Machine Learning

Lecture 09

Decision Trees

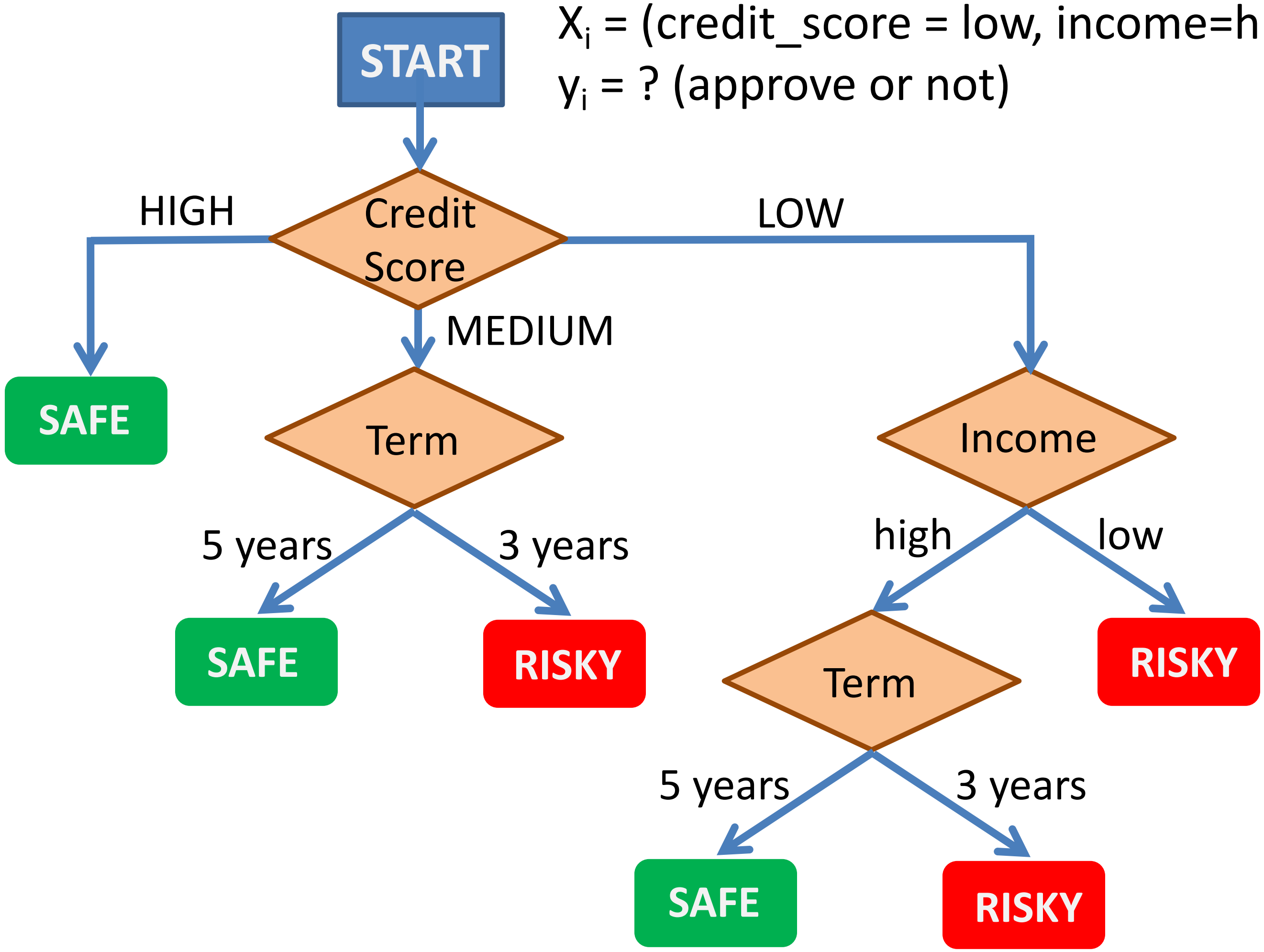
Intuition on Decision Trees

What makes a loan risky?

- Credit history, income, payback time?

X_i = (credit_score = low, income=high, term=short)

y_i = ? (approve or not)



Where do we start splitting the tree?

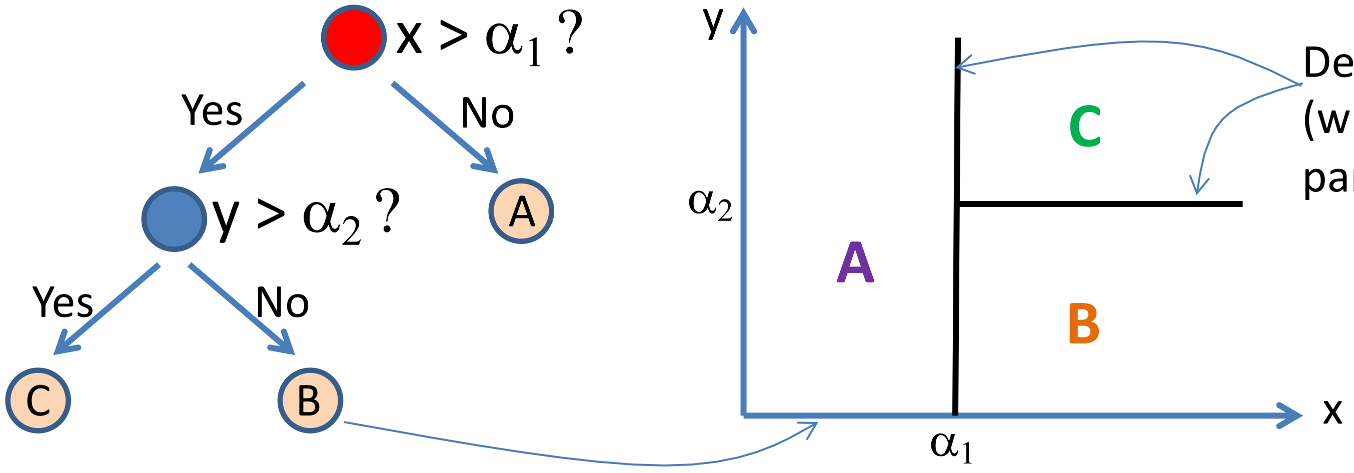
How do we split low/high?

Decision Tree learning

- Decision Tree (DT) Learning
  - A simple non-linear classifier that is nice and fast

Tree

Coordinate System (sample space)



Decision boundaries (which are always parallel to the axes)

● Root node

● Internal (decision) node (labeled with attributes-tests)

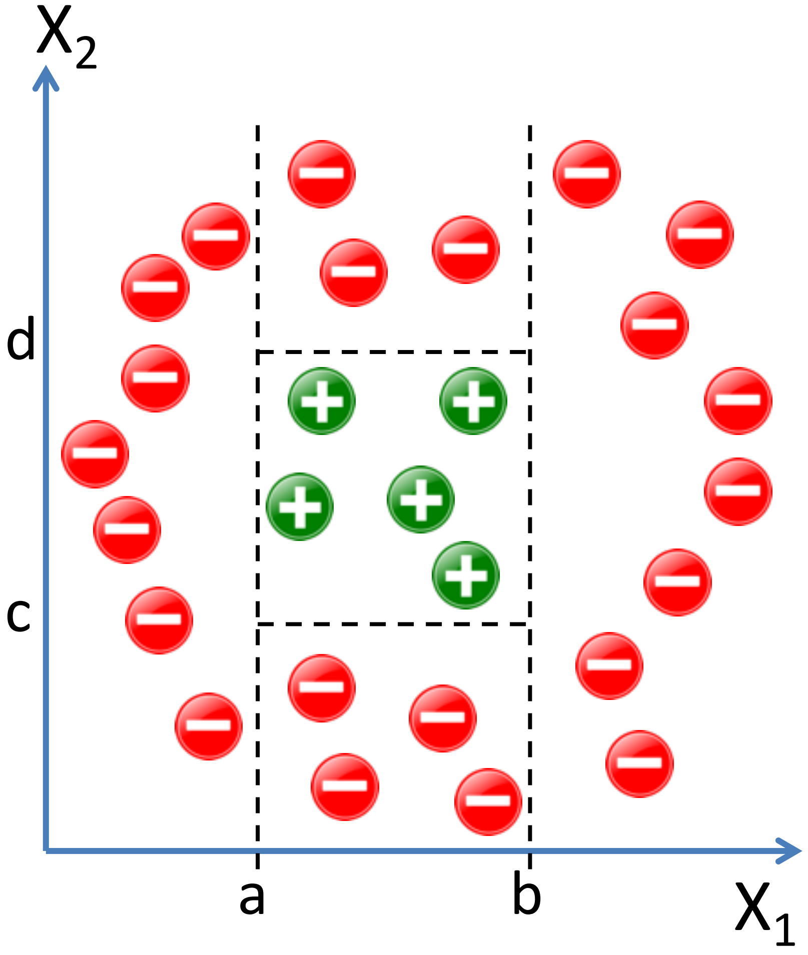
○ Leaf (terminal) node (labeled with classes)

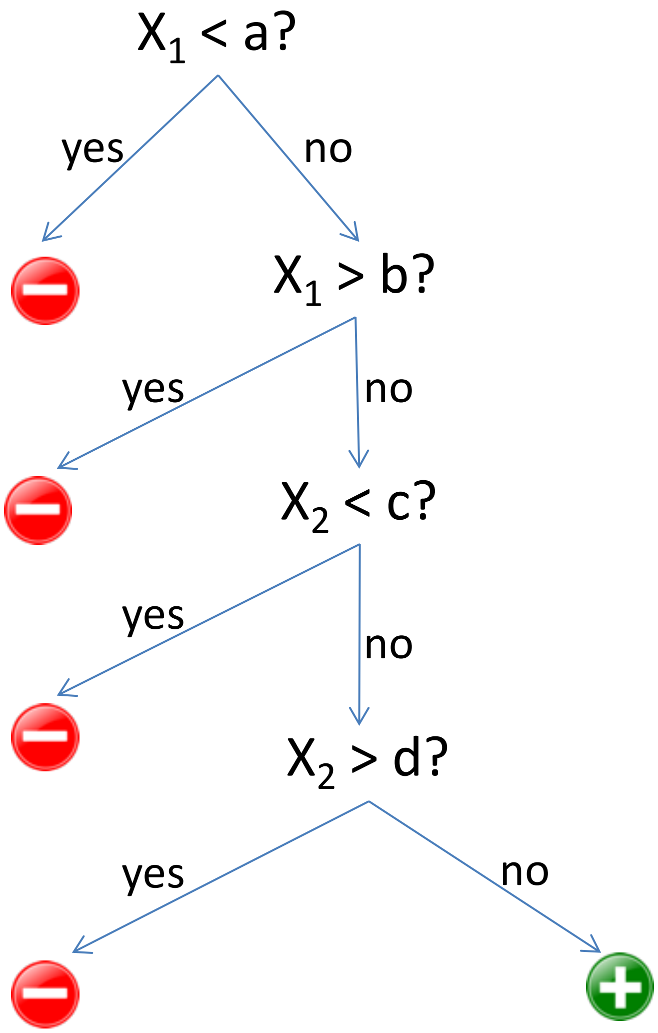
↘ Branch (Labeled with conditions or outcomes)

Split in the tree corresponds to a split in the coordinate system. DTs divide the feature space into axis-parallel rectangles and label each rectangle with one class.

DT example 1

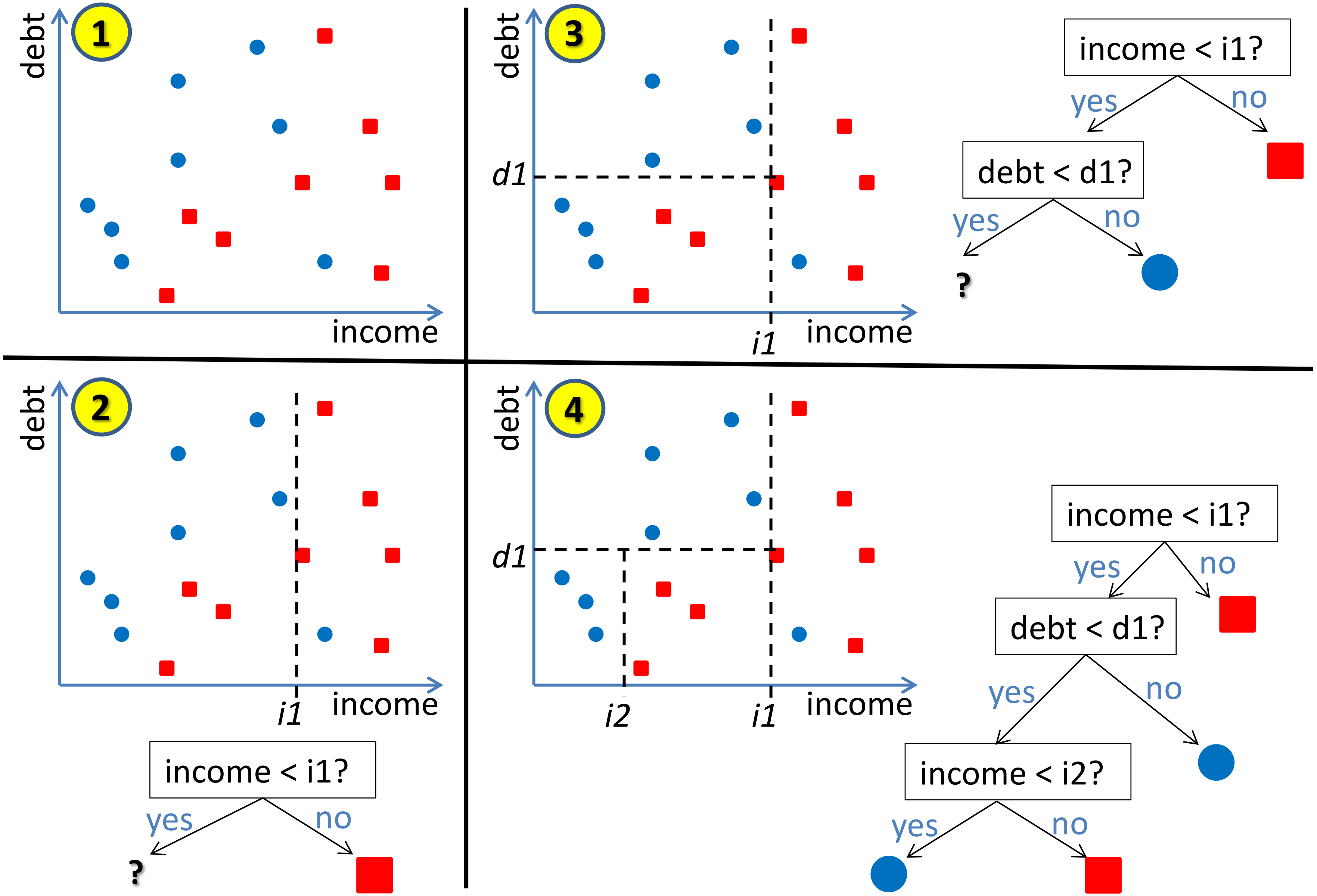
A divide-and-conquer scheme





Objective: Minimize error in each leaf (purity)

DT example 2



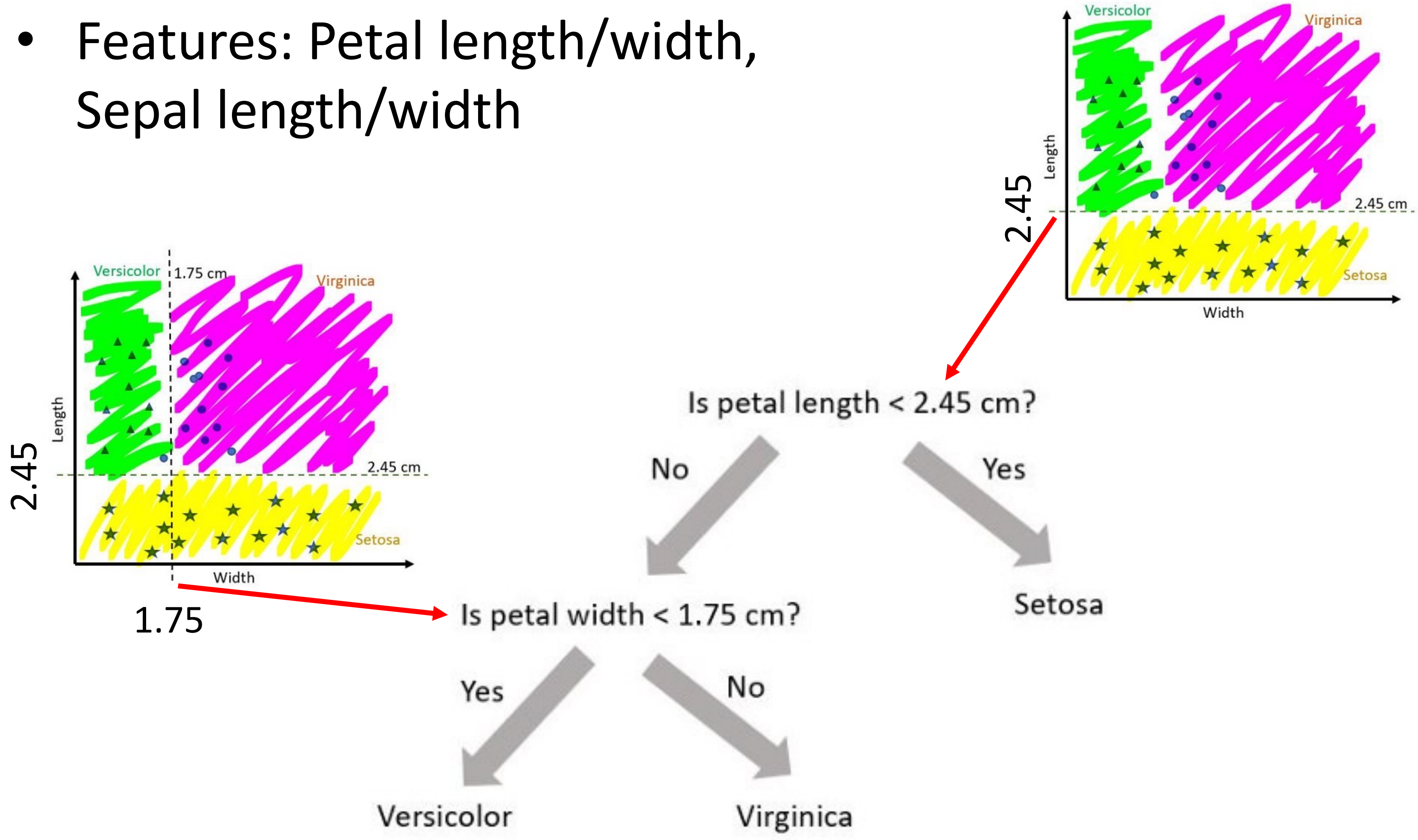
income < i1?
yes
no
debt < d1?
yes
no

income < i1?
yes
no

income < i1?
yes
no
debt < d1?
yes
no
income < i2?
yes
no

DT Example 3

- Iris Flower Classification
- Features: Petal length/width, Sepal length/width



Is petal length < 2.45 cm?

No

Yes

Setosa

Is petal width < 1.75 cm?

Yes

No

Versicolor

Virginica

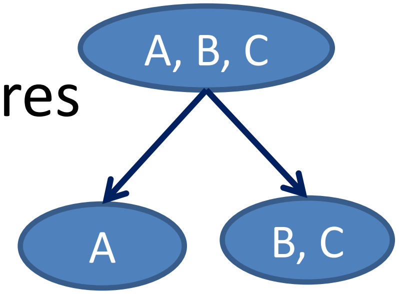
Ref: https://medium.com/x8-the-ai-community/decision-trees-an-intuitive-introduction-86c2b39c1a6c

Characteristics of DTs

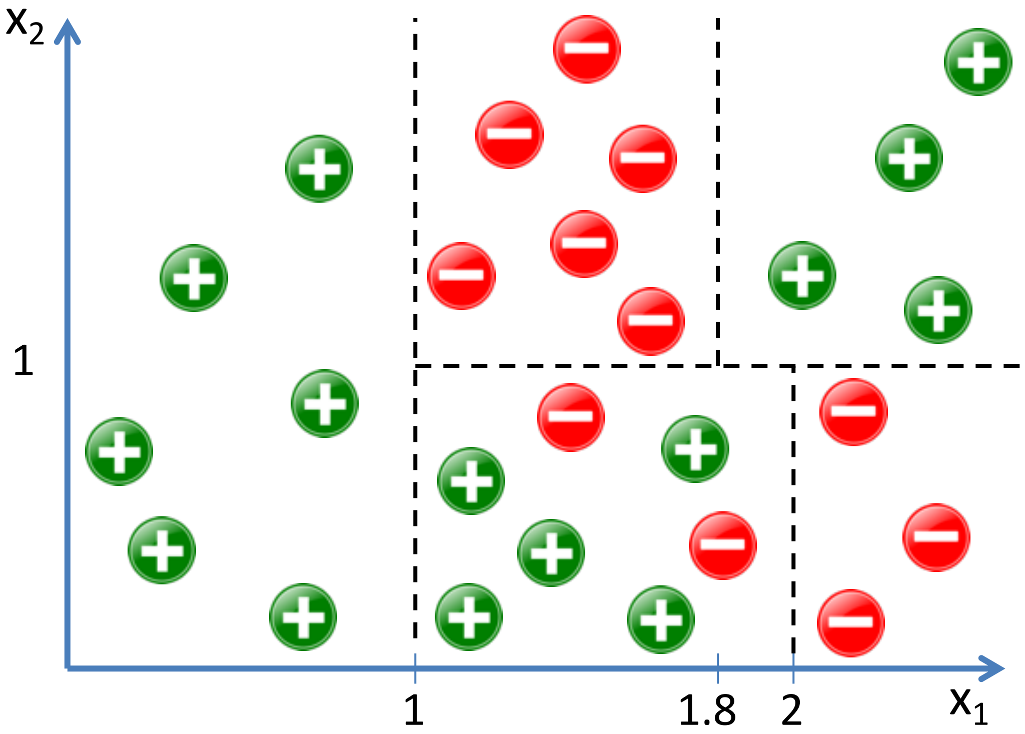
- Learning method: Supervised learning (discriminative, non-parametric)
- Types of DTs: ID3, C4.5, C5.0, CHAID, CART
- Handling of mixed data types: Yes
- Requires linearly separable data? No
- Classification ability: Binary and Multi-class
- Computationally demanding? No
- Ability for online classification: Yes
- Interpretability: High (white-box)
- When to consider: Medical diagnosis, text classification, credit risk analysis

Commonly used DTs

- **ID3**
  - Simple, created by R. Quinlan (1986)
  - Doesn’t handle missing values & susceptible to outliers
  - Doesn’t handle numerical features (categorical only)
  - Selection criterion: Entropy-based “Information Gain”
- **C4.5** (improved ID3)
  - Ability to handle numerical and categorical data
  - Handles missing data but susceptible to outliers
  - Selection criterion: Entropy-based “Information Gain”
- **CART** (Classification and Regression Trees)
  - Creates a binary tree (**decision nodes have exactly two branches**)
  - Handles both categorical & numerical features
  - Handles outliers and missing values
  - Selection criterion: "Gini index (impurity)"

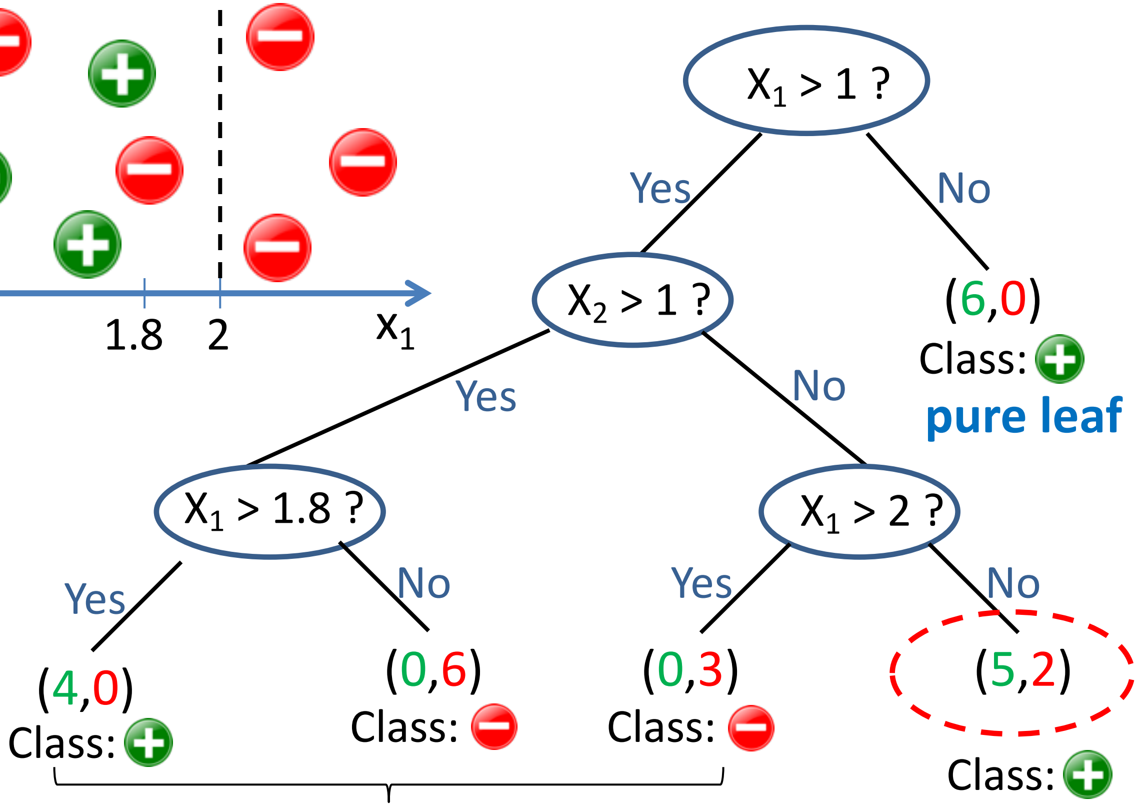


Classification tree



Binary tree: Minimize error in each leaf

$(x_1,y_1), (x_2,y_2), \ldots (x_n,y_n)$



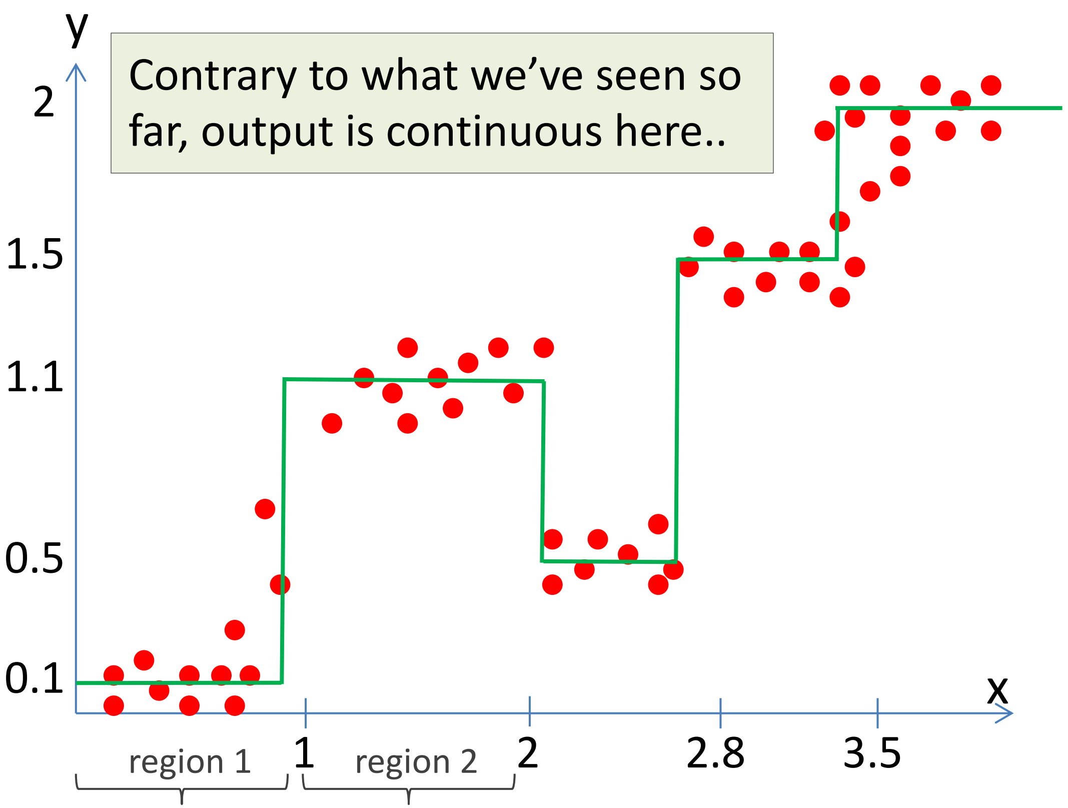
pure leaf

pure leaves

$x_i \in \Re 2$

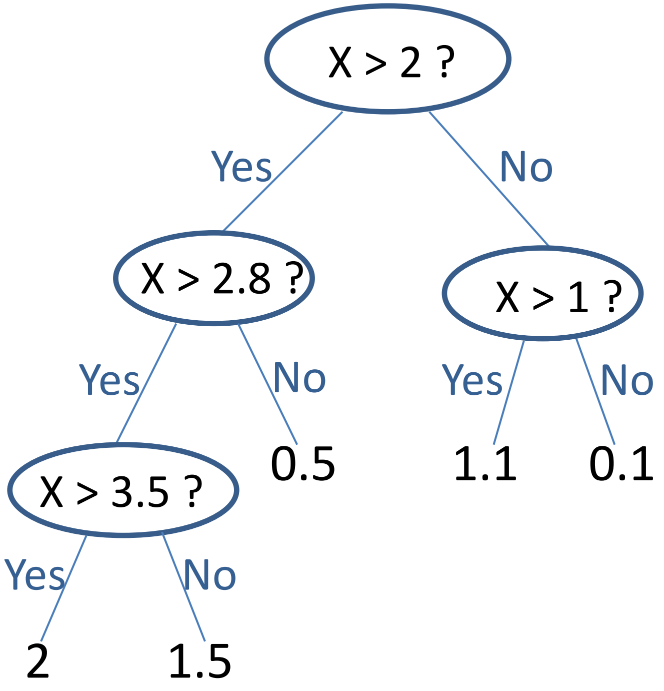
$y \in (+,-)$

Regression tree



Contrary to what we've seen so far, output is continuous here..

$$x_i \in \Re$$
$$y_i \in \Re$$



$$\hat{y} = \arg\min_{y} \sum_{i\in R} (\hat{y} - y_i)^2$$

$$\frac{\partial \sum_{i=1}^{m} (\hat{y} - y_i)^2}{\partial \hat{y}} = 0 \Rightarrow \hat{y} = \frac{1}{m} \sum_{i=1}^{m} y_i \left.\vphantom{\sum_{i=1}^{m}}\right\}$$

Average of the y_i's

We'll re-visit this in Ensemble Learning

Procedure

- Algorithm (divide-and-conquer)

0. Start at the root (with all training examples)

1. Select an attribute that best (?) separates the classes

2. Split it into subsets (child nodes)

– If the attribute value is categorical:

=> select category(ies)

– If the attribute is numerical:

=> select a threshold

3. Are they pure (all samples from the same class)?

– If yes: stop (training set is perfectly classified)

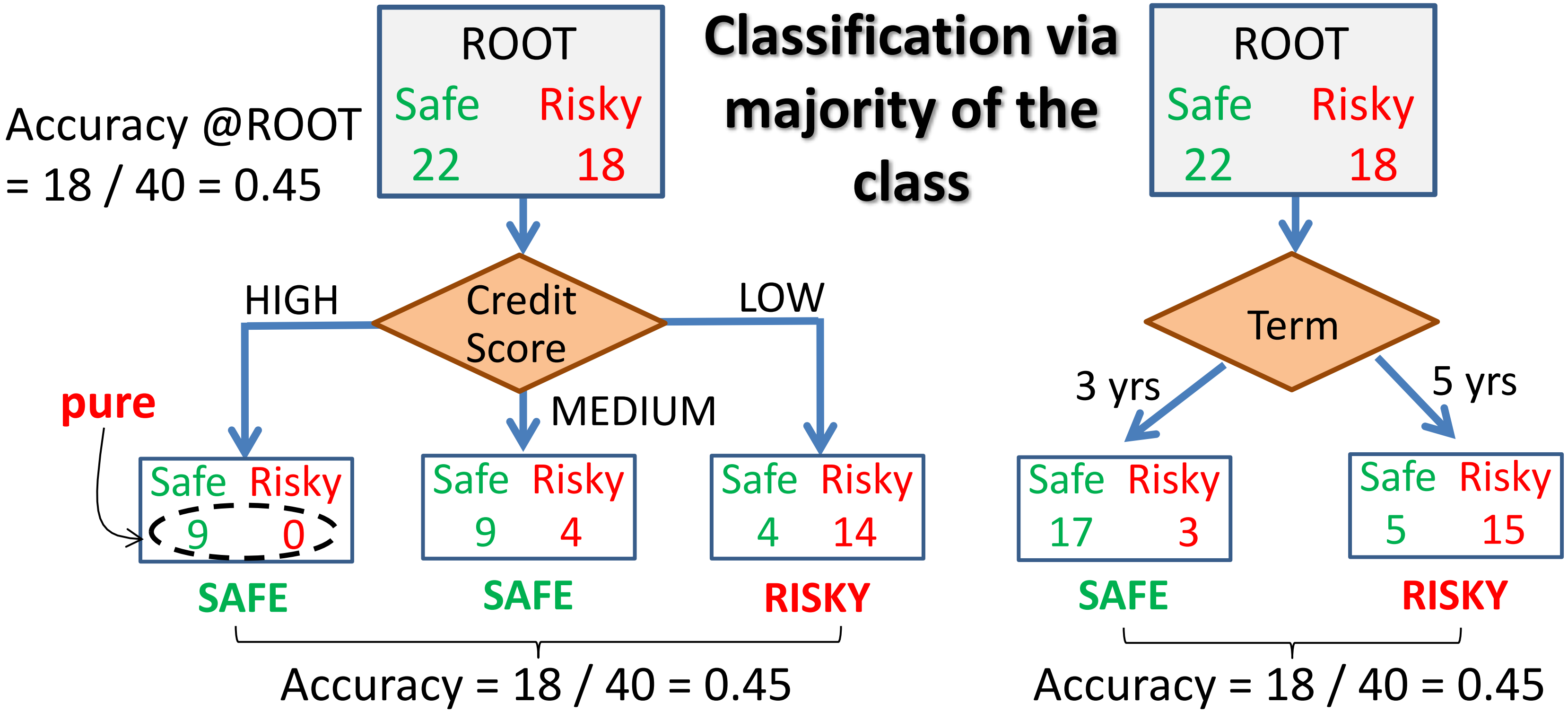
– If no: select an attribute and split further, go to (3)

Classification: When you have a new data point, start at the root and traverse down the tree to get to the subset this new data point belongs to

Challenges

- **Complications**
  - Attributes seldomly split a dataset perfectly
  - Attributes may be
    - Categorical: Binary (yes/no) or Multi-valued (hot, warm, cool)
    - Numerical
- **Questions**
  - Which feature to start with at the root (optimization on the attribute to pick)?
    - e.g. [income] or [debt]
  - Where to split the attribute from (break point for continuous attributes)?
    - income > xx?, debt > xx?
  - Sequence of other attributes down the tree
  - When to stop branching further?

Which attribute first?



- What if we want to start with "income" which is a numerical attribute? What would the splitting point be down the tree?
- We need some sort of a measure to decide which attribute to start splitting with. We'll use **purity** for this purpose.

Breakpoint for a numerical attribute

- Why use purity for splitting a tree?
  - The reason purity concept is used in a decision tree is because the **ultimate goal** is to **group similar observations into similar classes**, i.e., to tidy the data.
  - Classification error doesn’t always lead to the best long-term growth of the tree.
- How to split a numerical attribute?
  - Find all the breakpoints where the class labels associated with them change and pick the one that minimizes the purity measure. This leads us to the concepts of "**Gini index (impurity)**" and "**Information Gain**" which are also the fundamental methods for attribute selection in decision trees.

Gini index

- **Gini index** (or Gini impurity index) is a **criterion to minimize the probability of misclassification**
- If a data set S contains a set of possible examples from N classes, the Gini index G is defined as:

$$G(S) = 1 - \sum_{i=1}^{N} p_i^{\,2}$$ (want to minimize)

where ***p**$_i$* is the relative frequency (probability) of class "***i***" in ***S***

- After splitting S into two subsets, S_1 and S_2 with sizes N_1 and N_2, the Gini index of the split is computed as:

$$G_{split}(S) = \frac{N_1}{N} G(S_1) + \frac{N_2}{N} G(S_2)$$

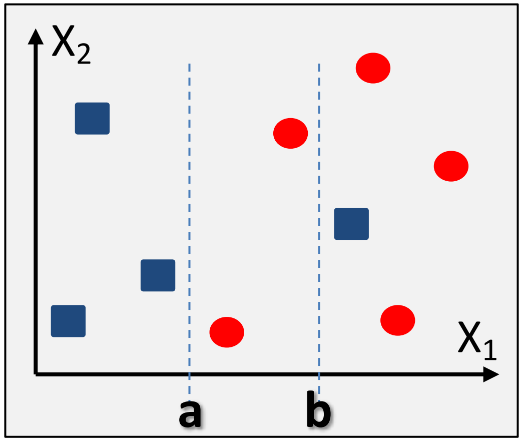
which corresponds to the **weighted average of each branch index**

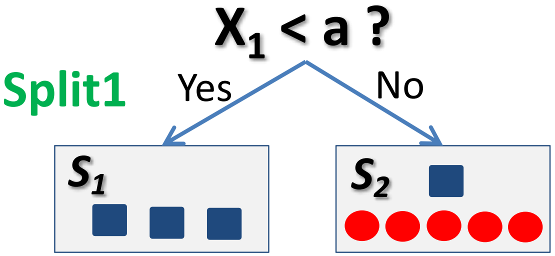
- The attribute providing the smallest Gini_{split}(S) is chosen to split the node

Gini index – example

$$\boxed{G(S) = 1 - \sum_{i=1}^{N} p_i^{\,2}}$$

$$G_{split}(S) = \frac{N_1}{N} G(S_1) + \frac{N_2}{N} G(S_2)$$





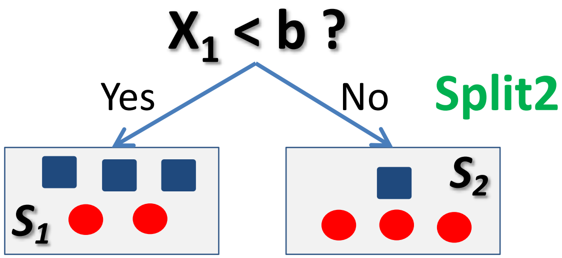
(3■ , 0●)   (1■ , 5●)

$$G(S_1) = 1 - \sum_{i=1}^{N} p_i^{\,2} = 1 - \left(\tfrac{3}{3}\right)^2 - \left(\tfrac{0}{3}\right)^2 = 0$$

$$G(S_2) = 1 - \sum_{i=1}^{N} p_i^{\,2} = 1 - \left(\tfrac{1}{6}\right)^2 - \left(\tfrac{5}{6}\right)^2 = 0.28$$

$$G_{Split1}(S) = \frac{N_1}{N} G(S_1) + \frac{N_2}{N} G(S_2)$$

$$G_{Split1}(S) = \frac{3}{9}\text{x}0 + \frac{6}{9}\text{x}0.28 = 0.19 \; ✓$$



(3■ , 2●)   (1■ , 3●)

$$G(S_1) = 1 - \sum_{i=1}^{N} p_i^{\,2} = 1 - \left(\tfrac{3}{5}\right)^2 - \left(\tfrac{2}{5}\right)^2 = 0.48$$

$$G(S_2) = 1 - \sum_{i=1}^{N} p_i^{\,2} = 1 - \left(\tfrac{1}{4}\right)^2 - \left(\tfrac{3}{4}\right)^2 = 0.38$$

$$G_{Split2}(S) = \frac{N_1}{N} G(S_1) + \frac{N_2}{N} G(S_2)$$

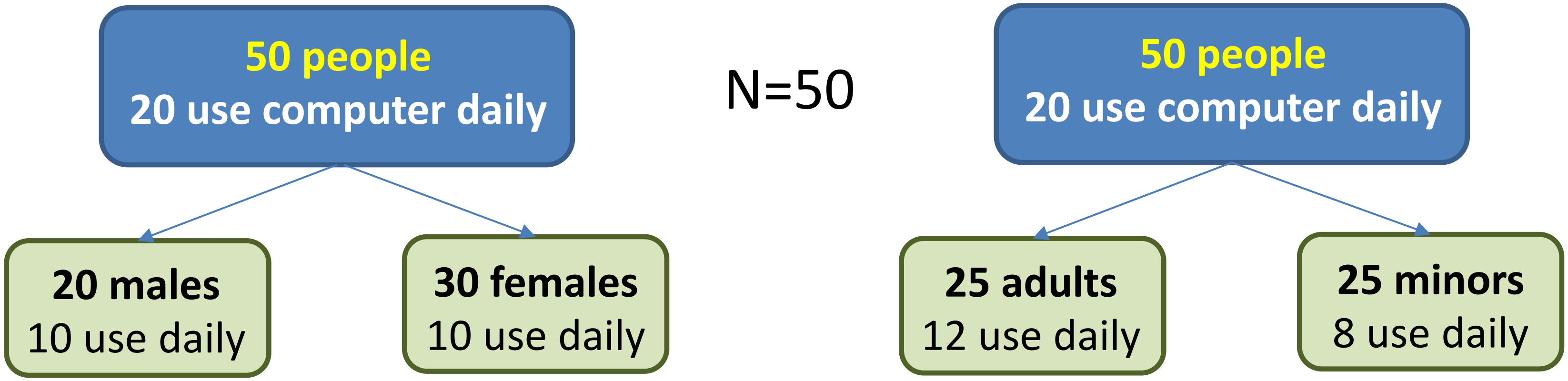
$$G_{Split2}(S) = \frac{5}{9}\text{x}0.48 + \frac{4}{9}\text{x}0.38 = 0.44$$

We choose Split1 over Split2 based on Gini index.

Gini index – example

SPLITTING ON GENDER

SPLITTING ON AGE



50 people
20 use computer daily

N=50

50 people
20 use computer daily

20 males
10 use daily

30 females
10 use daily

25 adults
12 use daily

25 minors
8 use daily

$p_1=10/20=0.5$
$p_2=10/20=0.5$
$\mathbf{G(S_1)=0.500}$

$p_1=10/30=0.33$
$p_2=20/30=0.67$
$\mathbf{G(S_2)=0.442}$

$p_1=12/25=0.48$
$p_2=13/25=0.42$
$\mathbf{G(S_1)=0.593}$

$p_1 = 8/25=0.32$
$p_2=17/25=0.68$
$\mathbf{G(S_2)=0.435}$

$$G(S)=1-\sum_{i=1}^{N} p_i^{\,2}$$

$$G_{split}(S)=\frac{N_1}{N}G(S_1)+\frac{N_2}{N}G(S_2)$$

$$G_{gender}(S)=\frac{20}{50}0.5+\frac{30}{50}0.442=0.465$$ ✓

$$G_{age}(S)=\frac{25}{50}0.593+\frac{25}{50}0.435=0.514$$

Based on Gini index, we choose 'GENDER' over 'AGE' to split on

Information Gain

- "Information Gain" is a common splitting criterion and it's based on a purity measure called "**Entropy**"

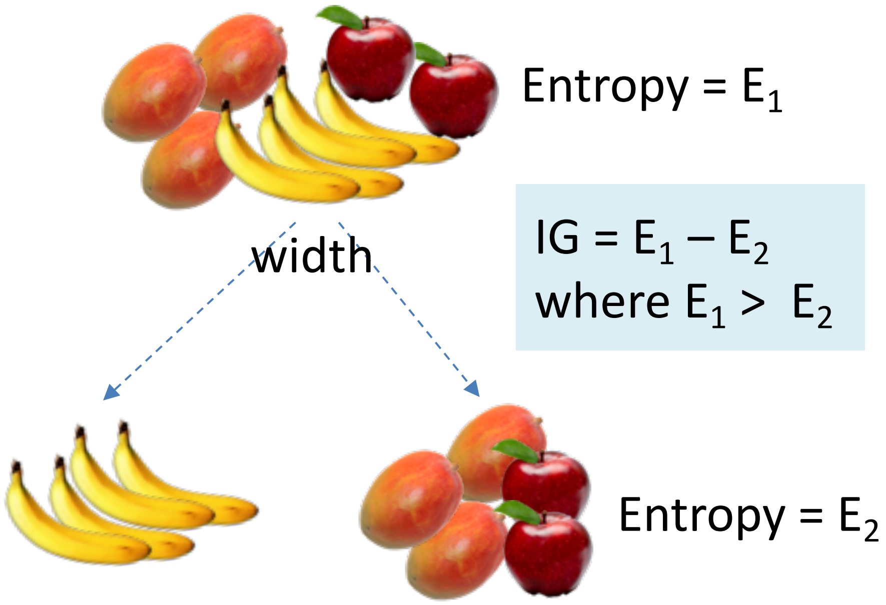
A random (disordered) dataset

→ High Entropy

A less random dataset

→ Less Entropy

- **Information Gain (IG)**: Measure of decrease in entropy after the split. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the purist branches)



IG = $E_1 - E_2$
where $E_1 > E_2$

Information Gain – cont'd

- How well each attribute splits a dataset into subsets with respect to a chosen target variable is measured by "purity".
- The most common splitting criterion is "**Information Gain**" based on a purity measure called "**Entropy**".
- **Entropy** is a probabilistic measure of uncertainty:

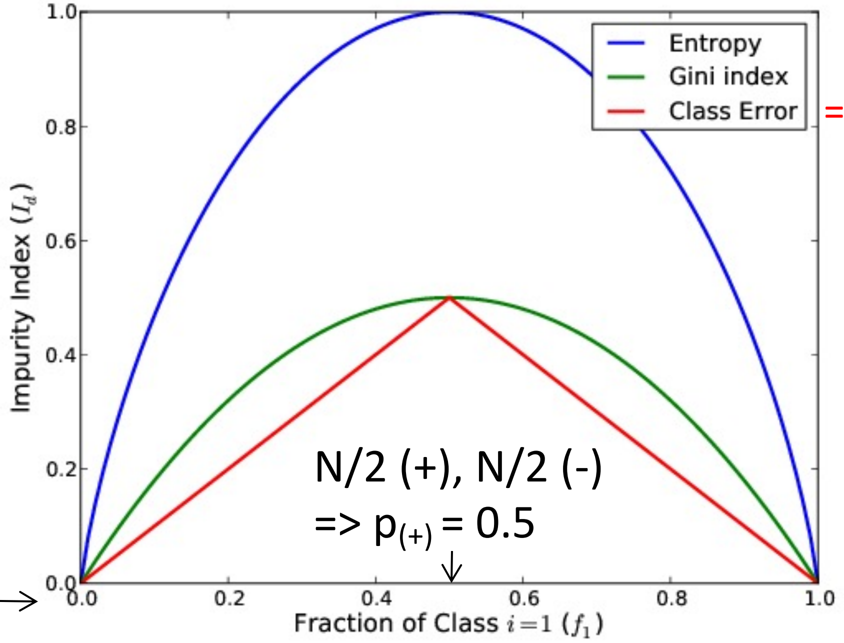
  More uncertainty => More entropy (deviation from purity)
- When we have a set of possible events in a dataset S as p_1, p_2, ... ,$p_{N,}$ we compute the **entropy** "H" to assess the uncertainty of the outcome as:

$$H(S) = -\sum_{i=1}^{N} p_i \log_2(p_i)$$

Information Gain – cont'd

- Entropy of a group in which all examples belong to the same class: $H(S) = -1\log_2(1) = 0$ (perfect purity, no uncertainty)
- Entropy of a group with 50% in either class: $H(S) = -0.5\log_2(0.5) - 0.5\log_2(0.5) = 1$ (maximum uncertainty)
- So, entropy attempts to maximize the mutual info in the DT by constructing an equal probability node
- Similar to entropy, the Gini index is maximal if the classes are perfectly mixed (a binary class):

$$G(S) = 1 - (p_1^{\ 2} + p_2^{\ 2})$$
$$= 1 - (0.5^2 + 0.5^2)$$
$$= 0.5$$



=1 – max {p_i}

N (+), 0 (-)
=> $p_{(+)}$ = 1

N/2 (+), N/2 (-)
=> $p_{(+)}$ = 0.5

0 (+), N (-)
=> $p_{(+)}$= 0

Information Gain – cont'd

- Using a decision algorithm, we start at the tree root and split the data on the feature that results in the largest **Information Gain (IG)** defined as follows:

  IG(attributeA) = info(parentNode) – info(childNodes)

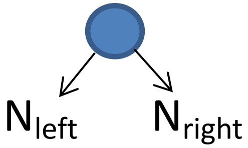
$$IG(S,A) = I(S) - \underbrace{\sum_{V \in Values(A)} \frac{N_V}{N_S} I(S_v)}_{\text{Expected value of the Entropy after S is partitioned using A}}$$

Entropy of the original collection before the split

S : Set of examples
V : Possible values of attribute A
S$_V$: Subset of S for which the attribute A has value V
N : Number of data points

- This tells us how much entropy of the training set we are reducing by selecting a particular feature/split

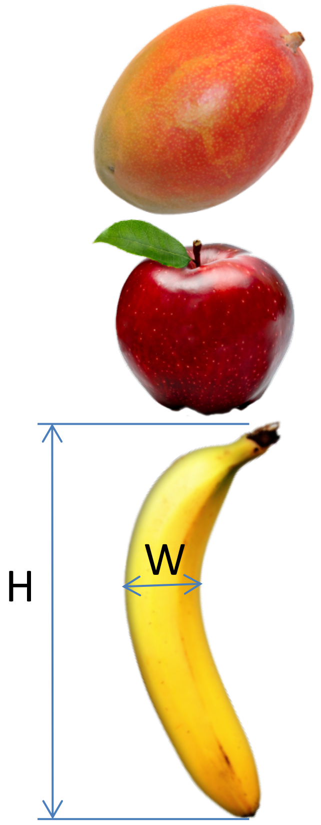
$$IG(S) = I(S) - \frac{N_{left}}{N} I(S_{left}) - \frac{N_{right}}{N} I(S_{right})$$



S, S_{left} and S_{right}: Data set of the parent, left child node and right child node where N_i is the number of data points in each branch

Entropy in action

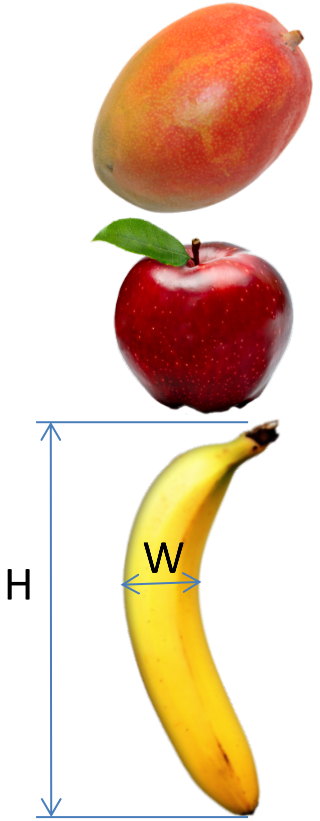
- Let’s create a DT for the example below:
  - – 2 features width, height
  - – 3 classes mango, apple, banana
  - – Data:



ID	Height (cm)	Width (cm)	Class
1	9	6	Mango
2	6	5.3	Apple
3	18	3.2	Banana
4	6.7	6.2	Apple
5	20	4	Banana
6	15	3.1	Banana
7	9	7.5	Apple
8	11.5	7.5	Mango
9	11	2.5	Banana
10	13	9	Mango

Entropy in action

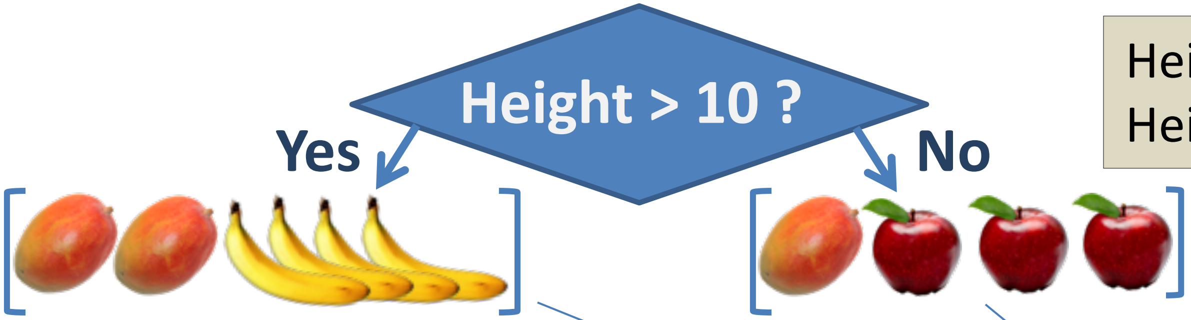
- Let’s create a DT for the example below:
  - 2 features width, height
  - 3 classes mango, apple, banana
  - Data:



ID	Height (cm)	Width (cm)	Class
1	9	7.5	Apple
2	6	5.3	Apple
3	6.7	6.2	Apple
4	9	6	Mango
5	11.5	7.5	Mango
6	13	9	Mango
7	18	3.2	Banana
8	15	3.1	Banana
9	11	2.5	Banana
10	20	4	Banana

Entropy in action – cont’d

- Let’s take the “height” with a threshold of “10 cm”



Height > 10 => Long
Height ≤ 10 => Short

$$I(S) = -\sum_{i=1}^{N} p_i \log_2(p_i)$$

entire dataset

$$IG(S) = I(S) - \frac{N_{left}}{N} I(S_{left}) - \frac{N_{right}}{N} I(S_{right})$$

$$IG(\textbf{Height}) = I\overset{\text{B M A}}{(4,3,3)} - \frac{6}{10} I\overset{\text{B M A}}{(4,2,0)} - \frac{4}{10} I\overset{\text{B M A}}{(0,1,3)}$$

$$I(4,3,3) = -(4/10)\log(4/10) - (3/10)\log(3/10) - (3/10)\log(3/10) = 1.571$$

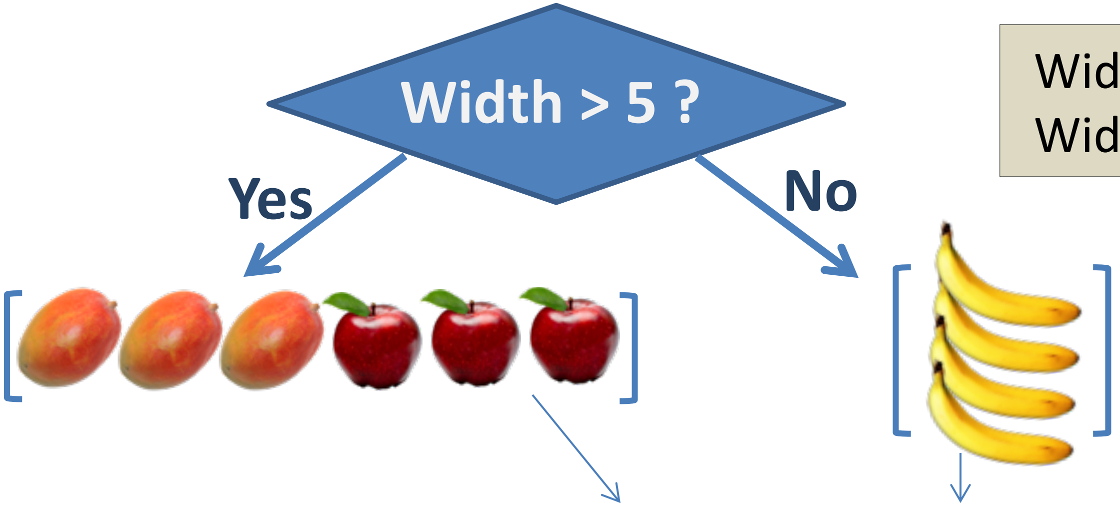
$$I(4,2,0) = -(4/6)\log(4/6) - (2/6)\log(2/6) - 0 = 0.918$$

$$I(0,1,3) = 0 - (1/4)\log(1/4) - (3/4)\log(3/4) = 0.811$$

$$IG(\textbf{Height}) = 1.571 - \frac{6}{10}0.918 - \frac{4}{10}0.811 = 0.696$$

Entropy in action – cont’d

- Now we try the “Width” with a threshold of “5”



Width > 5.0 => Wide
Width ≤ 5.0 => Narrow

$$IG(S) = I(S) \quad - \frac{N_{left}}{N} I(S_{left}) - \frac{N_{right}}{N} I(S_{right})$$

$$IG(\textbf{Width}) = \overset{\text{B M A}}{I(4,3,3)} - \frac{6}{10} \overset{\text{B M A}}{I(0,3,3)} - \frac{4}{10} \overset{\text{B M A}}{I(4,0,0)}$$

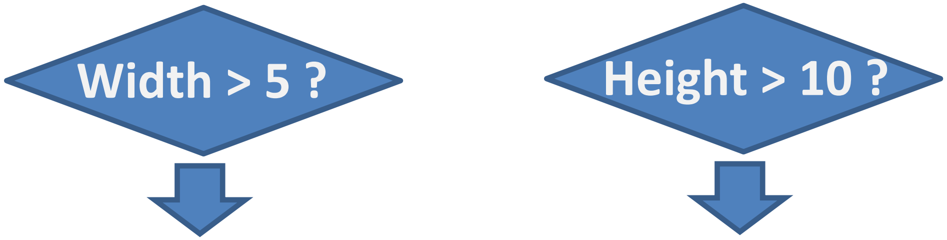
I(4,3,3) computed before

$$I(0,3,3) = 0 - (3/6)\log(3/6) - (3/6)\log(3/6) = 1.0$$

$$I(4,0,0) = (4/4)\log(4/4) - 0 - 0 = 0.0$$

$$IG(\textbf{Width}) = 1.571 - \frac{6}{10} 1.0 - \frac{4}{10} 0.0 = 0.971$$

Entropy in action – cont’d

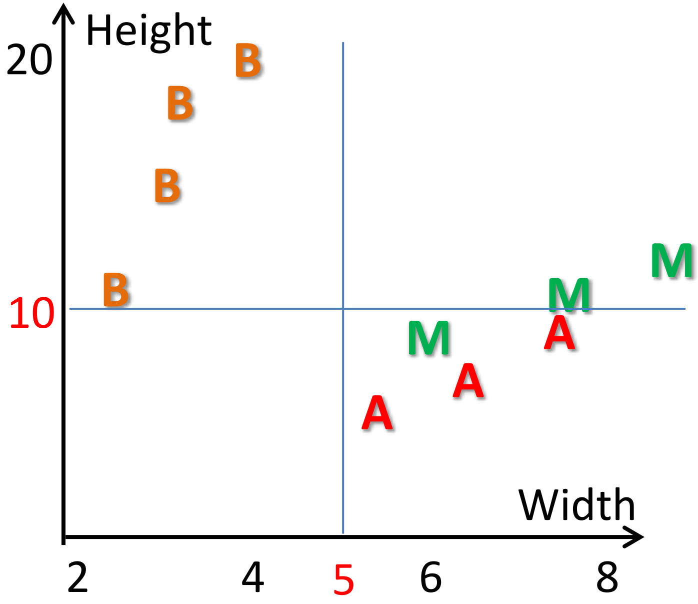


Width > 5 ?

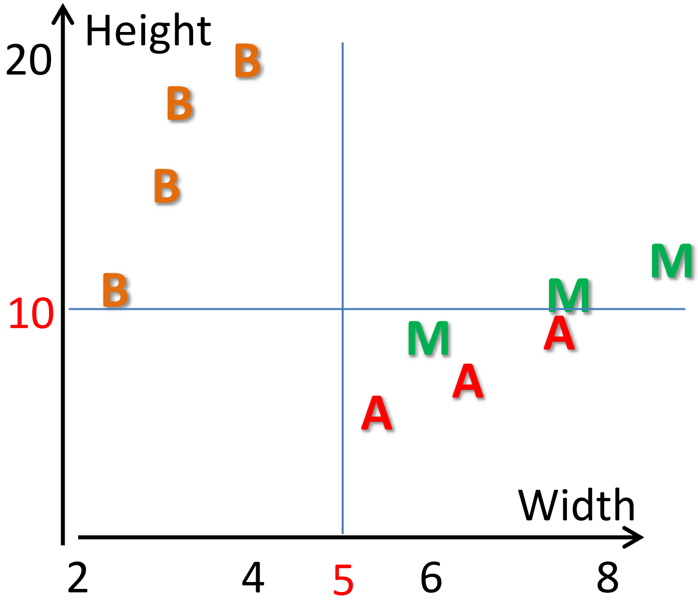
Height > 10 ?

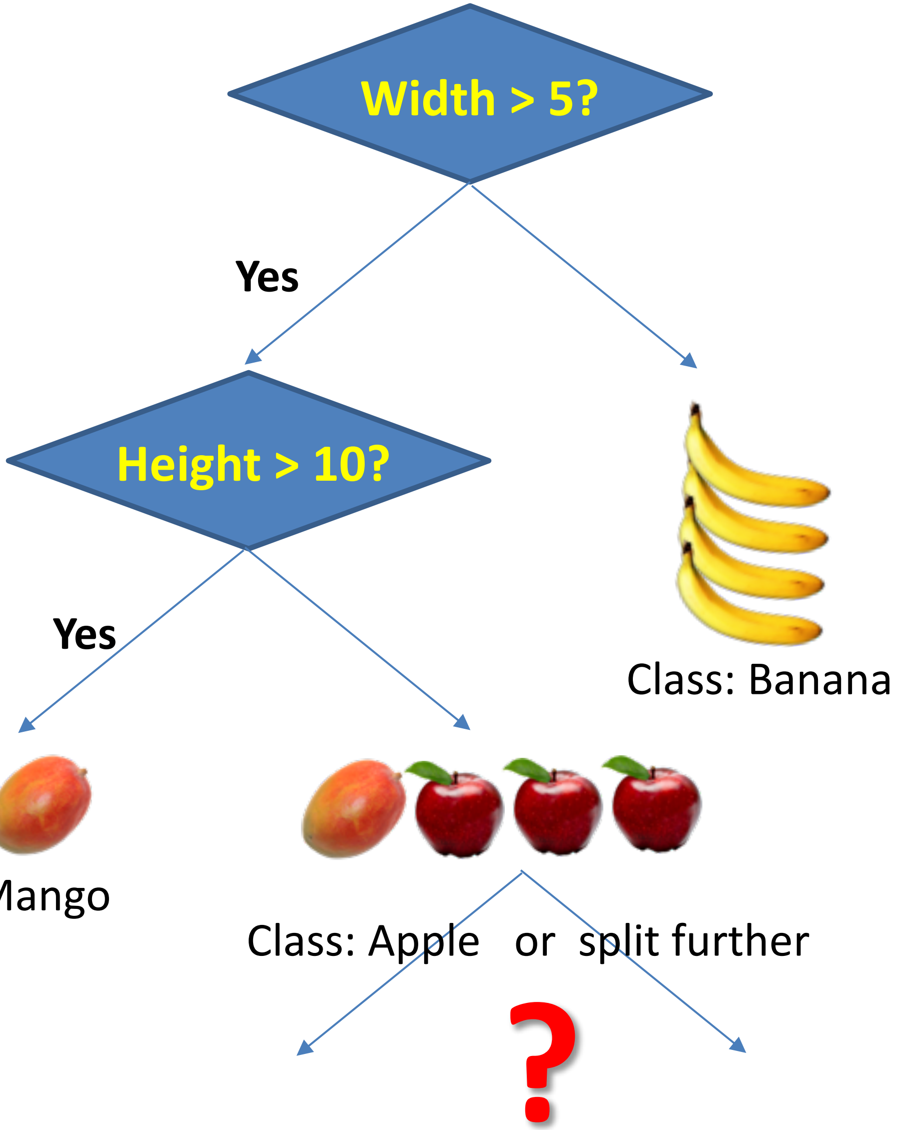
Information Gain: **0.971** **0.696**

- Lower entropy => Higher gain
- Feature “Width of 5.0” creates a better gain
- Now we can proceed with the left branch of “Width > 5.0?” and split it further
- **Question:** How do we know that splitting with another threshold (for either feature) won’t yield a better solution (higher gain)?



Entropy in action – cont'd





Width > 5?

Yes

Height > 10?

Yes

Class: Banana

Class: Mango

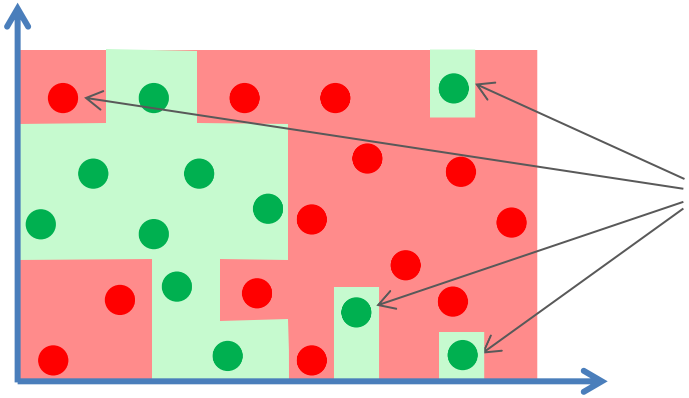
Class: Apple or split further

?

When do we stop growing the tree?

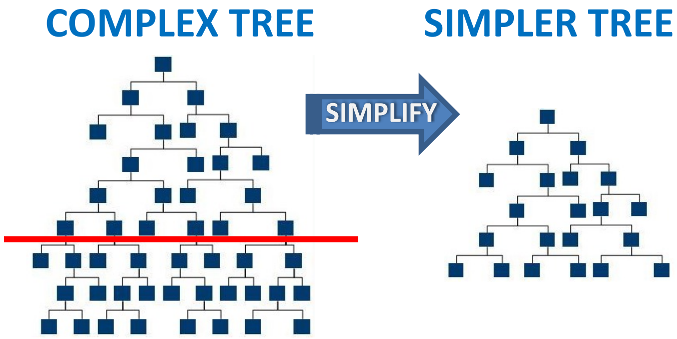
- You can stop your Decision Tree when:
  - your node contains only 1 class (perfect purity)
  - node contains less than N data points
  - max. depth is reached (number of splits allowed)
  - node purity is sufficient
  - you start to overfit
- Perfect classification on training examples is always possible for decision trees which leads to overfitting.
- Decision trees are very flexible. As the number of nodes increase, they can accommodate arbitrarily complex decision boundaries (zero training error) resulting in **overfitting**. This happens when you keep splitting the data until you get all singletons (all pure subsets).

Overfitting in DT's



These individual data points are probably noise in the dataset, but we capture these sample points by overfitting the tree

- How to avoid overfitting: Don't grow a tree that's too large
  - **Stopping early** (stop growing the tree when a statistically significant split is achieved – assume a threshold)
  - **Pruning** (start pruning the nodes by monitoring the validation error, then remove those that most improve it)



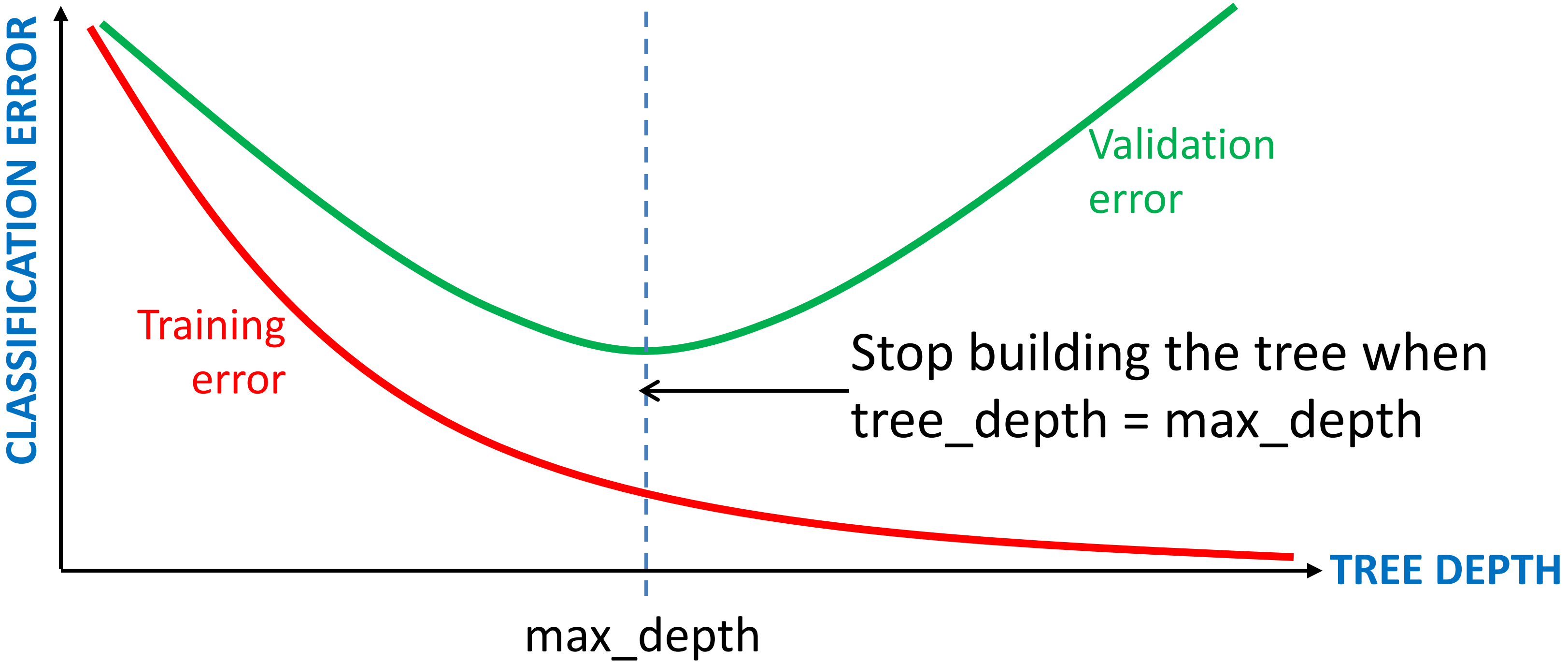
William of Occam, 13th Cent. Occam's razor: "All things being equal, the simplest solution tends to be the best one"

Prevent overfitting – Stopping early

- **Stopping early** : aka Pre-pruning the tree

1. Limit the **Tree Depth**

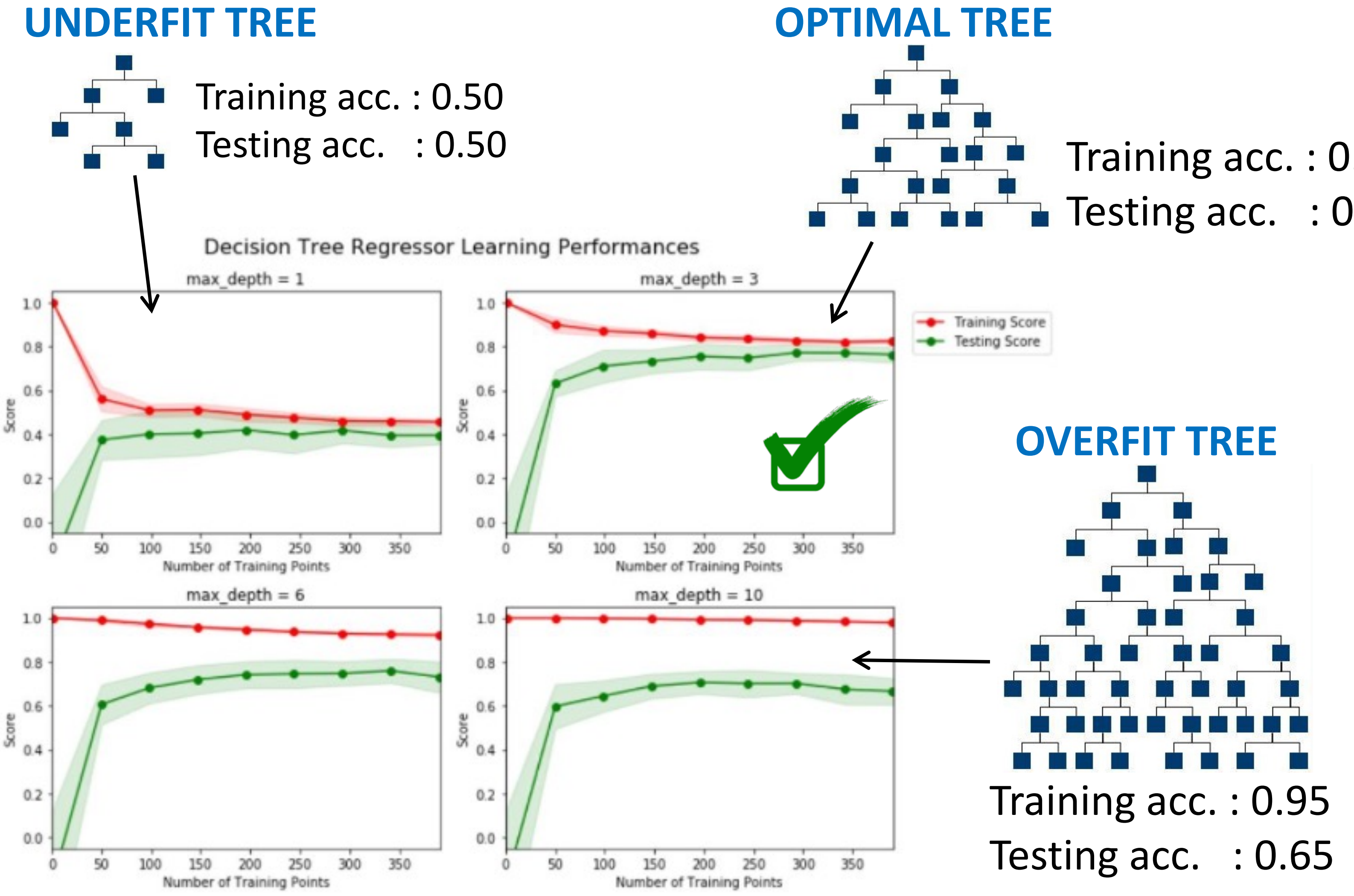
- We can use cross validation to detect the point where the validation error is minimum with respect to the model complexity (max tree depth)



Stop building the tree when tree_depth = max_depth

Prevent overfitting – Stopping early

- Testing and validation accuracies for different tree depths:



UNDERFIT TREE

Training acc. : 0.50
Testing acc. : 0.50

OPTIMAL TREE

Training acc. : 0.85
Testing acc. : 0.80

OVERFIT TREE

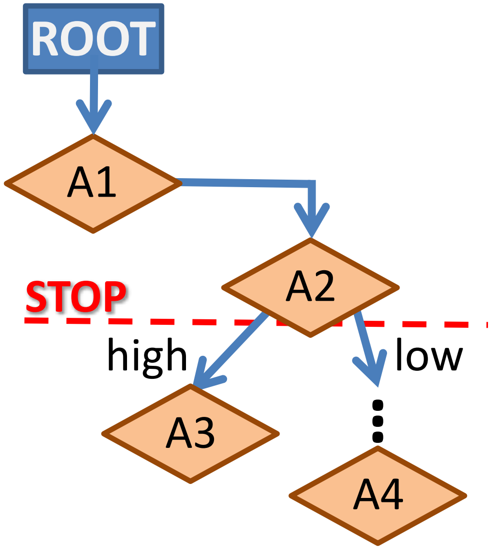
Training acc. : 0.95
Testing acc. : 0.65

Ref: https://www.linkedin.com/pulse/how-does-id3-algorithm-works-decision-trees-sagarnil-das/

Prevent overfitting – Stopping early

2. Use **Statistical Significance**

- Keep splitting until splitting further doesn't improve the accuracy

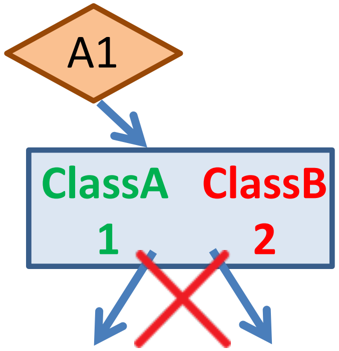


Split @A2	Classifica-tion error
(no split)	0.28
Split on A3	0.28
Split on A4	0.28

Use a threshold ε : Stop if the error does not improve more than, say 1% (ε=0.1)

Caveat: Careful with this as you might completely miss out on "good" splits that may occur right after the "useless" splits

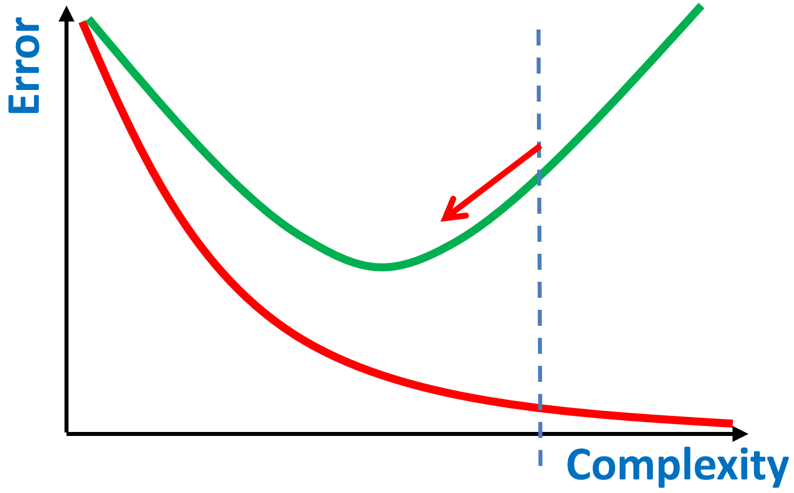
- Limit the **Number of Data Points in a Node**:
- Stop if the number of instances in a node becomes too small (when < N_{min}=10 to 100 depending on the size of the data set)



Doesn't make sense to split on such a small node!

Prevent overfitting – Pruning the tree

- **Tree pruning**: Identifying an removing the branches that reflect noise or outliers after the learning algorithm terminates (**post-pruning** the tree)



- We can go a little too far in the complexity of the tree built and then start backtracking until we get a reasonably simpler tree that has a better validation error.

Decision Trees – sklearn parameters

```
from sklearn.tree import DecisionTreeClassifier
Clf = DecisionTreeClassifier(max_depth= None,
                             max_features = None,
      min_samples_leaf = 2,min_samples_split = 1)
```

max_depth: This indicates how deep the tree can be. Deeper the tree, the more splits it has and captures more information about the data (def: None)

max_features: # of features to consider each time we look for the best split (defaults to None which uses all features). These features are selected randomly at each split. If you go deep enough, all features end up being used. Can be useful to reduce overfitting by reducing variance.

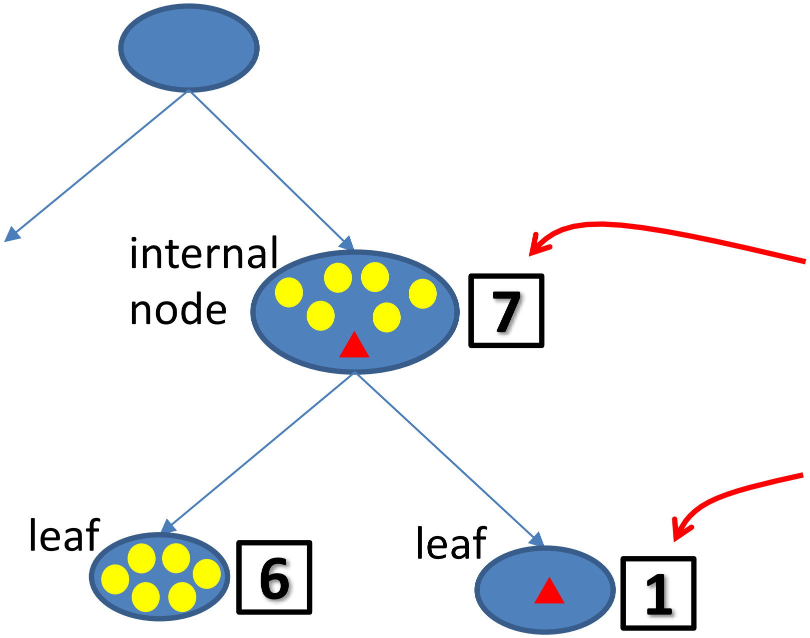
min_samples_split: minimum number of samples required to split an internal node. When we increase this parameter, the tree becomes more constrained as it has to consider more samples at each node (def: 2)

min_samples_leaf: minimum number of samples required to be at a leaf node. This parameter is similar to **min_samples_splits**, however, this describes the minimum number of samples at the leafs (def: 1)

Decision Trees w/ Python

- What's the difference between `min_samples_split` and `min_samples_leaf` ?
  - **min_samples_leaf** guarantees a minimum number of samples in a leaf, while **min_samples_split** can create arbitrary small leaves

```
Example:
min_samples_split = 5
min_samples_leaf  = 2
```



Continue splitting as 7 > min_samples_split

Less than min_samples_leaf, so the split will not be allowed (even if the internal node has 7 samples > 5)

Pros and Cons of DTs

- **Pros**
  - Observable, easy to understand & interpret (white box)
  - Compact (#nodes << D after pruning) and fast classification (O(depth))
  - Requires little data preparation
  - Able to handle both categorical & numerical data
  - Fast, robust and computationally cheap
  - Tolerance to irrelevant features
  - Some tolerance to correlational inputs
  - Performs well with large datasets (but generally poorly with many classes & small data – will yield interesting info even for small datasets)
  - Ability to deal with noisy or incomplete (missing) data
  - Outliers or linearly inseparable data are no problem

- **Cons**
  - Greedy (may not find the best tree)
  - Complex trees are hard to interpret
  - Can create over-complex trees that don't generalize well from the training data (can easily overfit – low bias/high variance)
  - Relatively less predictive in many cases (unbalanced data leads to biased trees)
  - Cannot work on linear combinations of features
  - Non-rectangular regions are not treated well

Decision Trees w/ Python

```
#using scikit-learn library
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

#md, mf, msl, mss and others are found via CV
Clf = DecisionTreeClassifier(max_depth=md,
                             max_features = mf,
                             min_samples_leaf = msl,
                             min_samples_split = mss)
Clf.fit(X_train, y_train)
accuracy_score(y_test, Clf.predict(X_test))
```