















menu

› Course Shortcuts

› Student Lounge

› Q&A

▼ Create Decision Trees

-  Module Introduction: Create Decision Trees
-  Notations for Machine Learning
-  Build a Decision Tree
-  Explore Decision Trees
-  Measure a Partition's Purity
-  Formalize Impurity Functions
-  Determine the Minimum Entropy Split
-  CART Algorithm
-  **CART Algorithm**
-  CART Cheat Sheet
-  Minimize Impurity of Regression Trees
-  Squared-loss Impurity
-  Linear Time Regression Trees
-  Implement Regression Trees
-  Module Wrap-up: Create Decision Trees

› Choose the Right Model

› Data Science in the Wild

› Course Resources

CART Algorithm

As discussed in the video, the CART algorithm builds a classification tree by iterating through all possible binary partitions of the training data based on a single feature threshold. It then picks the one partition that leads to the best weighted impurity of the resulting children. This procedure is repeated on each new dataset resulting from each partition until a stopping criterion is met. Here, we formalize this stopping criterion:

Base Cases:

The CART algorithm stops in exactly two cases:

- If all data points in the dataset share the same label, we stop splitting and create a leaf with label y .
- If all data points in the dataset share the same features, we create a leaf with the *most common label* y for classification and *average label* for regression.

Recursion:

If none of the base cases are met, for a dataset of size n , we try all features d and all possible splits $(n - 1)$, and select the split that minimizes the weighted impurity of the two branches of the split.

$$\frac{|S_L|}{|S|} I(S_L) + \frac{|S_R|}{|S|} I(S_R)$$

Here, I is the impurity function (Gini impurity, entropy etc.), and splits S_L and S_R are defined based on the feature $f \in \{1, \dots, d\}$ and threshold t that divides the data points into the two branches.

$$S_L = \{(\mathbf{x}, y) \in S : [\mathbf{x}]_f \leq t\}, S_R = \{(\mathbf{x}, y) \in S : [\mathbf{x}]_f > t\}$$

CART Pseudocode

Formally we can state the CART algorithm as:

```
procedure CART-ID3( $S = \{\mathbf{x}_i, y_i\}_i$ )  
  if all labels are equal to  $\bar{y}$  then                                ▷ Base Case #1  
    return Leaf[ $\bar{y}$ ]  
  else if all data points have the same features  $\bar{\mathbf{x}}$  then          ▷ Base Case #2  
    return Leaf[mode( $\{y : (\mathbf{x}, y) \in S\}$ )]                        ▷ Use mean(·) for regression  
  else                                                            ▷ Recursion on branches  
    ▷ Iterate through all possible features  $f$  and thresholds  $t$ , and find best partition  
     $f, t \leftarrow \arg \min_{f, t} \frac{|S_L|}{|S|} I(S_L) + \frac{|S_R|}{|S|} I(S_R)$   
     $S_L \leftarrow \{(\mathbf{x}, y) \in S : [\mathbf{x}]_f \leq t\}$   
     $S_R \leftarrow \{(\mathbf{x}, y) \in S : [\mathbf{x}]_f > t\}$   
    return Node[CART-ID3( $S_L$ ), CART-ID3( $S_R$ )]  
  end if  
end procedure
```

☆ Key Points

The CART algorithm splits the data repeatedly to maximally improve the purity of its children.

If all possible splits have been exhausted or the node is already pure, a node will not be split further.

Because the same procedure is used on every node in the tree structure, the algorithm is recursive.