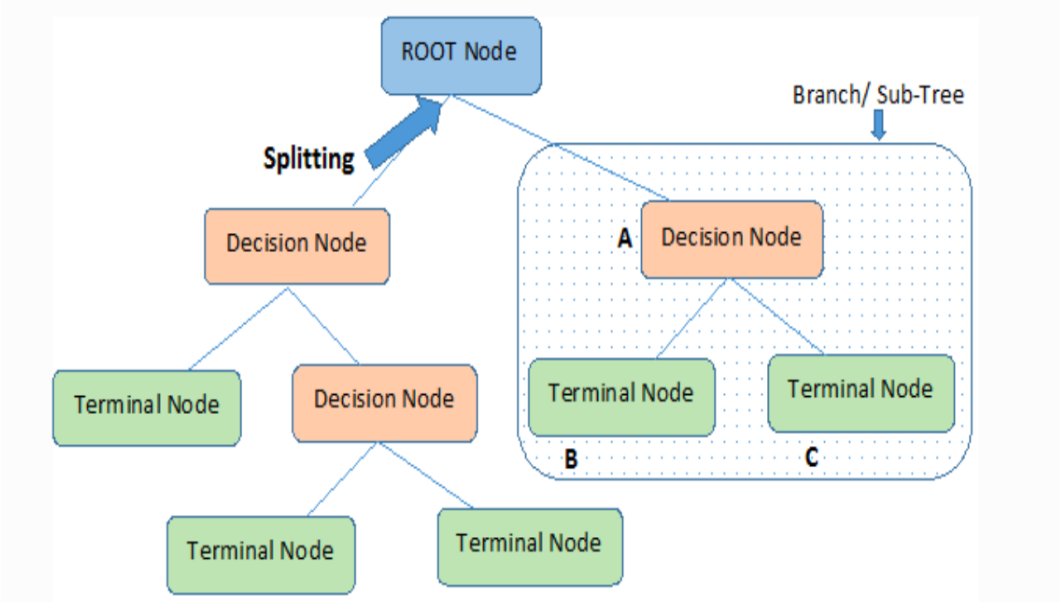
1 Introduction

Decision Trees are the foundation for many classical machine learning algorithms like Random Forests, Bagging, and Boosted Decision Trees. The idea was first proposed by Leo Breiman, a statistician at the University of California, Berkeley.

Now decision trees are widely used in many applications for predictive modeling. Sometimes decision trees are also referred to as CART, which is short for Classification and Regression Trees. Let's discuss in-depth how decision trees work, how they're built from scratch, and how we can implement them in R.

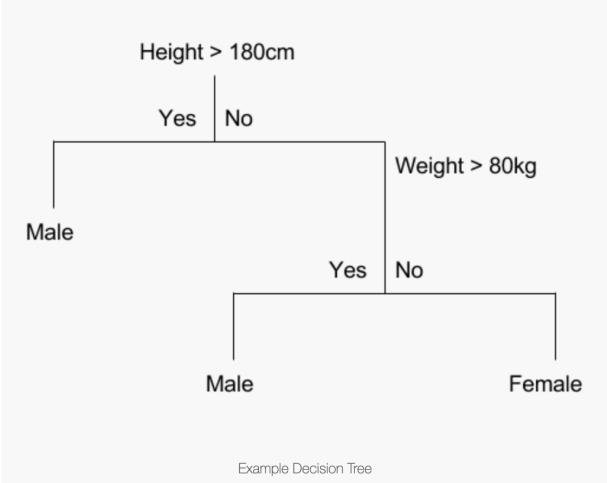
2 Definitions and Terminologies

2.1 What is a Decision Tree?



- **Root Node:** A root node is at the top of a tree. It represents entire population being analyzed and this can further get divided based on various features/attributes.
- **Splitting:** It is a process of dividing a node into two or more sub-nodes.
- **Decision Node:** When a sub-node splits into further sub-nodes, then it is called a decision node.
- **Leaf/Terminal Node:** A sub-node that does not split is called a Leaf or Terminal node.
- **Sub-Tree:** A sub section of entire tree is called branch or sub-tree.
- **Pruning:** Removing a sub-node from the tree is called pruning.

Example 1. Given a dataset with two attributes (e.g., height in centimeters and weight in kilograms) and the target variable (y) of sex as male or female, below is a crude example of a decision tree.



Making predictions: Given the input of [height = 165 cm, weight = 65 kg],

2.2 Types of Decision Trees

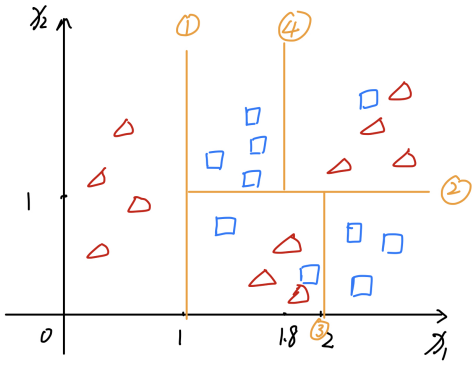
Based on the type of target variable, there are two types of decision tree:

1. **Classification Decision Tree:** Decision Tree which has categorical target variable is called as classification decision tree. In example 1, the target variable is "Male or Female".
2. **Regression Decision Tree:** Decision Tree that has continuous target variable is called as Regression Decision Tree.

In this lecture, we will focus on the Classification Decision Tree.

3 Creating a Decision Tree based on an existed splitting

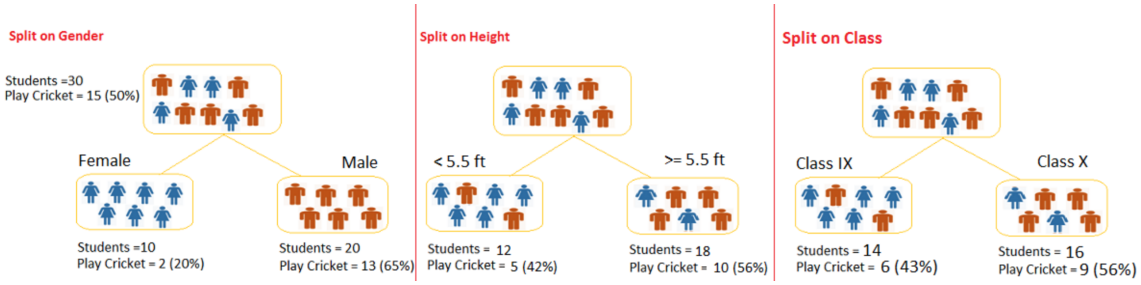
Question: Assume there exists a sequence of splitting on data set, how this can be related with a decision tree:



4 How to split

With more than one attribute taking part in the decision-making process, it is necessary to decide the relevance and importance of each of the attributes so that we can place the most relevant one at the root node and further traversing down by splitting the nodes.

Example 1. Let's say we have a sample of 30 students with three attributes: Gender (Male/ Female), Class (IX/ X), and Height. 15 out of these 30 play cricket in leisure time. Now, I want to create a CART model to predict who will play cricket during leisure period? **But what's the ROOT Node?** What Splitting measures can be used?



4.1 Gini Index

Used by the CART algorithm for classification trees, Gini Index/ Gini impurity (named after Italian mathematician Corrado Gini) is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

The Gini Index can be computed by summing the probability p_i of an item with label i being chosen times the probability $\sum_{k \neq i} p_k = 1 - p_i$ of a mistake in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category.

Definition 4.1 (Gini Index) *For a set of items with J classes, suppose $i \in \{1, 2, \cdots, J\}$, and let p_i be the fraction of items labeled with class i in the set, the Gini index for this set is given by:*

$$I_G(p) = \sum_{i=1}^{J} (p_i \sum_{k \neq i}^{J} p_k) = \sum_{i=1}^{J} p_i(1 - p_i) = 1 - \sum_{i=1}^{J} p_i^2$$

Example 2.

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Negative	High	Low	Down
Positive	Low	High	Up
Positive	High	High	Up
Negative	Low	High	Down
Positive	Low	Low	Down
Negative	High	High	Down
Negative	Low	High	Down
Positive	Low	Low	Down
Positive	High	High	Up

Table: Gini Index example

Attributes/Features	Gini Index
Past Trend	0.27
Open Interest	0.47
Trading Volume	0.34

Table 1: Gini Index attributes or features

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Positive	Low	High	Up
Positive	High	High	Up
Positive	Low	Low	Down
Positive	Low	Low	Down
Positive	High	High	Up

Table: Gini Index calculation for the Positive branch of Past Trend

Attributes/Features	Gini Index
Open Interest	0.33
Trading Volume	0

Table 2: Gini Index attributes or features

4.2 Other Measurements for Splitting

If we replace Gini Index I_G with "Information Gain" function, then the approach is called ID3. If we replace with "Gain Ratio" function, it will be C4.5 algorithm.

5 Implementation with CART in R programming

- How a CART model can be learned from training data.

  Greedy Splitting: All input variables and all possible split points are evaluated and chosen in a greedy manner based on Gini Index (e.g. the very best split point is chosen each time).

  Stopping Criterion: The recursive binary splitting procedure described above needs to know when to stop splitting as it works its way down the tree with the training data. The most common stopping procedure is to use a minimum count on the number of training instances assigned to each leaf node. If the count is less than some minimum then the split is not accepted and the node is taken as a final leaf node.

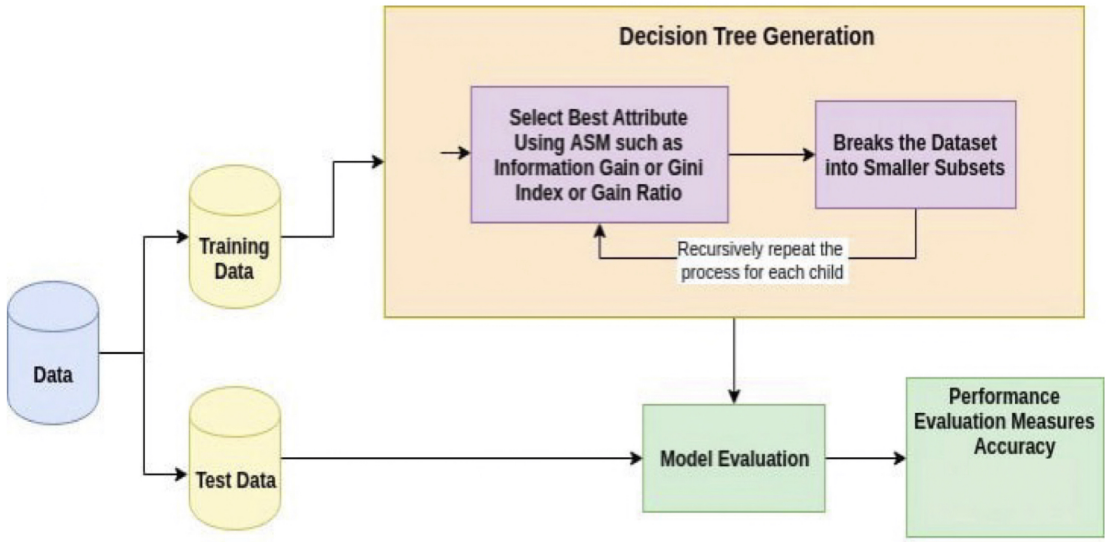
  Pruning The Tree: The stopping criterion is important as it strongly influences the performance of your tree. You can use pruning after learning your tree to further lift performance.

  The complexity of a decision tree is defined as the number of splits in the tree. Simpler trees are preferred. They are easy to understand (you can print them out and show them to subject matter experts), and they are less likely to overfit your data.

  The fastest and simplest pruning method is to work through each leaf node in the tree and evaluate the effect of removing it using a hold-out test set. Leaf nodes are removed only if it results in a drop in the overall cost function on the entire test set. You stop removing nodes when no further improvements can be made.

  More sophisticated pruning methods can be used such as cost complexity pruning (also called weakest link pruning) where a learning parameter (alpha) is used to weigh whether nodes can be removed based on the size of the sub-tree.

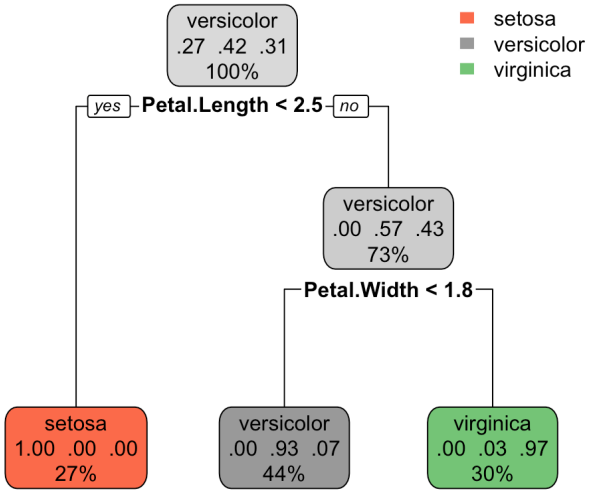
- How a learned CART model can be used to make predictions on unseen data.



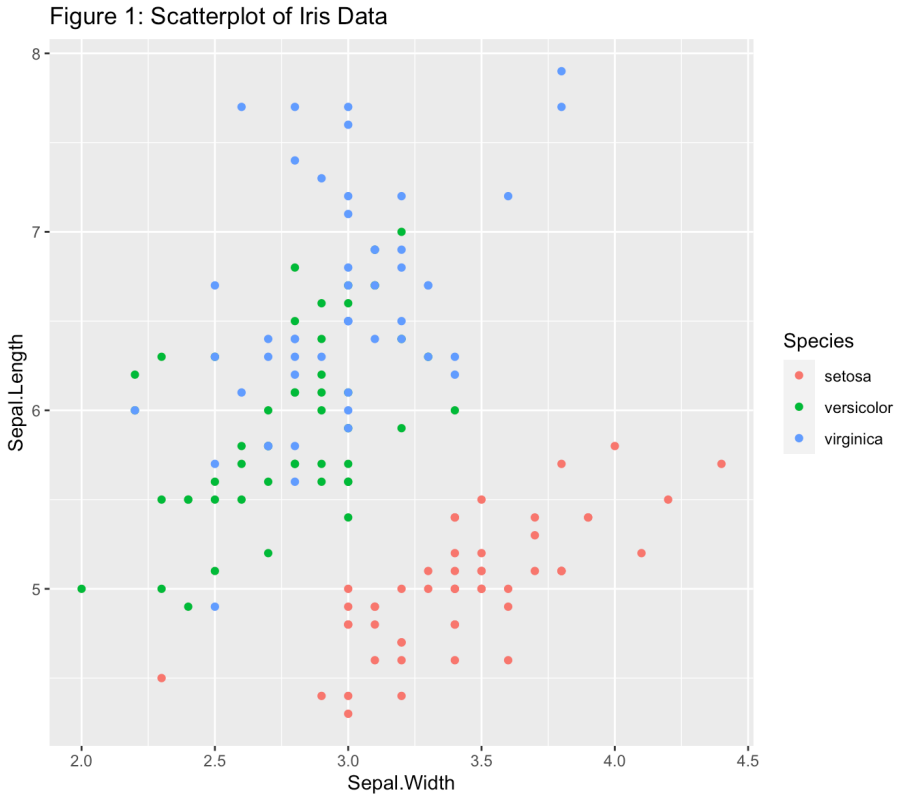
For R users, there are multiple packages available to implement CART such as ctree, rpart, tree etc. We will use a famous data set in R, called "iris". The iris dataset is a built-in dataset in R that contains measurements on 4 different attributes (Sepal Length, Sepal Width, Petal Length, and Petal Width) for 50 flowers from 3 different species (setosa, virginica, versicolor). So totally 150 samples. We will implement CART model to

classify the iris dataset, and report the prediction accuracy. We will also plot the classification decision tree.

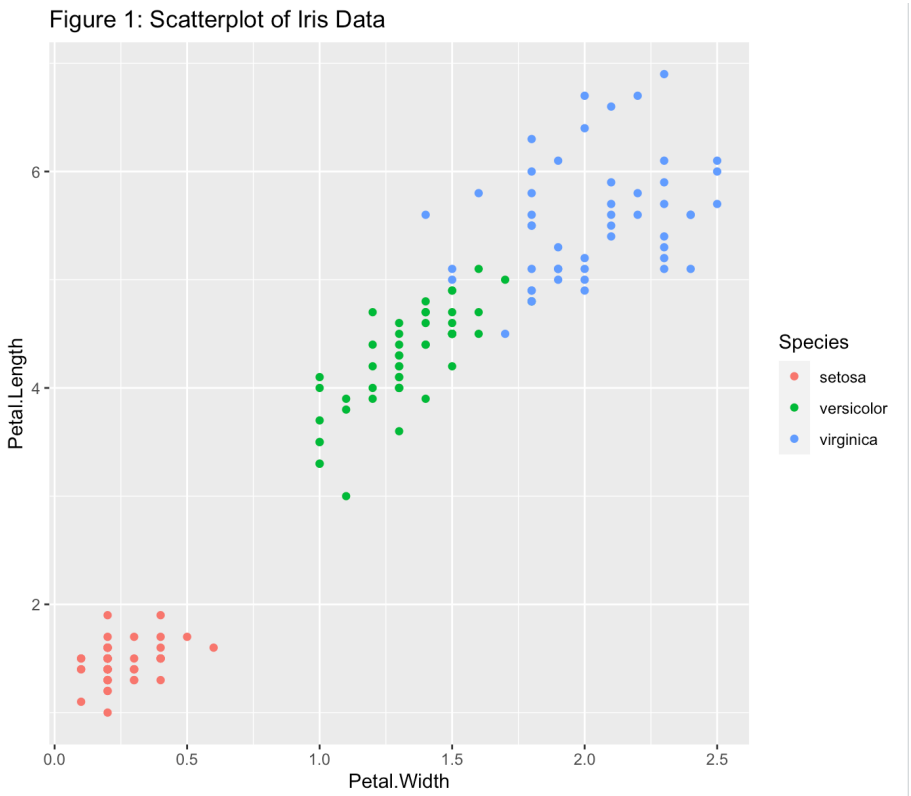
```
# Don't forget to install the following R packages first!
#install.packages("naniar")
#install.packages("rpart")
#install.packages("rpart.plot")
#install.packages("ggformula")

library(naniar) #naniar is a package to make it easier to summarise and handle missing values
data("iris")
#iris <- subset(iris,Species!="setosa") #remove "setosa" species
any_na(iris) #Dtree algorithms cannot take N/A value, we need to check the N/A value first.
n <- nrow(iris)
set.seed(1117) #specify seeds
#new <- iris[sample(n),]
t_idx <- sample(seq_len(n), size = round(0.7 * n)) #70 percent training data, 30 percent test data.
traindata <- iris[t_idx,]
testdata <- iris[ - t_idx,]
library(rpart) #"rpart" package is used to implement CART, the split default to Gini.
library(rpart.plot)
tree <- rpart(Species ~ ., data = traindata,
              method = "class") #change to anova for numerical

#pruning process, find the best stopping point.
printcp(tree) #find the best/optimal stopping point (CP, complexity paramater)
tree.pruned <- prune(tree,cp = tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"])

rpart.plot(tree.pruned)

future <- predict(tree.pruned, testdata, type="class")
future <- as.data.frame(future)
final <- cbind(future, testdata)
confusion <- table(final$Species,final$future, dnn = c("truth", "predicted"))
confusion
accuracy <- sum(diag(confusion)) / sum(confusion)
accuracy

#plot the iris dataset on "Sepal Length + Sepal Width" and "Petal Length + Petal Width" plane
library(ggformula)
scatterplot1=gf_point(Sepal.Length ~ Sepal.Width, data = iris, color = ~ Species) %>%
  gf_labs(title = "Figure 1: Scatterplot of Iris Data")
scatterplot2=gf_point(Petal.Length ~ Petal.Width, data = iris, color = ~ Species) %>%
  gf_labs(title = "Figure 1: Scatterplot of Iris Data")
scatterplot1
scatterplot2
```



Classification Decision Tree for iris



The data points locate not very regularly, it is difficult to split on this "Sepal" plane.



This picture clearly shows that we can split the dataset by simply using the length of petal and the width of the petal. See the following picture for the splitting points.