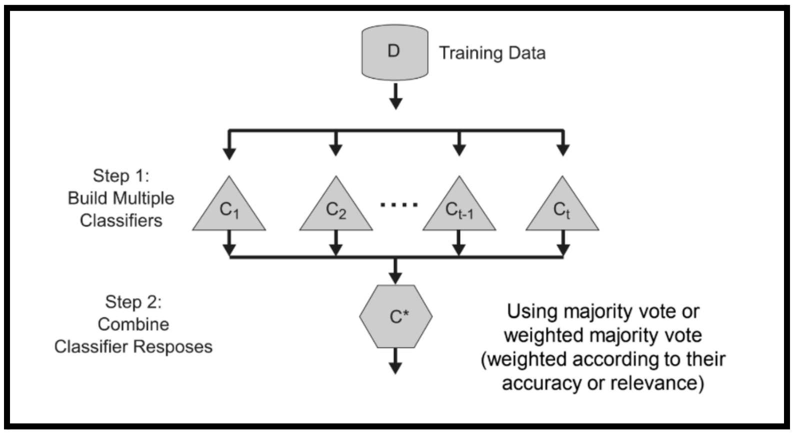
What is Ensemble Learning?

Ensemble learning is a machine learning technique where multiple models ("base models") are combined to solve a problem and improve overall performance.

Instead of relying on a single model, ensemble learning leverages the strengths of multiple models to make better predictions or classifications.

Why Use Ensemble Learning?

1. **Improved Accuracy**: Combining multiple models often results in better predictive accuracy than using a single model.
2. **Reduced Overfitting**: Ensemble methods reduce the risk of overfitting by averaging or combining predictions.
3. **Increased Robustness**: Ensembles are less sensitive to noise and errors in the data.


D
Training Data
Step 1: Build Multiple Classifiers
C_1
C_2
C_{t-1}
C_t
Step 2: Combine Classifier Resposes
C*
Using majority vote or weighted majority vote (weighted according to their accuracy or relevance)

Constructing Ensemble Classifiers:

Ensemble classifiers are built by combining multiple models to improve accuracy. To create these models, we can tweak four things: **training data**, **features**, **class labels**, or the **learning process**. Here's a simpler breakdown:

1. Changing the Training Data

We create different models by using different versions of the training data.

- **Bagging** (Bootstrap Aggregating):
    - Randomly pick subsets of the training data (with duplicates).
    - Train each model on a different subset.

    - Example: In Random Forests, many decision trees are trained on different data subsets.
- **Boosting**:
    - Train models one after the other.
    - Each new model focuses on fixing the mistakes made by the earlier ones.
    - Example: AdaBoost.

2. Changing the Input Features

Use different sets of input features to train each model.

- **Random Forests**:
    - Each tree gets a random set of features to work with.
    - This ensures that each model "sees" the data in a different way, making the ensemble stronger.

3. Changing the Class Labels

Change how class labels are treated to create multiple models.

- **Error-Correcting Output Coding (ECOC)**:
    - Break down a multi-class problem (e.g., predicting "red," "blue," or "green") into multiple smaller problems (e.g., "is it red?" or "is it blue?").
    - Combine the results to predict the original labels.

4. Changing the Learning Process

Introduce randomness or variations in the way models are trained.

- **Randomness in Neural Networks**:
    - Start with different random values (weights) for each neural network.
    - This ensures each model learns differently, even from the same data.

Bias-Variance Trade-Off, Overfitting, and Underfitting:

1. Bias

- **bias** refers to the error introduced by assuming a model is too simple to represent the underlying patterns in the data. It is a systematic error that prevents the model from capturing the complexity of the data.
- A **high bias** model makes the same type of errors consistently and fails to capture the complexity of the data.

2. Variance

- **Variance** is the error introduced by the model being **too sensitive** to small fluctuations in the training data.
- A **high variance** model tries to fit every detail of the training data, even noise, leading to poor performance on new data.

3. Underfitting

- **Definition**: When a model is **too simple** and cannot capture the underlying patterns in the data.
- **Causes**:
  - High bias.
  - Low variance.
  - Insufficient training time or features.
- **Signs**:
  - Poor performance on both training and test sets.

Example:

- A straight line to fit scattered, complex points.

4. Overfitting

- **Definition**: When a model is **too complex** and learns not only the patterns but also the noise in the training data.
- **Causes**:
  - High variance.
  - Low bias.
  - Training the model for too long.
- **Signs**:
  - Excellent performance on the training set but poor performance on the test set.

Example:

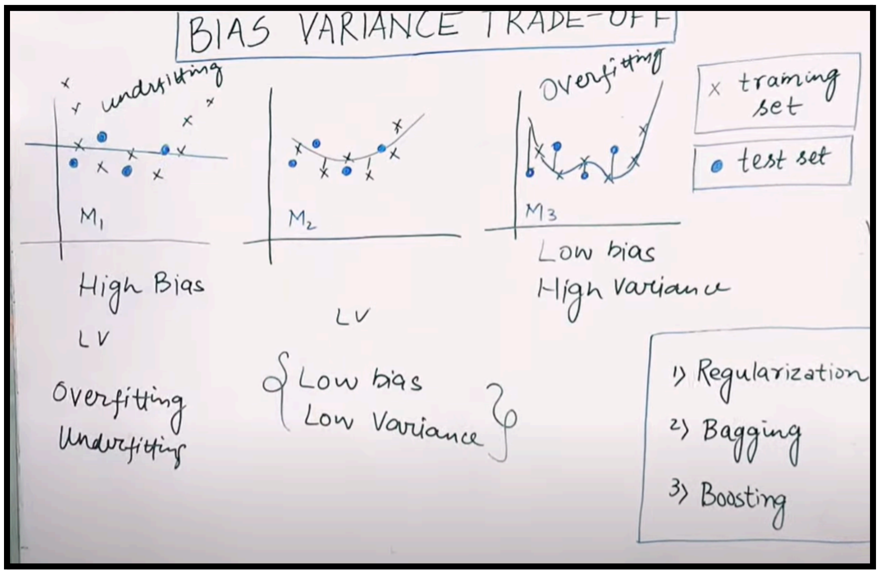
- A model that fits every single data point perfectly but fails to generalize to unseen data.

5. Bias-Variance Trade-Off

This trade-off explains the balance between bias and variance in a model's performance.

- **High Bias (Underfitting)**
- **High Variance (Overfitting)**
- **The Goal**:
  - Achieve **low bias** and **low variance**.
  - This is the "sweet spot" where the model generalizes well to unseen data.

    - By bagging, boosting.



Bagging

Bagging, or **Bootstrap Aggregating**, is a technique used in machine learning to make predictions more stable and accurate. It combines multiple models to work together so the final result is better than relying on just one model.

How Does Bagging Work?

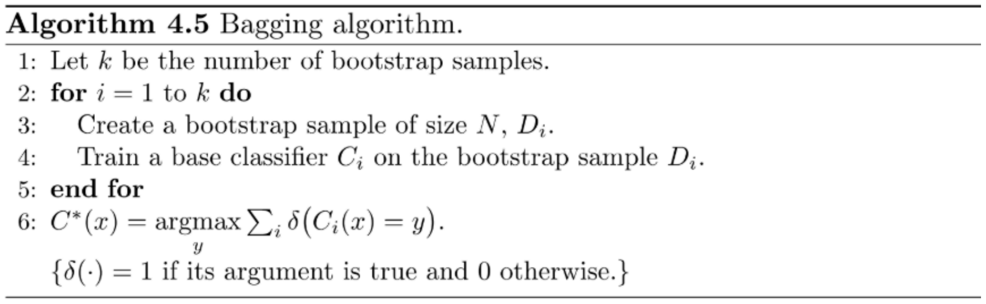
1. **Split the Data**:
    - Take random samples from your dataset (some data points may repeat).
    - Each sample is like a smaller version of your dataset.
2. **Train Multiple Models**:
    - Train a separate model (like decision trees) on each of these random samples.
3. **Combine Predictions**:
    - For classification: Take a **vote** from all models and pick the most popular answer.
    - For regression: Take the **average** of the predictions.

Why Use Bagging?

- **Reduces Overfitting**:
    - By combining models, Bagging avoids getting stuck in the noise of the training data.
- **Reduces Variance**:
    - If one model makes a bad prediction, others will correct it, making the overall prediction more stable.

Where is Bagging Used?

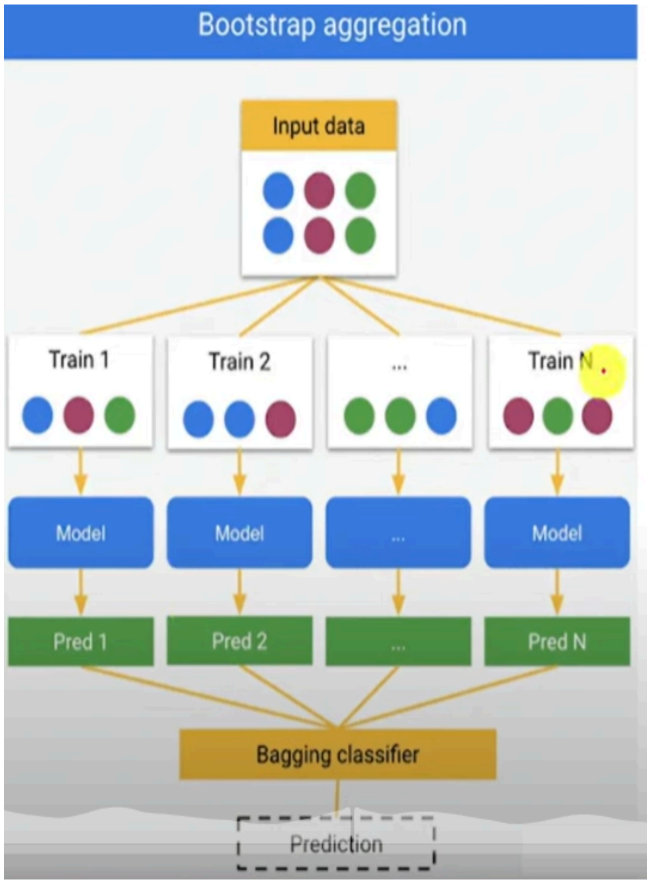
- Often used with **decision trees** because they can overfit easily.
- A well-known example is **Random Forest**, which uses Bagging to combine many decision trees.



Algorithm 4.5 Bagging algorithm.

```
1: Let $k$ be the number of bootstrap samples.
2: for $i = 1$ to $k$ do
3:     Create a bootstrap sample of size $N$, $D_i$.
4:     Train a base classifier $C_i$ on the bootstrap sample $D_i$.
5: end for
6: $C^*(x) = \underset{y}{\text{argmax}} \sum_i \delta(C_i(x) = y)$.
   $\{\delta(\cdot) = 1$ if its argument is true and 0 otherwise.$\}$
```

What is Happening in the Diagram?



1. **Input Data**:
    - The process starts with a single dataset (the input data, shown as colored circles).
2. **Bootstrap Sampling**:
    - Multiple **random subsets** of the input data are created (Train 1, Train 2, ..., Train N).
    - These subsets are created by **sampling with replacement**, meaning some data points can appear more than once in a subset while others might not appear at all.
3. **Train Models**:
    - Each subset is used to train a separate model (Model 1, Model 2, ..., Model N).
    - These models are independent of each other, and they might be decision trees or other types of models.
4. **Predictions**:
    - Each model makes a prediction (Pred 1, Pred 2, ..., Pred N).
5. **Combine Predictions**:
    - A **Bagging Classifier** aggregates the predictions:
        - **For Classification**: Uses majority voting (the prediction that most models agree on).
        - **For Regression**: Averages the predictions.
6. **Final Prediction**:

- The combined result becomes the final prediction.

Boosting

Boosting is a machine learning technique that combines **many weak models** (simple models that don't perform well individually) into a **strong model** that makes accurate predictions.

The key idea is to **focus on mistakes** made by earlier models and improve on them by training subsequent models.

How Does Boosting Work?

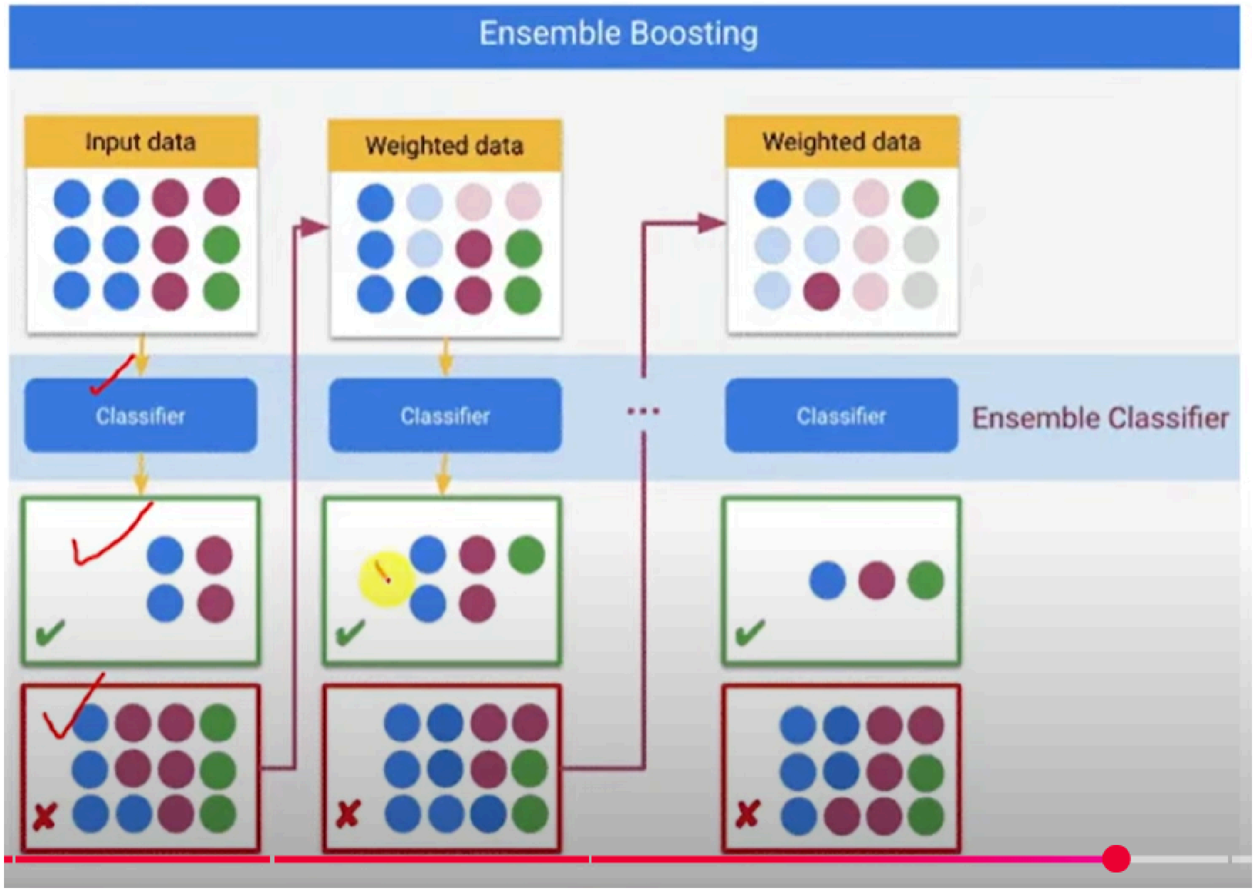
1. **Start with Training Data**:
    - Use the original dataset to train the first model (Classifier 1).
2. **Focus on Errors**:
    - After the first model makes predictions, it identifies which examples it got wrong.
    - Boosting increases the importance (weights) of those incorrectly predicted examples.
3. **Train Next Models**:
    - A second model (Classifier 2) is trained, but it focuses more on the difficult examples that the first model got wrong.
    - This process continues, with each new model focusing on correcting the mistakes of the previous ones.
4. **Combine Models**:
    - All the models are combined (using a weighted sum or voting) to make the final prediction.
    - This creates a strong model from several weak ones.

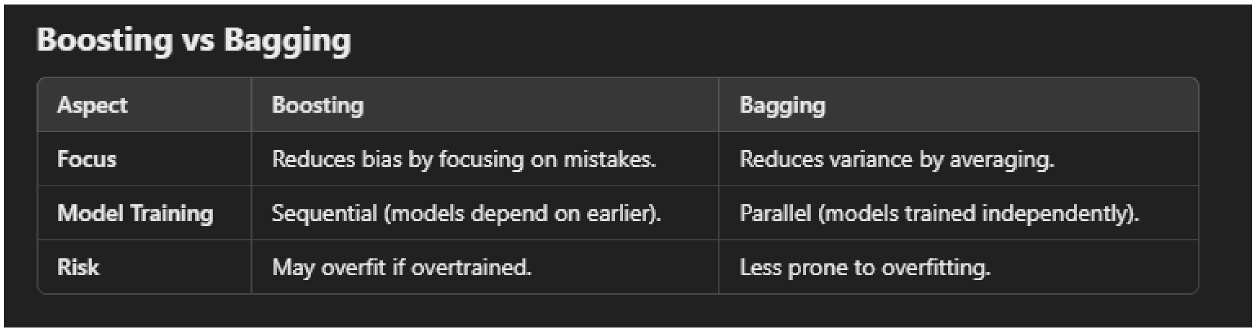
Key Features of Boosting

1. **Trains Models Sequentially**:
    - Unlike Bagging (parallel training), Boosting trains one model after another.
2. **Reduces Bias and Variance**:
    - By focusing on errors, Boosting reduces **bias** (helps avoid underfitting).
    - However, if overtrained, it can increase **variance** (risk of overfitting).
3. **Stops When...**:
    - Boosting continues until:
        - All training examples are correctly classified, or
        - A maximum number of models is reached.

Examples of Boosting Algorithms

1. **AdaBoost** (Adaptive Boosting):
    - Adjusts weights of misclassified examples at each step.





Boosting vs Bagging

Aspect	Boosting	Bagging
Focus	Reduces bias by focusing on mistakes.	Reduces variance by averaging.
Model Training	Sequential (models depend on earlier).	Parallel (models trained independently).
Risk	May overfit if overtrained.	Less prone to overfitting.

What is Random Forest?

- **Random Forest** is an ensemble learning method that combines the predictions of multiple decision trees to make a final prediction. It's called a "forest" because it consists of many decision trees.

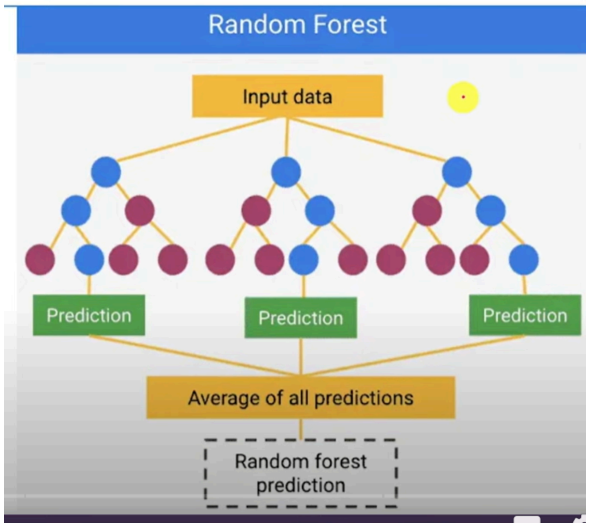
How Does Random Forest Work?

1. **Input Data**:
   - Start with a dataset containing features (inputs) and labels (outputs).

2. **Bootstrap Sampling**:
   - Random subsets of the training data are created (sampling with replacement).
   - Each subset is used to train a separate decision tree.
3. **Random Feature Selection**:
   - Each tree is built using a **random subset of features** instead of all features. This reduces correlation among trees and improves diversity.
4. **Train Decision Trees**:
   - Train multiple decision trees independently on these random subsets.
5. **Combine Predictions**:
   - For **classification**:
     - Each tree votes on the class, and the majority vote is the final prediction.
   - For **regression**:
     - The predictions of all trees are averaged.

Why Use Random Forest?

1. **Improved Accuracy**:
   - By combining multiple trees, the random forest reduces errors and improves predictions.
2. **Reduces Overfitting**:
   - Individual trees might overfit the data, but averaging their predictions helps avoid overfitting.
3. **Handles Large Datasets**:
   - Works well with both large datasets and high-dimensional data.
4. **Robustness**:
   - Less sensitive to noisy data and outliers.


Random Forest
Input data
Prediction
Prediction
Prediction
Average of all predictions
Random forest prediction