Supervised ML

Classification

Introduction

- **Data analysts** frequently need to predict a categorical outcome from a set of variables, e.g.,
  - Predicting whether an individual will repay a loan, given their demographics and financial history *(good credit risk/bad credit risk)*
  - Determining whether an ER patient is having a heart attack, based on their symptoms and vital signs *(heart attack/no heart attack)*
  - Deciding whether an email is spam, given the presence of key words, images, hypertext, header information, and origin *(spam/not spam)*

- **The goal** is to find an accurate method of classifying new cases from a set of predictors *(also called features)* into one of the two groups.

Supervised Learning

- **Supervised learning** starts with a set of observations containing values for both the predictor variables and the outcome.
  - The dataset is then *divided into a training sample and a test sample.*
  - A *predictive model* is developed using the data in the training sample and tested for accuracy using the data in the test sample.
  - Both samples are needed because classification techniques *maximize prediction for a given set of data.*
  - Estimates of their effectiveness will be *overly optimistic* if they're evaluated using the same data that generated the model.
  - By applying the classification rules *developed on a training sample* to a separate test sample, *you can obtain a more realistic accuracy estimate.*
  - An effective predictive model can be *used it to predict outcomes* in situations when only the predictor variables are known.

Data Set Information

- **Example:** Wisconsin Breast Cancer
  - From UCI Machine Learning Repository
  - The *goal* will be to develop a *model for predicting* whether *a patient has breast cancer* based on the characteristics of a fine-needle tissue aspiration (a tissue sample taken with a thin hollow needle from a lump or mass just under the skin)
  - The dataset contains 699 *fine-needle aspirate samples*
    - 458 (65.5%) are *benign*
    - 241 (34.5%) are *malignant*
  - The dataset contains *11 variables* and *doesn't include the variable names*.
  - Sixteen samples have *missing data* and are coded with a question mark (?).

Data Set Information

- **Example:** Wisconsin Breast Cancer Dataset
  - From UCI Machine Learning Repository
  - Data file: *breast_cancer.csv*
  - How to import in Python?
  - Variables

1. ID	*2. Clump thickness*
3. Uniformity of cell size	*4. Uniformity of cell shape*
5. Marginal adhesion	*6. Single epithelial cell size*
7. Bare nuclei	*8. Bland chromatin*
9. Normal nucleoli	*10. Mitoses*
11. Class	

Install the ucimlrepo package

```
pip install ucimlrepo
```

Import the dataset into your code

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
breast_cancer_wisconsin_original = fetch_ucirepo(id=15)

# data (as pandas dataframes)
X = breast_cancer_wisconsin_original.data.features
y = breast_cancer_wisconsin_original.data.targets

# metadata
print(breast_cancer_wisconsin_original.metadata)

# variable information
print(breast_cancer_wisconsin_original.variables)
```

Data Set Information

- **Example:** Wisconsin Breast Cancer
  - The first variable is an *ID variable* (which we will drop)
  - Last variable *(class)* contains the outcome *(coded* $2 =$ *benign,* $4 =$ *malignant)*
  - We will also exclude the observations containing *missing values.*
  - For each sample, nine *cytological characteristics* previously found to correlate with malignancy are also recorded.
  - Each of these variables is scored from 1 *(closest to benign)* to 10 *(most anaplastic)*.
  - No one predictor alone can distinguish between *benign* and *malignant* samples.
  - The *challenge* is to find a *set of classification rules* that can be used to *accurately predict* malignancy from some combination of these nine cell characteristics.

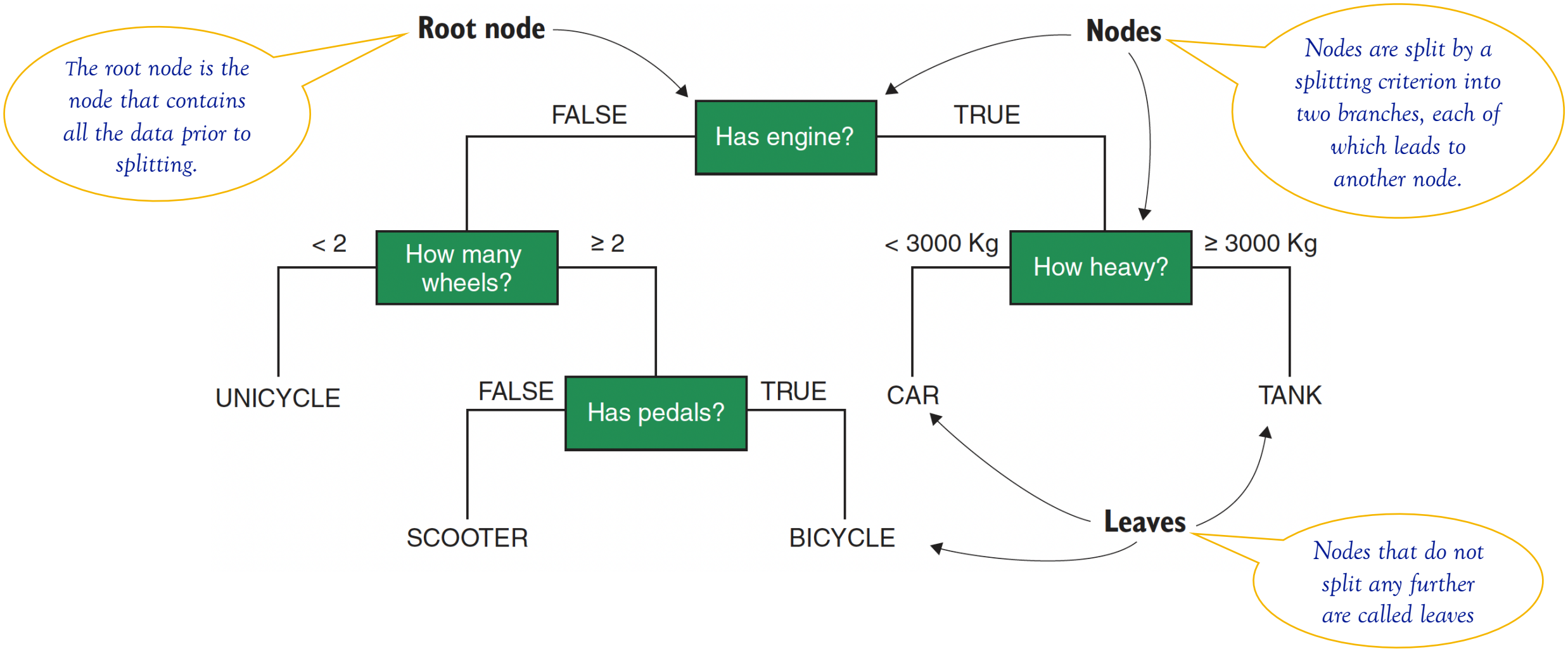
Decision Trees

- **Decision trees** are popular in data mining contexts.
  - DT learn a *sequence of questions* that separates cases into different classes.
  - Each question has a *binary answer*, and cases will be sent down the *left or right branch* depending on which criteria they meet.
  - There can be *branches within branches*; and once the model is learned, it can be *graphically represented as a tree*.
  - Have you ever played the game *20 questions*, where you have to guess what object, someone is thinking of by asking *yes-or-no questions?*
  - Another example is the game *Guess Who,* where you have to guess the other player's character by asking *questions about their appearance?*

Decision Trees

- **Example:**
  - Imagine that you want to create a model to represent the way people commute to work, *given features of the vehicle*.
  - You gather information on the vehicles, such as how many *wheels* they have, whether they have an *engine*, and their *weight*.
  - We can formulate this *classification process* as a *series of sequential questions*.
  - Every vehicle is *evaluated at each question* and *moves either left or right* in the model depending on how its features satisfy the question.

Decision Trees



Decision Trees

Tree-based models can be used for both classification and regression

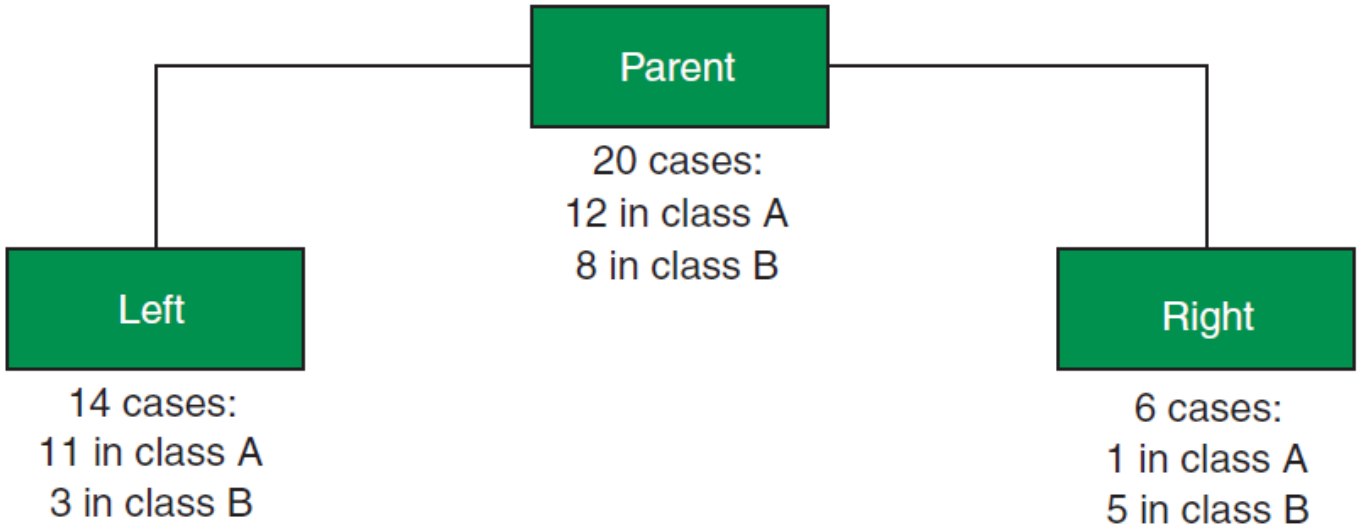
- *Classification trees*
- *Regression tress*
- To decide on the best features at each split two approaches are used:
  - *Information gain*
  - *Gini gain*
- *Gini gain* is slightly faster to compute, so we'll focus on it.

Gini Gain / Gini Index

- **Gini gain** is used to find the best split for a particular node when growing a decision tree
  - Gini index is used to measure the *impurity*.
  - **Impurity** is a measure of how *heterogeneous* the classes are within a node.
  - If a node contains only a *single class* (or a *leaf*), it would be said to be *pure*.
  - By estimating the *impurity* that would result from using each predictor variable for the next split, the algorithm can choose the feature that will result in the *smallest impurity*.
  - Put another way, the algorithm chooses the feature that will result in subsequent nodes that are as *homogeneous* as possible.

Gini Gain / Gini Index

- **Example:**
  - We have 20 cases in a parent node belonging to two classes, A and B. We split the node into two leaves based on some criterion.
  - In the left leaf, we have 11 cases from class A and 3 from class B.
  - In the right leaf, we have 5 from class B and 1 from class A.



Gini Gain / Gini Index

- **Example: We want to know the *Gini gain* of the split**
  - The Gini gain is the *difference* between the Gini index of the *parent node* and the Gini index of the *split*.
  - Looking at our example, the *Gini index* for any node is calculated as:

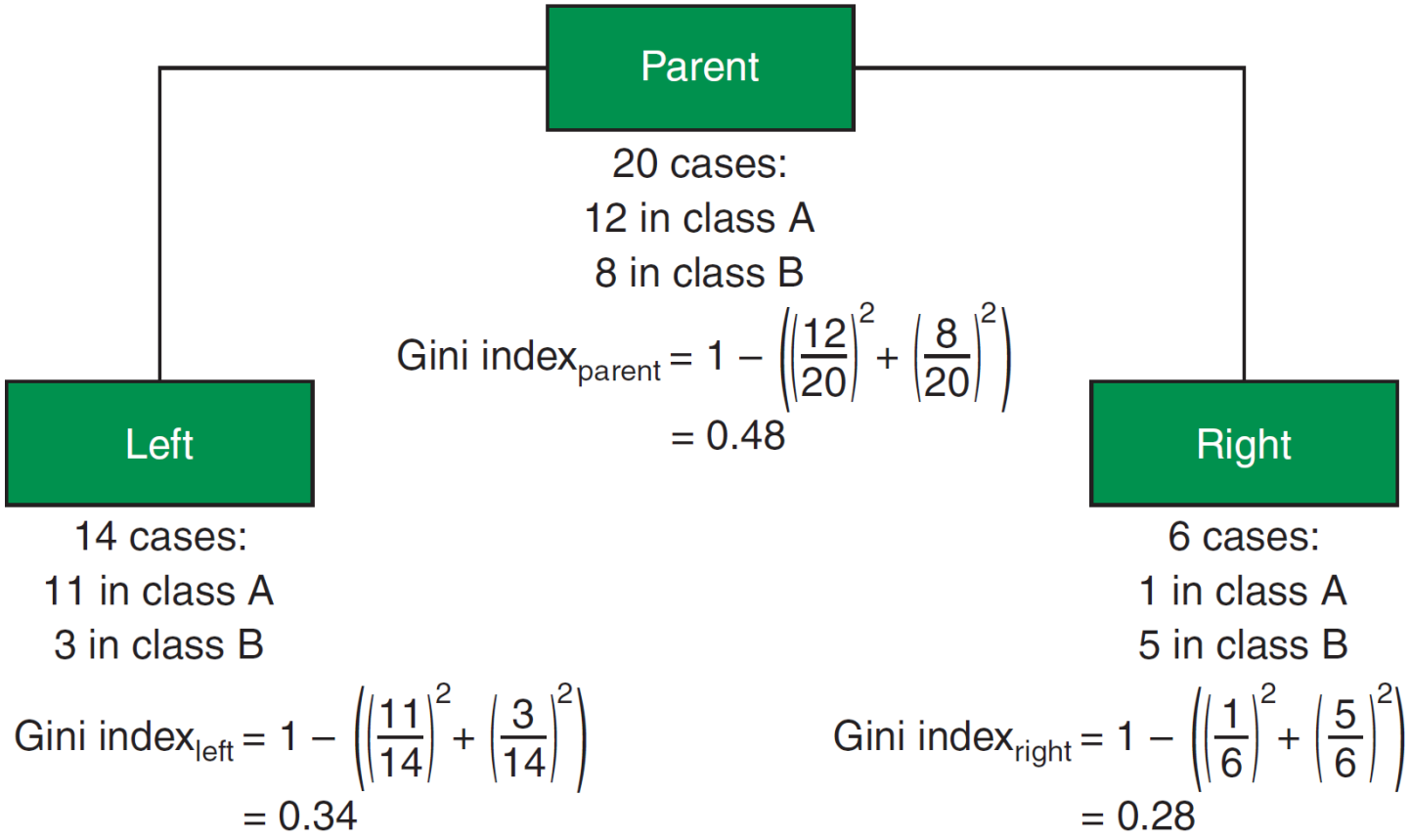
$$Gini\ index = 1 - (p(A)^2 + p(B)^2)$$

where $p(A)$ and $p(B)$ are the proportions of cases belonging to classes A and B, respectively.

Gini Gain / Gini Index

- **Example: We want to know the *Gini gain* of the split**

$$Gini\ index\ = 1 - (p(A)^2\ + p(B)^2)$$



Gini Gain / Gini Index

- **Example:**
  - Once we have the Gini indices for the left and right leaves, we can calculate the *Gini index* for the split as a whole.
  - Gini index of the split is the *sum of the left and right* Gini indices *multiplied* by the *proportion* of cases they accepted from the parent node.

$$Gini\ index_{split} = p(left) \times Gini\ index_{left} + p(right) \times Gini\ index_{right}$$

$$Gini\ index_{split} = \frac{14}{20} \times 0.34 + \frac{6}{20} \times 0.28 = 0.32$$

  - Gini gain (difference between the Gini indices of the *parent node* and the *split*)

$$Gini\ gain = 0.48 - 0.32 = 0.16$$

where 0.48 *is the Gini index of the parent*

Gini Gain / Gini Index

- **Gini gain** at a particular node is calculated for *each predictor variable*.
- The predictor that generates the *largest Gini gain* is used to split that node.
- This process is *repeated* for every node *as the tree grows*.
- Generalizing the Gini index to any number of classes:

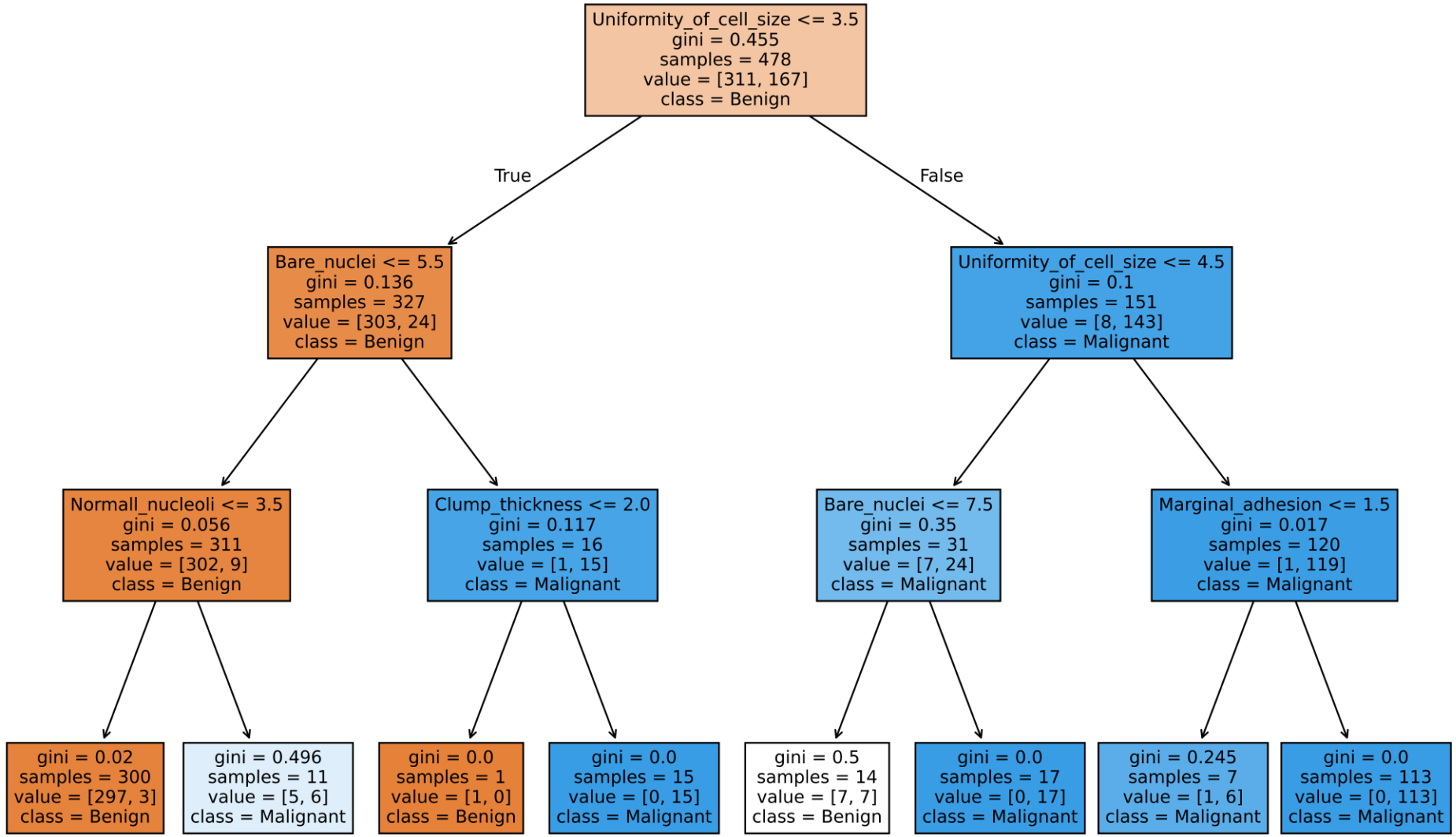
$$Gini\ index = 1 - \sum_{k=1}^{K} p(class_k)^2$$

which is just a fancy way of saying that we calculate $p(class_k)^2$ for each class from 1 to K (the number of classes), add them all up, and subtract this value from 1.

Strengths & Weaknesses

- **Strengths**
  - The intuition behind tree-building is quite *simple*, and *interpretable*.
  - It can handle *categorical and continuous predictor* variables.
  - It *makes no assumptions* about the distribution of the predictor variables.
  - It can *handle missing values* in sensible ways.
  - It can *handle continuous variables* on different scales.

- **Weaknesses**
  - Individual trees are very *susceptible to overfitting*—so much so that they are rarely used.


Decision Tree for Breast Cancer Classification
Uniformity_of_cell_size <= 3.5
gini = 0.455
samples = 478
value = [311, 167]
class = Benign
True
False
Bare_nuclei <= 5.5
gini = 0.136
samples = 327
value = [303, 24]
class = Benign
Uniformity_of_cell_size <= 4.5
gini = 0.1
samples = 151
value = [8, 143]
class = Malignant
Normall_nucleoli <= 3.5
gini = 0.056
samples = 311
value = [302, 9]
class = Benign
Clump_thickness <= 2.0
gini = 0.117
samples = 16
value = [1, 15]
class = Malignant
Bare_nuclei <= 7.5
gini = 0.35
samples = 31
value = [7, 24]
class = Malignant
Marginal_adhesion <= 1.5
gini = 0.017
samples = 120
value = [1, 119]
class = Malignant
gini = 0.02
samples = 300
value = [297, 3]
class = Benign
gini = 0.496
samples = 11
value = [5, 6]
class = Malignant
gini = 0.0
samples = 1
value = [1, 0]
class = Benign
gini = 0.0
samples = 15
value = [0, 15]
class = Malignant
gini = 0.5
samples = 14
value = [7, 7]
class = Benign
gini = 0.0
samples = 17
value = [0, 17]
class = Malignant
gini = 0.245
samples = 7
value = [1, 6]
class = Malignant
gini = 0.0
samples = 113
value = [0, 113]
class = Malignant

Questions?