Random Forest

Hao Le Vu

IUH

2025

Types of Ensemble Methods

1. Bagging

- Stands for *Bootstrap Aggregating*.
- Build multiple models in parallel using random subsets of data.
- Final prediction: average (regression) or majority vote (classification).
- Example: Random Forest.

Types of Ensemble Methods

2. Boosting

- Models are built sequentially, each focusing on previous errors.
- Adaptive weights are assigned to misclassified samples.
- Example: AdaBoost, Gradient Boosting, XGBoost.

Types of Ensemble Methods

3. Stacking

- Combine predictions of multiple diverse models using a meta-model.
- The meta-model learns to optimally weight the base models.
- Example: Using Logistic Regression as a meta-learner.

Random Forest Classification: Concept

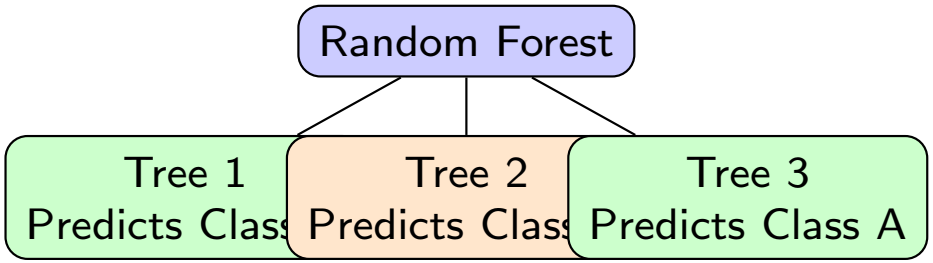
Definition

- **Random Forest** is an **ensemble learning algorithm** that builds multiple decision trees and combines their predictions.
- It uses the principle of **bagging** (Bootstrap Aggregating).
- Each tree votes for a class, and the final prediction is made by **majority voting**.

Key Idea

- Reduce overfitting of a single decision tree by averaging multiple diverse trees.
- Diversity comes from random sampling of both **data** and **features**.

Random Forest: Overview Illustration



Final Prediction: Class A (majority vote)

Multiple decision trees vote to produce a robust final prediction.

How Random Forest Works

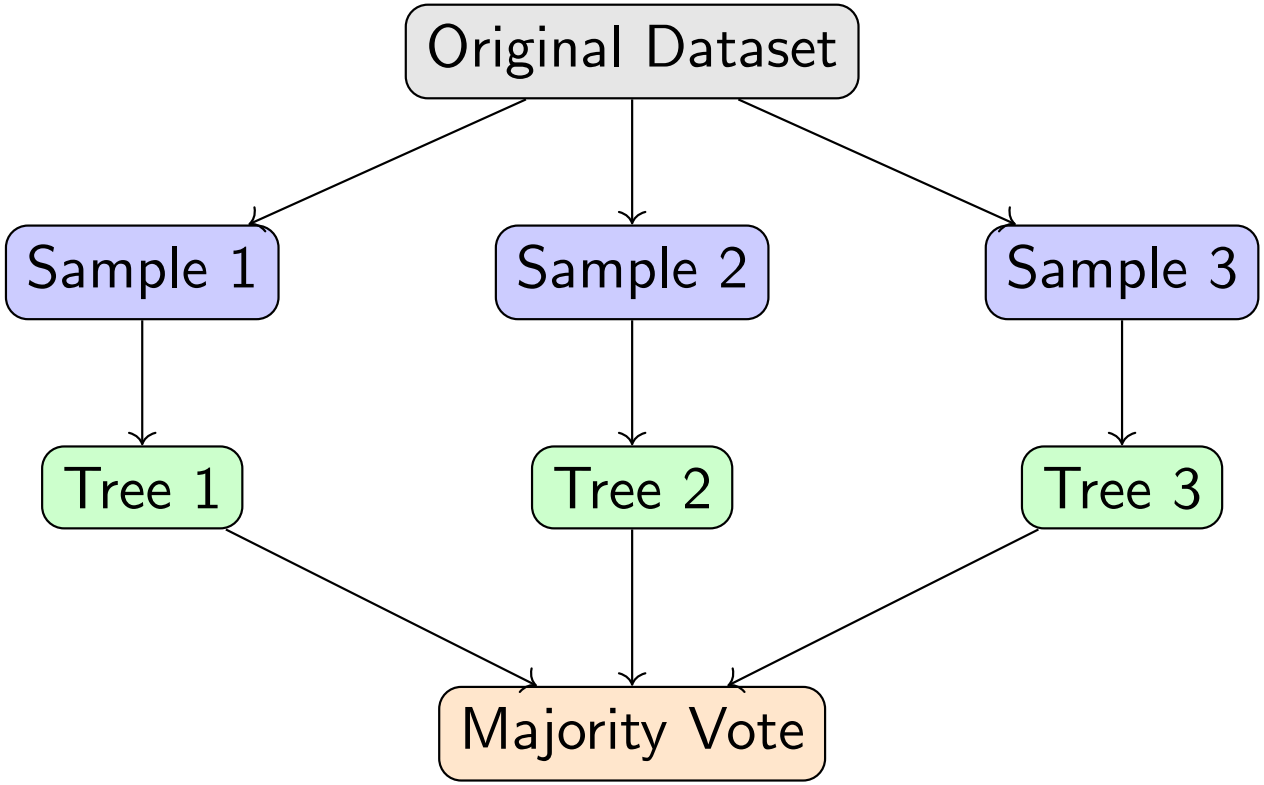
Training Process

1. Create B bootstrap samples from the original dataset.
2. Train a decision tree on each sample:
   - At each split, randomly select a subset of features.
   - Find the best split using only this subset.
3. Repeat until B trees are built.

Prediction Process

- Each tree outputs a class prediction.
- Final prediction is based on **majority voting**.

Random Forest: Workflow Visualization



Random Forest uses bootstrapped samples and feature randomness to grow diverse trees.

Why Random Forest Improves Performance

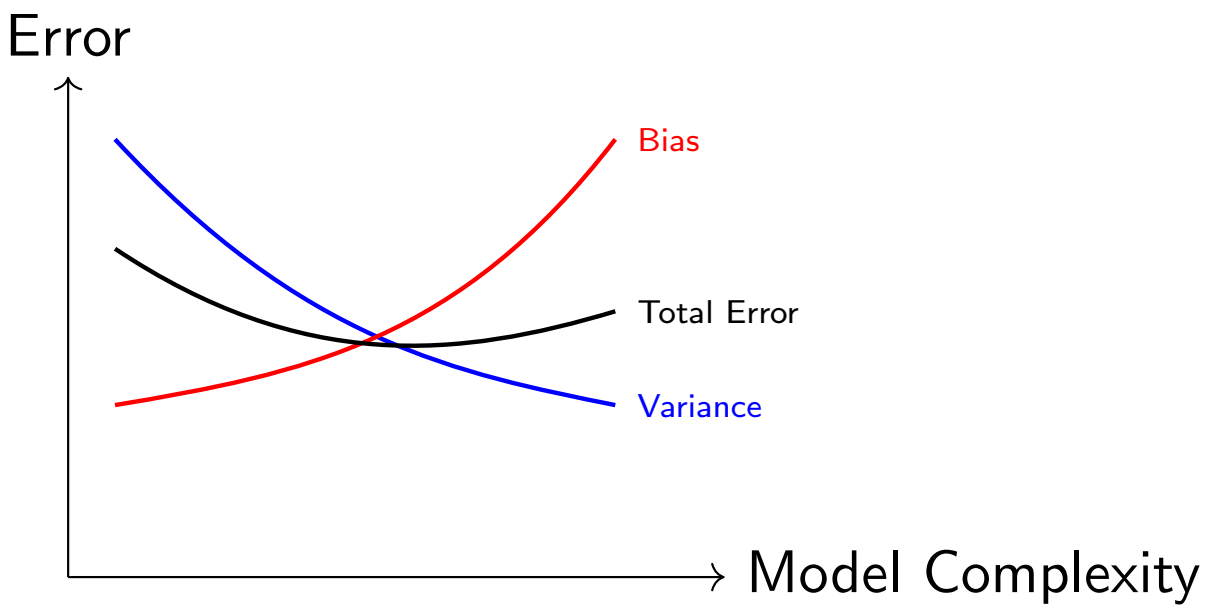
Ensemble Principle

- Combining multiple weak learners (trees) results in a stronger overall model.
- Errors from individual trees tend to cancel each other out.
- Works well when trees are diverse and uncorrelated.

Bias-Variance Tradeoff

- A single deep decision tree has:
  - Low bias
  - High variance (overfits easily)
- Random forest reduces variance without significantly increasing bias.

Bias-Variance Tradeoff Visualization



Random Forest balances bias and variance by combining multiple trees.

Random Forest: Key Hyperparameters

Tree Structure

- **n_estimators:** Number of trees in the forest.
- **max_depth:** Maximum depth of each tree.
- **min_samples_split:** Minimum samples required to split a node.
- **min_samples_leaf:** Minimum samples required in a leaf node.

Feature Randomness

- **max_features:** Number of features to consider at each split.
- **bootstrap:** Whether to use bootstrap sampling (True/False).

Other

- **random_state:** Reproducibility.
- **class_weight:** Handle class imbalance.

Random Forest: Advantages and Limitations

Advantages

- High accuracy and robustness.
- Reduces overfitting compared to a single decision tree.
- Works well with both numerical and categorical features.
- Handles missing data effectively.

Limitations

- Less interpretable than a single decision tree.
- Can be computationally expensive for very large datasets.
- Not ideal for datasets with extremely high dimensionality unless tuned carefully.

Python Example: Random Forest Classification

```python
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

# Load dataset
iris = load_iris()
X, y = iris.data, iris.target

# Split into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Train Random Forest
model = RandomForestClassifier(n_estimators=100, max_depth=3, random_state=42)
model.fit(X_train, y_train)

# Evaluate
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```