Decision Trees, Random Forest

Created and Edited by Amanda Magzal

06

Decision Trees

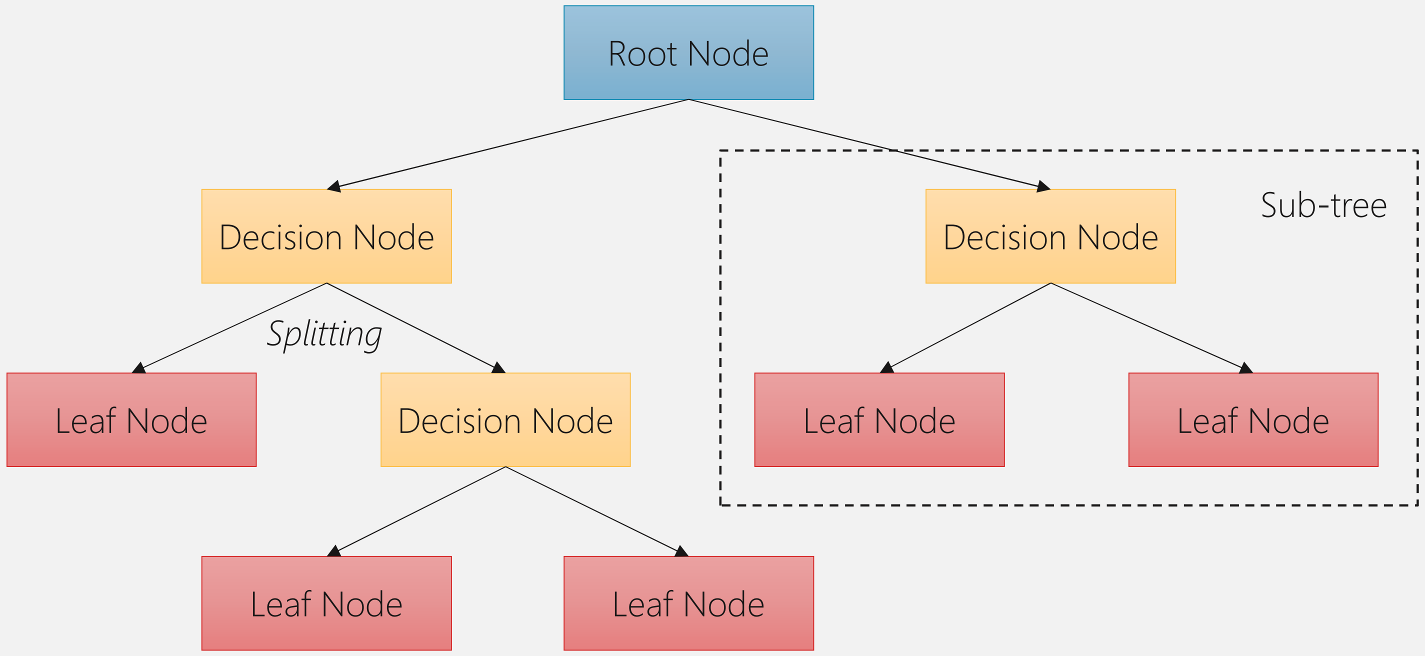
Easy to work with.

Work with all types of data.

Prone to overfit.

Terminology

- Root node: the first decision node of the tree that includes the entire dataset.
- Leaf node: a terminal node that includes the final output of the tree.
- Splitting: the process of dividing a decision node into sub-nodes according to a given condition.
- Sub-tree/Branch: a tree formed by splitting a node.



Classification Tree

1. Calculate the impurity for every feature.

2. Select the feature that has the lowest impurity score and split accordingly.

It is common to use the gini impurity:

$$\text{gini} = 1 - \sum_{k=1}^{K} (p_k)^2$$

Where $\boldsymbol{p}$ is the proportion of observations of category $\boldsymbol{k}$.

- Small impurity scores reflect nodes in which most observations belong to the same class.

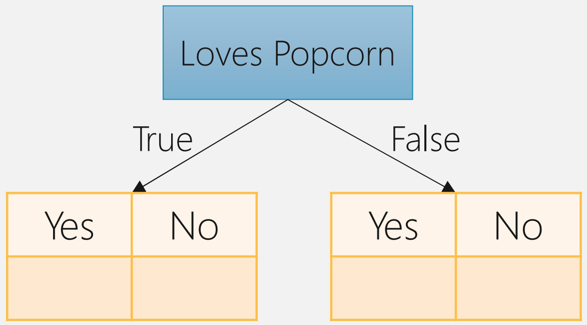
Building a Classification Tree

Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

Given a simple dataset with 3 features (loves popcorn, loves soda and age), we want to build a classification tree to predict whether a person loves movies.

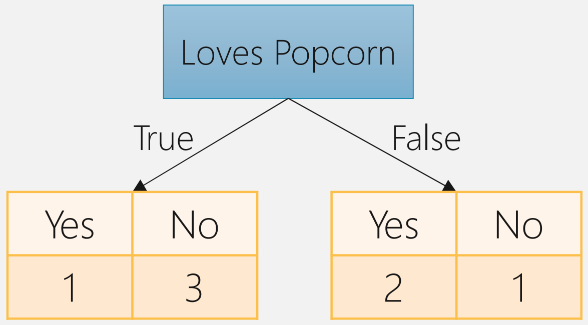
Select the root node

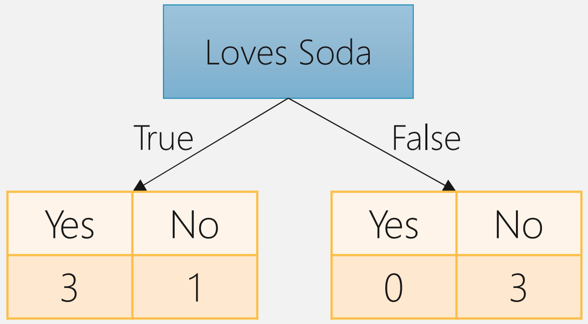
Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



Check how well each feature predicts

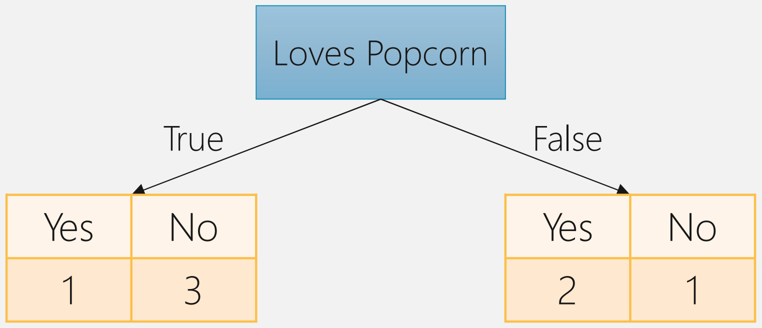
Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No





Calculate the gini impurity

Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

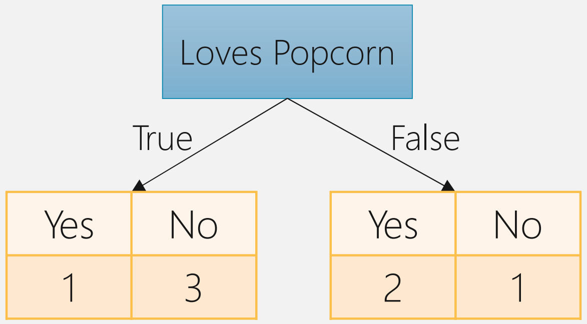


$$\text{gini} = 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2 = 0.375 \quad \text{gini} = 1 - \left(\frac{2}{2+1}\right)^2 - \left(\frac{1}{2+1}\right)^2 = 0.444$$

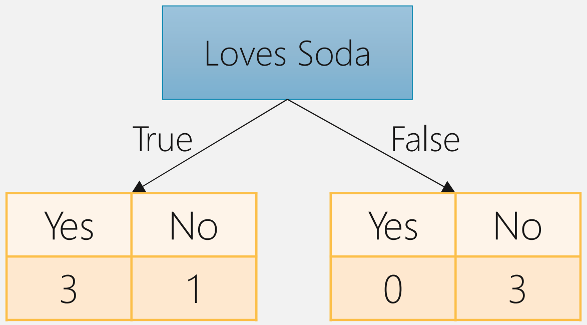
$$\text{Total gini} = \left(\frac{4}{4+3}\right) \cdot 0.375 + \left(\frac{3}{4+3}\right) \cdot 0.444 = 0.405$$

Calculate the gini impurity

Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



gini = 0.405



gini = 0.214

Dealing with a numeric feature

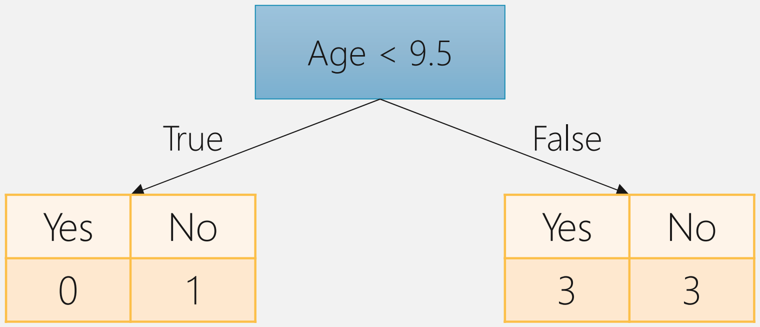
Age	Loves Movies
7	No
12	No
18	Yes
35	Yes
38	Yes
50	No
83	No

9.5	gini = 0.429
15	gini = 0.343
26.5	gini = 0.476
36.5	gini = 0.476
44	gini = 0.343
66.5	gini = 0.429

1. Sort the observations by age.

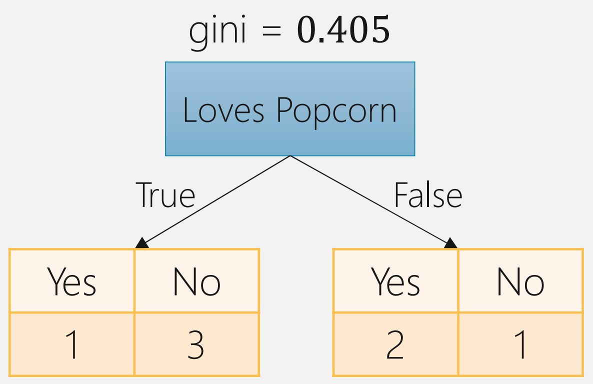
2. Calculate the average age for all adjacent people.

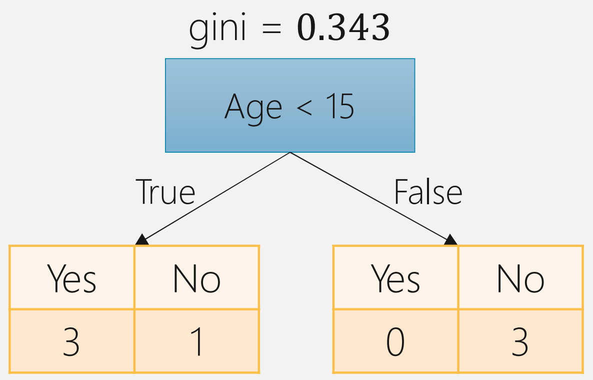
3. Calculate the gini impurity value for each average age.

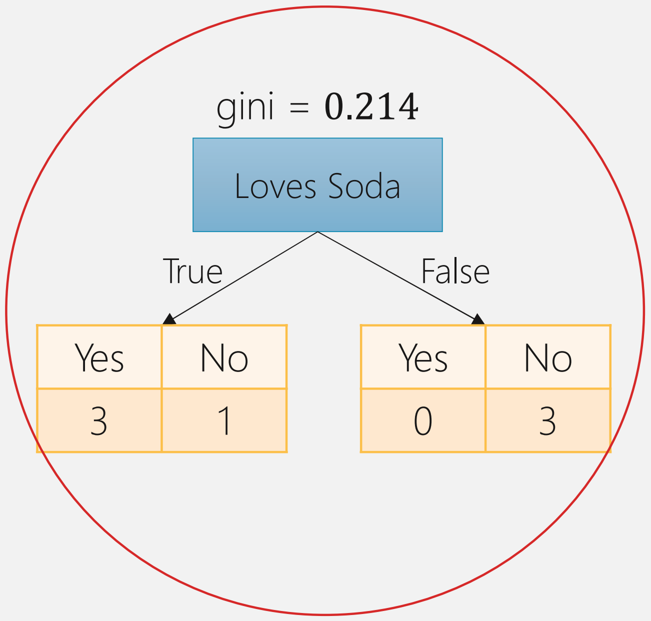


Select the root node

Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

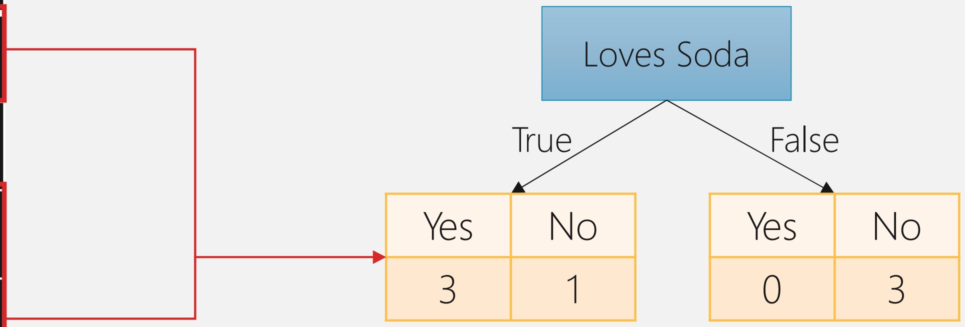






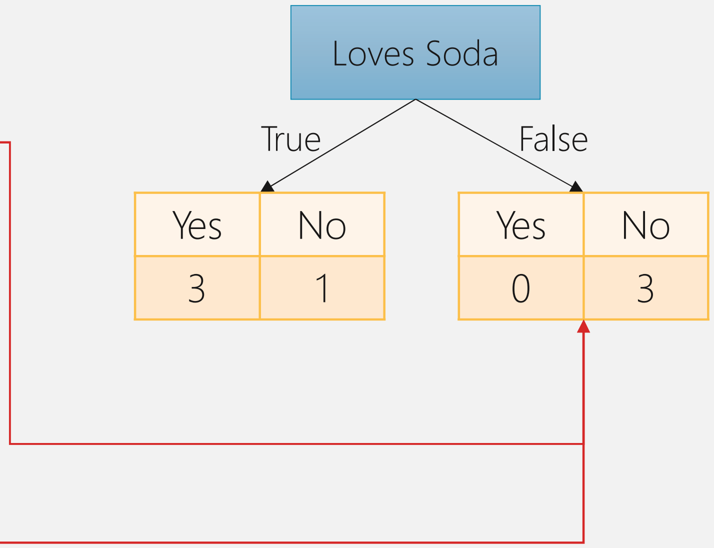
Split the dataset

Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



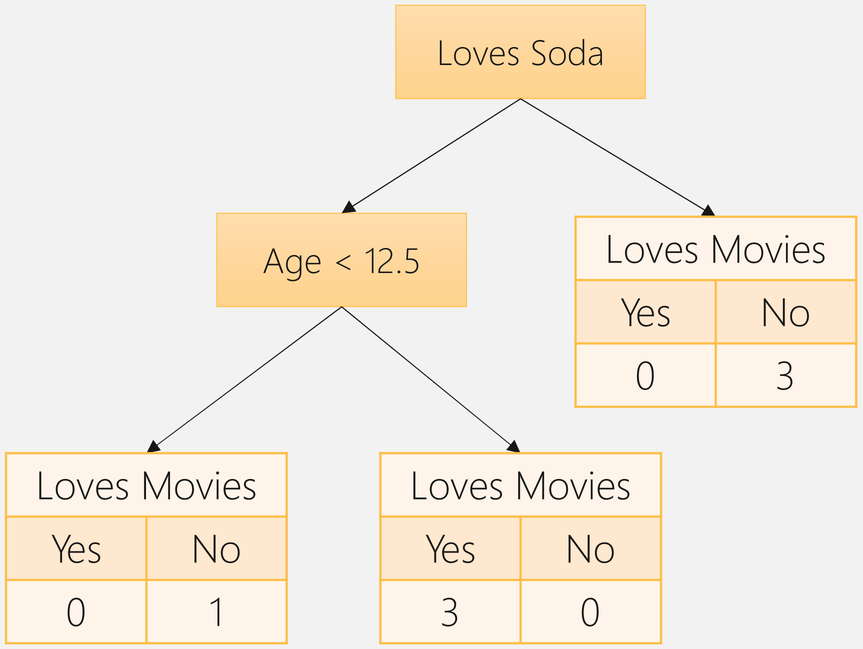
Split the dataset

Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



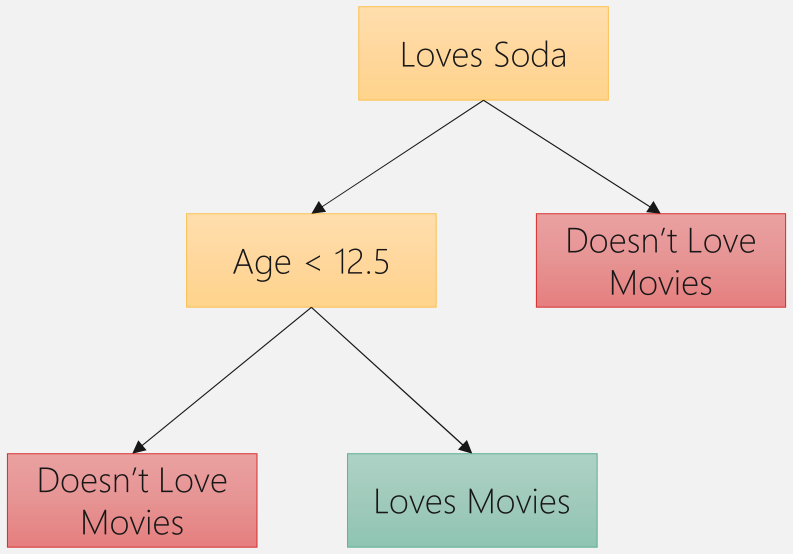
Continue building the tree

Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



The final decision tree

Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



Regression Tree

1. Calculate the SSR for every feature.

2. Select the feature that has the lowest SSR and split accordingly.

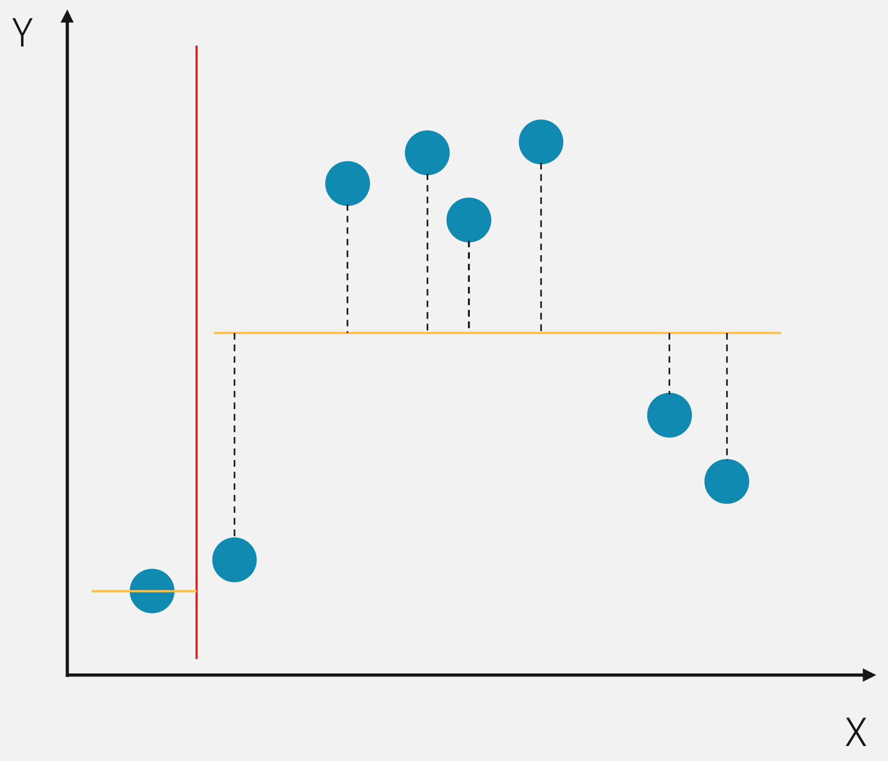
The SSR is calculated as:

$$\text{SSR} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Where $\boldsymbol{y_i}$ is the true value and $\hat{\boldsymbol{y}}_i$ is the predicted value (the average of the relevant observations).

- The final prediction is typically the average of the leaf's observations.

Building a Regression Tree

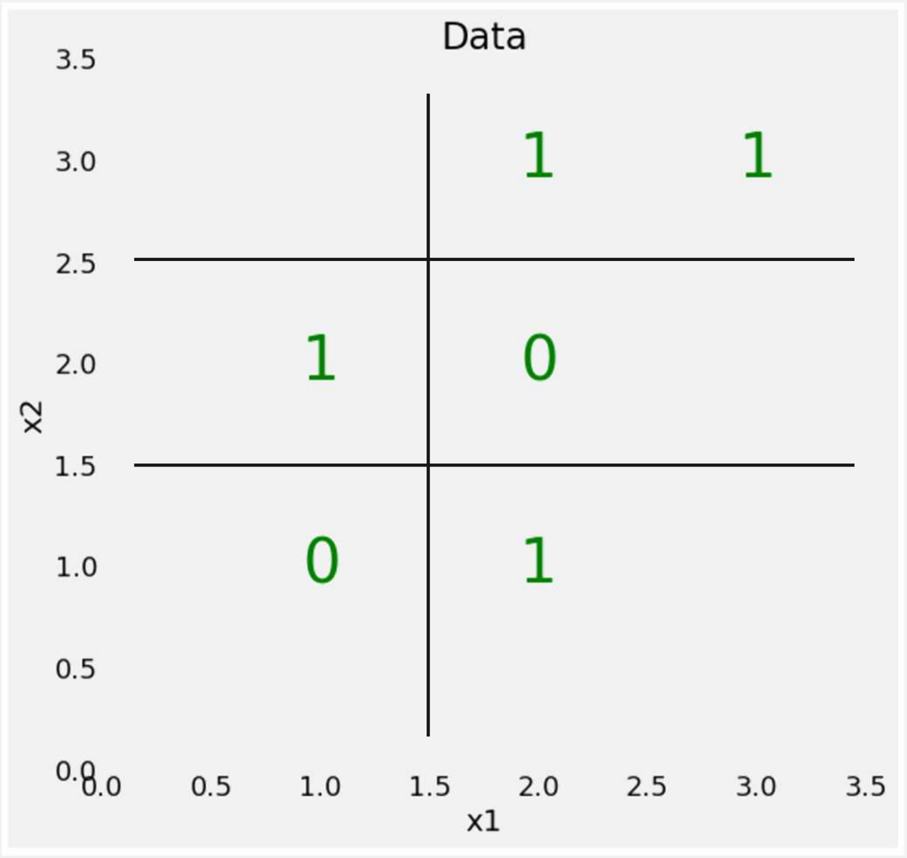


1. Sort the observations by feature X.

2. Calculate the average for all adjacent values of X.

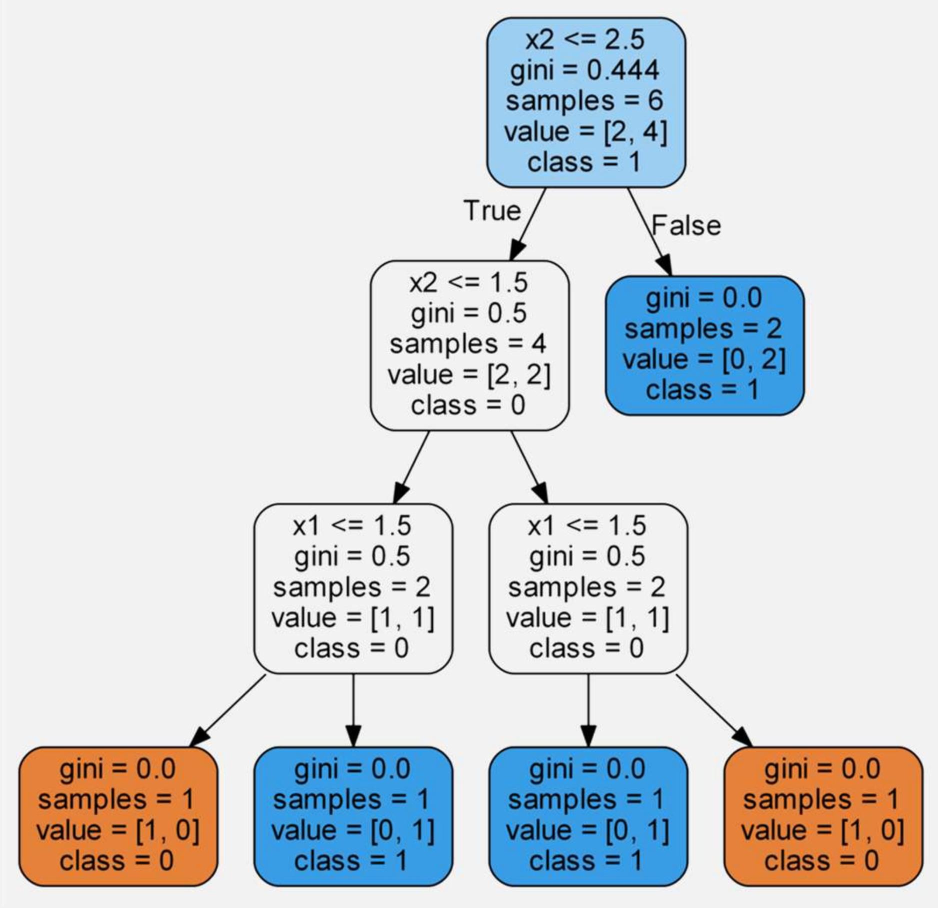
3. Calculate the SSR value for each split.

4. Select the split that yields the lowest SSR.



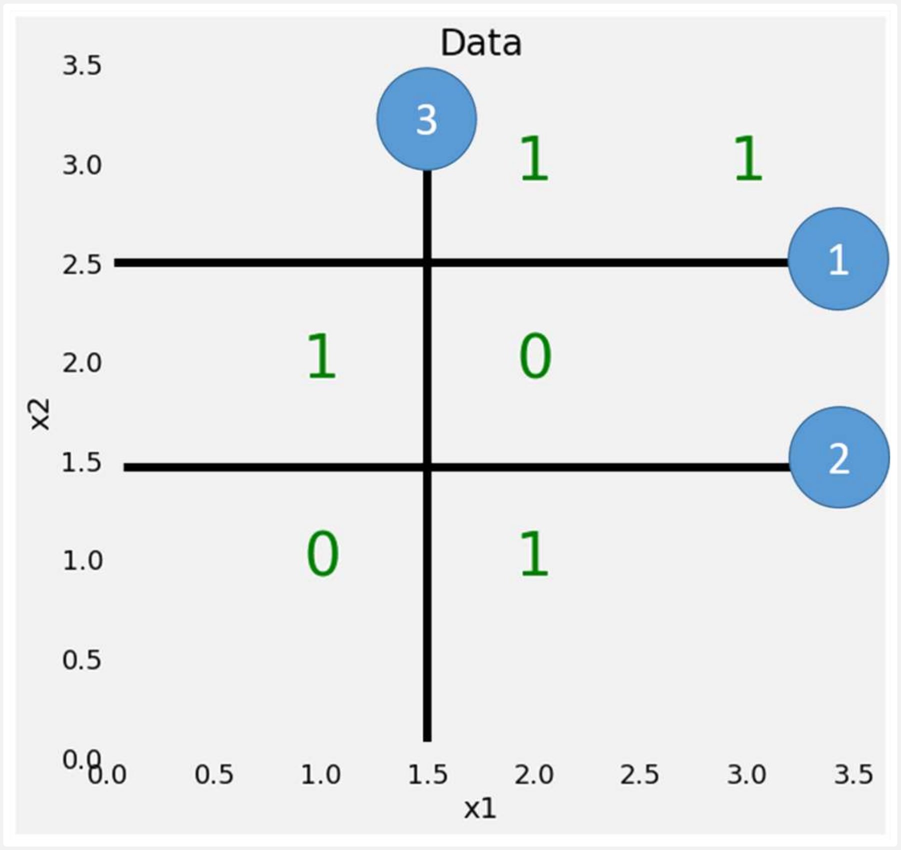
Example

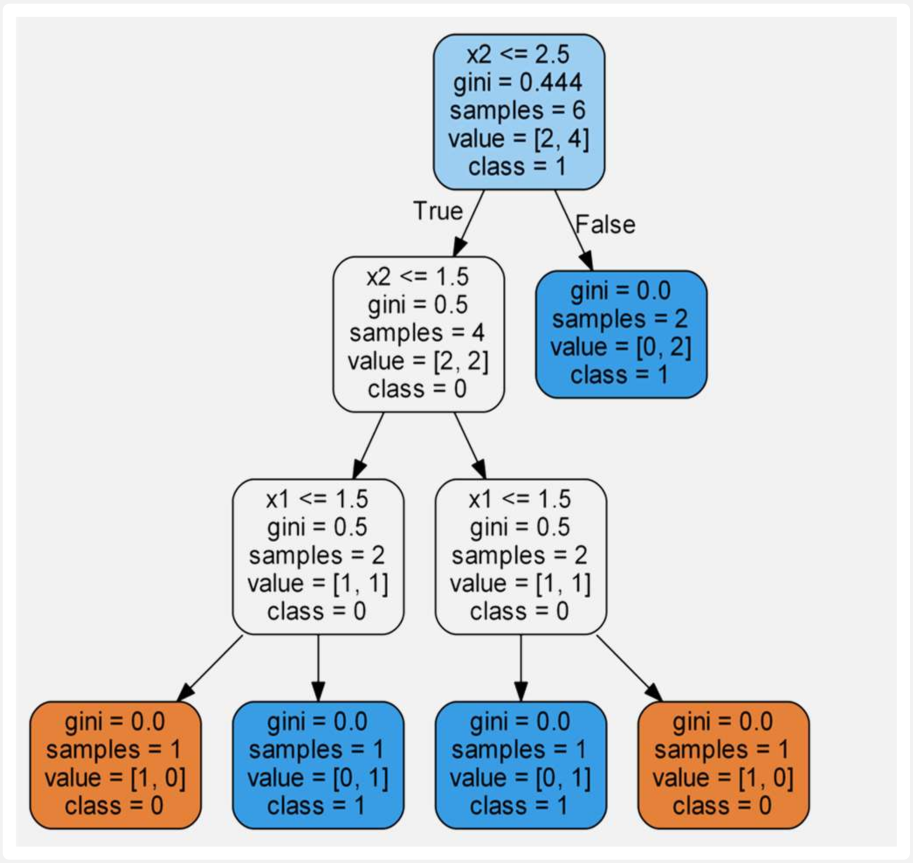
- Our dataset includes two features (x1, x2) and 6 observations. Every observation belongs to a class (0 or 1).
- Our goal is to build a decision tree to be able to predict to which class an observation belongs, given its features.
- We'll do so by drawing straight lines to divide the observations into boxes which we'll call nodes.



Nodes Information

1. Statement: a condition that splits the node's data into two or more separate sub-trees.
2. gini: the gini impurity of the node.
3. samples: the number of observations in the node.
4. value: the number of observations for each class.
5. class: the majority class. In case of a leaf node, that would be the model's prediction.


Data
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
x1
x2
3
1
2
1
1
1
0
0
1


x2 <= 2.5
gini = 0.444
samples = 6
value = [2, 4]
class = 1
True
False
x2 <= 1.5
gini = 0.5
samples = 4
value = [2, 2]
class = 0
gini = 0.0
samples = 2
value = [0, 2]
class = 1
x1 <= 1.5
gini = 0.5
samples = 2
value = [1, 1]
class = 0
x1 <= 1.5
gini = 0.5
samples = 2
value = [1, 1]
class = 0
gini = 0.0
samples = 1
value = [1, 0]
class = 0
gini = 0.0
samples = 1
value = [0, 1]
class = 1
gini = 0.0
samples = 1
value = [0, 1]
class = 1
gini = 0.0
samples = 1
value = [1, 0]
class = 0

Pruning

Removing some of the leaves and replacing the split with a leaf that includes more observations.

Decision trees have the tendency to overfit the data. One method to prevent that is pruning.

For regression trees:

1. Calculate the SSR of each tree by summing the SSRs of each leaf of the tree.

2. Add a tree complexity penalty (αT) to the SSR score, where α is a tuning parameter and T is the number of leaves.

3. Select the tree with the lowers overall tree score.

- The parameter α is selected using cross validation.

Decision Tree Hyper-Parameters

`criterion (classification):` *`{“gini”, “entropy”}, default=”gini”`*
The function to measure the quality of a split.

`criterion (regression):` *`{“squared_error”, “friedman_mse”, “absolute_error”, “poisson”}, default=”squared_error”`*
The function to measure the quality of a split.

`max_depth:` *`int, default=None`*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than `min_samples_split` samples.

`min_samples_split:` *`int or float, default=2`*
The minimum number of samples required to split an internal node.

`min_samples_leaf:` *`int or float, default=1`*
The minimum number of samples required to be at a leaf node.

Resources

Classification Trees

Regression Trees

Decision Trees

Pruning Regression Trees

Decision Trees

What is Bootstrap?

Bootstrapping is a statistical procedure that resamples a single dataset to create many simulated samples.

How Bootstrapping Works

Each original data point has an equal probability of being randomly drawn and included in the resampled datasets.

The data points are drawn with replacement. i.e. each data point can be selected more than once for the same resampled dataset.

The resampled datasets are of the same size as the original dataset.

The central assumption for bootstrapping is that the original sample accurately represents the actual population.

Creating a Bootstrapped Dataset

Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

$\longrightarrow$

Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	18	Yes
Yes	Yes	38	Yes
Yes	No	50	No
Yes	No	50	No

What is Bagging?

Bootstrapping the data and using the aggregate to make a decision.

- Create n trees using a different bootstrapped datasets and classify a new sample according to the majority vote.

Random Forest

Easy to work with.

Work with all types of data.

More accurate than a single tree.

Creating a Random Forest

Build n trees:

1. Using a bootstrapped dataset (different for each tree).
2. Considering a random subset of features at each step.

Making Predictions using a RF

Classification:

1. Run the sample down each tree and keep track of the predictions.
2. The final output is the one that received most of the votes.

Regression:

1. Run the sample down each tree and keep track of the predictions.
2. The final output is the average of all predictions.

Estimating the Accuracy of a RF

Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

Out-of-bag dataset

Loves Popcorn	Loves Soda	Age	Loves Movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	18	Yes
Yes	Yes	38	Yes
Yes	No	50	No
Yes	No	50	No

Out-of-bag Error

- We can measure how accurate our random forest is by the proportion of out-of-bag samples that were correctly classified.
- Out-of-bag Error: the proportion of out-of-bag samples that were incorrectly classified.

Random Forest Hyper-Parameters

`n_estimators:` *`int, default=100`*
The number of trees in the forest.

`bootstrap:` *`bool, default=True`*
Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.

- The same parameters used for decision trees can be used for random forests.

Resources

Random Forest

Random Forest

Random Forest

Random Forest in Pyhton

Random Forest Classifier

Random Forest Regressor