# Chapter 18

# Classification and Regression Trees Tutorial

Decision trees are a flexible and very powerful machine learning method for regression and classification predictive modeling problems. In this chapter you will discover how to implement the CART machine learning algorithm from scratch step-by-step. After completing this chapter you will know:

- How to calculate the Gini index for a given split in a decision tree.
- How to evaluate different split points when constructing a decision tree.
- How to make predictions on new data with a learned decision tree.

Let's get started.

## 18.1 Tutorial Dataset

In this tutorial we will work through a simple binary (two-class) classification problem for CART. To keep things simple we will work with a two input variables ($X1$ and $X2$) and a single output variable ($Y$). This is not a real problem but a contrived problem to demonstrate how to implement the CART model and make predictions. The example was designed so that the algorithm will find at least two split points in order to best classify the training dataset. The raw data for this problem is as follows:

```
X1            X2            Y
2.771244718   1.784783929   0
1.728571309   1.169761413   0
3.678319846   2.81281357    0
3.961043357   2.61995032    0
2.999208922   2.209014212   0
7.497545867   3.162953546   1
9.00220326    3.339047188   1
7.444542326   0.476683375   1
10.12493903   3.234550982   1
6.642287351   3.319983761   1
```

Listing 18.1: CART Tutorial Data Set.

When visualized as a two-dimensional scatter plot the dataset looks as follows:

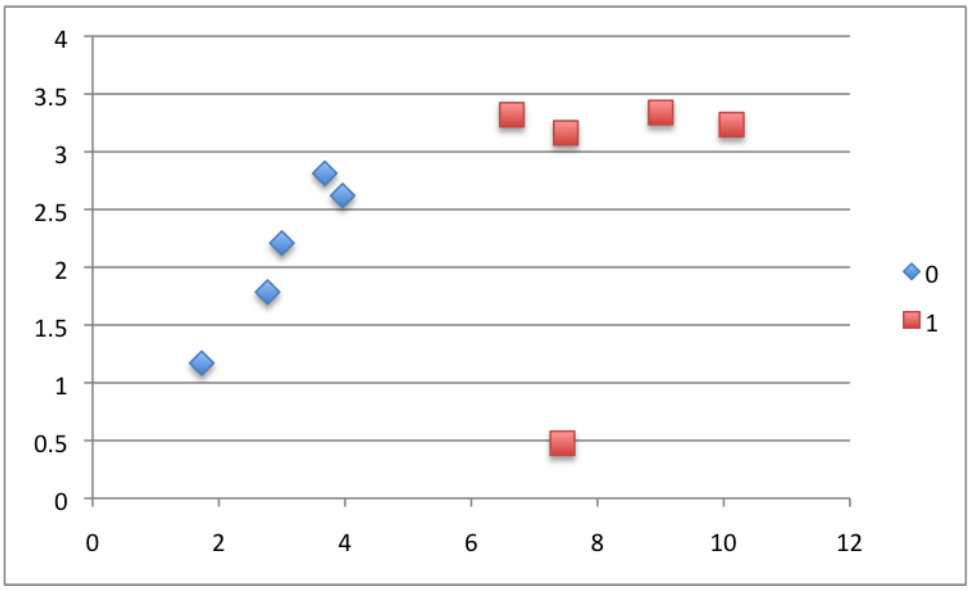


Figure 18.1: Classification And Regression Trees Tutorial Dataset.

## 18.2 Learning a CART Model

The CART model is learned by looking for split points in the data. A split point is a single value of a single attribute, e.g. the first value of the $X1$ attribute 2.771244718. Partitioning data at a split point involves separating all data at that node into two groups, left of the split point and right of the split point. If we are working on the first split point in the tree, then all of the dataset is affected. If we are working on say a split point one level deep, then only the data that has filtered down the tree from nodes above and is sitting at that node is affected by the split point.

We are not concerned with what the class value is of the chosen split point. We only care about the composition of the data assigned to the LEFT and to the RIGHT child nodes of the split point. A cost function is used to evaluate the mix of classes of training data assigned to each side of the split. In classification problems the Gini index cost function is used.

### 18.2.1 Gini Index Cost Function

We calculate the Gini index for a child node as follows:

$$G = 1 - ({p_1}^2 + {p_2}^2) \tag{18.1}$$

Where $p_1$ is the proportion of instances in the node with class 1 and $p_2$ is the proportion of instances in the node for class 2. We will always have two groups, a LEFT and RIGHT group because we are using a binary tree. And we know from our dataset that we only have two classes.

The proportion of data instances of a class is easy to calculate. If a LEFT group has 3 instances with class 0 and 4 instances with class 1, then the proportion of data instances with

class 0 would be $\frac{3}{7}$ or 0.428571429. To get a feeling for Gini index scores for child nodes, take a look at the table below. It provides 7 different scenarios for mixes of 0 and 1 classes in a single group.

| Class 0 | Class 1 | Count | Class 0/Count | Class 1/Count | Gini |
|---|---|---|---|---|---|
| 10 | 10 | 20 | 0.5 | 0.5 | 0.5 |
| 19 | 1 | 20 | 0.95 | 0.05 | 0.095 |
| 1 | 19 | 20 | 0.05 | 0.95 | 0.095 |
| 15 | 5 | 20 | 0.75 | 0.25 | 0.375 |
| 5 | 15 | 20 | 0.25 | 0.75 | 0.375 |
| 11 | 9 | 20 | 0.55 | 0.45 | 0.495 |
| 20 | 0 | 20 | 1 | 0 | 0 |

Listing 18.2: Sample Gini calculations.

It is easy to visualize these scenarios for one group, but the concepts translate to summing the proportions across the LEFT and RIGHT groups. You can see when the group has a 50-50 mix in the first row, that Gini is 0.5. This is the worst possible split. You can also see a case where the group only has data instances with class 0 on the last row and a Gini index of 0. This is an example of a perfect split.

The calculation of the Gini index for a chosen split point involves calculating the Gini index for each child node and weighting the scores by the number of instances in the parent node, as follows:

$$G = ((1 - (g1_1{}^2 + g1_2{}^2)) \times \frac{n_{g1}}{n}) + ((1 - (g2_1{}^2 + g2_2{}^2)) \times \frac{n_{g2}}{n}) \tag{18.2}$$

Where $G$ is the Gini index for the split point, $g1_1$ is the proportion of instances in group 1 for class 1, $g1_2$ for class 2, $g2_1$ for group 2 and class 1, $g2_2$ group 2 class 2, $n_{g1}$ and $n_{g2}$ are the total number of instances in group 1 and 2 and $n$ are the total number of instances we are trying to group from the parent node.

Our goal in selecting a split point is to evaluate the Gini index of all possible split points and greedily select the split point with the lowest cost. Let's make this more concrete by calculating the cost of selecting different data points as our split point.

### 18.2.2 First Candidate Split Point

The first step is to choose a split that will become the stump or root node of our decision tree. We will start with the first candidate split point which is the $X1$ attribute and the value of $X1$ in the first instance: $X1 = 2.7712$.

- **IF** $X1 < 2.7712$ **THEN** LEFT
- **IF** $X1 \geq 2.7712$ **THEN** RIGHT

Let's apply this rule to each $X1$ value in our training dataset. Below is the answer we get for each numbered instance in the dataset:

| X1 | Y | Group |
|---|---|---|
| 2.771244718 | 0 | RIGHT |
| 1.728571309 | 0 | LEFT |
| 3.678319846 | 0 | RIGHT |
| 3.961043357 | 0 | RIGHT |

```
2.999208922    0   RIGHT
7.497545867    1   RIGHT
9.00220326     1   RIGHT
7.444542326    1   RIGHT
10.12493903    1   RIGHT
6.642287351    1   RIGHT
```

Listing 18.3: Separation of Training Dataset by Candidate Split.

How good was this split? We can evaluate the mixture of the classes in each of the LEFT and RIGHT nodes as a single cost of choosing this split point for our root node. The LEFT group only has one member, where as the RIGHT group has 9 members. Starting with the LEFT group, we can calculate the proportion of training instances that have each class:

- $Y = 0$: $\frac{1}{1}$ or 1.0
- $Y = 1$: $\frac{0}{1}$ or 0.0

The RIGHT group is more interesting as it is comprised of a mix of classes (we are probably going to get a high Gini index).

- $Y = 0$: $\frac{4}{9}$ or 0.4444
- $Y = 1$: $\frac{5}{9}$ or 0.5555

We now have enough information to calculate the Gini index for this split:

$$Gini(X1 = 2.7712) = ((1 - (\frac{1}{1}^2 + \frac{0}{1}^2)) \times \frac{1}{10}) + ((1 - (\frac{4}{9}^2 + \frac{5}{9}^2)) \times \frac{9}{10}) \tag{18.3}$$

or

$$Gini(X1 = 2.7712) = 0.4444 \tag{18.4}$$

### 18.2.3 Best Candidate Split Point

We can evaluate each candidate split point using the process above with the values from $X1$ and $X2$. If we look at the graph of the data, we can see that we can probably draw a vertical line to separate the classes. This would translate to a split point for $X1$ with a value around 0.5. A close fit would be the value for $X1$ in the last instance: $X1 = 6.6422$.

- **IF** $X1 < 6.6422$ **THEN** LEFT
- **IF** $X1 \geq 6.6422$ **THEN** RIGHT

Let's apply this rule to all instances, below we get the assigned group for each numbered data instance:

```
X1            Y   Group
2.771244718   0   LEFT
1.728571309   0   LEFT
3.678319846   0   LEFT
3.961043357   0   LEFT
2.999208922   0   LEFT
7.497545867   1   RIGHT
9.00220326    1   RIGHT
7.444542326   1   RIGHT
10.12493903   1   RIGHT
6.642287351   1   RIGHT
```

Listing 18.4: Separation of Training Dataset by Bets Split.

There are 5 instances in each group, this looks like a good split. Starting with the LEFT group, we can calculate the proportion of training instances that have each class:

- $Y = 0$: $\frac{5}{5}$ or 1.0
- $Y = 1$: $\frac{0}{5}$ or 0.0

The RIGHT group has the opposite proportions.

- $Y = 0$: $\frac{0}{5}$ or 0.0
- $Y = 1$: $\frac{5}{5}$ or 1.0

We now have enough information to calculate the Gini index for this split:

$$Gini(X1 = 6.6422) = ((1 - (\frac{5}{5}^2 + \frac{0}{5}^2)) \times \frac{5}{10}) + ((1 - (\frac{0}{5}^2 + \frac{5}{5}^2)) \times \frac{5}{10}) \tag{18.5}$$

or

$$Gini(X1 = 6.6422) = 0.0 \tag{18.6}$$

This is a split that results in a pure Gini index because the classes are perfectly separated. The LEFT child node will classify instances as class 0 and the right as class 1. We can stop there. You can see how this process could be repeated for each child node to build up a more complicated tree for a more challenging dataset.

## 18.3 Making Predictions on Data

We can now use this decision tree to make some predictions for all of the training instances. But we have already done that when we calculated the Gini index above. Instead, let's classify some new data generated for each class using the same distribution. Here is the test dataset:

```
X1            X2            Y
2.343875381   2.051757824   0
3.536904049   3.032932531   0
2.801395588   2.786327755   0
3.656342926   2.581460765   0
2.853194386   1.052331062   0
```

```
8.907647835    3.730540859    1
9.752464513    3.740754624    1
8.016361622    3.013408249    1
6.58490395     2.436333477    1
7.142525173    3.650120799    1
```

Listing 18.5: Test Dataset.

Using the decision tree with a single split at, $X1 = 6.6422$ we can classify the test instances as follows:

```
Y   Prediction
0   0
0   0
0   0
0   0
0   0
1   1
1   1
1   1
1   0
1   1
```

Listing 18.6: Predictions for Test Dataset.

The result is a near perfect classification accuracy of 90% accurate.

## 18.4 Summary

In this chapter you discovered exactly how to construct a CART model and use it to make predictions. You learned how to:

- Calculate the Gini index for a candidate split in a decision tree.
- Evaluate any candidate split points using Gini index.
- How to create a decision tree from scratch to make predictions.
- How to make predictions on new data using a learned decision tree.

You now know how to implement CART from scratch. In the next chapter in you will discover the Naive Bayes machine learning algorithm for classification.