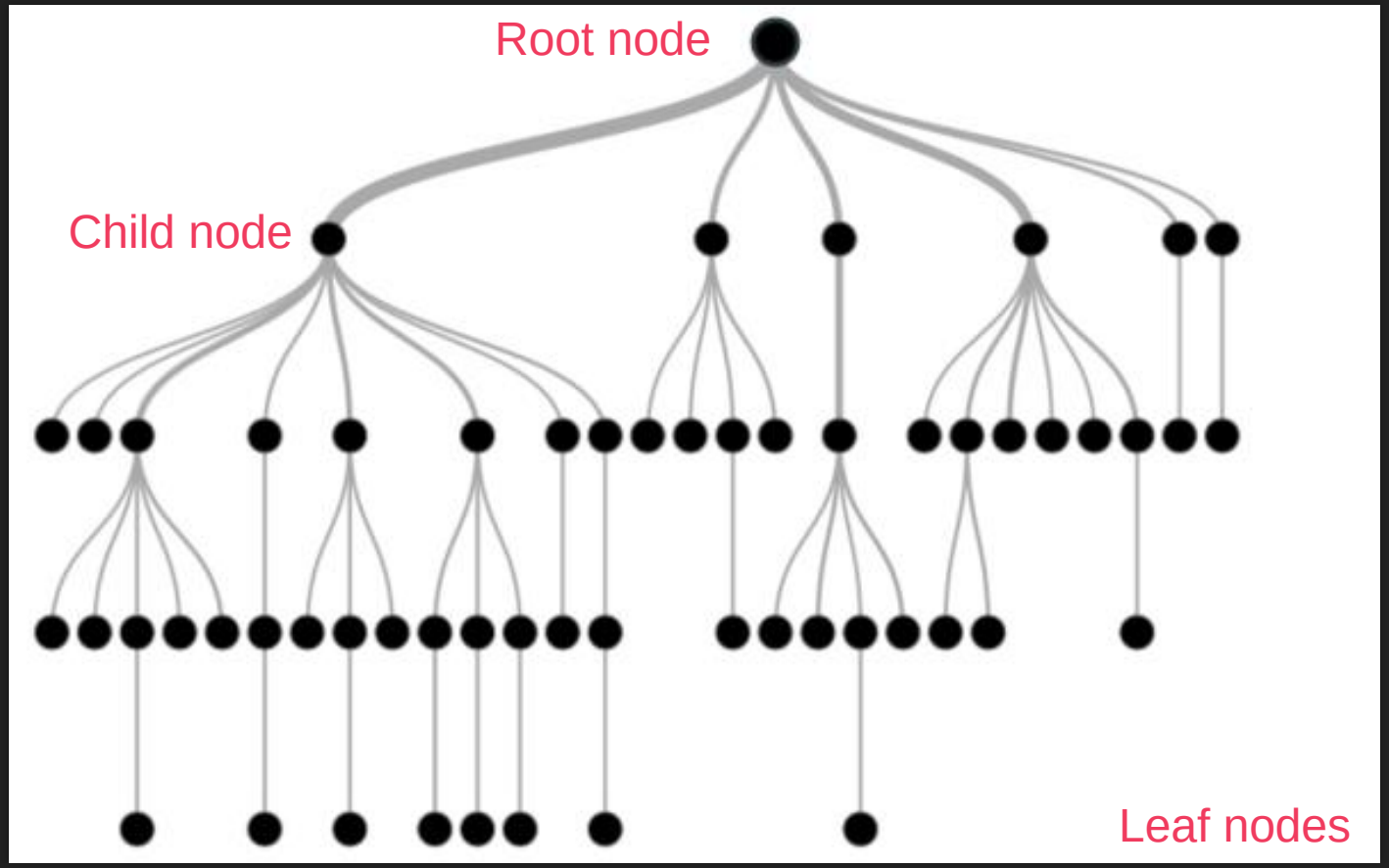
Trees in Machine Learning

VIP Machine Learning Course

Decision Tree

- A machine learning model used for both classification and regression tasks.
- In this method, the decision-making and learning processes on the data are performed in the form of a tree.
- This algorithm partitions the features of the given dataset using a cost function.
- One of the important parameters of this algorithm is the depth of the tree, which can be adjusted appropriately to prevent overfitting.
- During learning, the tree grows and includes irrelevant features. Therefore, a process called pruning is used to remove extra branches (inappropriate decisions made on the data) during optimization.

General Structure of a Decision Tree



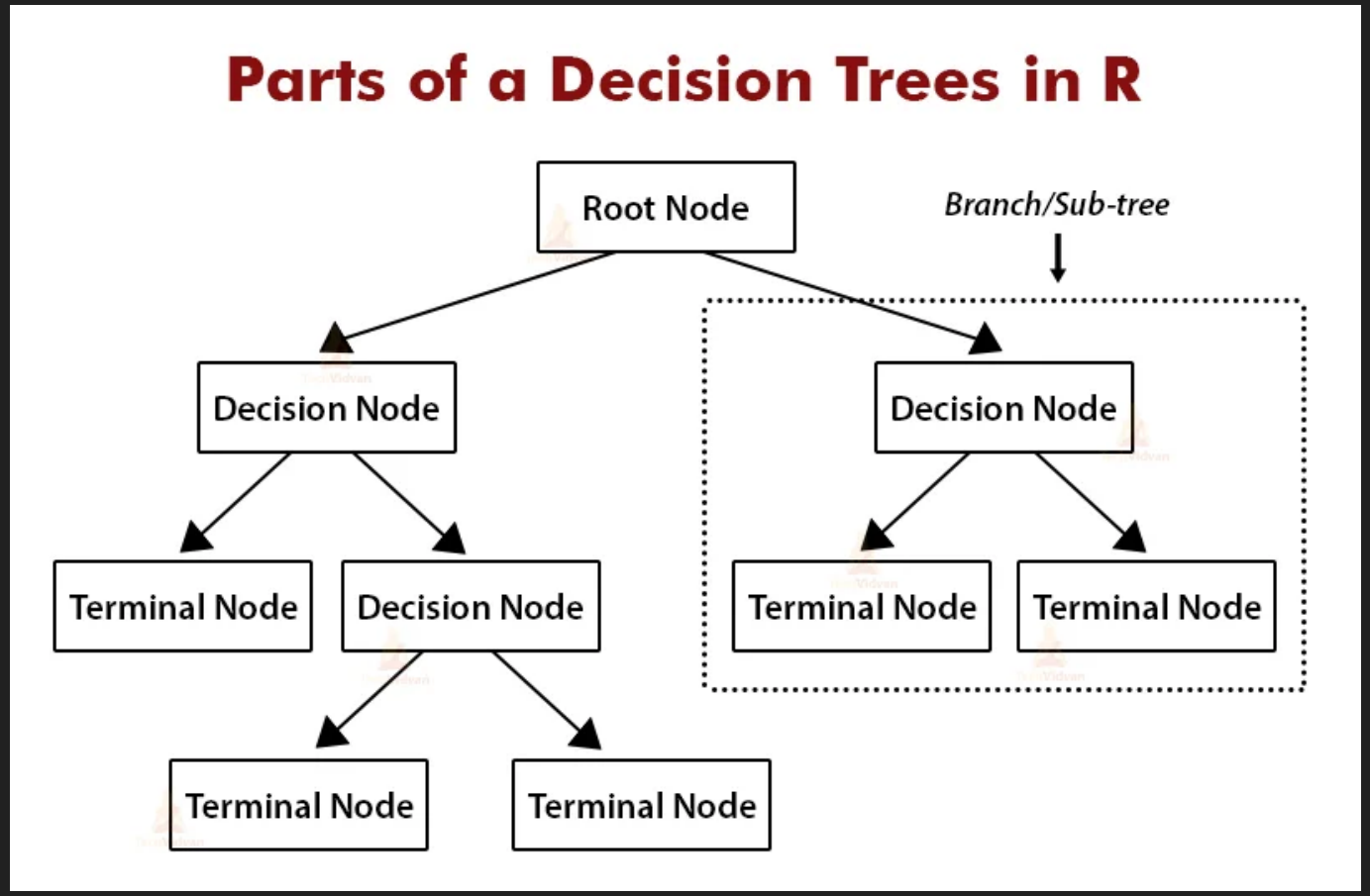
Decision Tree

- This algorithm has a suitable simplicity for displaying the decision-making process, making it usable.
- However, this simplicity can be reduced as the number of decisions and the depth of the tree increase.
- Moreover, the decision tree algorithm can easily suffer from the problem of overfitting.

Some terms in Decision Tree

- Root Node: The root node is the first and topmost node of the tree, representing the data of the problem and can be divided into two or more homogeneous sets.
- Splitting: The process of dividing nodes into two or more sub-nodes. Decision Node: A node that is split into multiple sub-nodes.
- Leaf/Terminal Node: The end nodes of each branch that cannot be split further (they have no children).
- Pruning: Sometimes parts of the tree need to be removed. Pruning is the process of removing certain decision nodes and is essentially the opposite of splitting.
- Branch/Sub-Tree: Refers to all sub-sections of the decision tree.
- Parent and Child: A node that is split into sub-nodes is called the parent node, and the resulting sub-nodes are called children.

Some terms in Decision Tree

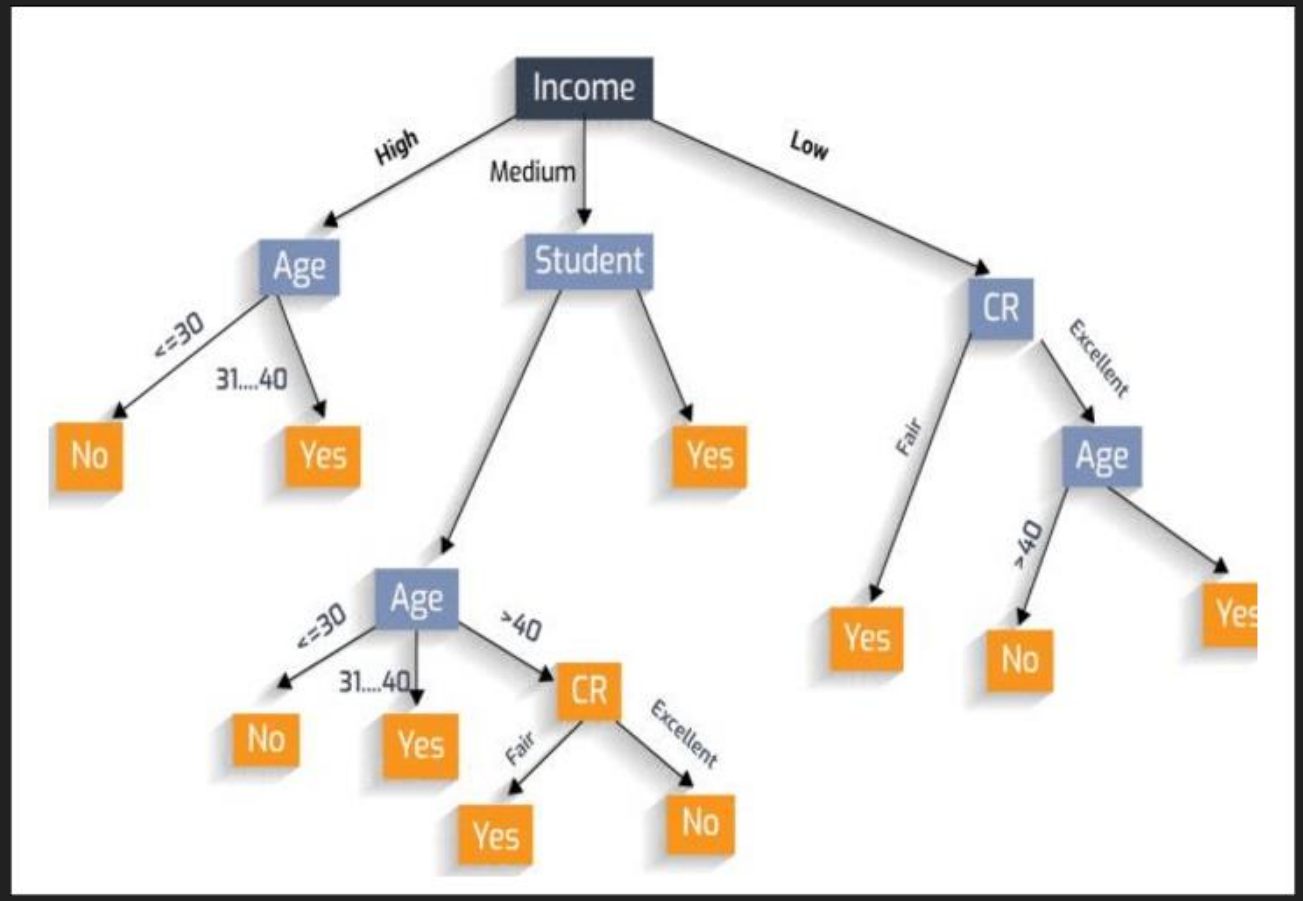


https://techvidvan.com/tutorials/decision-tree-in-r/

Steps in a Decision Tree

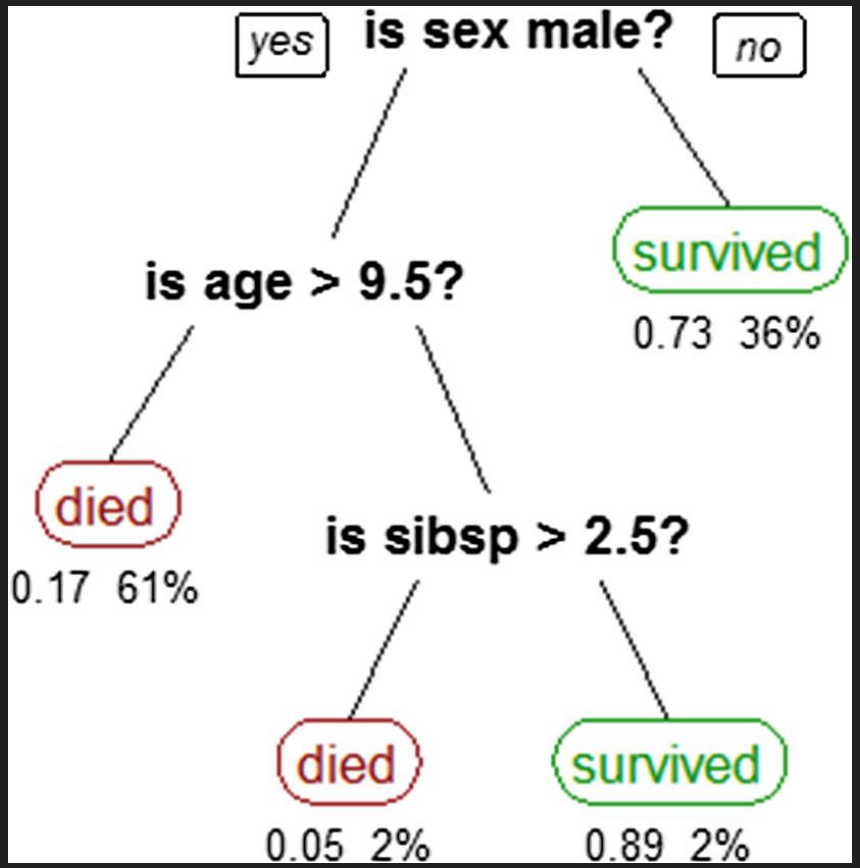
- Step 1: The decision tree algorithm starts from the root node, which includes the entire dataset for the problem.
- Step 2: Using the "Attribute Selection Measure" (ASM) method, the best feature in the dataset is selected.
- Step 3: Split the root node into subsets that contain possible and suitable values for the best features.
- Step 4: Generate decision tree nodes that include the best features.
- Step 5: Using the subsets created from the dataset in Step 3, new decision trees are recursively created. This process continues until no further classification of nodes is possible, and the final node is obtained as the leaf or terminal node.

A Simple Example of a Decision Tree



https://www.researchgate.net/figure/Example-on-Decision-Tree-43_fig2_350386944

The example of the Titanic ship in Decision Tree



https://www.researchgate.net/figure/A-decision-tree-predicting-the-survival-of-a-passenger-on-the-Titanic-using-the-sex-age_fig7_335664576

Feature selection evaluation in Decision Tree

- One of the important topics in decision trees is how to make decisions to select the best feature at the root node and subsequent nodes. To address this issue, there exists an algorithm known as Attribute Selection Measure (ASM) or feature selection evaluation.

- First method: Information Gain (sometimes referred to as the Entropy method in some sources).
- Second method: Gini Index.

The Entropy method in Decision Tree

- This method operates based on changes in entropy after partitioning a dataset according to its features.
- It calculates how much information a feature provides about a class.
- Based on the information values obtained, nodes are split and a decision tree is constructed.
- Decision trees always aim to maximize information gain, thus prioritizing features that provide more information.Entropy serves as a measure of impurity within a feature, indicating the randomness of data.
- The formula below represents the calculation of information gain:
- Read more about Information gain here.

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

The Gini Index method in Decision Tree

- The Gini Index is a measure of impurity that determines the impurity relative to the purity of features used in the CART (Classification and Regression Tree) algorithm, which is utilized for both classification and regression problems.
- The Gini Index is specifically applied in CART for classification tasks.
- For regression tasks in CART, the Sum Squared Error function is used instead of the Gini Index.
- A feature with a lower Gini Index is preferred over a feature with a higher Gini Index when constructing the decision tree.
- The formula for calculating the Gini Index is as follows:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

- In decision tree construction using CART, features with lower Gini Index values are prioritized as they provide greater purity in classifying instances.

The Gini Index method in Decision Trees

- In the Gini formula, if all data belongs to one class, it indicates that the data is pure. The Gini value always ranges between 0 and 1.
- A Gini index of 0 means all data points belong to one class, while a Gini index of 1 indicates the data points are evenly distributed among different classes.
- Additionally, pi in this formula represents the probability of a data point belonging to the class being examined.
- Read more about Gini Index here.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Recursive binary splitting in Decision Trees

- To facilitate branch splitting or applying splitting in decision trees, we rely on a mechanism that uses recursive binary splitting to organize these divisions in a way that prevents the tree from growing excessively in size.
- In this approach, at various split points, a cost function is evaluated, and the split is performed using the optimal cost value, typically the minimum cost value.
- In the example of the Titanic dataset, when the algorithm begins, it assesses a large number of features, calculating the cost function for each. Eventually, the "Sex" feature, which in this example has the lowest cost, is selected.
- This algorithm is essentially a recursive and greedy approach because it aims to choose the best predictive feature or classifier at the root node.

The cost of splitting in Decision Trees

- In regression problems, the following formula is used for calculation:

$$\text{Sum of Squares Error} \rightarrow \quad SSE = \sum (y - y')^2$$

- In classification problems, the following formula is used for calculation, where higher values for Gini Gain imply a better quality of that split or division.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Stopping criteria in Decision Trees

- To prevent the decision tree from growing excessively and to mitigate overfitting, it's important to know when to stop the splitting process.
- First method: By setting a minimum number of training instances required in each leaf node. For example, in the case of Titanic, we might specify that only nodes with at least 10 passengers can be further split to decide their survival status. Any leaf node with fewer than 10 passengers would not be expanded further.
- Second method: Setting a maximum depth limit for the tree. The maximum depth is the longest path from the root to a leaf node.

Pruning Decision Trees

- Pruning can enhance the performance of a decision tree by eliminating branches that have less importance relative to the problem's objective. This reduction in complexity improves the predictive power and accuracy of the algorithm while reducing overfitting. Pruning can start either from the root or from the leaves of the tree.
- Reduced Error Pruning: This is the simplest type of pruning, starting from the leaves and removing the node with the best class prediction at each leaf. This approach continues until further pruning would decrease the accuracy of the model.
- Cost Complexity Pruning: This method is more complex than Reduced Error Pruning and involves using a parameter called Alpha to determine how nodes are pruned based on the size of subtrees. It is also known as Weakest Link Pruning, focusing on removing the weakest links in the tree structure to optimize its complexity.

Decision Tree in Scikit-Learn

- In the scikit-learn library, you can use DecisionTreeClassifier for classification tasks. This algorithm is suitable for both binary classification (output labels include 0 and 1) and multi-class classification (e.g., labels 1, 2, 3, 4).
- For regression tasks, you can use DecisionTreeRegressor in the scikit-learn library.

The fundamental issue with Decision Trees

- In Decision Trees, accuracy typically increases with each split, and more splits can lead to higher accuracy. However, during machine learning training, it's possible for the model to easily become overfitted, and it's not always clear when overfitting occurs.
- It's crucial to use cross-validation to prevent this issue. Additionally, optimizing hyperparameters can help address overfitting.
- Due to variance issues, Decision Trees can be unstable, meaning small changes in data can result in entirely different trees.
- Increasing the number of features increases the complexity of the tree. In cases with a high number of features, it's advisable to reduce the problem's dimensions first. Principal Component Analysis (PCA) can be used for this purpose.

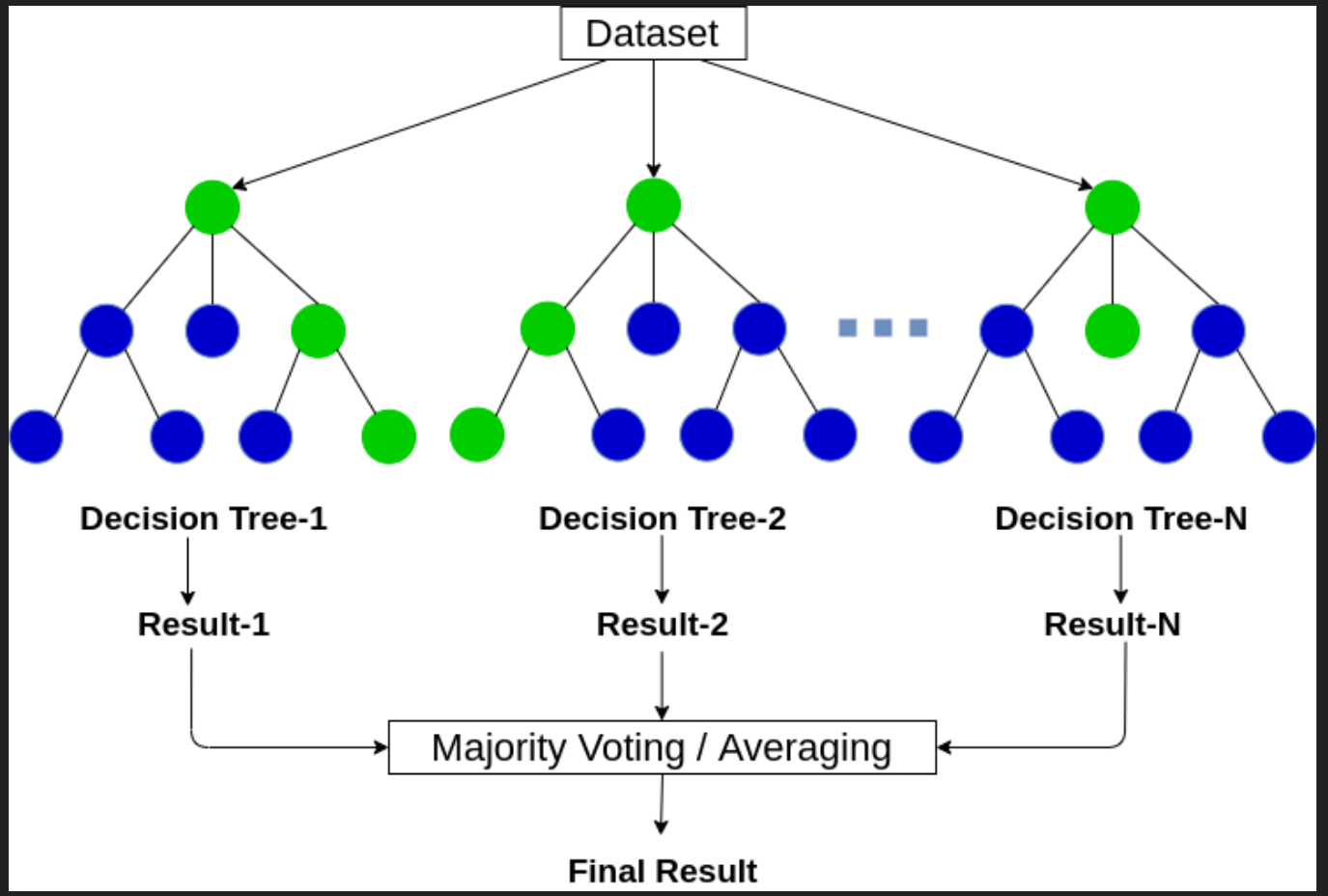
Random Forest

- A Random Forest is a collection of Decision Trees. It typically provides good results on data (even without optimal parameter tuning!).
- You can adjust the number of trees in the forest (n_estimators).
- You can also specify the maximum number of features used in building the Decision Trees.
- However, you cannot control how the algorithm randomly selects features or how data points are assigned to each tree.
- In this algorithm, increasing the number of trees usually increases accuracy, but after a while, the accuracy stabilizes.
- Unlike a single Decision Tree, the model created by a Random Forest does not suffer from high bias issues, and the model's variance is typically low.

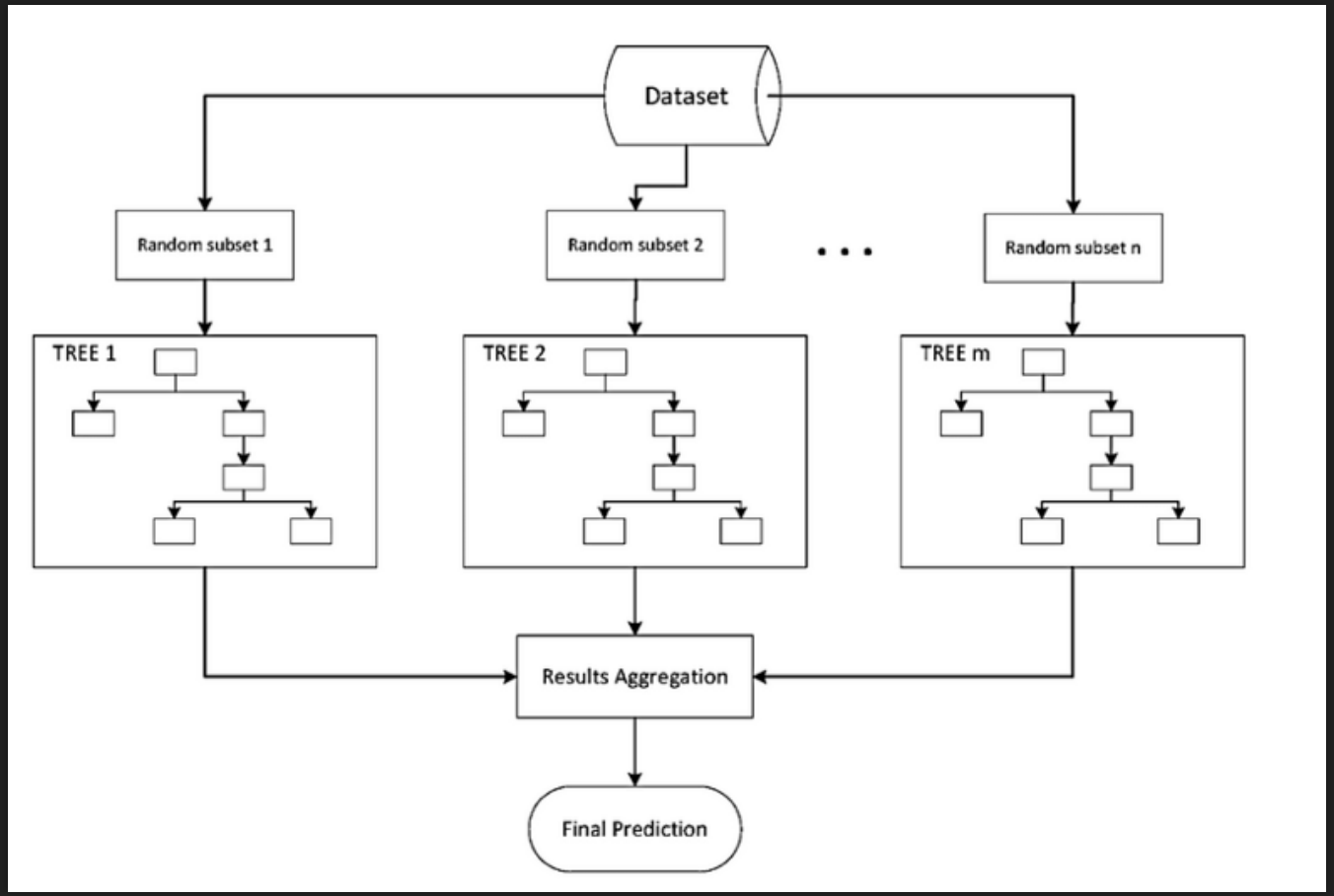
Bagging and Ensemble in Random Forest

- In machine learning, when we combine several machine learning models together and derive a new optimized model based on this combination, we say that an Ensemble operation has occurred.
- In the context of Random Forest, we are indeed witnessing Ensemble on Decision Trees. This algorithm attempts to create an optimized Random Forest by combining Decision Trees.
- The operation of BAGging (Bootstrap AGGregating) is a fundamental part of how Random Forest operates and is a type of Ensemble algorithm. In Random Forest, data is divided into several smaller and random subsets through bootstrapping. After this stage, a Decision Tree is applied to each subset, and ultimately, the outputs of all Decision Trees are aggregated together. The final output is obtained through a process known as aggregation, which typically involves majority voting or averaging at this stage. This process is clearly visible on the next page.

Random Forest



Bagging and Ensemble in Random Forest



When is Decision Tree used?

- For simple and interpretable models
- For models without parameters
- When we are not concerned about feature selection or regularization and issues like multicollinearity
- When overfitting is not a significant concern!

When is Random Forest used?

- When we seek higher accuracy (and are not concerned with the ease of model interpretation and scrutiny)
- When we aim to reduce variance (addressing the issue of overfitting). It should be noted that this model performs better than Decision Trees on new data that exhibit unexpected differences from the training data (the algorithm's accuracy on unexpected data is superior to that of Decision Trees).
- The main limitation of this algorithm is that it runs very slowly when dealing with very large datasets, numerous trees, or deep trees. This makes it nearly impractical to use this algorithm for solving certain real-world problems.

Additional resources

https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/
https://www.kaggle.com/code/prashant111/bagging-vs-boosting
https://www.upgrad.com/blog/random-forest-vs-decision-tree/#:~:text=A%20decision%20tree%20combines%20some,forest%20model%20needs%20rigorous%20training
https://mlu-explain.github.io/random-forest/
https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f
https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification/#:~:text=Bagging%20is%20an%20ensemble%20algorithm,used%20in%20each%20data%20sample.

The End

Thank you for your attention. I wish you pleasant times ahead.