*Random Forest*

* A random forest is an ensemble learning method that builds multiple decision trees and combines their outputs to improve prediction accuracy.
* Each tree in the forest, a random subset of features is considered (not all features).
* For classification: it uses majority voting across trees.
* For regression: it uses average prediction of all trees.
* **How it works?**
* **Bootsrapping:** (Sampling with replacement)
* Create multiple random datasets (samples) from the original training data.
* Each sample may have repeated rows.
* **Random Feature Selection at Splits:**
* At each node of every tree, select a random subset of features to find the best split.
* Introduces diversity among trees.
* **Train Multiple Decision Trees**
* Each tree is trained on its own bootstrapped date and random feature.
* Trees are built independently.
* **Aggregate Prediction:**
* Classification: Majority Vote
* Regression: Average of all tree outputs.
* **Important Hyperparameters:**
* N\_estimators: Number of Trees.
* Max\_features: Number of features considered per split.
* Max\_depth: Maximum depth of a tree.
* Min\_samples\_split: Min samples to split a node.
* Bootstrap: Whether bootstrapping is used.
* Random\_state: For reproductibility
* **Pros and Cons:**
* High accuracy on many datasets.
* Reduces overfitting compared to single decision trees.
* Handles missing values & outliers well.
* Feature importance insight.
* Works well on both small and large datasets.
* Slower to train and predict (many trees).
* Less interpretable (“black box” model)
* May require more memory and computation.
* Can still overfit on very noisy data of not turned.
* **Important:**
* Random Forest is less prone to high variance than a single decision tree.
* Adding more trees usually improves stability but increases computational cost.
* Feature importance is one of its biggest interpretability advantages.
* **When & Where Random Forest is Used?**
* You want high accuracy:
* Especially when the data is noisy or contains non-linear relationships.
* It performs well out-of-the-box without much tuning.
* You have a mix of Feature types:
* Works with both numerical and categorical features.
* Can handle missing values reasonably well.
* You want to avoid Overfitting:
* Decision Trees can memorize the training data.
* Random Forest combats this by averaging across many trees, making it more generalizable.
* You care about feature importance:
* Random forest gives you a solei way to rank features by importance, which helps with explainability and feature selection.
* You’re working on Problems like:
* Customer Churn Prediction
* Loan default classification
* Medical Risk Assessment
* Price Range Classification
* Spam detection
* Any tabular data classification or regression
* In general, Random Forest is better in most real-world cases because it:
* Reduces overfitting
* Is more accurate and robust
* Can handle noisy, complex data better
* But, if interpretability and speed matter most (e.g., you want to explain your model to non-tech people), we should go with a decision tree.
* For quick prototyping, a Decision tree is faster.

|  |  |  |
| --- | --- | --- |
| Criteria | Decision Tree | Random Forest |
| Simplicity | Easier to understand | Harder to interpret (many trees) |
| Speed | Faster to train | Slower (many trees to train) |
| Overfitting Risk | Very high | Much lower (averages out noise) |
| Accuracy | Good on simple problems | Better on complex/real-world data |
| Stability | Sensitive to data changes | More stable and robust |
| Feature Insight | Limited | Built-in feature importance |