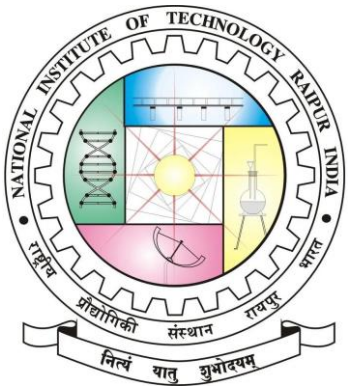


National Institute of Technology Raipur

Data Warehousing and Data Mining
Term Paper

Decision Trees and Random Forest Algorithm

By

Kunal Sachdeva

Roll No. 19115045

(6th Semester Computer Science & Engineering 2020-21)

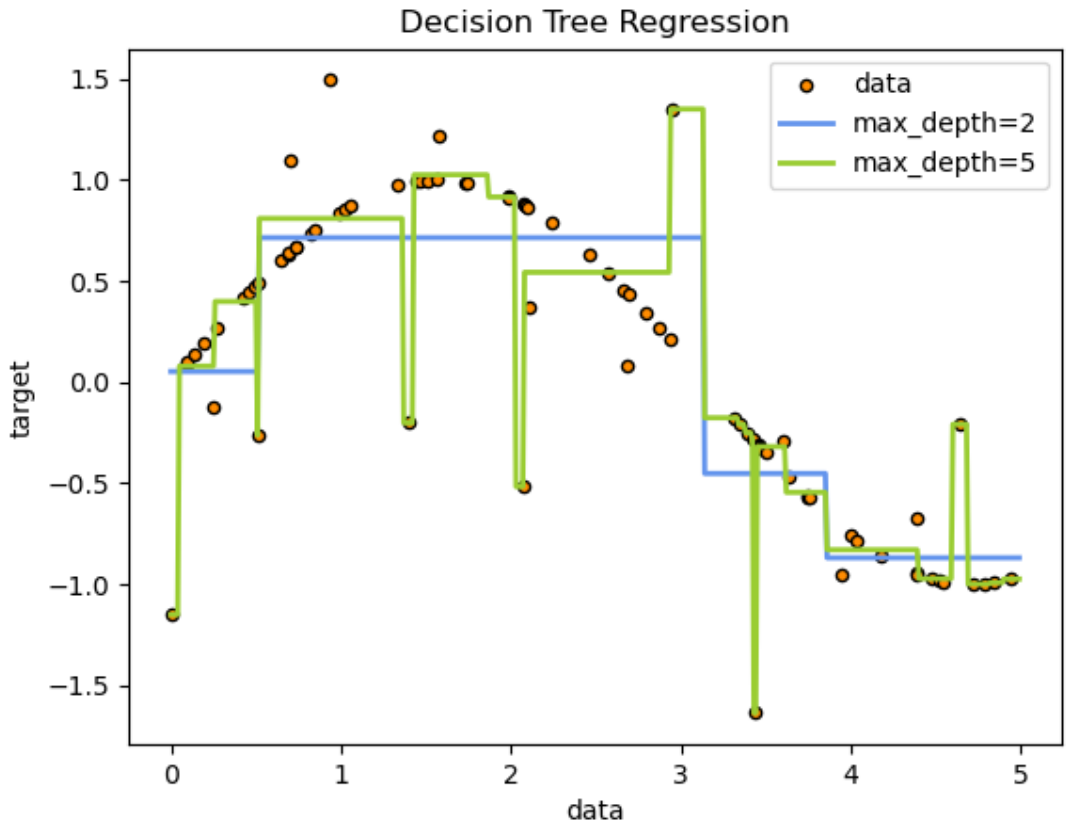
Abstract

For classification and regression, Decision Trees (DTs) are a non-parametric supervised learning method. The goal is to learn simple decision rules from data attributes to develop a model that predicts the value of a target variable. A tree is an approximation to a piecewise constant. Random Forest is a supervised machine learning algorithm that is commonly used to solve classification and regression problems. It creates decision trees from various samples, using the majority vote for classification and the average for regression.

Introduction

The most powerful and widely used tool for categorization and prediction is the decision tree. A decision tree is a flowchart-like tree structure in which each internal node represents an attribute test, each branch reflects the test's conclusion, and each leaf node (terminal node) stores a class label.

In the example below, decision trees use a series of if-then-else decision rules to estimate a sine curve using data. The decision criteria become more complex as the tree grows deeper, and the model becomes more accurate.



Important Terminology related to Tree based Algorithms:
Let's have a look at some of the basic terms associated with decision trees:
Root Node: The root node symbolizes the total population or sample, which is then separated into two or more homogeneous groups.
Splitting: The process of dividing a node into two or more sub-nodes is known as splitting.
Decision Node: A decision node is formed when a sub-node splits into more sub-nodes.
Leaf/Terminal Node: Nodes that do not split are referred to as Leaf/Terminal nodes.
Pruning: Pruning refers to the process of removing sub-nodes from a decision node. The splitting process can be described as the polar opposite of splitting.
Branch / Sub-Tree: A branch or sub-tree is a portion of a larger tree.
Parent and Child Node: A node that is divided into sub-nodes is considered the parent node of sub-nodes, while sub-nodes are the children of the parent node.

Regression Trees vs Classification Trees

- We all know that the decision tree's terminal nodes (or leaves) are at the bottom. This means that decision trees are usually drawn backwards, with the leaves at the bottom and the roots at the top.
- Let's have a look at the main differences and similarities between classification and regression trees:
- When the dependent variable is continuous, regression trees are utilized. When the dependent variable is categorical, classification trees are utilized.
- The value obtained by terminal nodes in the training data in the case of a regression tree is the mean response of observations falling in that region. As a result, if an unknown data observation falls within that range, we'll generate a prediction based on the mean value.
- In the case of a classification tree, the mode of observations falling in that region is the value (class) obtained by the terminal node in the training data. As a result, if an unknown data observation falls within that range, we'll create a prediction based on the mode value.
- The predictor space (independent variables) is divided into discrete and non-overlapping sections by both trees. These regions can be thought of as high-dimensional boxes or boxes for the sake of simplicity.
- Recursive binary splitting is a top-down greedy strategy used by both trees. It's called 'top-down' because it starts at the top of the tree, when all of the observations are in one place, and separates the predictor space into two new branches as it descends the tree. It's called greedy because the algorithm only worries about the present split (searches for the best

variable available) and ignores future splits that would result in a better tree.

- This process of splitting continues until a user-defined halting requirement is met. For example, once the number of observations per node falls below 50, we can tell the algorithm to halt.
- The splitting process produces fully developed trees in both circumstances until the halting criteria are met. The fully grown tree, on the other hand, is prone to overfit data, resulting in low accuracy on unknown data. This results in 'pruning.' Pruning is one of the methods for dealing with overfitting. In the following section, we'll learn more about it.

Advantages and Disadvantages of Decision Trees

Decision trees provide the following advantages:

- They are simple to learn and interpret. It is possible to visualize trees.
- It Doesn't necessitate a lot of data preparation. Other procedures frequently necessitate data standardization, the creation of dummy variables, and the removal of blank values. This module, however, does not handle missing values.
- The cost of using the tree (that is, predicting data) is proportional to the number of data points needed to train it.
- Able to work with both numerical and category information. For the time being, however, the scikit-learn implementation does not handle categorical variables. Other methods are often reserved for examining datasets with only one type of variable. For additional information, see algorithms.
- Capable of dealing with difficulties with many outputs.
- The model is based on a white box. If a circumstance can be observed in a model, Boolean logic can simply describe the situation. In contrast, the findings of a black box model (such as an artificial neural network) may be more difficult to decipher.
- Statistical tests can be used to validate a model. As a result, the model's dependability may be accounted for.
- Performs well even if the underlying model from which the data were created violates some of its assumptions.

The following are some of the disadvantages of decision trees:

- Decision-tree learners may produce overly complicated trees that may not adequately generalize data. This is referred to as overfitting. To avoid this

problem, mechanisms like as pruning, establishing the minimum number of samples required at a leaf node, and setting the maximum depth of the tree are required.

- Decision trees are inherently insecure, as slight changes in the data might result in an entirely new tree being created. The use of decision trees within an ensemble helps to solve this difficulty.
- Decision tree predictions are neither smooth or continuous, but piecewise constant approximations, as seen in the diagram above. As a result, extrapolation is difficult for them.
- Under numerous characteristics of optimality and even for simple notions, the issue of learning an optimal decision tree is known to be NP-complete. As a result, practical decision-tree learning algorithms rely on heuristic algorithms like the greedy algorithm, in which each node makes locally optimal judgments. Such algorithms can't promise that they'll give you the best decision tree in the world. This can be avoided by using an ensemble learner to train several trees, with the features and samples being randomly sampled and replaced.
- Some topics, like as XOR, parity, and multiplexer difficulties, are difficult to grasp because decision trees do not clearly describe them.
- If some classes dominate, decision tree learners build biased trees. As a result, prior to fitting using, it is recommended to balance the dataset.

Construction of Decision Tree

By separating the source set into subgroups based on an attribute value test, a tree can be "trained." Recursive partitioning is the process of repeating this method on each derived subset. When all of the subsets at a node have the same value of the target variable, or when splitting no longer adds value to the predictions, the recursion is complete. Because the building of a decision tree classifier does not necessitate domain expertise or parameter selection, it is well suited to exploratory knowledge discovery.

High-dimensional data can be handled via decision trees. The accuracy of the decision tree classifier is generally good. A common inductive strategy to learning classification information is decision tree induction.

Decision Tree Representation:

Instances are classified using decision trees by sorting them down the tree from the root to a leaf node, which provides the classification. As indicated in the above diagram, an instance is classified by starting at the root node of the tree, checking the attribute specified by this node, and then progressing along the tree branch according to the attribute value. The subtree rooted at the new node is then

processed in the same way.

The decision tree in the preceding graphic categorizes a given morning based on whether it is suitable for tennis play and returns the classification associated with that leaf. (In this situation, the answer is Yes or No).

The instance, for example, would be sorted down the decision tree's leftmost branch and so classed as a negative instance.

(Outlook = Rain, Temperature = Hot, Humidity = High, Wind = Strong)

To put it another way, a decision tree is a disjunction of conjunctions of constraints on attribute values of instances.

(Outlook = Sunny ^ Humidity = Normal) v (Outlook = Overcast) v (Outlook = Rain ^ Wind = Weak)

Pruning

Pruning a tree can improve its performance even further. It entails eliminating branches that make use of low-importance attributes. By lowering the complexity of the tree, we can improve its predictive power by reducing overfitting.

Pruning can begin either at the base or at the leaves. The most basic pruning approach starts with leaves and removes each node in that leaf with the most popular class; this change is preserved if it does not degrade accuracy. Reduced error pruning is another name for it. More advanced pruning approaches, such as cost complexity pruning, use a learning parameter (alpha) to determine whether nodes can be deleted based on the sub-size. tree's Weakest link pruning is another term for this.

Random Forest Algorithm

Random forests, also known as random choice forests, are an ensemble learning method for classification, regression, and other tasks that works by building a large number of decision trees during training. For classification tasks, the random forest's output is the class chosen by the majority of trees.

The mean or average prediction of the individual trees is returned for regression tasks. Random decision forests address the problem of decision trees overfitting their training set. Random forests outperform decision trees in most cases, but they are less accurate than gradient enhanced trees. Data features, on the other

hand, can have an impact on their performance.

One of the most essential characteristics of the Random Forest Algorithm is that it can handle data sets with both continuous and categorical variables, as in regression and classification. For classification difficulties, it produces superior results.

Assumptions for Random Forest

Because the random forest combines numerous trees to forecast the dataset's class, some decision trees may correctly predict the output while others may not. However, when all of the trees are combined, the proper result is predicted. As a result, two assumptions for a better Random Forest classifier are as follows:

- The dataset's feature variable should have some actual values so that the classifier can predict accurate outcomes rather than a guess.
- Each tree's predictions must have very low correlations.

The Random Forest algorithm has the following steps:

- Step 1: In Random Forest, n number of random records are selected from a data collection with k number of records.
- Step 2: For each sample, individual decision trees are built.
- Step 3: Each decision tree produces a result.
- Step 4: For classification and regression, the final output is based on Majority Voting or Averaging, respectively.

Important Random Forest Features

1. Diversity- When creating an individual tree, not all attributes/ variables/ features are evaluated, and each tree is unique.
2. Immune to the curse of dimensionality- The feature space is minimized because each tree does not consider all of the features.
3. Parallelization- Each tree is built from scratch using different data and properties. This means we can fully utilize the CPU to create random forests.
4. Train-Test split- In a random forest, we don't need to separate the data into train and test because the decision tree will always miss 30% of the data.
5. Stability- The result is stable because it is based on majority voting/ averaging

---

Difference Between Decision Tree & Random Forest

Although a random forest is a collection of decision trees, its behavior differs significantly.

Decision trees	Random Forest
When decision trees are allowed to grow without supervision, they are prone to overfitting.	Because random forests are built from subsets of data and the final output is based on average or majority rating, overfitting is avoided.
In terms of calculation, a single decision tree is faster.	It is, on the whole, slower.
When a decision tree receives a data set with features as input, it will create a set of rules for prediction.	Random forest selects data at random, forms a decision tree, and averages the results. It does not rely on any formulas.