive for large datasets
Interpretability	Low (black-box ensemble)	Moderate (depends on kernel and dimensionality)

Handling of non-linearity	Naturally handled through decision trees	Requires kernel trick
Overfitting Control	Reduced by averaging across trees	Controlled by regularization parameter (C)
Training Speed	Faster for medium datasets	Slower for large datasets
Memory Usage	High (stores many trees)	Low (depends on number of support vectors)
Accuracy	High for most tabular datasets	Very high for high-dimensional data
Best Used When	Dataset is large, with mixed data types	Dataset is smaller, well-separated, and continuous

5. Real-World Case Study Examples:

a) Random Forest in Healthcare:
Predicting cardiovascular disease using patient records (features like age, cholesterol, blood pressure). The algorithm combines multiple tree decisions to ensure high recall and minimize false negatives.

b) SVM in Image Recognition:
Classifying handwritten digits (MNIST dataset) using SVM with RBF kernel. The model maps 28×28-pixel data into high-dimensional space for accurate boundary separation between digits.

c) Random Forest in Agriculture:
Used for predicting soil fertility levels based on pH, moisture, and nutrient content, providing recommendations for crop selection.

d) SVM in Cybersecurity:
Network intrusion detection — SVMs classify network traffic as normal or malicious using statistical and time-based features.

6. Conclusion:

Both **Random Forest** and **SVM** are highly effective supervised learning algorithms, each excelling under different conditions.

- **Random Forest** is better suited for **large, noisy, and tabular datasets** where interpretability is secondary but performance is crucial.

- **SVM** is ideal for **high-dimensional, well-structured datasets** where a clear margin exists between classes.

In modern ML pipelines, the choice between RF and SVM often depends on:

- **Data size and dimensionality**
- **Computational resources**
- **Need for interpretability**
- **Nature of the decision boundary**

In practice, both models are complementary — Random Forest for robustness and feature importance, and SVM for mathematical elegance and strong theoretical guarantees.