Data Mining & Machine Learning

Yong Zheng

Illinois Institute of Technology
Chicago, IL, 60616, USA

ILLINOIS TECH
College of Computing

Review

- Supervised vs Unsupervised Learning
- Classification Task
- Classification Algorithms
  - KNN
  - Naïve Bayes
  - Decision Tree
  - Logistic Regression
  - SVM
  - Neural Networks
  - Ensemble Methods

Decision Tree Learning

- Decision Tree Learning: Basics
- Decision Tree Learning: Feature Selection
- Decision Tree Learning: Overfitting and Pruning
- Extension: Regression Tree

Decision Tree Learning

- Decision Tree Learning: Basics
- Decision Tree Learning: Feature Selection
- Decision Tree Learning: Overfitting and Pruning
- Extension: Regression Tree

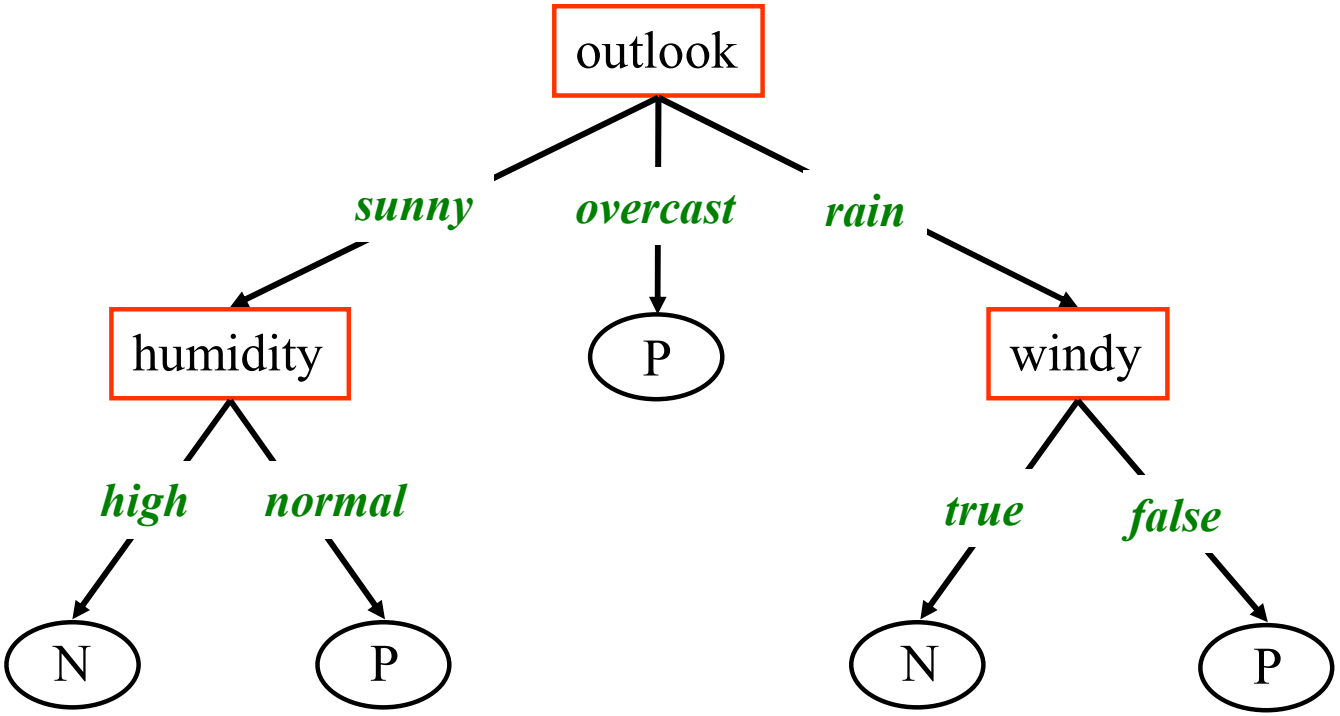
Decision Trees

- **A decision tree is a flow-chart-like tree structure**

Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Decision Trees

- **How it works?**
  - We learn and build a tree structure based on the training set
  - After that, we are able to make predictions based on the tree
  - Example, Is it good to play golf? *{sunny, windy, high humidity}*
  - Question: How to learn such a tree? There could be many possible trees



Decision Trees

- **Example: "is it a good day to play golf?"**
  - a set of attributes and their possible values:

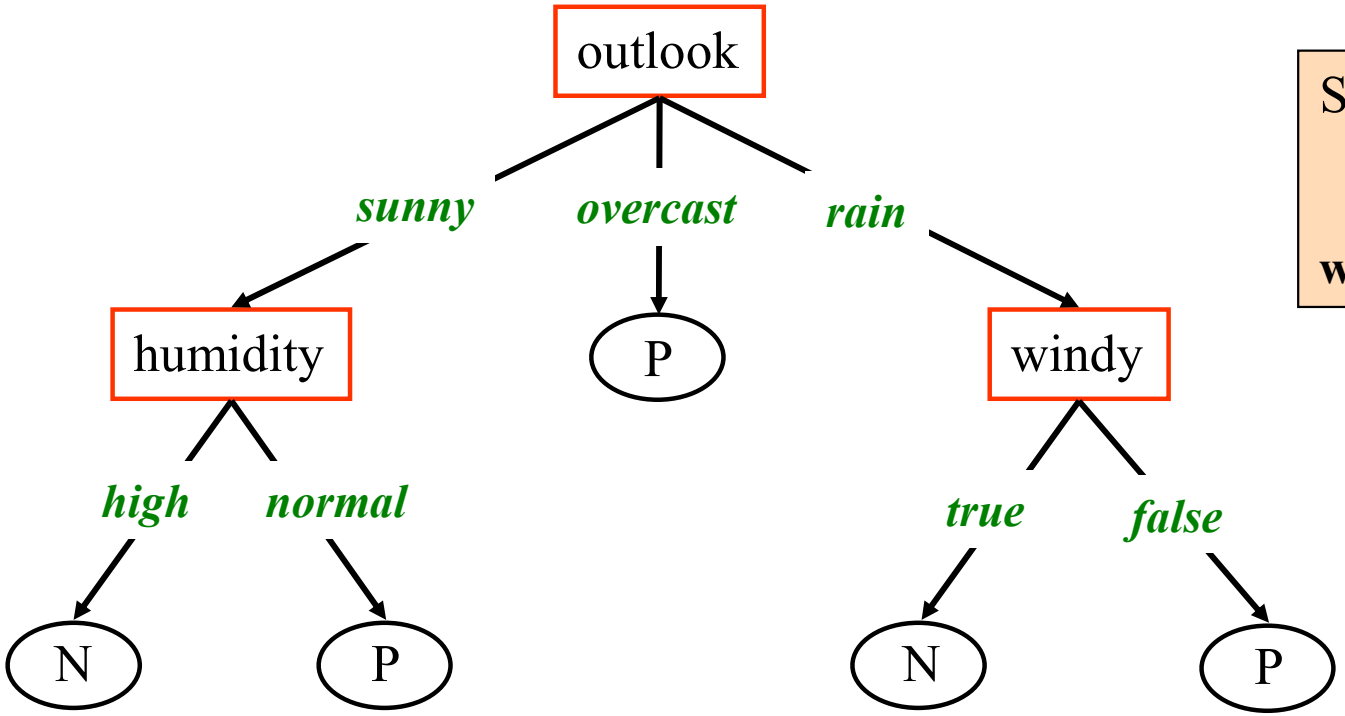
outlook	sunny, overcast, rain
temperature	cool, mild, hot
humidity	high, normal
windy	true, false

In this case, the target class is a binary attribute, so each instance represents a positive or a negative example.

Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Using Decision Trees for Classification

- **Example (a decision tree to decide whether to go on a picnic):**



So a new instance:

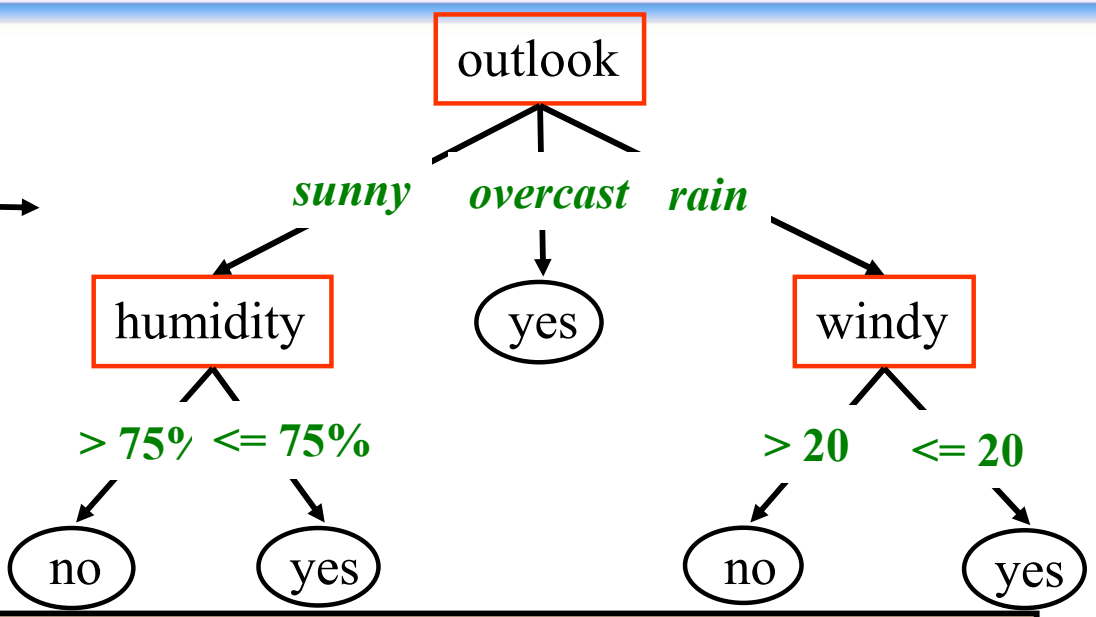
<rainy, hot, normal, true>: ?

will be classified as "noplay"

- Root node: the top of the tree, e.g. the node "Outlook"
- Parent and Children nodes: outlook as the parent, "humidity" and "windy" are children nodes
- Leaf node: P, N are the leaf nodes which do not have children and we can reach a leaf node to get the predictions

Decision Trees and Decision Rules

If attributes are continuous, it should be converted into a nominal variable



Each path in the tree represents a decision rule:

Rule1:
If (outlook="sunny") AND (humidity<=0.75)
Then (play="yes")

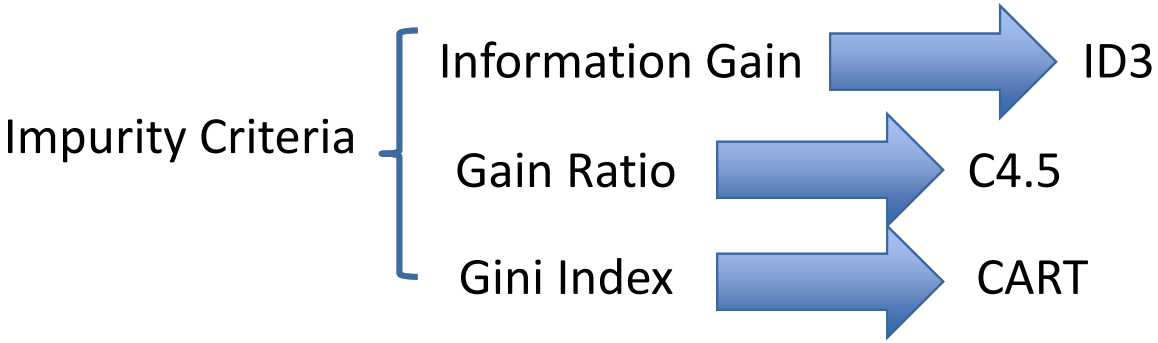
Rule2:
If (outlook="rainy") AND (wind>20)
Then (play="no")

Rule3:
If (outlook="overcast")
Then (play="yes")

• • •

Tree-Based Learning

- **Popular Tree-Based Learning Techniques**
  - **ID3**, or Iternative Dichotomizer, was the first of these three Decision Tree techniques implementations developed by Ross Quinlan (Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106.)
  - **C4.5**, Quinlan's next iteration. The new features (versus ID3) are: (i) accepts both continuous and discrete features; (ii) handles incomplete data points; (iii) solves over-fitting problem by (very clever) bottom-up technique usually known as "pruning"; and (iv) different weights can be applied the features that comprise the training data.
  - **CART**, or Classification And Regression Trees. The CART implementation is very similar to C4.5; the one notable difference is that CART constructs the tree based on a numerical splitting criterion recursively applied to the data.



Top-Down Decision Tree Generation

- The basic approach usually consists of two phases:
  - Tree construction
    - At the start, all the training examples are at the root
    - Partition examples are recursively based on selected attributes
  - Tree pruning
    - remove tree branches that may reflect noise in the training data and lead to errors when classifying test data
    - improve classification accuracy
- Basic Steps in Decision Tree Construction
  - Tree starts a single node representing all data
  - If sample are all same class then node becomes a leaf labeled with class label
  - Otherwise, ***select feature*** that best separates sample into individual classes.
  - Recursion stops when:
    - Samples in node belong to the same class (majority)
    - There are no remaining attributes on which to split

Decision Tree Learning

- Decision Tree Learning: Basics
- Decision Tree Learning: Feature Selection
- Decision Tree Learning: Overfitting and Pruning
- Extension: Regression Tree

Choosing the "Best" Feature

- Feature selection is the key component in decision trees: deciding what features of the data are relevant to the target class we want to predict.
- Popular impurity measures in decision tree learning
  - Information Gain: Used in ID3. We introduce it on the next page
  - Gain Ratio: improvement over information gain. It is used in C4.5
  - Gini Index: Used in CART. It is a measure of how often a randomly chosen element from the set would be incorrectly labelled if it was randomly labelled according to the distribution of labels in the subset.

Understand Entropy & Information Gain

- The decision tree is built in a top-down fashion, but the question is how do you choose which attribute to split at each node? The answer is find the feature that best splits the target class into the purest possible children nodes (ie: nodes that don't contain a mix of both male and female, rather pure nodes with only one class).
- For instance, in our previous example on recognition of tigers and lions, we have features like stripes, weights, size, color, etc. But, you may notice that, “stripes” is the key feature to distinguish tigers and lions!

Understand Entropy & Information Gain

- This measure of information is called purity.
- Entropy is a measure of impurity. Information Gain which is the difference between the entropies before and after.
- Information_Gain = Entropy_before split - Entropy_after split
- Entropy_before split = Impurity before the split
- Entryopy_after split = Impurtiy after the split
- The larger an information gain value (by a feature) is, the feature should be used to split the instances as the “best” node.

Trees Construction Algorithm (ID3)

- Decision Tree Learning Method (ID3)
  - Input: a set of training examples ***S***, a set of features ***F***
  - 1. If every element of ***S*** has a class value "yes", return "yes"; if every element of ***S*** has class value "no", return "no"
  - 2. Otherwise, choose the **best feature** ***f*** from ***F*** (if there are no features remaining, then return failure);
  - 3. Extend tree from ***f*** by adding a new branch for each attribute value of ***f***
    - 3.1. Set ***F'*** = ***F*** – {***f***},
  - 4. Distribute training examples to leaf nodes (so each leaf node ***n*** represents the subset of examples S_n of ***S*** with the corresponding attribute value
  - 5. Repeat steps 1-5 for each leaf node ***n*** with S_n as the new set of training examples and F' as the set of attributes until we finally label all the leaf nodes
- Main Question:
  - how do we choose the best feature at each step?

Note: ID3 algorithm only deals with categorical attributes, but can be extended (as in C4.5) to handle continuous attributes

Choosing the "Best" Feature

- Feature selection is the key component in decision trees: deciding what features of the data are relevant to the target class we want to predict.
- Use Information Gain to find the "best" (most discriminating) feature
- Assume there are two classes, *P* and *N* (e.g, P = "yes" and N = "no")
  - Let the set of instances S (training data) contains p elements of class P and n elements of class N
  - The amount of information, needed to decide if an arbitrary example in S belongs to P or N is defined in terms of entropy, I(p,n):

$$I(p,n) = -\Pr(P)\log_2 \Pr(P) - \Pr(N)\log_2 \Pr(N)$$

  - Note that Pr(P) = p / (p+n) and Pr(N) = n / (p+n)

Choosing the "Best" Feature

- More generally, if we have *m* classes, and s_1, s_2, …, s_m are the number of instances of *S* in each class, then the entropy is:

$$I(s_1, s_2, \cdots, s_m) = -\sum_{i=1}^{m} p_i \log_2 p_i$$

- where p_i is the probability that an arbitrary instance belongs to the class $_i$.

Choosing the "Best" Feature

- Now, assume that using attribute A a set *S* of instances will be partitioned into sets $S_1, S_2, \ldots, S_v$ each corresponding to distinct values of attribute A.
  - **If S_i contains p_i cases of *P* and n_i cases of *N*, the entropy, or the expected information needed to classify objects in all subtrees S_i is**

$$E(A) = \sum_{i=1}^{v} \Pr(S_i) I(p_i, n_i)$$

where,

$$\Pr(S_i) = \frac{|S_i|}{|S|} = \frac{p_i + n_i}{p + n}$$

The probability that an arbitrary instance in S belongs to the partition S_i

- The encoding information that would be gained by branching on *A*:

$$Gain(S, A) = E(S) - E(A)$$

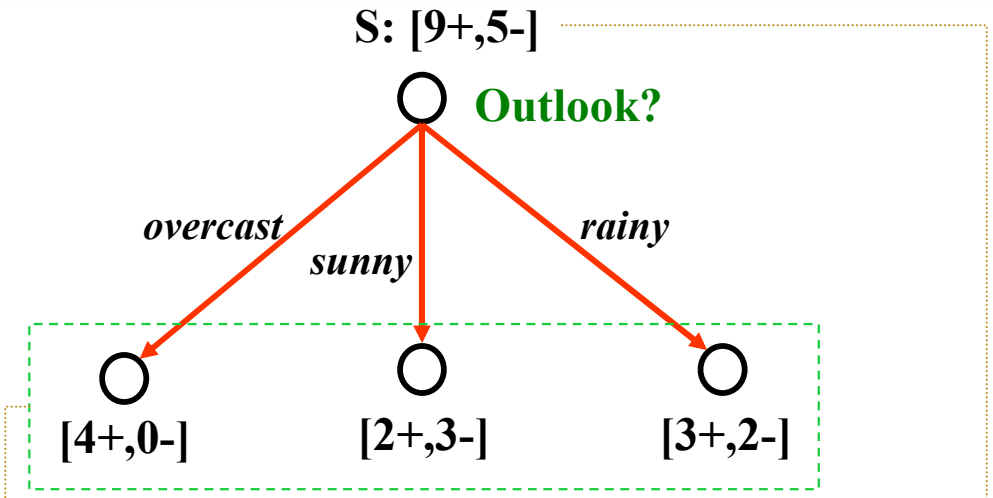
- At any point we want to branch using an attribute that provides the highest information gain.

We use attribute A to split node S
= Entropy before splitting – Entropy after splitting by using feature A

Attribute Selection - Example

- The "Golf" example: what attribute should we choose as the root?

Day	outlook	temp	humidity	wind	play
D1	sunny	hot	high	weak	No
D2	sunny	hot	high	strong	No
D3	overcast	hot	high	weak	Yes
D4	rain	mild	high	weak	Yes
D5	rain	cool	normal	weak	Yes
D6	rain	cool	normal	strong	No
D7	overcast	cool	normal	strong	Yes
D8	sunny	mild	high	weak	No
D9	sunny	cool	normal	weak	Yes
D10	rain	mild	normal	weak	Yes
D11	sunny	mild	normal	strong	Yes
D12	overcast	mild	high	strong	Yes
D13	overcast	hot	normal	weak	Yes
D14	rain	mild	high	strong	No



$I(9,5) = -(9/14).\log(9/14) - (5/14).\log(5/14)$
$= 0.94$

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

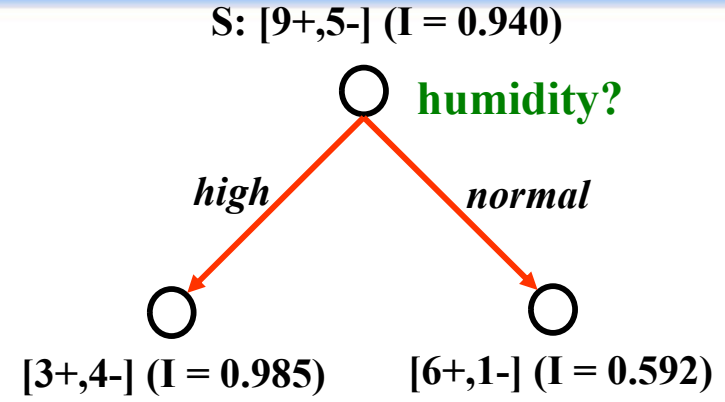
$I(2,3) = -(2/5).\log(2/5) - (3/5).\log(3/5)$
$= 0.97$

$I(3,2) = -(3/5).\log(3/5) - (2/5).\log(2/5)$
$= 0.97$

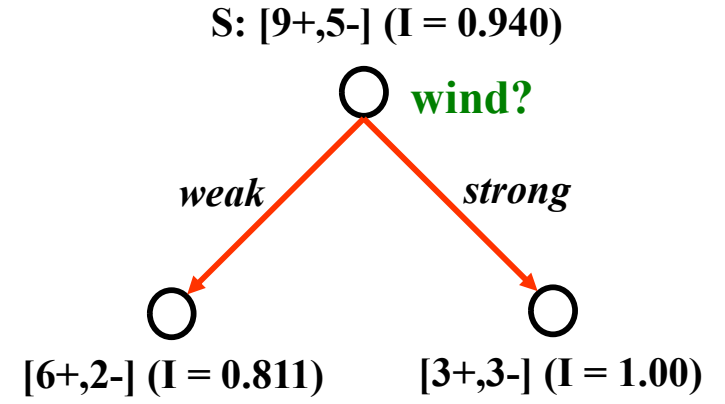
$Gain(outlook) = .94 - (4/14)*0 - (5/14)*.97 - (5/14)*.97$
$= .24$

Attribute Selection - Example (Cont.)

Day	outlook	temp	humidity	wind	play
D1	sunny	hot	high	weak	No
D2	sunny	hot	high	strong	No
D3	overcast	hot	high	weak	Yes
D4	rain	mild	high	weak	Yes
D5	rain	cool	normal	weak	Yes
D6	rain	cool	normal	strong	No
D7	overcast	cool	normal	strong	Yes
D8	sunny	mild	high	weak	No
D9	sunny	cool	normal	weak	Yes
D10	rain	mild	normal	weak	Yes
D11	sunny	mild	normal	strong	Yes
D12	overcast	mild	high	strong	Yes
D13	overcast	hot	normal	weak	Yes
D14	rain	mild	high	strong	No



Gain(humidity) = .940 - (7/14)*.985 - (7/14)*.592 = .151

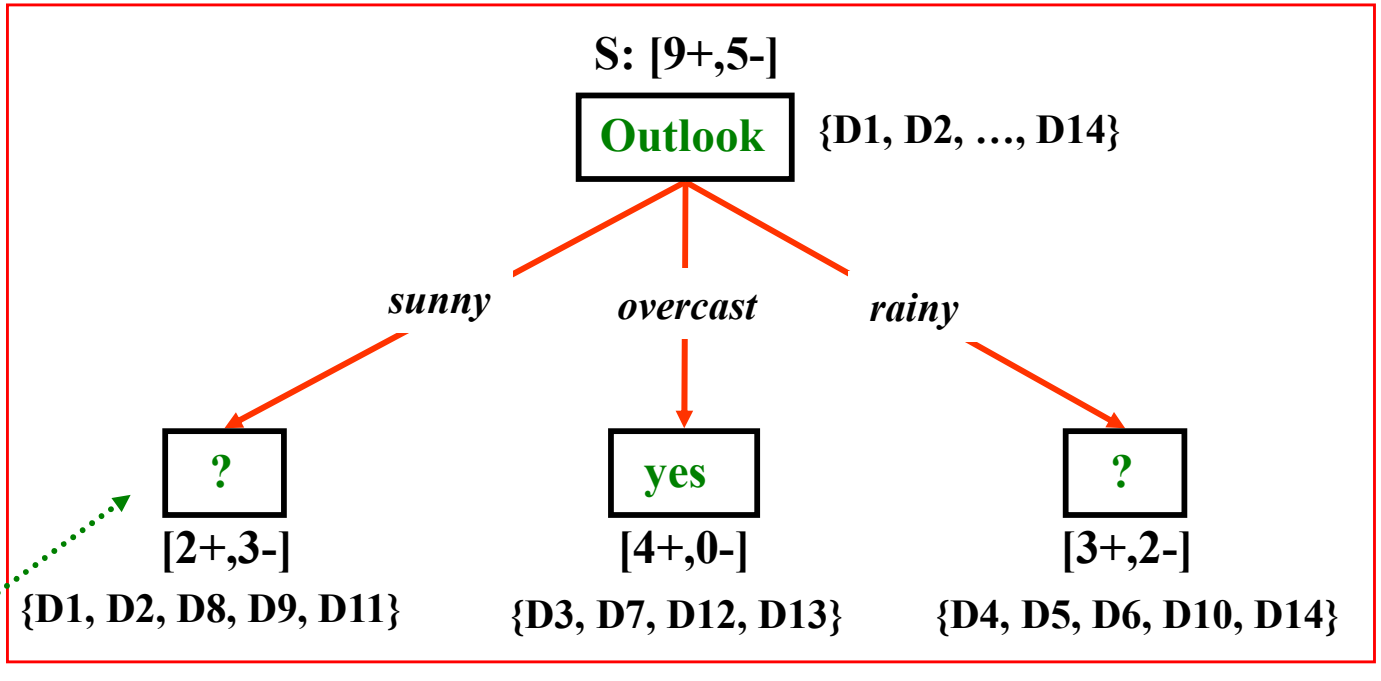


Gain(wind) = .940 - (8/14)*.811 - (8/14)*1.0 = .048

So, classifying examples by humidity provides more information gain than by wind. Similarly, we must find the information gain for "temp". In this case, however, you can verify that outlook has largest information gain, so it'll be selected as root

Attribute Selection - Example (Cont.)

- Partially learned decision tree; which attribute is the next?



S_{sunny} = {D1, D2, D8, D9, D11}

Gain(S_{sunny}, humidity) = .970 - (3/5)*0.0 - (2/5)*0.0 = .970

Gain(S_{sunny}, temp) = .970 - (2/5)*0.0 - (2/5)*1.0 - (1/5)*0.0 = .570

Gain(S_{sunny}, wind) = .970 - (2/5)*1.0 - (3/5)*.918 = .019

Other Criteria

- Information Entropy and Information Gain
  it is used in ID3
  Gain(S, A) = E(S) – E(A)
- Drawbacks of Information Gain
  - If an attribute has large number of values, it is more possible to be pure by using this attribute
  - IG will introduce biases for these attributes which have large number of values
  - The bias will further result in overfitting problem

Other Criteria

- Information Entropy and Information Gain

  it is used in ID3

  Gain(S, A) = E(S) – E(A)

- Gain Ratio

  It was first introduced to C4.5 classification

  Gain Ratio = Gain (S, A)/Info(A) $\text{Info}(A) = \sum_{i=1}^{y} \Pr(S_i) Log_2 \Pr(S_i)$

- Gini Index

  It was used in the CART algorithm

  https://en.wikipedia.org/wiki/Decision_tree_learning#Gini_impurity

Decision Trees

- Same questions in Decision Trees:

1). Any data requirements?

Categorical data can be used directly; numeric data may be transformed to categorical ones. Numeric data can be automatically utilized or transformed in C4.5

2). Is there a learning/optimization process

Yes, we are going to learn a tree structure.

3). Overfitting in DT?

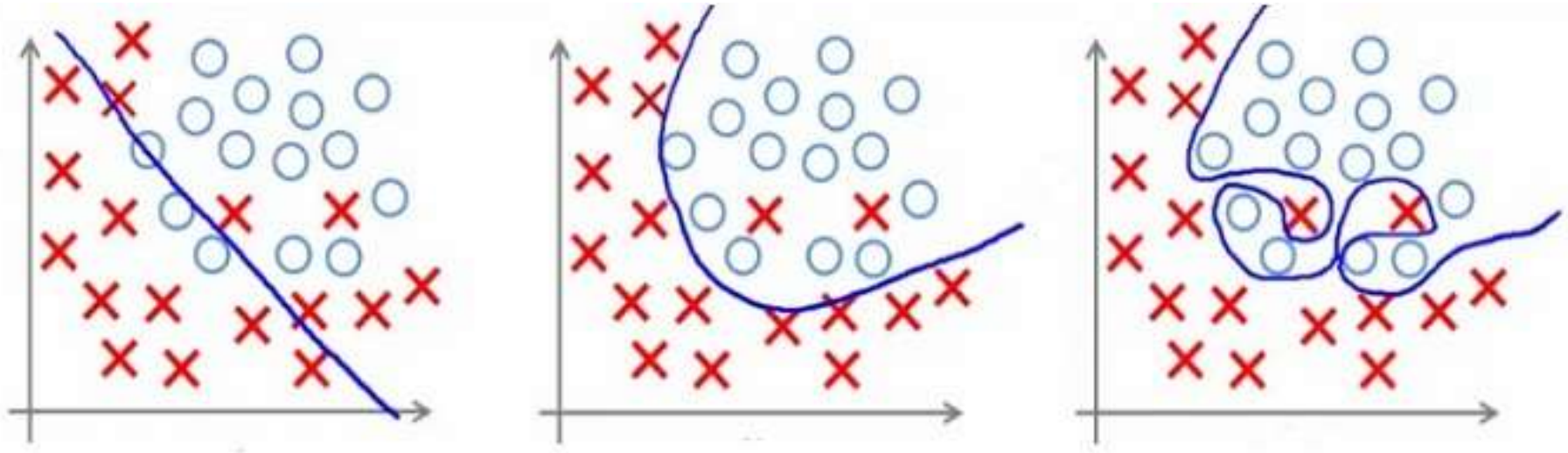
More details in the next page.

Decision Tree Learning

- Decision Tree Learning: Basics
- Decision Tree Learning: Feature Selection
- Decision Tree Learning: Overfitting and Pruning
- Extension: Regression Tree

Overfitting Problem

Problem: The model is over-trained by the training set; it may show a high accuracy on training set, but significantly worse performance on test set.



Under-fitting

(too simple to explain the variance)

Appropriate-fitting

Over-fitting

(forcefitting -- too good to be true)

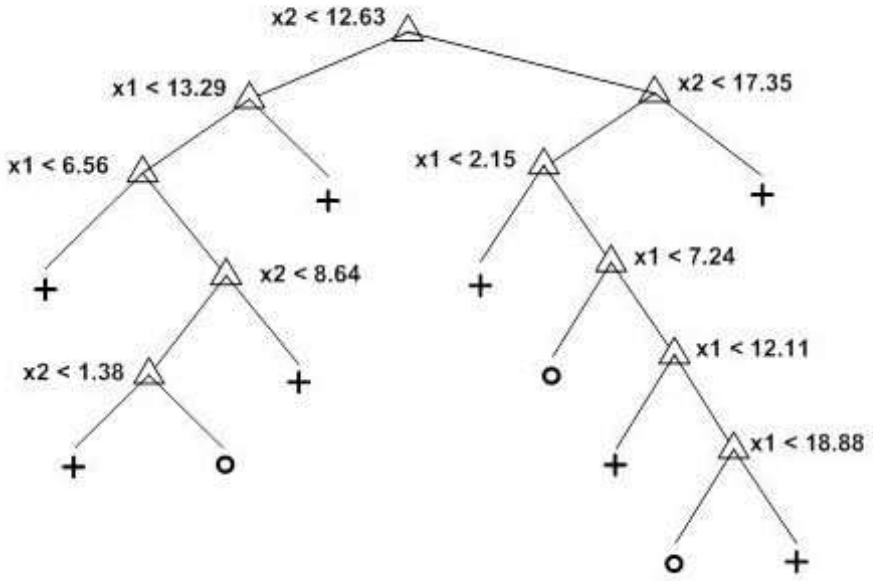
Overfitting Problem

Let's see an example

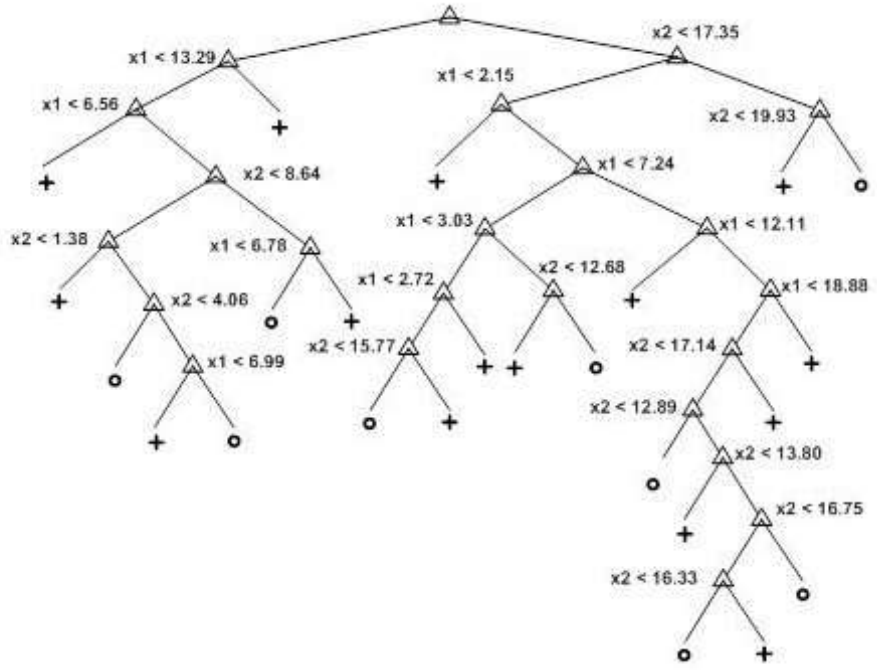
- You worked classifications on a data set. The data set is big, so you used hold-out evaluation. You build a model based on training, and evaluate the model based on the testing set. Finally, you get a 99% classification accuracy on the testing set. Is this an example of overfitting?
- How about N-fold cross validations?

Overfitting in Decision Trees

- Overfitting is a common problem in decision trees



Decision Tree with 11 leaf nodes

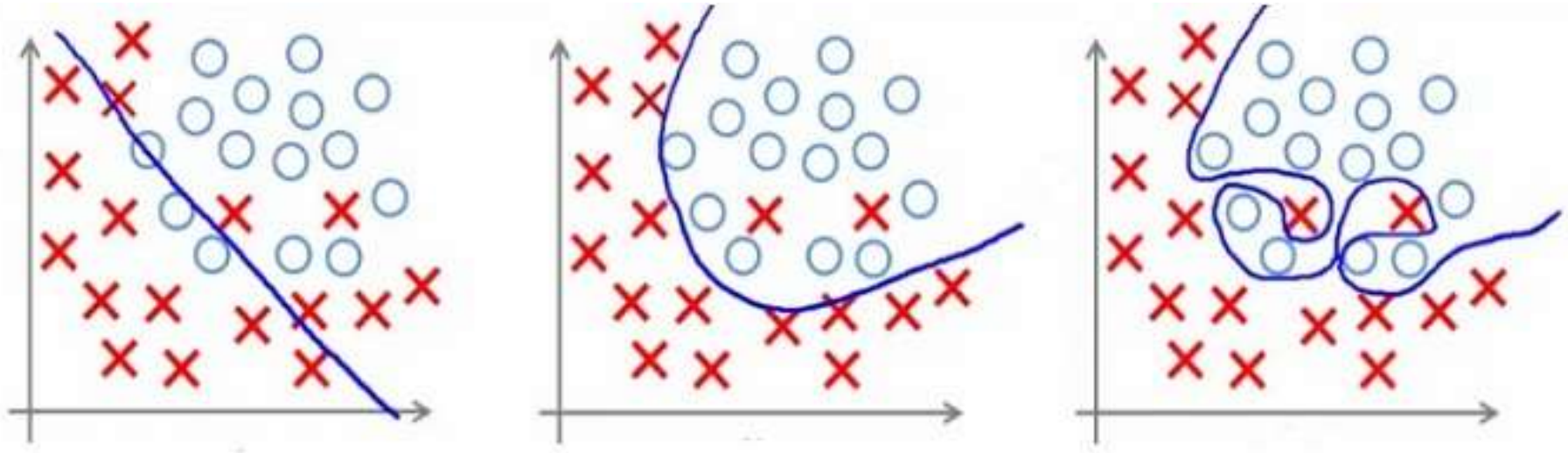


Decision Tree with 24 leaf nodes

Which tree is better?

Overfitting Problem

Problem: The model is over-trained by the training set; it may show a high accuracy on training set, but significantly worse performance on test set.



Under-fitting

(too simple to explain the variance)

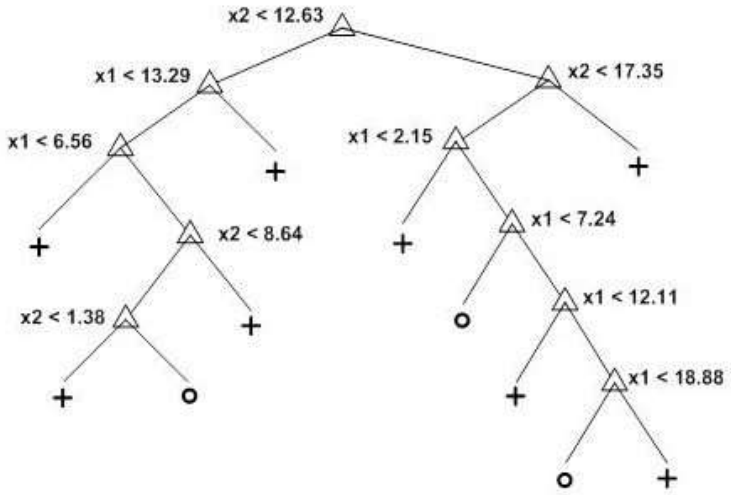
Appropriate-fitting

Over-fitting

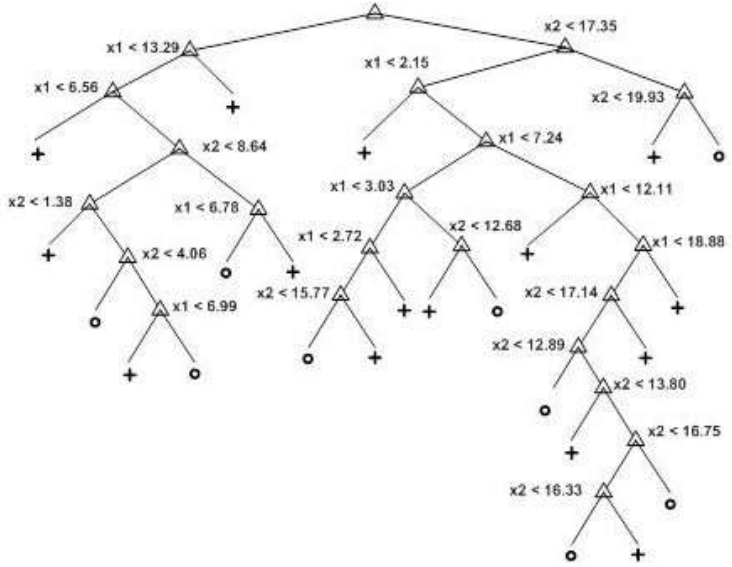
(forcefitting -- too good to be true)

Overfitting in Decision Trees

- Overfitting in DT: A tree may obtain good results on training, but bad on testing
- The tree may be too specific with many branches & leaf nodes



Decision Tree with 11 leaf nodes



Decision Tree with 24 leaf nodes

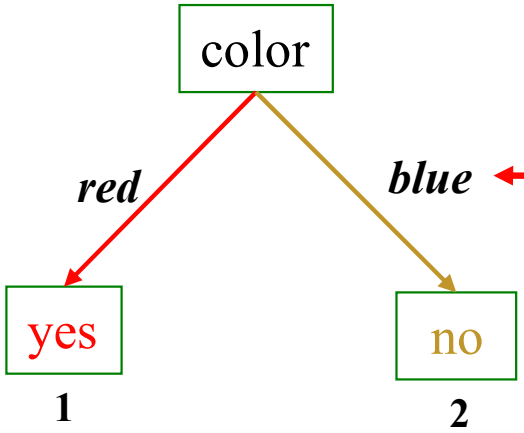
Which tree is better?

Solution: Tree Pruning

- A tree generated may over-fit the training examples due to noise or too small a set of training data
- Two approaches to alleviate over-fitting:
  - (Stop earlier): Stop growing the tree earlier
  - (Post-prune): Allow over-fit and then post-prune the tree
- Example of the Stop-Earlier:
  - Examine the classification metric (such as accuracy) at each node. Stop the splitting process if the metric meets pre-defined value
  - Use Minimum Description Length (MDL) principle: halting growth of the tree when the encoding is minimized.

Post-Pruning the Tree

- A decision tree based on the training data may need to be pruned
  - over-fitting may result in branches or leaves based on too few examples
  - pruning is the process of removing branches and subtrees that are generated due to noise; this improves classification accuracy
- Subtree Replacement: merge a subtree into a leaf node
  - At a tree node, if the accuracy without splitting is higher than the accuracy with splitting, replace the subtree with a leaf node; label it using the majority class



Suppose with test set we find 3 red "no" examples, and 2 blue "yes" example. We can replace the tree with a single "no" node. After replacement there will be only 2 errors instead of 5.

Summary

	KNN Classifier	Naïve Bayes Classifier	Decision Trees
Principle	Find the KNN by distances; Assign the major class label;	Calculate conditional probability; compare probability of each label given an example	Build a tree by top-down fashion. The node is selected by "best" feature
Assumptions	No	Features are conditional independent with labels	No
Feature Types	Categorical data should be converted to binary values	Numeric values may be converted to categorical ones	Numeric values may be converted to categorical ones
Feature Normalization	Yes	No	No
Overfitting	Lazy learner; Parameters	Imbalance classes	Tree Pruning

Decision Tree Learning

- Decision Tree Learning: Basics
- Decision Tree Learning: Feature Selection
- Decision Tree Learning: Overfitting and Pruning
- Extension: Regression Tree

Regression Tree

- Decision tree, by default, was developed for classifications where the target variable is a nominal variable
- The tree-based method can also be extended to predict or estimate a numerical variable – it is the technique of Regression Tree

Regression Tree

- To further understand regression tree, let's compare decision classification tree vs regression tree

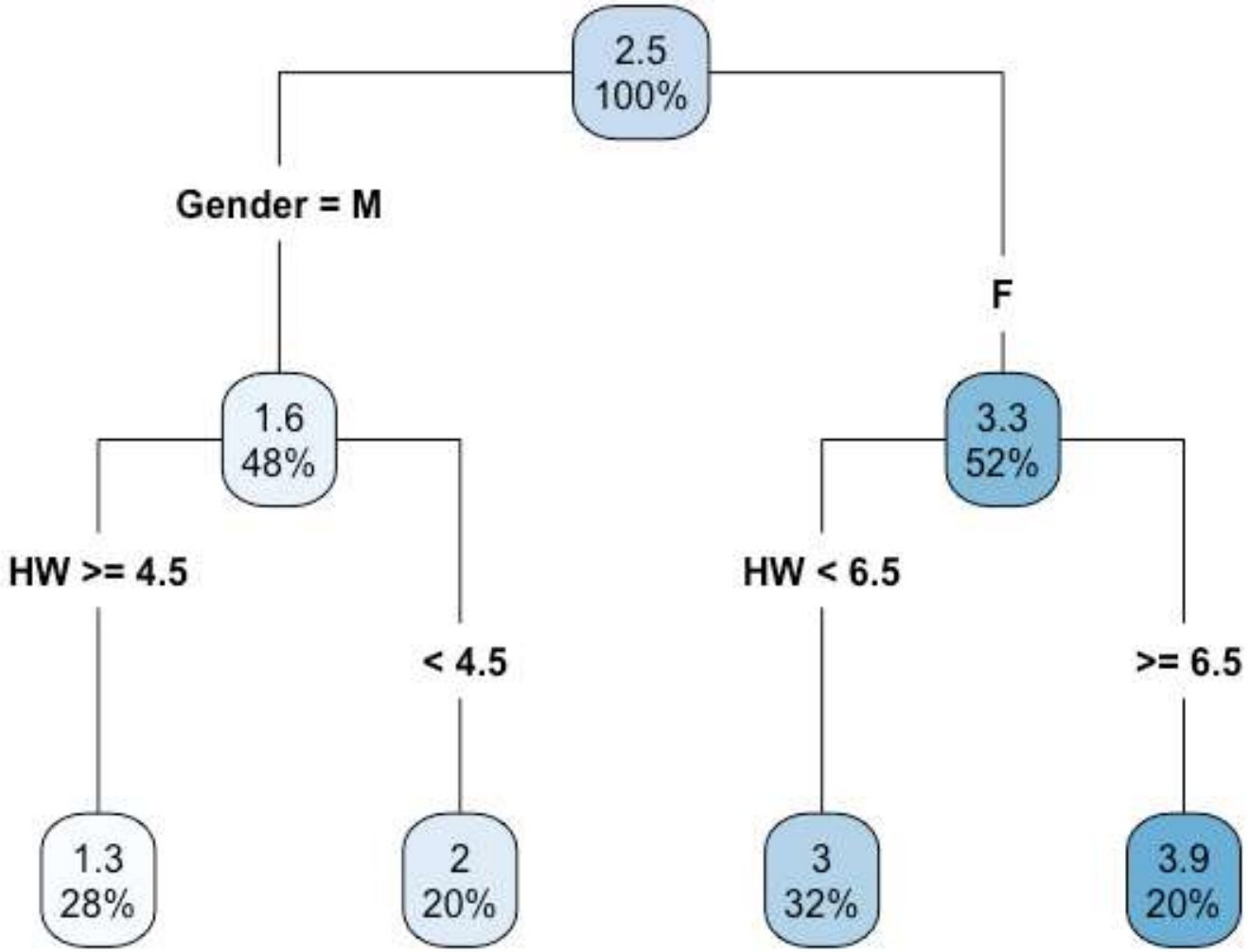
	Classification Tree	Regression Tree
Target variable	Nominal	Numerical
Output in Leaf	Nominal label	Numerical value (mean of set)
Impurity	IG or Gini index	MSE (mean squared error)
Branches	Could be more than two	Binary

Comparison between classification trees and regression trees

Regression Tree

- How it works
  - How to split the space to create branch/trees
    Everytime, it iterates all possible value or categories in each feature to create a binary split
    The impurity is MSE. We want to find the best split which makes lowest MSE in each split
    We continue the splitting until it meets stopping criteria
  - How to output a numerical value in leaf node
    The value is the mean of values in a splitted group

Regression Tree



Summary

- Decision Tree
  - How it works? A feature selection process
  - Different impurity criteria
  - Solutions for overfitting
    - Stop-earlier
    - Post-pruning
- Regression Tree
  - An adaption of decision tree to address regression problems (i.e., predicting a numerical variable)