**Decision Trees VS Random Forests**

**Understanding Ensemble Learning and Overfitting Control**

**Introduction**

We shall learn Decision Tree-like methods and Random Forest, which is an ensemble technique aimed at improving upon the classical tree methods by reducing overfitting and increasing robustness ((Breiman, 2001) In this tutorial, we are going to implement both methods using the Breast Cancer Wisconsin data set. You will learn about the decision tree working principles, its overfitting inability, and how Random Forests become more stable and accurate in an ensemble way.

By the end of this tutorial, you'll be able to:

* Explain how decision trees work (with examples)
* Understand entropy and Gini impurity
* Identify why trees can overfit
* Train a Random Forest and compare it to a single tree
* Visualize decision paths and evaluate performance
* Apply both models to a real-world medical dataset

**About the Dataset: Breast Cancer Wisconsin (Diagnostic)**

The Breast Cancer Wisconsin (Diagnostic) Dataset, a popular benchmark in medical machine learning research, has been used for this tutorial. The database includes a total of 569 records, which were made by digitizing images of fine needle aspirate (FNA) taken from breast masses. It is made up of 30 numerical attributes regarding characteristics of the cell nuclei (e.g. radius, texture, perimeter, area, smoothness) for each record. The target variable is a class binary indicating one of the two possible outcomes: malignant (1) or benign (0). It has a rich feature set and a balanced class distribution, which makes the dataset increasingly suitable for tree-based models. It exists in the built-in datasets of scikit-learn, thus can easily be loaded and put into use without the external downloads (Dua & Graff, 2017).

* *You can load it directly using:*

from sklearn.datasets import load\_breast\_cancer

**Section 1: Understanding Decision Trees**

A decision tree will be defined as a flowchart in which each internal node represents a decision taken about a feature, each branch indicates the outcome of a decision, and each terminal leaf node represents a class label or a final decision (Quinlan, 1986). The tree will continue to recursively partition the data into smaller subsets trying to maximize a pureness of the classes with respect to impurities like entropy or Gini index to give informative split points (Han et al. 2012).

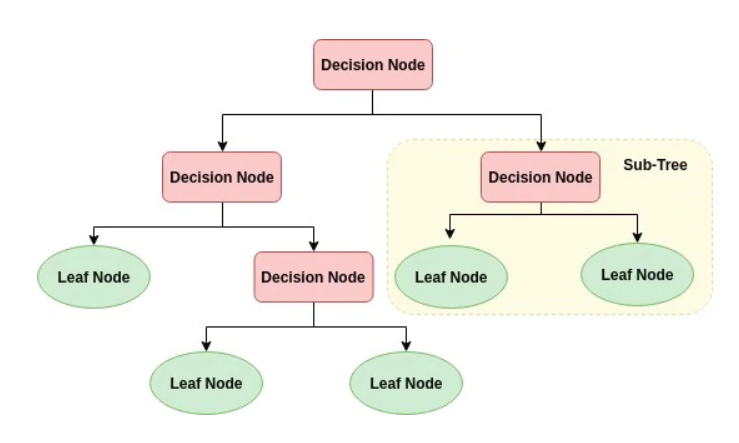
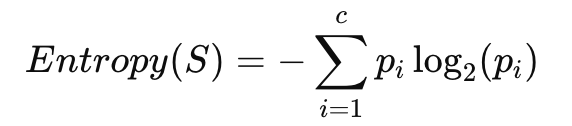


Figure Illustration of a basic decision tree with root, branches, and leaves

**Splitting Logic and Impurity Measures**

**Common Metrics for Splitting**

1. **Entropy**

****

Entropy measures disorder. The more mixed the classes are, the higher the entropy.

**2. Information Gain**

A black text on a white background

AI-generated content may be incorrect.

This metric calculates how much entropy is reduced by splitting the data on feature AAA.

**3. Gini Impurity**

A black and white math equation

AI-generated content may be incorrect.

Gini is simpler and faster than entropy and is the default in scikit-learn’s CART-based Decision Trees (Breiman et al., 1984).

**Limitations of Decision Trees**

Although intuitive and fast, decision trees have drawbacks:

|  |  |
| --- | --- |
| **Problem** | **Explanation** |
| **Overfitting** | Deep trees can memorize the training set instead of generalizing |
| **Instability** | Small changes in data may produce different trees |
| **Poor generalization** | Especially when data is noisy or imbalanced |

This is why **ensemble methods** were introduced — to combine multiple weak learners into a strong one.

**Section 2: Understanding Random Forests**

A Random Forest is a powerful ensemble learning technique that constructs and merges several decision trees to generate a final output. Each individual tree is trained using a randomly chosen portion of the dataset and a random selection of features (Breiman, 2001).

This approach enhances the robustness and generalization ability of the model, making it more stable and less prone to overfitting.

**How It Works:**

1. **Bootstrap sampling**: Random samples are drawn with replacement.
2. **Random feature selection**: Only a few features are used at each split.
3. **Aggregation**: For classification, predictions are combined using majority vote; for regression, the average is taken.

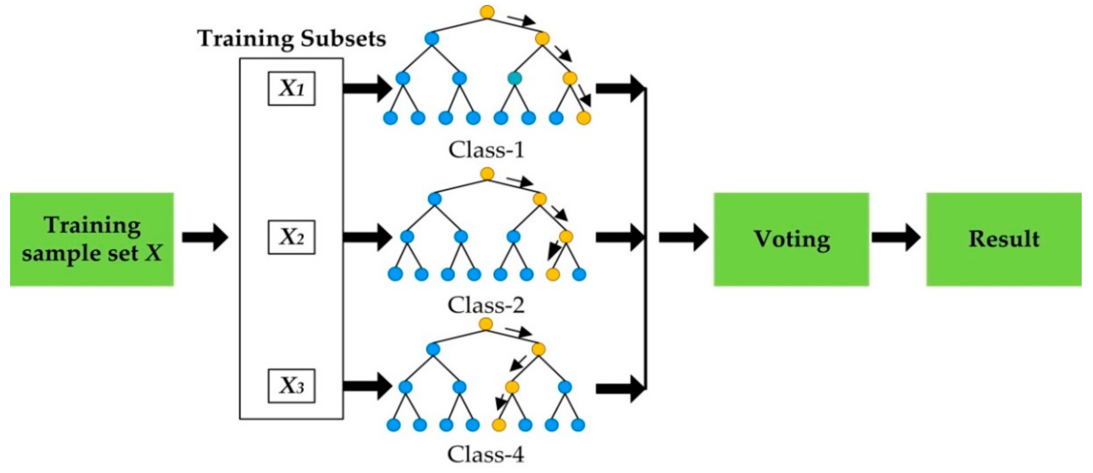


Figure Diagram showing multiple trees trained on subsets of data and voting on final prediction.

**Why It Works**

Randomness helps create **decorrelated trees**, meaning that errors made by one tree are not likely to be repeated by others. Combining these reduces variance and overfitting, while still maintaining low bias (Dietterich, 2000).

**Decision Tree vs. Random Forest – Comparison Table**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Decision Tree** | **Random Forest** |
| Accuracy | Moderate (can overfit) | High (ensemble effect) |
| Variance | High | Low |
| Interpretability | High | Moderate (multiple trees) |
| Training Time | Fast | Slower (more computation) |
| Overfitting Risk | High | Low |

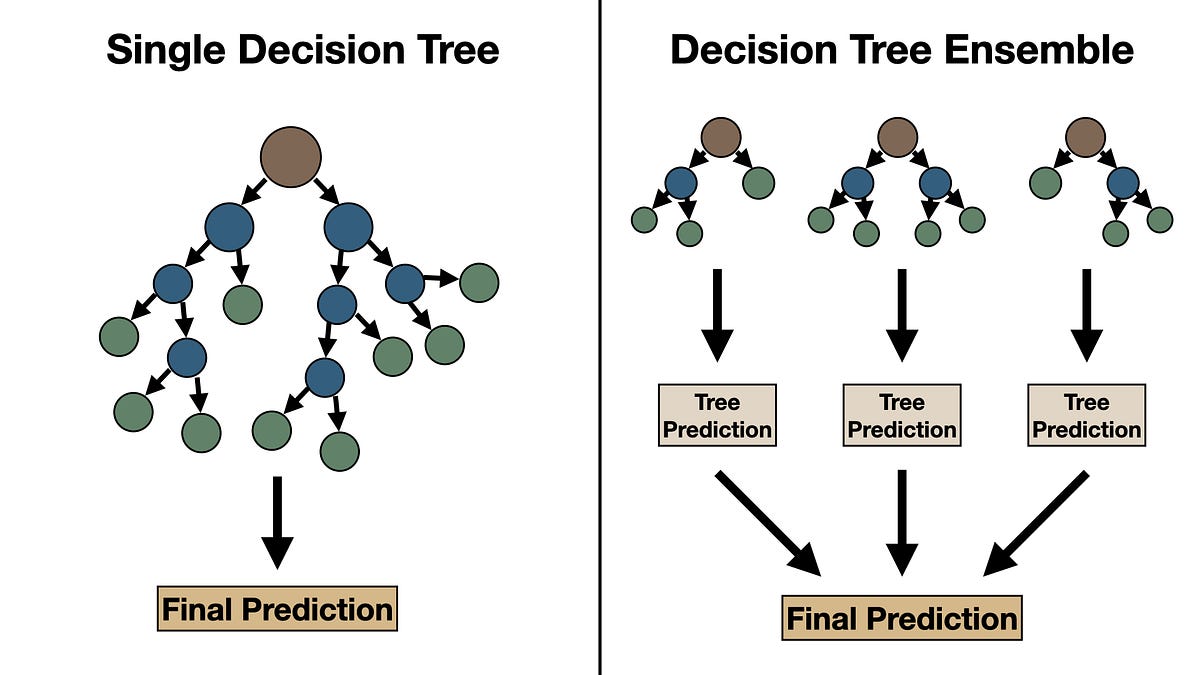


Figure Side-by-side visual of single tree vs. ensemble voting mechanism.

**Summary Before Coding**

* Decision Trees are intuitive and interpretable.
* However, they are often overfit and are sensitive to noise.
* Random Forests solve this by combining many decision trees with added randomness, resulting in better generalization and accuracy.

Next, we’ll implement both models on a real dataset and compare their performance step by step.

**

**Section: Step-by-Step Implementation – Decision Tree vs. Random Forest**

We’ll use the **Breast Cancer Wisconsin dataset**, available directly in scikit-learn, to compare the performance of a Decision Tree and a Random Forest.

**Step 1: Import Required Libraries**

**A screen shot of a computer code

AI-generated content may be incorrect.**

Figure Screenshot of Colab cell importing essential libraries

**Step 2: Load and Explore the Dataset**

A screenshot of a computer code

AI-generated content may be incorrect.

A table with numbers and text

AI-generated content may be incorrect.

Figure Colab screenshot of the first few rows of the dataset

**Dataset Summary**

df.describe()

A table with numbers and letters

AI-generated content may be incorrect.

Figure Statistical summary showing mean, std, min, max of each feature

**Step 3: Split and Scale the Data**

A screenshot of a computer program

AI-generated content may be incorrect.We standardized the data to ensure all features are on the same scale, which is essential for tree ensembles to perform optimally.

**Step 4: Train a Decision Tree Classifier**

**A black text on a white background

AI-generated content may be incorrect.**

Figure Screenshot of Decision Tree training in Colab.

**Step 5: Train a Random Forest Classifier**

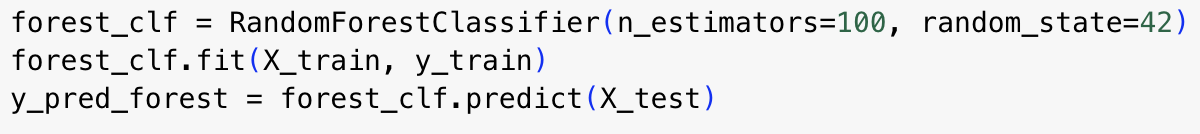


Figure Colab screenshot showing Random Forest training

**Step 6: Compare Model Performance**

A close-up of a code

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure Terminal output showing higher precision, recall, and F1-score for Random Forest

**Step 7: Confusion Matrix Visualization**

A computer screen shot of a code

AI-generated content may be incorrect.

A group of squares with numbers and a number

AI-generated content may be incorrect.

Figure Side-by-side confusion matrix comparison — Random Forest shows fewer misclassifications

**Step 8: Visualize a Single Tree**



A diagram of a tree

AI-generated content may be incorrect.

*Decision tree visualization showing top-level splits*

**References**

* Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5-32.
* Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press.
* Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1), 81–106.
* Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques*. Elsevier.
* Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1–15). Springer.
* Dua, D. and Graff, C. (2017). *UCI Machine Learning Repository*. Irvine, CA: University of California, School of Information and Computer Science. <https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)>

**🔗 Repository Links:**

|  |  |
| --- | --- |
| **Component** | **Link Placeholder** |
| notebook.ipynb | https://github.com/Rehman132/Individual-assignment-Machine-learning-tutorial/blob/main/decision\_tree\_random\_forest\_tutorial.ipynb |
| README.md | <https://github.com/your-username/random-forest-tutorial> |
| LICENSE (MIT) | <https://github.com/your-username/random-forest-tutorial/blob/main/LICENSE> |
| Final Report (.pdf/.docx) | <https://github.com/your-username/random-forest-tutorial/blob/main/tutorial.pdf> |