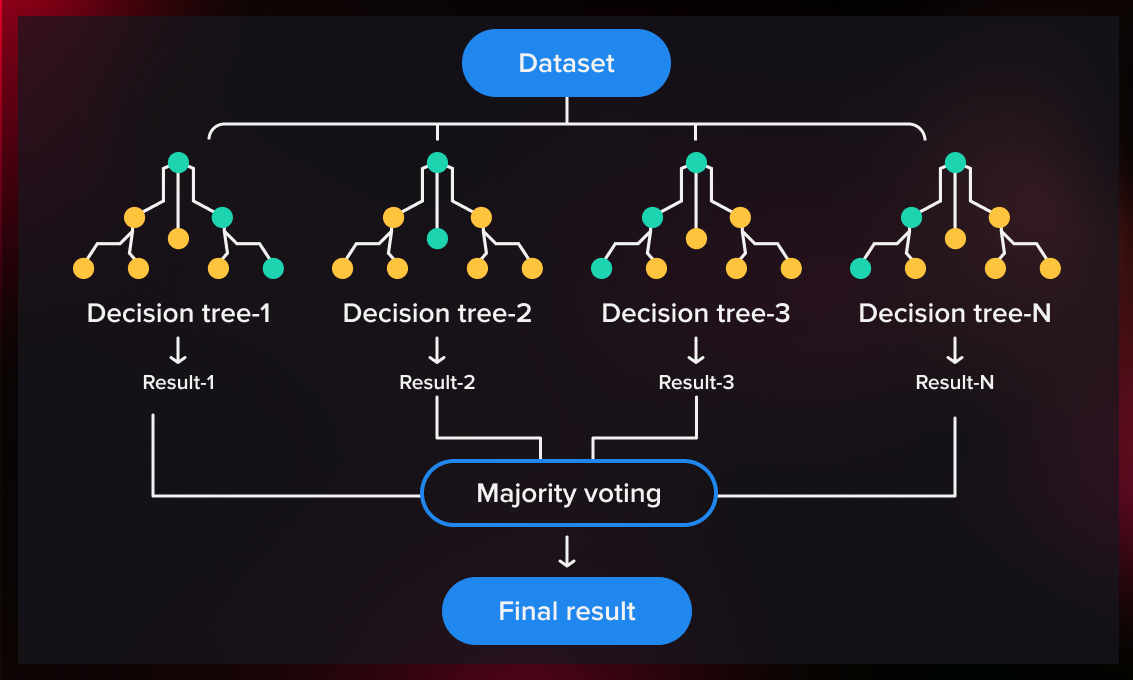
**Random Forest Algorithm**

*Advantages, Disadvantages, and Implementation in Machine Learning*



*Name: Plamen Svetoslavov*

*Course: Math Concepts for Developers*

*Subject: Project – Analysis of a Machine Learning Algorithm*

**History**

**The Random Forest algorithm**, a cornerstone in machine learning, was developed by Leo Breiman and Adele Cutler in the early 2000s. Building upon Breiman's earlier work on bagging (bootstrap aggregating) and the random subspace method introduced by Tin Kam Ho in 1995, they combined these concepts to create an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Breiman and Cutler's innovation addressed the limitations of **single** **decision trees**, particularly their tendency to overfit the training data. By introducing randomness in both the data samples (through bootstrapping) and the feature selection at each split in the tree, Random Forests reduce variance and improve predictive accuracy. This methodology not only enhances performance but also provides measures of feature importance, aiding in the interpretability of the model.

In 2006, "Random Forests" was registered as a trademark by Breiman and Cutler, reflecting the significance of their contribution to the field. Since then, Random Forests have become widely adopted due to their robustness, ease of use, and versatility in handling both classification and regression tasks across various domains.

**What is a Decision tree?**

Because a random forest comprises multiple decision trees, it is appropriate to first outline the decision tree algorithm. A decision tree begins with an initial question, for example, “Should I go surfing?” Subsequent questions, such as “Is there a long-period swell?” or “Is the wind offshore?”, are posed to refine the decision. Each question represents a decision node that partitions the data into subsets. As each split occurs, the observation moves closer to a final outcome, represented by a leaf node. Observations satisfying a condition follow the "Yes" branch, while those that do not follow the alternative.

The objective of a decision tree is to find the best splits. The most common methodology used is the **Classification and Regression Tree (CART)** algorithm, which selects splits based on metrics such as:

* ***Gini Impurity:***

Where is the probability of class in node and is the number of classes.

* ***Entropy (Used for Information Gain):***
* **Mean Squared Error (MSE) – used in regression trees:**

*Although decision trees are a fundamental supervised learning method, they are susceptible to bias and overfitting. However, assembling multiple decision trees into a random forest significantly enhances predictive performance, particularly when the trees are uncorrelated.*

**Ensemble methods**

**Ensemble learning methods** involve combining multiple classifiers, such as decision trees, and aggregating their predictions to determine the most common outcome. Two of the most recognized ensemble techniques are bagging, also known as bootstrap aggregation, and boosting. In 1996, **Leo Breiman** introduced the [bagging technique](https://link.springer.com/content/pdf/10.1007/BF00058655.pdf), where random samples are drawn from the training set with replacement—allowing the same data points to be selected multiple times. After creating several such samples, separate models are trained independently, and depending on the task—regression or classification—their outputs are averaged or majority-voted to produce a more accurate prediction. This method is widely used to minimize variance in noisy datasets.

**Random Forest Algorithm**

**A Random Forest** is formally defined as a collection of tree-structured classifiers {h(x, Θk)}, where each tree is trained on a randomly drawn subset of the data and features. The random vectors Θk are independent and identically distributed, ensuring that each tree brings unique structure and decisions to the ensemble. The final prediction is determined by aggregating the outputs of all trees, either through majority vote (classification) or average (regression).

**The random forest algorithm** builds upon the bagging approach by combining both data sampling and feature randomness to create an ensemble of less correlated decision trees. Feature randomness—also known as feature bagging or the [*"random subspace method"*](https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf)—involves selecting a random subset of features for each tree, promoting diversity among the trees.

**This is one of the major differences between standard decision Trees and random forests. While traditional decision trees evaluate all** available features when splitting nodes, random forests consider only a random selection.  
For example, if we think back to the "Should I surf?" scenario, the questions I might ask to reach a decision may differ significantly from those another person would consider. By capturing this natural variability within the data, random forests help reduce the risks of overfitting, bias, and high variance, leading to more reliable and accurate predictions.

*Before we start with the Random Forest algorithm, let's see how a standard decision tree works.*

**Sandart decision tree**

|  |  |  |
| --- | --- | --- |
| **Weather** | **Windy** | **Surf / Decision** |
| Sunny | No | **No** |
| Sunny | Yes | **No** |
| Overcast | No | **Yes** |
| Rainy | No | **Yes** |
| Rainy | Yes | **No** |
| Overcast | Yes | **Yes** |

**Choose the method by which we will split the data:**

* Gini impurity (most commonly used method)
* Information Gain

We choose the best feature -> **Weather**.  
Build the first level of the tree:

Weather = Sunny -> Surf = No  
Weather = Sunny -> Surf = No  
**The two rows are "No"; it is a clean node, and the result is "No".**

Weather = Overcast -> Surf = Yes  
Weather = Overcast -> Surf = Yes  
***The two rows are "Yes", it is a clean node and the result is "Yes".***

Weather = Rainy -> Surf = Yes  
Weather = Rainy -> Surf = No  
**Mixed classes. The node is not clean, and we need to use another feature.  
This creates the next level of the tree.**

Since the Rainy group still has mixed classes, we selected the best available feature, which is Windy, to further split the data.

The next node is **Windy**:

Windy = No -> Surf = Yes

Windy = Yes -> Surf = No

**After the split based on Windy, all resulting nodes become clean.  
The tree is now fully built, and predictions are made by traversing from the root node to the appropriate leaf node based on the input data.**

***The goal of every decision tree is to split the data in such a way that each final node (leaf) contains examples with the same output.***

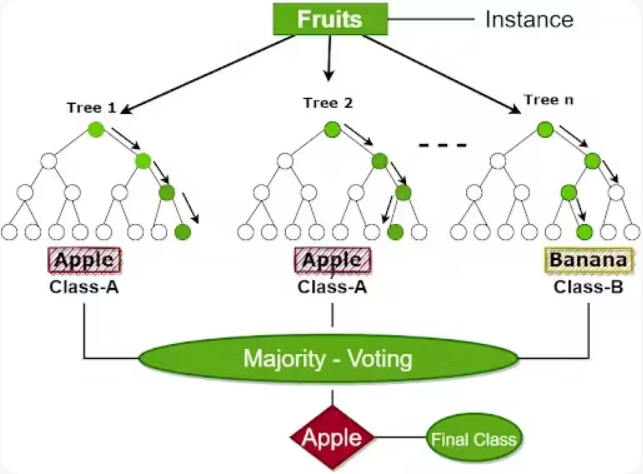
**Summary**

*In a standard Decision Tree, the final prediction is made by a single tree, which is trained entirely on the data provided. The model uses all available features and examples to create a sequence of decision rules that best split the data into pure groups. Although simple and easy to interpret, a standard Decision Tree is prone to overfitting, especially when trained on small or noisy datasets.*

**How the Random Forest Algorithm Works**

**The Random Forest algorithm** builds multiple decision trees during training and combines their outputs to make more accurate and stable predictions. Each tree is trained on a different random subset of the training data **(a technique called *bagging*)** and uses a random subset of features when splitting nodes (*feature randomness*). This randomness ensures that the trees are diverse and less likely to make the same errors.

**Example** – Imagine a dataset consisting of images of various fruits. This dataset is used to train a Random Forest classifier. Each decision tree in the forest is trained on a different subset of the data. During the training process, each tree makes its own predictions. When a new image is introduced, the Random Forest determines the final classification based on the majority vote from all the trees.



When making a prediction:

* In classification tasks, each tree "votes" for a class, and the majority vote is selected as the final output.
* In regression tasks, the average of all tree outputs is taken.

*This ensemble approach reduces overfitting compared to a single decision tree and improves generalization, making Random Forest a powerful and reliable algorithm for classification and regression problems.*

**Key Characteristics of Random Forest**

Random Forest is an ensemble algorithm that uses multiple decision trees to make more stable and accurate predictions. Its key characteristics include bagging (bootstrapping random subsets of data with replacement) and random feature selection when splitting nodes (feature randomness). This reduces correlation between trees and improves the model’s generalization. Random Forest is suitable for both classification and regression tasks.

According to Breiman’s original paper (2001), the accuracy of a Random Forest depends on two key factors: the **strength** of the individual trees and the **correlation** between them. The strength measures how well each tree performs on its own, while the correlation indicates how similarly the trees make errors. The ideal scenario is to have strong trees that are uncorrelated, as this leads to lower generalization error. This balance is what gives Random Forest its power and stability as an ensemble method.

**Training Process in Random Forest**

Training a Random Forest begins by generating multiple random subsets of the original training data using bootstrapping. A separate decision tree is built for each subset. When splitting a node, the tree randomly selects a subset of the

available features and only uses those to determine the best split. This process increases diversity among the trees, resulting in better general performance. Unlike standard decision trees, trees in a Random Forest are typically not pruned.

**Making Predictions with Random Forest**

Once the forest is trained, predictions are made by aggregating the outputs of all trees. For classification tasks, each tree casts a vote for a class, and the class with the majority vote becomes the final prediction. In regression tasks, the average of all predictions is taken as the final result. While individual trees may be weak predictors, their combined output forms a strong and reliable model.

**Advantages**

Random Forest offers many advantages. It is highly resistant to overfitting, especially when the number of trees is sufficiently large. It performs well even with large and noisy datasets. One of its most helpful features is the automatic estimation of feature importance, which assists in data analysis. Additionally, Random Forest can handle missing values and maintain accuracy even with incomplete data.

**Another important advantage** is that Random Forest is highly robust to noise. Even when a small percentage of class labels in the training data are incorrect, Random Forest tends to maintain stable performance. This is because it does not focus excessively on individual data points, unlike algorithms like Adaboost which can overfit to noisy labels.

**Disadvantages**

Despite its strengths, Random Forest has some limitations. Training and inference can be slow, particularly when the number of trees is large. The model is also harder to interpret compared to a single decision tree. Furthermore, it may require significant memory and computational resources.

**Out-of-Bag (OOB) Evaluation**

One of Random Forest's unique features is the use of Out-of-Bag (OOB) evaluation. Since each tree is trained on a bootstrap sample, roughly one-third of the data is not used for that particular tree. These unused observations—OOB data—can serve as a validation set to assess model accuracy without needing a separate test set.

Where is the set of trees that did not use sample

This acts as an internal validation method without needing a separate test set.

**Feature Importance**

Random Forest provides a built-in way to determine the importance of each feature. This is done by measuring how much each feature contributes to impurity reduction or by comparing model accuracy before and after shuffling the values of a specific feature. This is useful for feature selection and understanding which variables influence the output the most.

**Where:**

* : decrease in impurity at split
* : all splits using feature in tree

**Hyperparameters**

Random Forest includes several important hyperparameters that can be tuned to improve performance:

* n\_estimators: number of trees in the forest
* max\_features: number of features considered when looking for the best split
* max\_depth: maximum depth of the trees
* min\_samples\_split and min\_samples\_leaf: minimum number of samples required to split a node or form a leaf
* bootstrap: whether to use bootstrap sampling
* Tuning these parameters is essential to balance accuracy and computational efficiency.

**Applications of Random Forest**

Random Forest is widely used in various domains:

* **Medicine** – disease diagnosis based on symptoms and lab results
* **Finance** – credit risk assessment and fraud detection
* **Computer Vision** – image classification
* **Natural Language Processing** – text analysis and automatic categorization
* **Industry** – defect prediction, machinery maintenance forecasting, etc.