Decision Trees : Classification and Regression Trees (CART)

Ayush Singhal & Gaurav Shukla

National Institute of Science Education and Research (NISER)
Bhubaneswar

February 17, 2023

CART : Classification and Regression Trees

- A decision tree is a tree-like model that is used for making decisions. It consists of nodes that represent decision points, and branches that represent the outcomes of those decisions. The decision points are based on the values of the input variables, and the outcomes are the possible classifications or predictions.
- A decision tree is constructed by recursively partitioning the input data into subsets based on the values of the input variables. Each partition corresponds to a node in the tree, and the partitions are chosen so as to minimize the impurity of the resulting subsets.

Introduction
CART

- The **CART** (Classification and Regression Trees) algorithm is a decision tree-based algorithm that can be used for both classification and regression problems in machine learning. It works by recursively partitioning the training data into smaller subsets using binary splits.
- CART is a powerful and popular algorithm due to its ability to handle both categorical and continuous features, its interpretability, and its ability to capture non-linear relationships between features and the target variable.
- The Classification and Regression Trees (CART) algorithm always creates a binary tree, which means that each non-terminal node has two child nodes. This is in contrast with other tree-based methods, which may allow multiple child nodes

- The algorithm works by recursively partitioning the training data into smaller subsets using binary splits. The tree starts at the root node, which contains all the training data, and recursively splits the data into smaller subsets until a stopping criterion is met.
- At each node of the tree, the algorithm selects a feature and a threshold that best separates the training data into two groups, based on the values of that feature. This is done by choosing the feature and threshold that maximizes the information gain or the Gini impurity, which are measures of how well a split separates the data.
- The process continues recursively, with each node in the tree splitting the data into two smaller subsets, until a stopping criterion is met. The stopping criterion could be a maximum depth for the tree, a minimum number of instances in each leaf node, or other criteria.
- Once the tree is built, it can be used to make predictions by traversing the tree from the root node to a leaf node that corresponds to the input data. For regression problems, the prediction is the average of the target values in the leaf node. For classification problems, the prediction is the majority class in the leaf node.

- Calculate the Gini impurity for the entire dataset. This is the impurity of the root node.
- For each input variable, calculate the Gini impurity for all possible split points. The split point that results in the minimum Gini impurity is chosen.
- The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
- Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
- The resulting tree is the decision tree.

Gini Impurity

Introduction

- Gini impurity is a measure of the quality of a split in a decision tree algorithm. It measures the probability of misclassification of a random sample. The Gini impurity measures how often a randomly chosen element from the set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset.
- The Gini impurity of a node can be calculated as follows:

$$G = 1 - \sum_{i=1}^{c} p_i^2$$

where c is the number of classes, and p_i is the probability of a randomly chosen element in the node being labeled as class i.

Gini Impurity

Example

- To illustrate how the Gini impurity is calculated, consider the following example. Suppose we have a binary classification problem, where we want to predict whether a person will buy a particular product based on their age and income.

Age	Income	Buy Product?
30	20000	Yes
40	50000	Yes
20	30000	No
50	60000	No
60	80000	Yes

- We want to build a decision tree to predict whether a person will buy the product based on their age and income. Suppose we want to split the data based on age, with a threshold of 35. The left child node will contain the data where age is less than or equal to 35, and the right child node will contain the data where age is greater than 35.

Gini Impurity

Example

- The Gini impurity for the left child node can be calculated as follows:

$$G_{\text{left}} = 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = 0.5$$

  There are two data points in the left child node, one of which buys the product, and one of which does not.
- Therefore, the probability of a randomly chosen element being labeled as "Yes" is 0.5, and the probability of it being labeled as "No" is also 0.5.
- The Gini impurity for the right child node can be calculated as follows:

$$G_{\text{right}} = 1 - (\frac{2}{3})^2 - (\frac{1}{3})^2 \approx 0.444$$

  There are three data points in the right child node, two of which buy the product, and one of which does not.
- Therefore, the probability of a randomly chosen element being labeled as "Yes" is $2/3$, and the probability of it being labeled as "No" is $1/3$.

Gini Impurity

Example

- The overall Gini impurity for the split can be calculated as a weighted average of the Gini impurities for the child nodes, where the weights are proportional to the number of data points in each node.

$$G_{\text{split}} = \frac{2}{5} G_{\text{left}} + \frac{3}{5} G_{\text{right}} \approx 0.48$$

- The decision tree algorithm will try different thresholds and different features to find the split that minimizes the Gini impurity.
- Here is an example Python code that calculates the Gini impurity for a binary classification problem:

```
import numpy as np

def gini_impurity(labels):
    n = len(labels)
    classes = np.unique(labels)
    gini = 0.0
    for c in classes:
        p = np.sum(labels == c)
```

CART

Example

Let's consider an example of using the CART algorithm for a binary classification problem.

- Suppose we have a dataset of patients with information about their age, gender, blood pressure, and cholesterol level, and whether or not they have heart disease. We want to build a decision tree to predict whether a new patient will have heart disease based on their age, gender, blood pressure, and cholesterol level.
- To start, we calculate the Gini impurity of the entire dataset. Suppose there are 500 patients in the dataset, and 200 of them have heart disease, and 300 of them do not. The Gini impurity is:

$$G = 1 - (\frac{200}{500})^2 - (\frac{300}{500})^2 = 0.48$$

- Next, we consider each input variable and each possible split point. Suppose we choose age as the first split variable. We consider all possible split points and calculate the Gini impurity for each split. Suppose the split point that results in the minimum Gini impurity is 50 years.

CART

Example

- We split the data into two subsets: patients who are 50 years old or younger, and patients who are older than 50. We create two new nodes for these subsets and calculate the Gini impurity for each node.
- Suppose the first node contains 300 patients, of which 100 have heart disease and 200 do not. The Gini impurity of this node is:

$$G_1 = 1 - (\frac{100}{300})^2 - (\frac{200}{300})^2 = 0.44$$

- Suppose the second node contains 200 patients, of which 100 have heart disease and 100 do not. The Gini impurity of this node is:

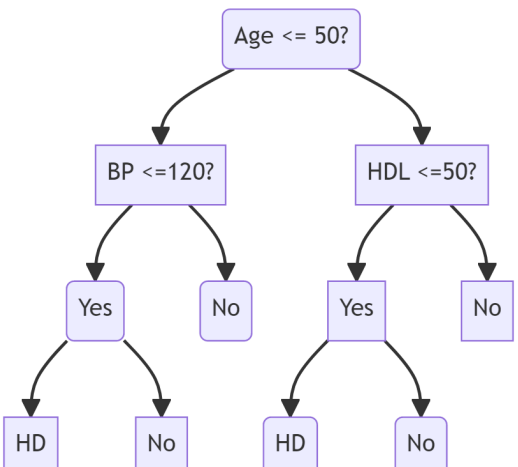
$$G_2 = 1 - (\frac{100}{200})^2 - (\frac{100}{200})^2 = 0.5$$

- We choose the split that results in the minimum Gini impurity, which is the split on age at 50.
- We create two new nodes for the subsets and continue the process until we meet a stopping criterion.

CART

Example

- Suppose we set the stopping criterion to be a maximum tree depth of 2. The resulting decision tree would look like this:



- This decision tree can be used to predict whether a new patient will have heart disease based on their age, blood pressure, and cholesterol level.

CART

Code Sample

- In this example, we're using the first two features of the iris dataset (sepal length and sepal width) for simplicity. We're building a decision tree classifier with a maximum depth of 2 and fitting it to the data. We're then making predictions for new data and printing the predictions.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
# Load the iris dataset
iris = load_iris()
# Use the first two features for simplicity
X = iris.data[:, :2]
y = iris.target
# Create a decision tree classifier with a maximum depth of 2
clf = DecisionTreeClassifier(max_depth=2)
# Fit the classifier to the data
clf.fit(X, y)
# Make predictions for new data
new_data = [[5.1, 3.5], [7.2, 3.6], [6.0, 2.2]]
predictions = clf.predict(new_data)
# Print the predictions
print(predictions)
```

CART

Benefits

- One of the significant benefits of the tree-based method, especially CART, is that the decision-making process closely resembles how humans make decisions. This makes it easy to understand and accept the results obtained from the tree-style decision-making process. This intuitive explanatory power is a crucial reason why the tree-based method is likely to remain relevant.
- Another advantage of CART is that it allows diverse types of input data, unlike many linear combination methods like logistic regression or support vector machines. CART can handle mixed input data, including numerical variables like price or area and categorical variables like house type or location. This flexibility makes CART a preferred tool of choice in a variety of applications. CART's ability to handle a range of input data types, coupled with its ease of interpretability, makes it a powerful algorithm for many machine learning tasks.

CART

Advantages and Disadvantages

Advantages

- It is a simple and intuitive algorithm that is easy to understand and interpret.
- It can handle both numerical and categorical data.
- It can handle missing values by imputing them with surrogate splits.
- It can handle multi-class classification problems by using an extension called the multi-class CART.

Disadvantages

- It tends to overfit the data, especially if the tree is allowed to grow too deep.
- It is a greedy algorithm that may not find the optimal tree.
- It may be biased towards predictors with many categories or high cardinality.
- It may produce unstable results if the data is sensitive to small changes or noise.

References

- Classification and Regression Trees by Leo Breiman, Jerome Friedman, Charles J. Stone and R.A. Olshen.
- COS 402 Lecture 2: Classification and Decision Trees , Sanjeev Arora and Elad Hazan (Fall 2016)
- SNU Lectures on ML Lecture 9: Classification and Regression Tree (CART) , Hyeong In Choi (Fall 2017)