Ensemble Learning

References:

- Wikipedia -Decision Tree
- Geeksforgeeks - Iterative Dichotomiser 3 (ID3) Algorithm From Scratch ⭐
- Geeksforgeeks - Implementing CART (Classification And Regression Tree) in Python
- Geeksforgeeks - ML | Bagging classifier
- Geeksforgeeks - Random Forest Algorithm in Machine Learning
- Quora - Does random forest select a subset of features for every tree or every node?
- Geeksforgeeks - Boosting in Machine Learning | Boosting and AdaBoost ⭐

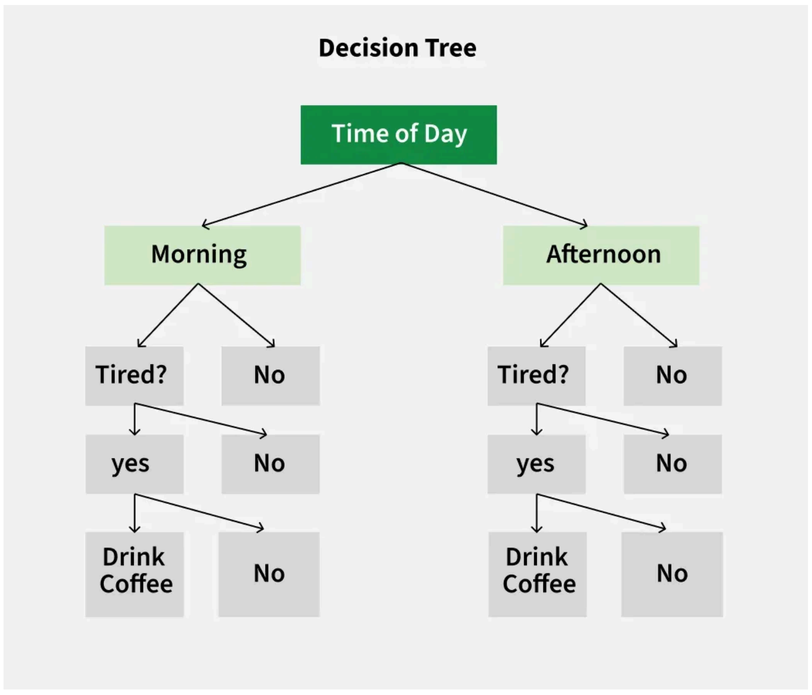
Ensemble learning is a supervised machine-learning technique that combines multiple models to build reliable models and prediction accurate model. It works by combining the strengths of multiple model to create a model that is robust and less likely to overfit the data.

There are two main types of ensemble methods:

- **Bagging**: Models are trained independently on different subsets of the data, and their results are averaged or voted on.
- **Boosting**: Models are trained sequentially, with each one learning from the mistakes of the previous model.

Decision Tree

In ensemble learning, the most popular weak learner is **Decision Tree**. it is a flowchart-like structure in which each internal node represents a **test on an attribute**, each branch represents the **outcome of the test**, and each leaf node represents a **class label** (decision taken after computing all attributes). The paths from root to leaf represent classification rules.



ID3

ID3 (Iterative Dichotomiser 3), proposed by Ross Quinlan in 1986, is a classical implementation of decision tree algorithm. The algorithm works as follows:

1. **Selecting the Best Attribute**:
    - Iterate over all attributes, and for each attribute, iterate over all possible values, calculate the entropy, and select the attribute and threshold value that results in the most significant information gain when used for splitting the data.

2. **Creating Tree Nodes**:
    - The chosen attribute is used to split the dataset into subsets based on its distinct values.
    - For each subset, ID3 recurses to find the next best attribute to further partition the data, forming branches and new nodes accordingly.
3. **Stopping Criteria**:
    - The recursion continues until one of the stopping criteria is met, such as when all instances in a branch belong to the same class or when all attributes have been used for splitting.
4. **Handling Missing Values**:
    - ID3 can handle missing attribute values by employing various strategies like attribute mean/mode substitution or using majority class values.

> A quick review of information theory:
>
> - **Entropy**: A measure of disorder or uncertainty in a set of data. For a set S with classes $\{c_1, c_2, \ldots, c_n\}$, the entropy is calculated as:
>
> $$H(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$
>
>   where p_i is the proportion of instances of class c_i in the set.
> - **Information Gain**: A measure of how well a certain quality reduces uncertainty:
>
> $$IG(S, A) = H(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} \cdot H(S_v)$$
>
>   where S is the dataset, A is the attribute, S_v is the subset of S with the value of A equal to v, and $values(A)$ is the set of possible values of A.
> - **Gain Ratio**: An improvement on Information Gain that considers the inherent worth of characteristics that have a wide range of possible values:
>
> $$GR(S, A) = \frac{IG(S, A)}{\sum_{v \in values(A)} \frac{|S_v|}{|S|} \cdot \log_2 \frac{|S_v|}{|S|}}$$
>
>   where $H(S)$ is the entropy of the dataset.

Disadvantages of ID3:

- Only supports classification problems.
- Sensitive to noise, which may lead to overfitting.

C4.5

C4.5 uses a modified version of information gain called the **gain ratio** to reduce the bias towards features with many values.

It handles continuous attributes by first sorting the attribute values and then selecting the midpoint between adjacent values as a potential split point. The split that maximizes information gain or gain ratio is chosen.

While effective for both discrete and continuous attributes, C4.5 may still struggle with noisy data and large feature sets.

CART

CART (Classification and Regression Trees) is a widely used decision tree algorithm that is used for classification and regression tasks. The algorithm works as follows:

1. **Start with All Data**: CART begins with the entire dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. For classification, y_i is a categorical variable, while for regression, y_i is a continuous variable.
2. **Find the Best Split**: Iterate over all features, and for each feature, find the best threshold value that splits the data into two subsets. The best split is the one that results in the highest information gain or variance reduction (for classification and

regression, respectively).

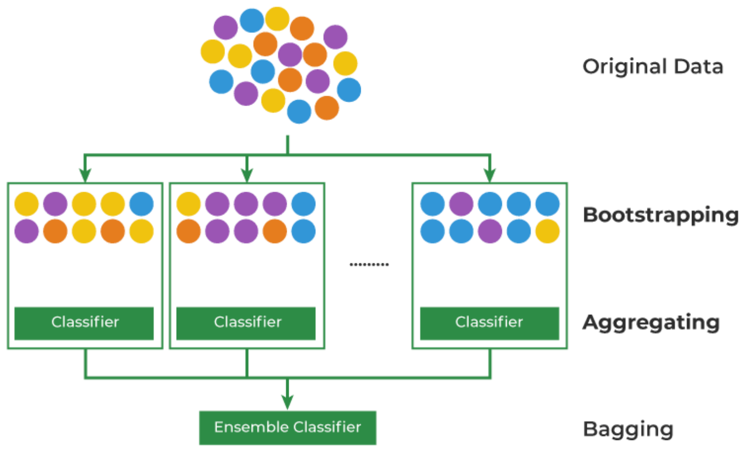
3. **Recursively Build the Tree**: The process of splitting is repeated for each subset, creating branches and sub-branches in the tree. This continues until the subsets are pure enough (i.e., they are homogeneous) or the tree reaches a certain size limit.
4. **Make Predictions**: Start at the top of the tree, and follow the branches based on given input data and the splitting criteria of internal nodes, until we reach a leaf node. The value or category in the leaf node is the prediction.

> **Gini Impurity**: Given a dataset $D = \{(x_1, c_1), (x_2, c_2), \ldots, (x_n, c_n)\}$, the Gini impurity is defined as:
>
> $$\text{Gini}(D) = 1 - \sum_{i=1}^{n} p_i^2$$
>
> where p_i is the proportion of instances of class c_i in the dataset D.

> **Variance**: Given a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, the variance of the dataset's target variable is defined as:
>
> $$\text{Variance}(D) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2$$
>
> where $\bar{y}$ is the mean of the target variable.

Bagging

Bagging (Bootstrap aggregating) is a type of ensemble learning in which multiple base models are trained independently and parallelly on different subsets of training data. Each subset is generated using bootstrap sampling in which data points are picked at randomly with replacement. In bagging classifier the final prediction is made by aggregating the predictions of all base model using majority voting.



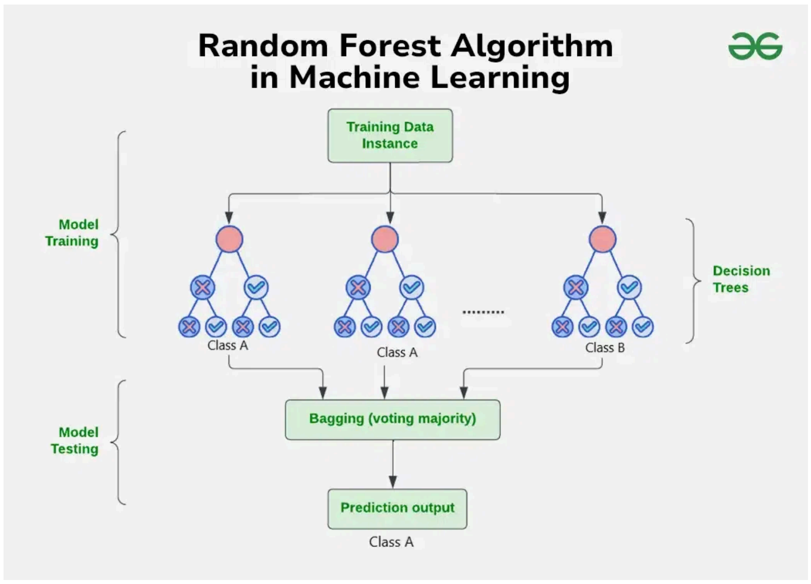
> Bootstrap: A statistical method for estimating the population parameter, such as the mean, by sampling the data **with replacement**.

Random Forest

A **Random Forest** is a collection of decision trees that work together to make predictions. It builds multiple decision trees, where each tree is trained on a random subset of the data and **a random subset of features**.

For each internal node, the tree randomly chooses a subset of features to iterates over, and for each feature, the tree chooses the best threshold value that splits the data into two subsets. Each tree has access to the full feature set, and the feature subsets can differ within a tree.

The final prediction is made by aggregating the predictions of all the trees.



Extra Trees

Extra Trees is an extension of Random Forest that uses a more randomized algorithm to build the trees.

Random Forest iterates over a subset of features, and for each feature, it chooses the best threshold value that splits the data into two subsets. However, Extra Trees randomly choose the threshold of feature value, so the generalization can be better.

Boosting

Boosting is an ensemble learning technique that sequentially combines multiple weak classifiers to create a strong classifier. Boosting models are trained sequentially with each model correcting the errors of its predecessor. The final prediction is made by adding the predictions of all the models.

Feature	**Boosting**	**Bagging**
Combination Type	Combines predictions of different weak models	Combines predictions of the same type of model
Goal	Reduces **bias**	Reduces **variance**
Model Dependency	New models depend on previous models' errors	All the models have the same weightage
Weighting	Models are weighted based on performance	All models have equal weight.

AdaBoost

AdaBoost (Adaptive Boosting) is a statistical classification meta-algorithm formulated by Yoav Freund and Robert Schapire in 1995, who won the 2003 Gödel Prize for their work.

AdaBoost assigns a weight to each training sample. In each iteration, data points with higher error are assigned higher weights, and the weak learner aims to minimize the weighted error in the next iteration.

The classical scenario of AdaBoost is **binary classification**. The algorithm works as follows:

> **Pseudocode of AdaBoost for Binary Classification**
>
> Given:
>
> - Samples:
>
> $$x_1, x_2, \dots, x_n \in \mathbb{R}^d$$
>
> - Target variable:
>
> $$y_1, y_2, \dots, y_n \in \{-1, 1\}$$

- Initial weights:

$$w_1^{(1)} = w_2^{(1)} = \cdots = w_n^{(1)} = 1/n$$

- Classifier: A function that maps input features to a real value, where the sign of the output indicates the class label.

$$f(x) : \mathbb{R}^d \to \mathbb{R}$$

- Error function:

$$E(y_i, f(x_i)) = \mathrm{e}^{-y_i f(x_i)}$$

- Weak learner: A function that maps input features to a binary output.

$$h(x) : \mathbb{R}^d \to \{-1, 1\}$$

For $t = 1, 2, \ldots, T$:

- Choose $h_t(x)$:
  - Find the weak learner h_t that minimizes the weighted sum error:

$$e_t = \sum_{i=1}^{n} w_i^{(t-1)} I(y_i \neq h_t(x_i))$$

  - Choose $\alpha_t = \frac{1}{2}\ln\left(\frac{1-e_t}{e_t}\right)$
  - Add to Ensemble model:

$$f_t(x) = f_{t-1}(x) + \alpha_t h_t(x)$$

  - Update weights:
    - $$w_i^{(t+1)} = w_i^{(t)} e^{-\alpha_t y_i h_t(x_i)}, \quad i = 1, 2, \ldots, n$$
    - Renormalize weights such that $\sum_{i=1}^{n} w_i^{(t+1)} = 1$

Diadvantage:

- Sensitive to anomalies, which may gain higher weights.

Gradient Boosting

Gradient Boosting is a type of ensemble learning technique. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.

Given a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}, x_i \in \mathbb{R}^d, y_i \in \mathbb{R}$. To solve the regression problem, we construct an ensemble model:

$$f(x) = \sum_{t=1}^{T} h_t(x)$$

where $h_t(x) : \mathbb{R}^d \to \mathbb{R}$ is the t-th weak learner.

Additionally, denote $f_t(x) = \sum_{t=1}^{T} h_t(x)$ as the t-th ensemble model. We have $\hat{y}_i^{(t)} = f_t(x_i)$.

In the t-th iteration, the algorithm trains a new weak learner h_t. The objective is to minimize the loss function:

$$\begin{aligned}
\sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t)}) &= \sum_{i=1}^{n} L\Big(y_i, \hat{y}_i^{(t-1)} + h_t(x_i)\Big) \\
&= \sum_{i=1}^{n} \left[L(y_i, \hat{y}_i^{(t-1)}) + h_t(x_i) \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} \right] \\
&= \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)}) + \sum_{i=1}^{n} h_t(x_i) \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} \\
&= \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)}) + \sum_{i=1}^{n} g_i h_t(x_i) \qquad (g_i \overset{\text{def}}{=} \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}})
\end{aligned}$$

The first term is fixed in the t-th iteration, so we can discard it. The second term comes from the first-order Taylor expansion of the loss function. Denote $\mathbf{g} = (g_1, g_2, \ldots, g_n)$, $\mathbf{h} = (h_t(x_1), h_t(x_2), \ldots, h_t(x_n))$. The objective becomes minimizing $\sum_{i=1}^{n} g_i h_t(x_i) = \mathbf{g} \cdot \mathbf{h}$.

The best direction of $\mathbf{h}$ is the negative gradient of the loss function. i.e., $\mathbf{h}^* \propto \mathbf{g}$. If we choose a learning rate η, we can update the ensemble model as follows:

$$f_t(x_i) = f_{t-1}(x_i) + h_t(x_i) = f_{t-1}(x_i) + \eta \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}$$

In particular, if the loss function is squared loss, the objective becomes:

$$\begin{aligned}
\sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t)}) &= \sum_{i=1}^{n} L\Big(y_i, \hat{y}_i^{(t-1)} + h_t(x_i)\Big) \\
&= \sum_{i=1}^{n} \Big(y_i - \hat{y}_i^{(t-1)} - h_t(x_i)\Big)^2
\end{aligned}$$

To minimize the objective, we can simply let $h_t(x_i) = y_i - \hat{y}_i^{(t-1)}$. So we are fitting the residual of the previous model.