Decision Tree Algorithm

Define: A decision tree is a supervised machine learning algorithm that resembles a flowchart. It starts with a root node and branches out into decision nodes, which represent feature tests. Each branch leads to another node or a leaf node. Leaf nodes represent the final prediction or classification.

How does a decision tree algorithm work?

A decision tree algorithm works by repeatedly splitting the data into smaller and smaller subsets based on feature values. At each step, the algorithm chooses the best feature to split the data, aiming to maximize information gain. This process continues until the data is divided into pure subsets, or a stopping criterion is met, such as reaching a maximum depth or a minimum number of samples.

Entropy:

A measure of randomness or impurity in the dataset.

$$Entropy = \sum_{i=1}^{c} -P_i * log_2(P_{i)}$$

pi should be probaablility of ith class

Interpretation:

- Entropy = 0: Data is pure (all samples belong to one class).
- Higher entropy: More mixed or uncertain data.

Information Gain (IG):

Derived from entropy and shows how much a split improves purity.

Purpose: Helps in selecting the best feature for splitting by maximizing the reduction in uncertainty.

Gini Index

Definition: A measure of impurity or diversity within the dataset.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Interpretation:

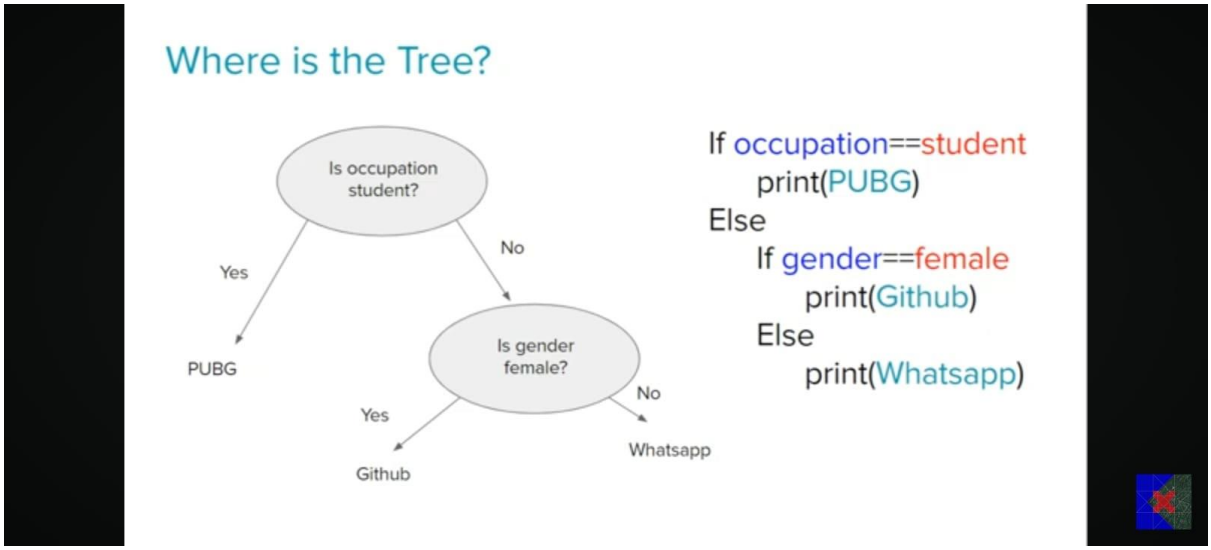
Gini = 0: Pure node (all samples in one class).

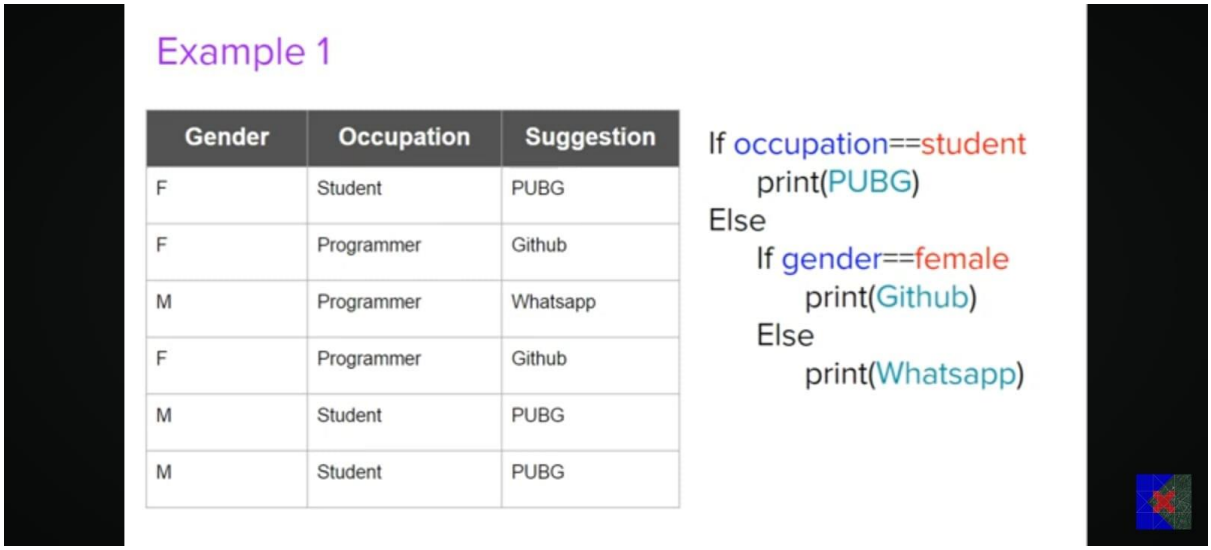
Higher Gini: Higher impurity.

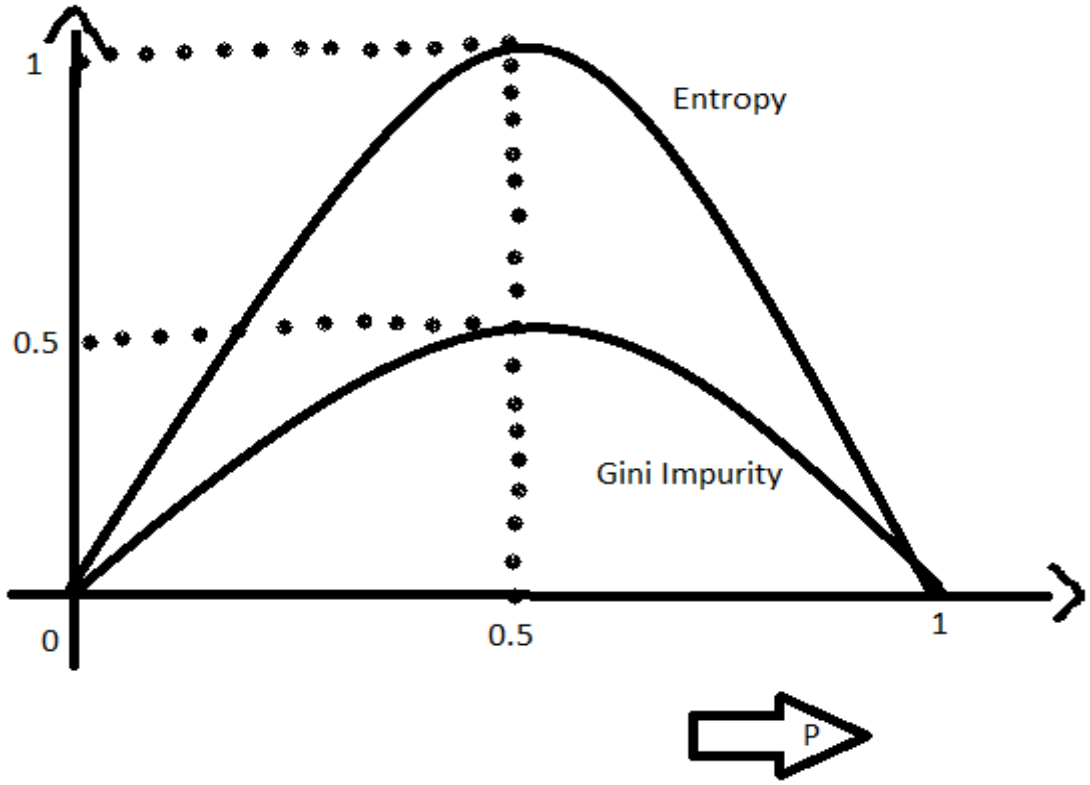
Usage: Often used in CART (Classification and Regression Trees) because it is computationally efficient compared to entropy.

Key Differences (Quick Recall):

- **Entropy vs. Gini**:
    - Entropy is slightly more complex computationally as it uses logarithms.
    - Gini is faster and often preferred in practice but generally leads to similar splits as entropy.







$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

$$Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

However, Gini Impurity is generally more computationally efficient than entropy. The graph of entropy increases up to 1 and then starts decreasing, while Gini Impurity only goes up to 0.5 before decreasing, thus requiring less computational power. The range of entropy is from 0 to 1, whereas the range of Gini Impurity is from 0 to 0.5.

MAIN GOAL:

- For a **good split**, **minimize Entropy and Gini Index to reduce impurity**, and **maximize Information Gain** to improve class separation, ensuring purer child nodes.

For a **better split in regression** using a decision tree, **minimize Mean Squared Error (MSE)** or Variance to ensure the child nodes have more homogeneous target values.

What are the advantages and disadvantages of decision trees?

Advantages:

- Can handle both numerical and categorical data.
- No need for feature scaling or normalization.
- Non-parametric, making it versatile for different types of data.

Disadvantages:

- Prone to overfitting, especially with deep trees.
- Sensitive to noisy data and small changes in the dataset.
- May not generalize well on unseen data.

What is pruning, and why is it important?

Pruning refers to the process of removing sections of a decision tree that provide little predictive power, typically branches that have little data.

Why is it important?

- Pruning reduces overfitting by simplifying the tree and making it more generalizable to new data.
- It improves model performance by removing unnecessary complexity, thus enhancing interpretability and reducing variance.

Prevent A Decision Tree From overfitting?

- **Limit the tree depth** by setting a maximum depth to avoid creating overly complex trees.
- **Set a minimum number of samples per leaf** to ensure each leaf has enough data to make reliable predictions.
- **Use pruning** to remove branches that have little predictive value.
- **Apply cross-validation** to tune the model and ensure it generalizes well to unseen data. These techniques help keep the model simple and improve its ability to generalize to new data.

Explain the difference between bagging and boosting.

Bagging (Bootstrap Aggregating) builds multiple independent models (e.g., decision trees) on different random subsets of the training data, and the final prediction is made by averaging or voting.

It focuses on reducing variance and preventing overfitting.

Boosting, on the other hand, builds models sequentially, where each new model corrects the errors of the previous one.

It focuses on reducing bias and improving the model's accuracy by giving more weight to misclassified data points in each iteration.

Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of predictions. It builds a collection of decision trees during training and merges their results to make a final decision.

How does Random Forest reduce overfitting compared to a single decision tree?

- Random Forest reduces overfitting by **averaging the predictions of multiple decision trees,** each trained on different subsets of the data.
- This process, known **as bagging** (REDUCED VARIANCE), helps to smooth out the model's predictions, making it less sensitive to noise and outliers.
- Additionally, by selecting random subsets of features for each split, it prevents individual trees from becoming too complex and overfitting to the training data.

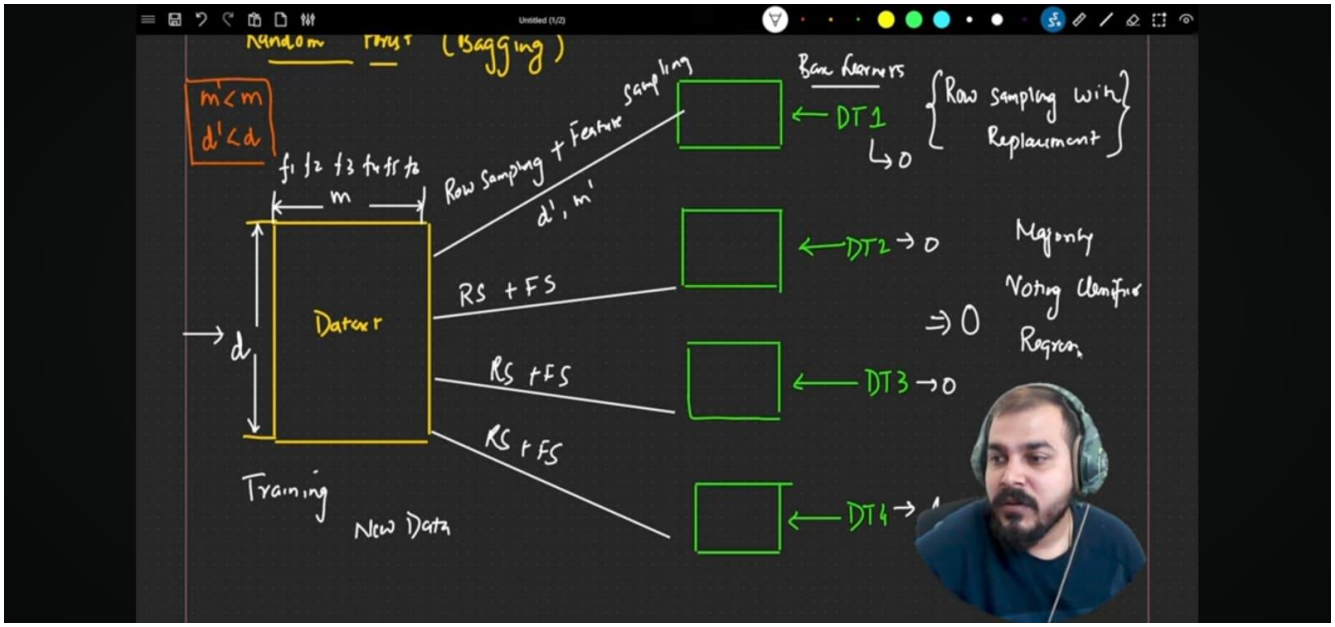
What are out-of-bag (OOB) errors in random forests?

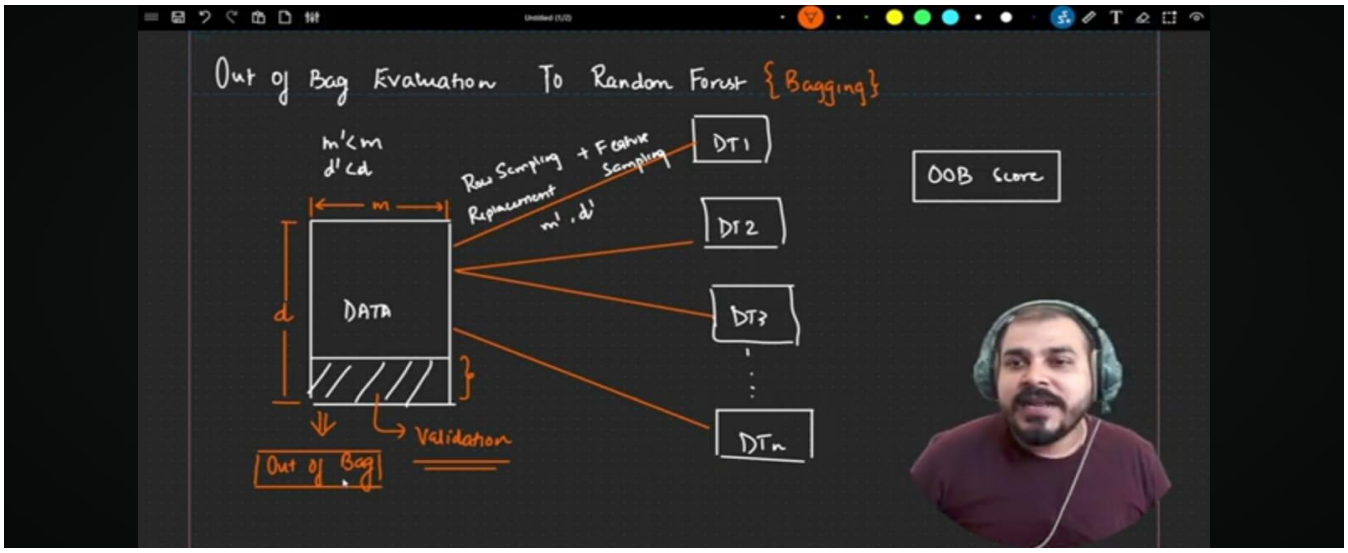
- Out-of-bag (OOB) errors are the predictions made using **the training samples that were not selected for a specific tree during the bootstrap sampling process**.
- Since each tree is trained on a random subset of data, around **one-third of the data is left out (OOB data**) and can **be used to estimate the model's performance**, providing an unbiased error estimate without the need for a **separate validation set**.

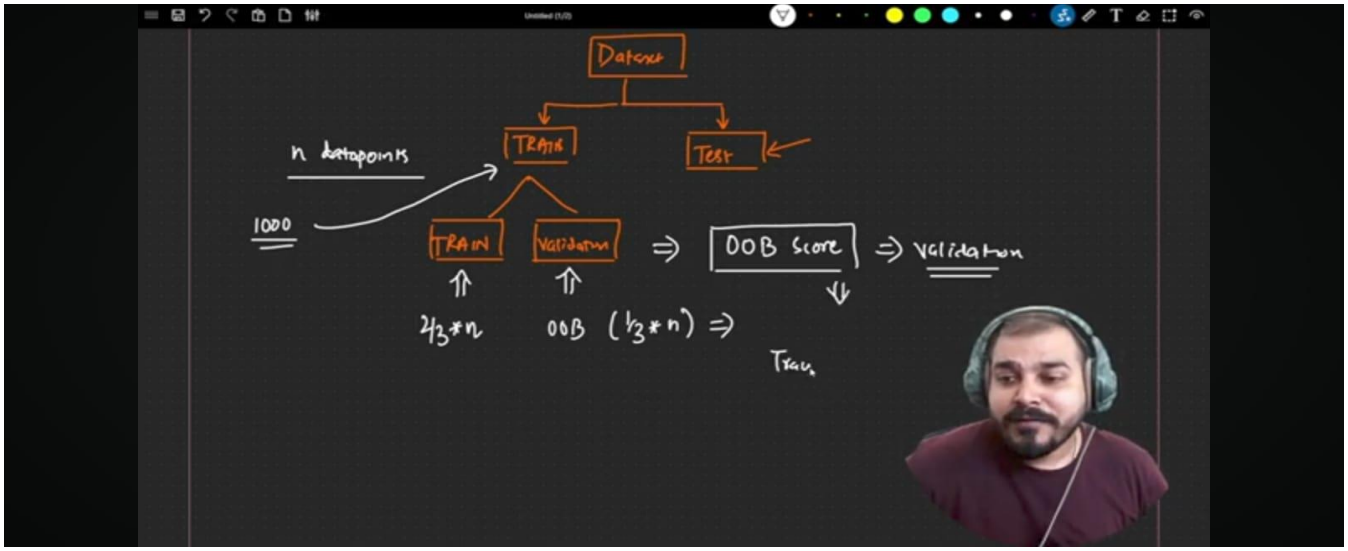
Can random forests handle missing data? How?

Yes, Random Forest can handle missing data. During training, if a data point has missing values, the algorithm can use surrogate splits, which are **alternative splits based on other features**, to make the decision.

Additionally, since multiple trees are built, each tree can handle missing data in different ways, ensuring that the final prediction remains reliable.


Random Forest (Bagging)
m<m
d'<d
f1 f2 f3 f4 f5 f6
m
d
Dataset
Row Sampling + Feature Sampling
d', m'
RS + FS
RS + FS
RS + FS
Training
New Data
Base Learners
DT1 → 0
DT2 → 0
DT3 → 0
DT4 →
{Row sampling with Replacement}
⇒ 0
Majority Voting Classifier
Regressor


Out of Bag Evaluation To Random Forest {Bagging}
m'<m
d'<d
m
d
DATA
Row Sampling + Feature Sampling
Replacement
m' . d'
DT1
DT2
DT3
DTn
OOB Score
Validation
Out of Bag


Dataset
TRAIN
Test
n datapoints
1000
TRAIN
Validation
⇒
OOB Score
⇒ Validation
2/3*n
OOB (1/3*n) ⇒

Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm used for both classification and regression tasks. The main idea behind SVM is to find the hyperplane that best separates the data into different classes (for classification) or fits the data (for regression).

Hyperplanes, Support Vectors, and Margins:

Hyperplane: It's an *n*on-dimensional space, a hyperplane is a flat affine subspace of one dimension less (e.g., a line in 2D, a plane in 3D). In SVM, the goal is to find the optimal hyperplane that separates the classes.

Support Vectors: These are the data points closest to the hyperplane. These points are crucial in defining the position and orientation of the hyperplane. In essence, SVM relies on these points to form the optimal boundary.

Margin: The margin is the distance between the hyperplane and the closest support vectors. SVM aims to **maximize this margin, as a larger margin leads to better generalization on unseen data. The larger the margin, the lower the model's variance and better its performance.**

Linear vs Non-Linear Classification:

Linear Classification: When the data is linearly separable, SVM finds a straight line (in 2D) or hyperplane (in higher dimensions) to separate the two classes. This is the simplest case where SVM works by finding a hyperplane that maximizes the margin between the two classes.

Non-Linear Classification: When data is not linearly separable (i.e., no straight line can divide the classes), SVM uses a kernel trick to map the data into a higher-dimensional space where a hyperplane can separate the data. This transformation allows SVM to handle complex, non-linear decision boundaries.

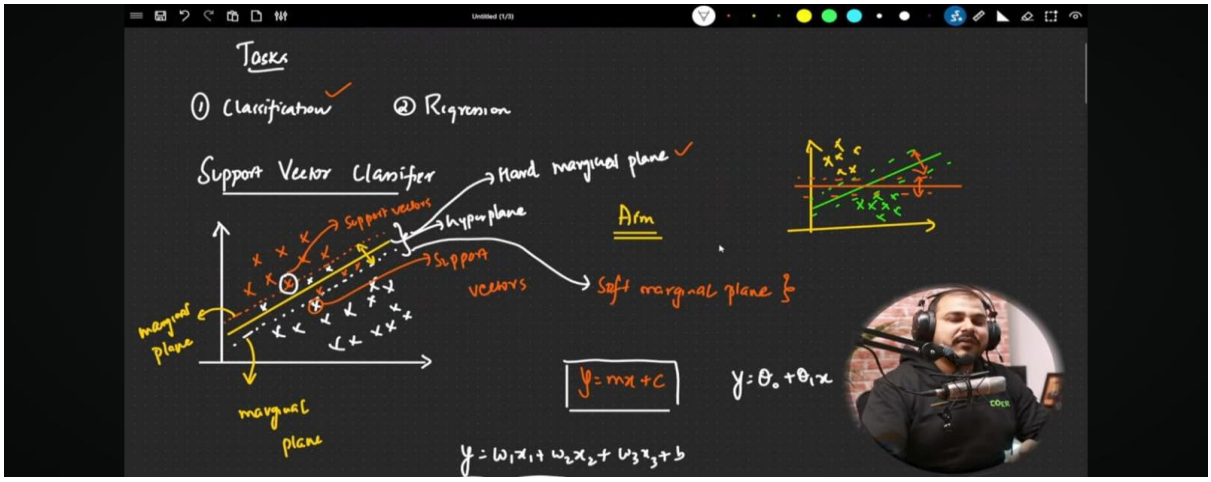
In SVM, the kernel function transforms the data into a higher-dimensional space where a linear hyperplane can be used to separate the data, even if it's not linearly separable in the original space. This transformation is done efficiently using the kernel trick, which avoids explicit computation of the transformation. Common kernels include:

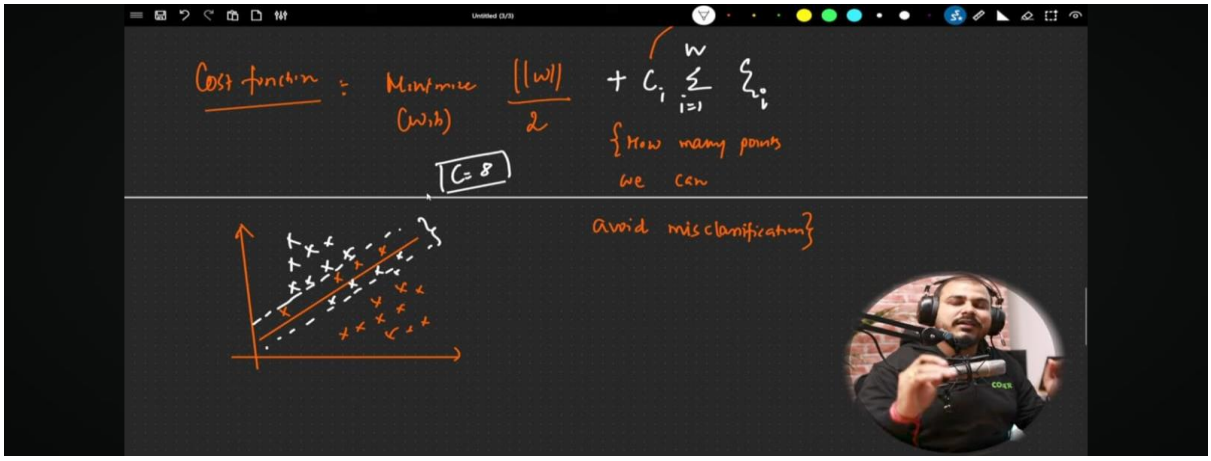
- Linear: For linearly separable data.
- RBF: For complex, non-linear decision boundaries.
- Polynomial: For polynomial decision surfaces.

The **kernel** in **SVM** at the**Linear**, **Polynomial**, and **Radial Basis Function (RBF)**, each suited for different types of data separability.

Support Vector Classification (SVC)

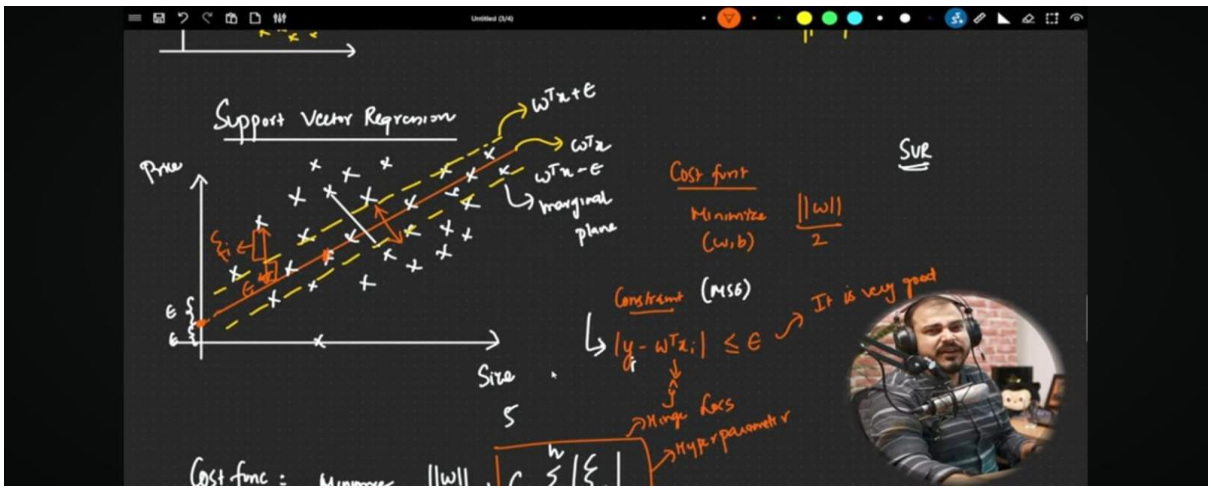
- **Definition**: SVC is used for classifying data into two classes by finding the hyperplane that best separates the classes with the maximum margin. The goal is to correctly classify new data points by making the decision boundary as wide as possible while maintaining accuracy.
- **Cost Function**: The cost function involves two terms:
    - Margin Maximization: Maximize the distance between the hyperplane and the support vectors.
    - Regularization (C): Controls the trade-off between maximizing the margin and minimizing classification errors. A higher CCC values result in fewer misclassifications but may lead to overfitting.
- **Advantages**:
    - Works well for both linear and non-linear data using kernels (e.g., RBF, polynomial).
    - Effective in high-dimensional spaces.
    - Robust to overfitting, especially in high-dimensional spaces.
- **Disadvantages**:
    - Computationally expensive for large datasets, especially with non-linear kernels.
    - Sensitive to kernel choice and parameter tuning (e.g., C, gamma).
    - Not suitable for very large datasets.

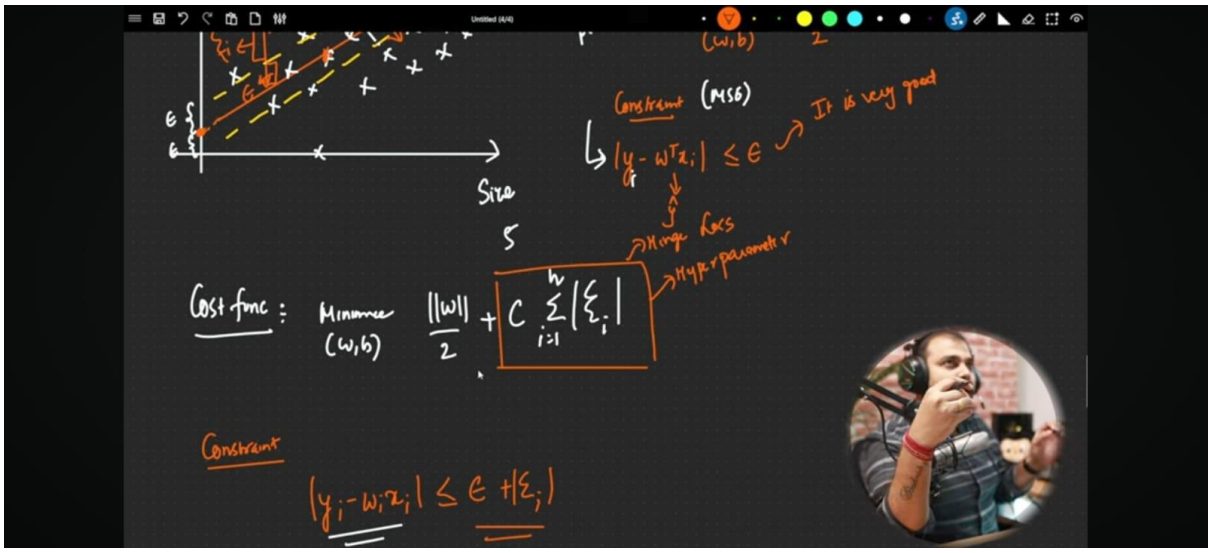




Support Vector Regression (SVR)

- **Definition**: SVR is used for regression tasks where the goal is to find a function that deviates from the true values by no more than a specified margin (epsilon, ϵ\epsilonϵ) while being as flat as possible. The focus is on minimizing errors within the margin.
- **Cost Function**: The cost function for SVR involves:
  - **Epsilon Insensitive Loss**: Data points within the margin (epsilon) are ignored (no penalty).
  - **Regularization**: Similar to SVC, the parameter CCC controls the trade-off between maximizing the margin and minimizing the error.
- **Advantages**:
  - Works well for **non-linear regression** tasks using kernel functions.
  - Robust to outliers, as it does not penalize errors within the margin.
  - Effective in high-dimensional spaces.
- **Disadvantages**:
  - **Sensitive to choice of CCC and ϵ\epsilonϵ**; requires careful tuning.
  - **Computationally expensive** for large datasets.
  - May not perform well if there is a lot of noise in the data.





Comparison between SVM and Logistic Regression:

- **SVM**: Finds a **hyperplane** that maximizes the margin between classes, works well for both **linear and non-linear** data (using kernels). It is more flexible and powerful, especially for complex decision boundaries.
- **Logistic Regression**: Models the probability of class membership using a **sigmoid function** and assumes a **linear relationship** between features and the target variable. It is simpler and faster for linearly separable data.

In short, **SVM** is more suited for complex, non-linear problems, while **Logistic Regression** is effective for simpler, linearly separable data.