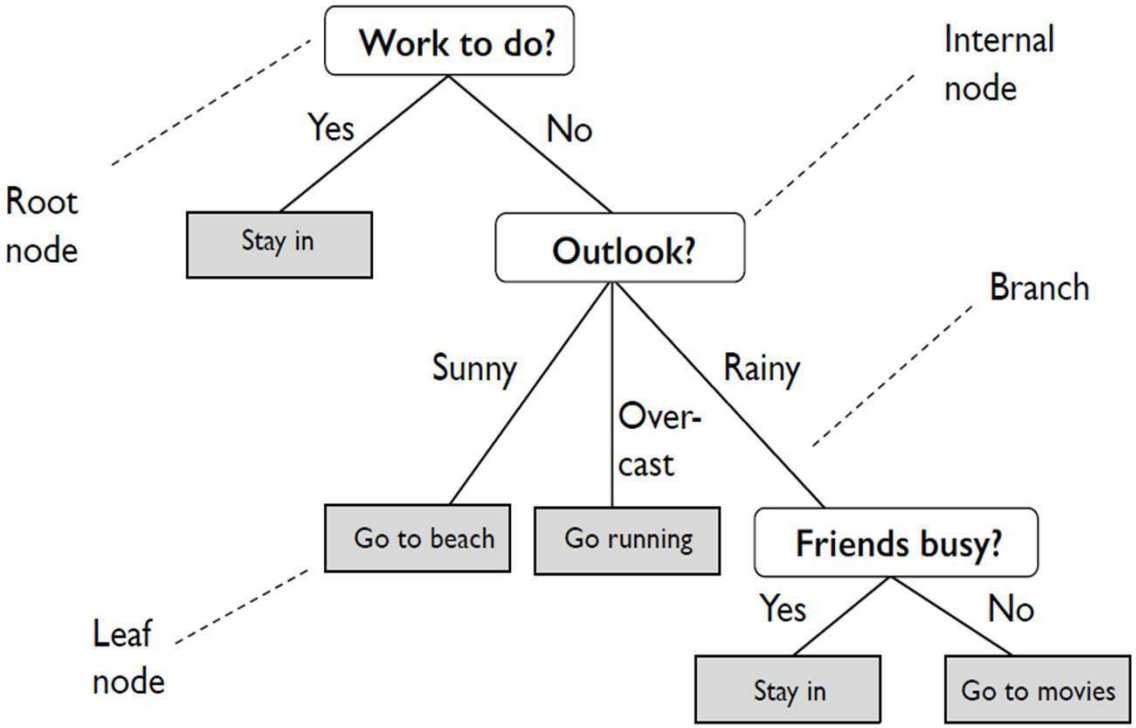
Decision Tree

Fatemeh Mansoori

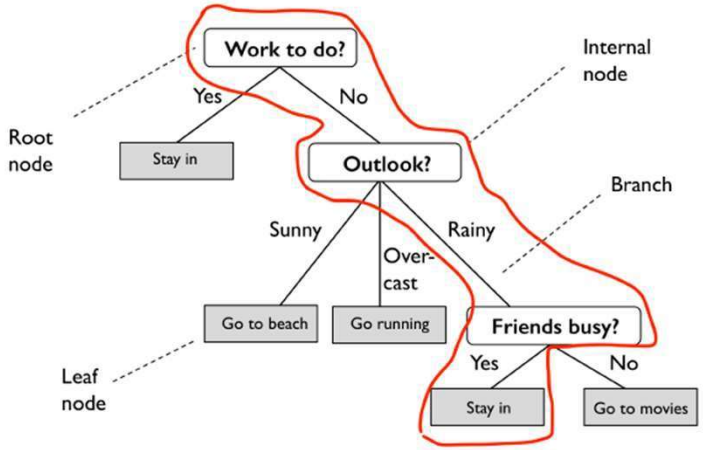
Decision Tree Terminology



Decision Tree algorithms can be considered as an iterative, top-down construction
method for either classifier or regressor.

Considering only binary (or Boolean) features, at each node, how many potential splits
to be evaluated given that the dataset has *m* features?

Decision Trees as Rulesets



IF _____ ___ _____

___ _____

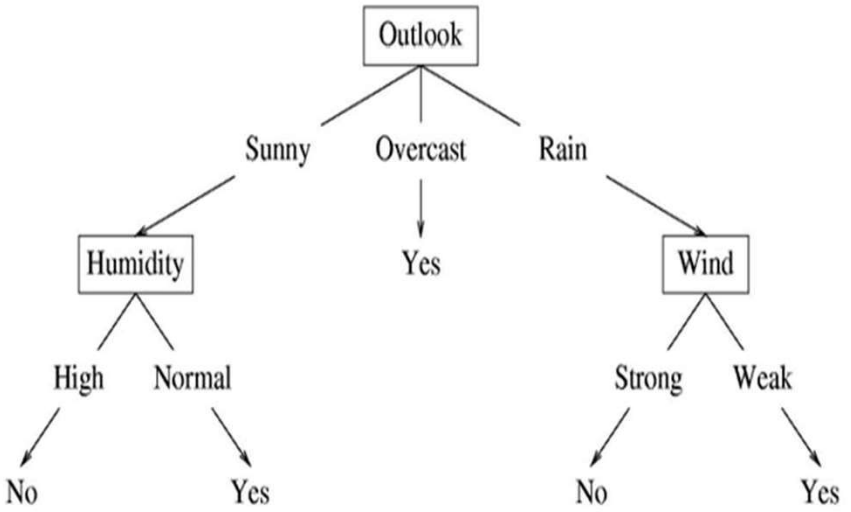
THEN _____ ___ _____

Sample Dataset

- Columns denote features X_i
- Rows denote labeled instances $\langle \boldsymbol{x}_i, y_i \rangle$
- Class label denotes whether a tennis game was played

$\langle \boldsymbol{x}_i, y_i \rangle$

Predictors				Response
Outlook	**Temperature**	**Humidity**	**Wind**	**Class**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No



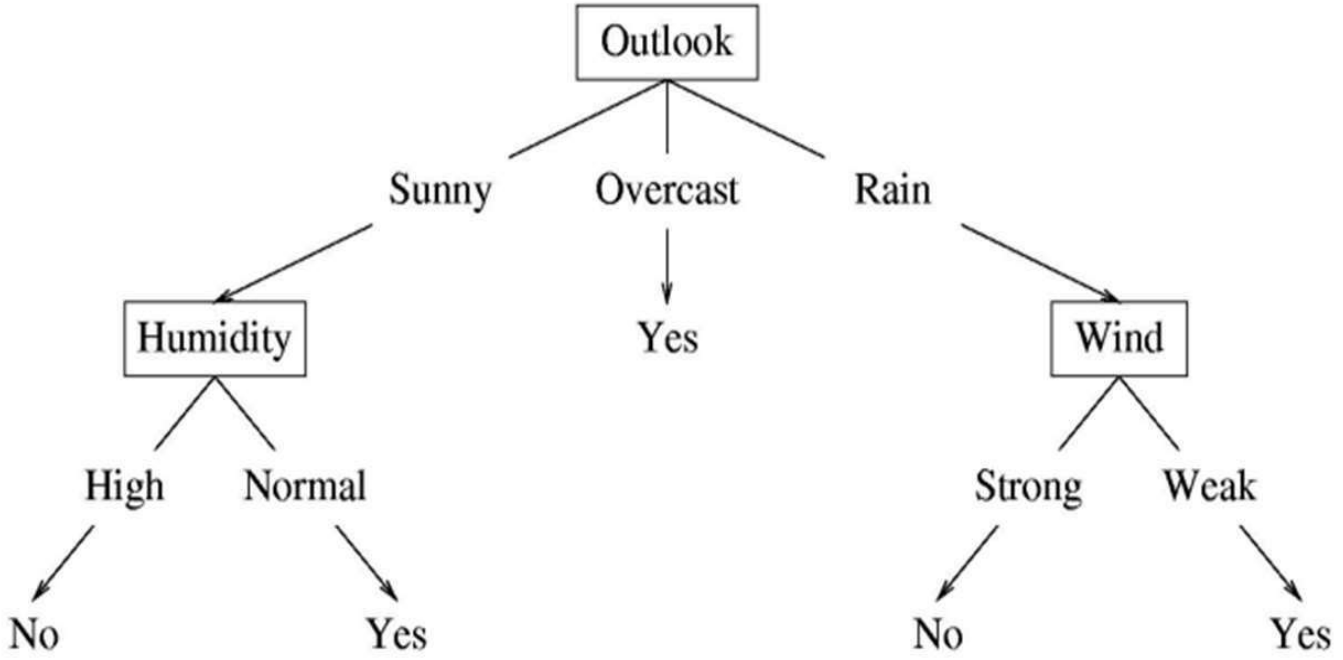
Each internal node: test one attribute X_i

Each branch from a node: selects one value for X_i

Each leaf node: predict Y (or $p(Y \mid \boldsymbol{x} \in \text{leaf})$)

Decision Tree

- A possible decision tree for the data:

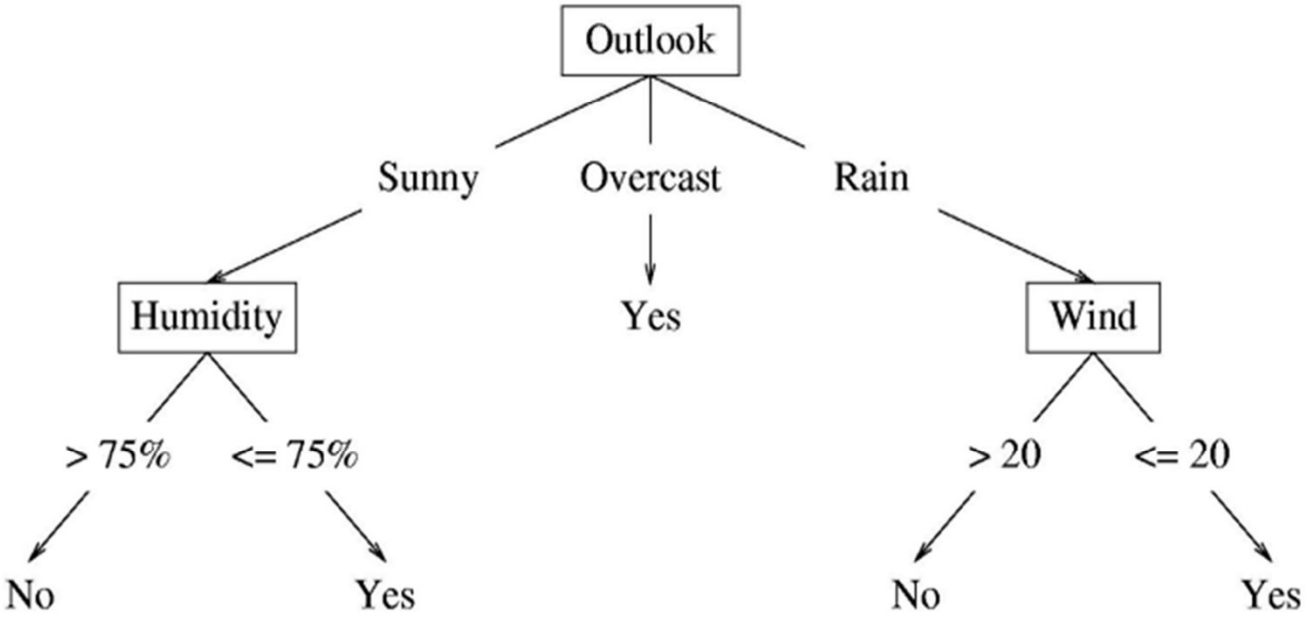


- What prediction would we make for

  <outlook=sunny, temperature=hot, humidity=high, wind=weak> ?

Decision Tree

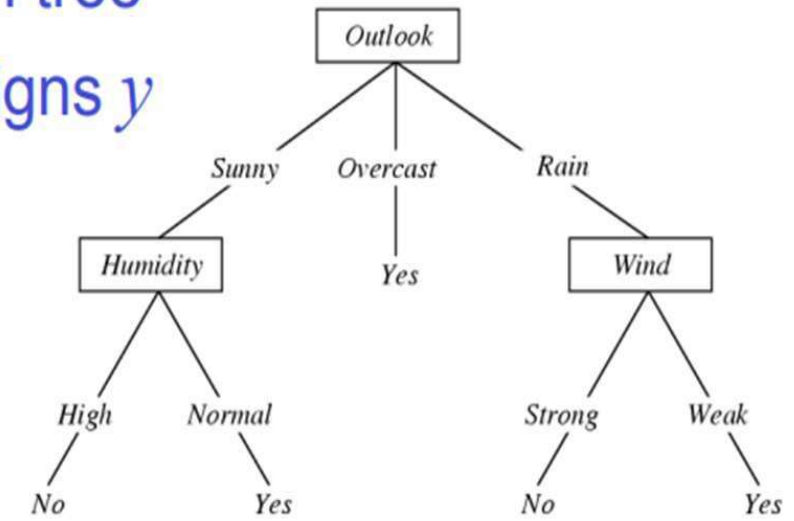
- If features are continuous, internal nodes can test the value of a feature against a threshold



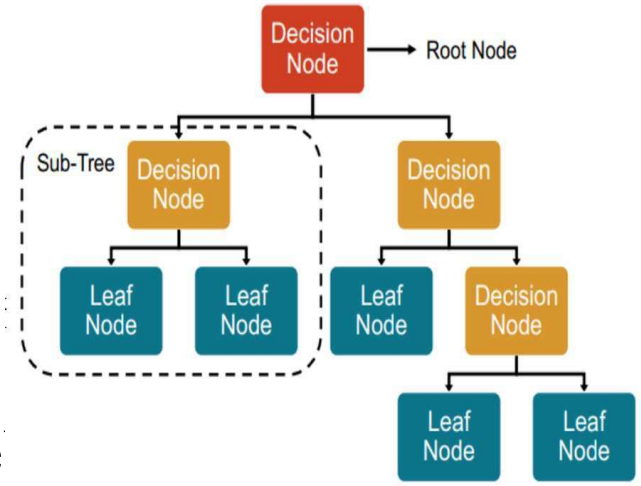
Decision Tree Learning

Problem Setting:

- Set of possible instances X
  - each instance x in X is a feature vector
  - e.g., <*Humidity=low, Wind=weak, Outlook=rain, Temp=hot*>
- Unknown target function $f : X \rightarrow Y$
  - Y is discrete valued
- Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$
  - each hypothesis h is a decision tree
  - trees sorts x to leaf, which assigns y



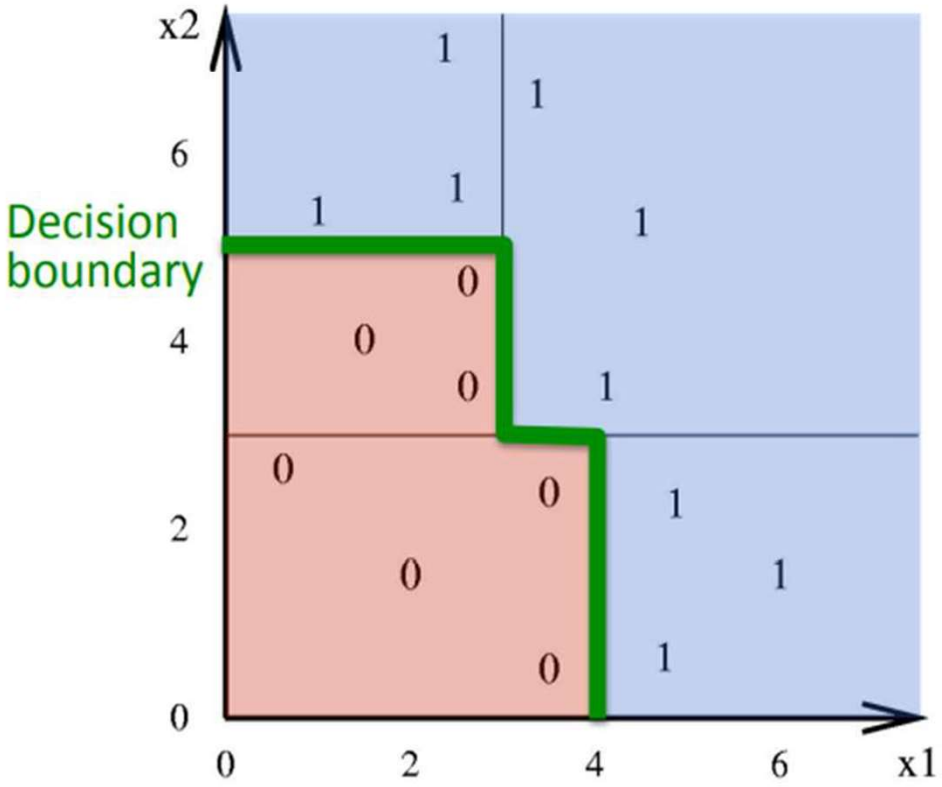
Terminology

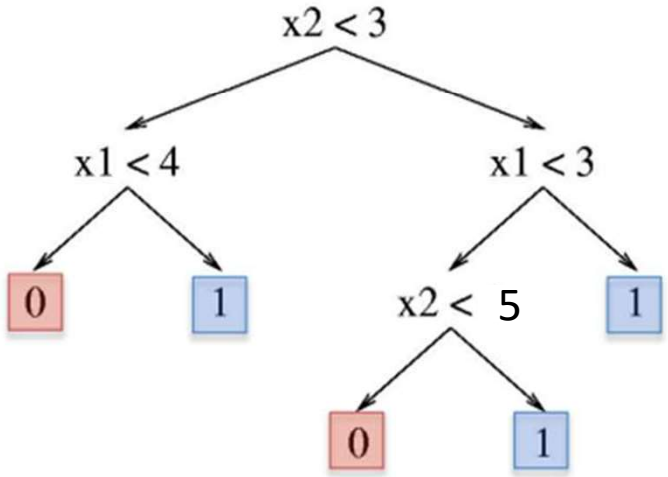


- Root Node: No incoming edge, zero, or more outgoing edges.
- Internal Node: One incoming edge, two (or more outgoing edges.
- Leaf Node: Each leaf node is assigned a class label if nodes are pure; otherwise, the class
- label is determined by majority vote.
- Parent & Child Node: If a node is split, we refer to that given node as the parent node, and the resulting nodes are called child nodes.

Decision Tree – Decision Boundary

- Decision trees divide the feature space into axis-parallel (hyper-)rectangles
- Each rectangular region is labeled with one label
  - or a probability distribution over labels

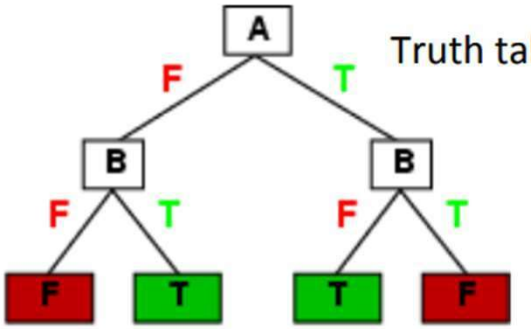




Expressiveness

- Decision trees can represent any boolean function of the input attributes

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Truth table row → path to leaf

- In the worst case, the tree will require exponentially many nodes

Which Tree is the best?

- Smallest decision tree that correctly classifies all of the training examples is best
- Finding the provably smallest decision tree is NP-hard
- Instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small

Algorithm

Algorithm 2 Constructing DT

```
1:  procedure FINDTREE(S, A)                    ▷ Input: S (samples), A (attributes)
2:      if A is empty or all labels in S are the same then
3:          status ← leaf
4:          class ← most common class in S
5:      else
6:          status ← internal
7:          a ← bestAttribute(S, A)             ▷ The attribute value test
8:          LeftNode ← FindTree(S(a > t), A − {a})   ▷ t (threshold)
9:          RightNode ← FindTree(S(a ≤ t), A − {a})
10:     end if
11: end procedure
```

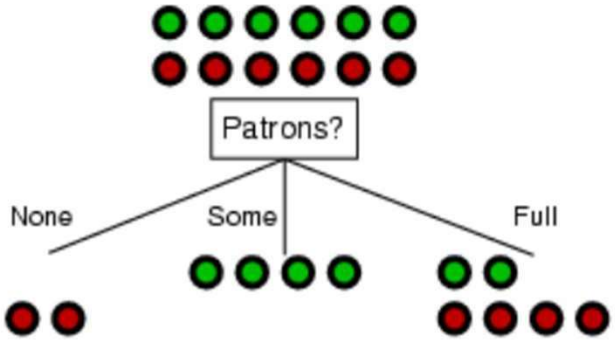
Choosing the Best Attribute

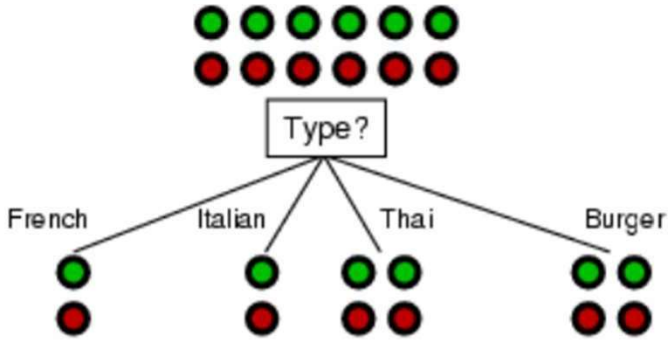
Key problem: choosing which attribute to split a given set of examples

- Some possibilities are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose the attribute with the smallest number of possible values
  - **Most-Values:** Choose the attribute with the largest number of possible values
  - **Max-Gain:** Choose the attribute that has the largest expected *information gain*
    - i.e., attribute that results in smallest expected size of subtrees rooted at its children
- The ID3 algorithm uses the Max-Gain method of selecting the best attribute

Choosing an Attribute

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



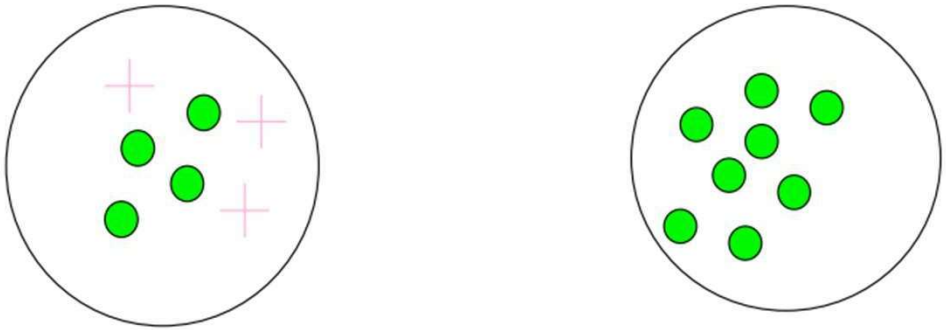


Which split is more informative: *Patrons?* or *Type?*

Information Gain

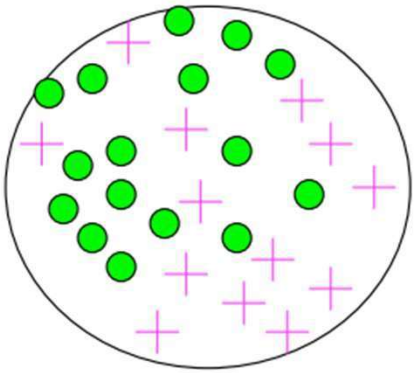
Impurity/Entropy (informal)

– Measures the level of **impurity** in a group of examples

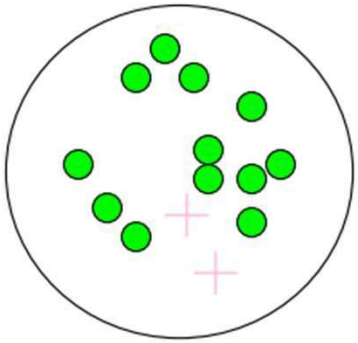


Impurity

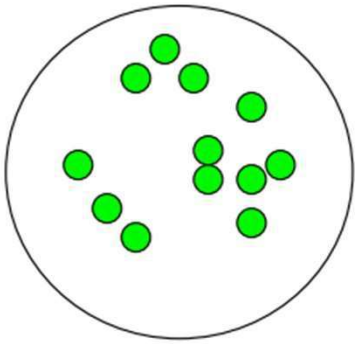
Very impure group



Less impure



Minimum impurity



Entropy: a common way to measure impurity

Entropy $H(X)$ of a random variable X

# of possible values for X

$$H(X) = -\sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

$H(X)$ is the expected number of bits needed to encode a randomly drawn value of X (under most efficient code)

Why? Information theory:

- Most efficient code assigns $-\log_2 P(X=i)$ bits to encode the message $X=i$
- So, expected number of bits to code one random X is:

$$\sum_{i=1}^{n} P(X = i)(-\log_2 P(X = i))$$

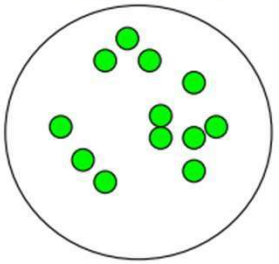
2-Class Cases:

Entropy $H(x) = -\sum_{i=1}^{n} P(x=i)\log_2 P(x=i)$

- What is the entropy of a group in which all examples belong to the same class?
  - entropy = - 1 $\log_2 1$ = 0

  not a good training set for learning

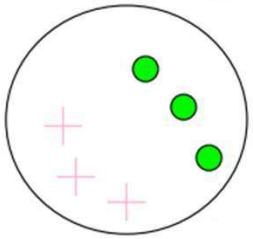
Minimum impurity



- What is the entropy of a group with 50% in either class?
  - entropy = -0.5 $\log_2 0.5$ – 0.5 $\log_2 0.5$ =1

  good training set for learning

Maximum impurity



Information Gain

- We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.
- Information gain tells us how important a given attribute of the feature vectors is.
- We will use it to decide the ordering of attributes in the nodes of a decision tree.

From Entropy to Information Gain

- **Entropy** measures the uncertainty in a specific distribution.

$$H(X) = -\sum_{x_i \in x} P(x_i) \log P(x_i)$$

- **Information Gain (IG)**

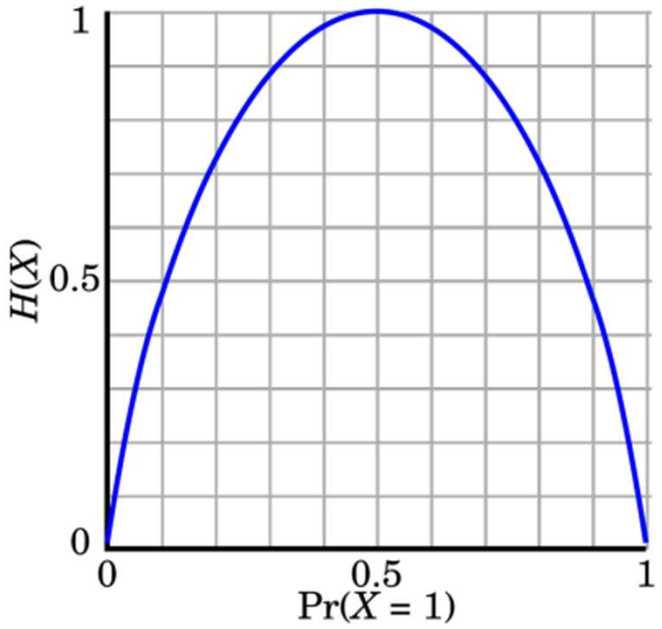
$$\text{Gain}(S, A) = H_S(Y) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H_{S_V}(Y)$$

A: variable used to split samples

Y: target variable

S: samples, S_v: subset of S where $A = v$

$H_S(Y)$: entropy of Y over S



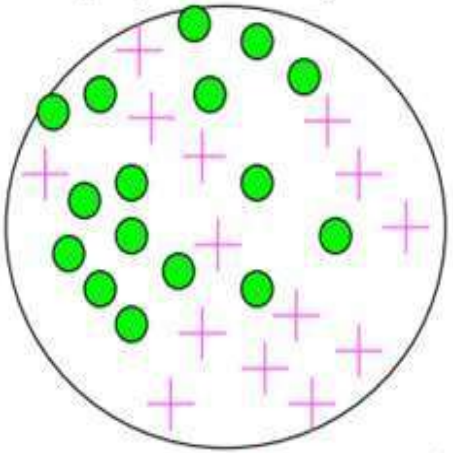
Adopted from Wikipedia

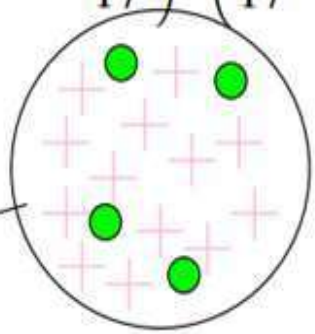
Calculating Information Gain

Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$

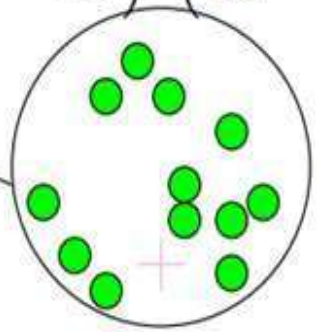
Entire population (30 instances)





17 instances

child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$



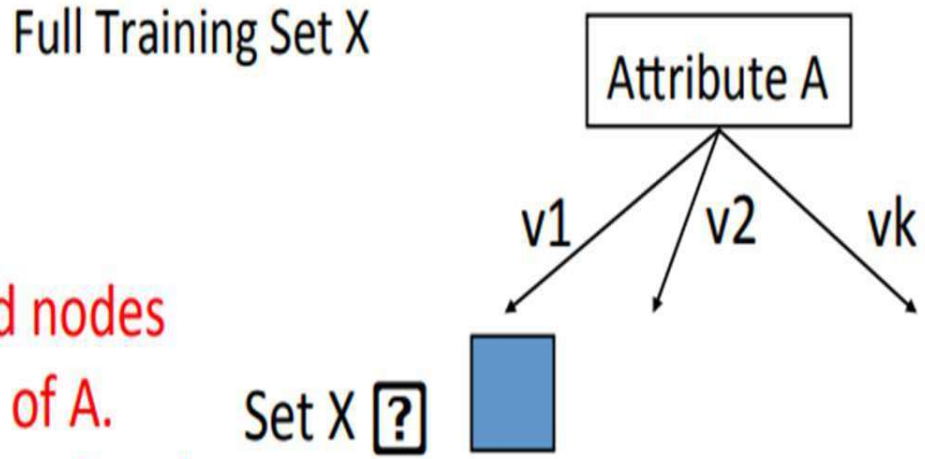
13 instances

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain= 0.996 - 0.615 = 0.38

Using Information Gain to Construct a Decision Tree



Choose the attribute A with highest information gain for the full training set at the root of the tree.

Construct child nodes for each value of A. Each has an associated subset of vectors in which A has a particular value.

repeat recursively till when?

Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenr
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

Gini Index

- Gini Impurity: Let X be a random variable over a discrete space Ω with probability mass function P. Then the Gini coefficient of X is:

$$G(X) = \sum_{x \in X} \mathbb{P}(x)(1 - \mathbb{P}(x)) = 1 - \sum_{x \in X} \mathbb{P}(x)^2$$

Algorithm

- ID3: Iterative Dichotomizer 3
  - It can only be used for discrete features, and cannot handle numeric features!
  - It supports multi-category splits.
  - It does not apply any pruning, and is prone to overfitting.
  - It results in short and wide trees.
  - It maximizes information gain/minimizes entropy.

- C4.5
  - It supports Continuous and discrete features (continuous feature splitting is very expensive because must consider all possible ranges).
  - It can handles missing attributes (ignores them in information gain computation)
  - It performs post-pruning

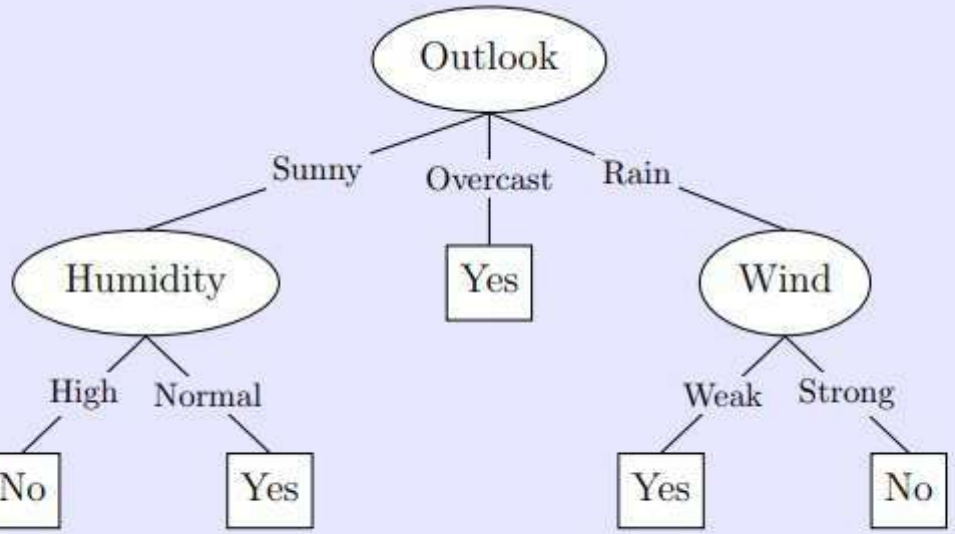
Decision Tree for Regression

- It is the same as classification with a few subtle modifications:
- The main difference is that instead of predicting a class in each node, it predicts a value.
- The prediction for each node is simply the average target value of the all training instances associated to this leaf node.
- Instead of trying to split the training set in a way that minimizes impurity, algorithm now tries to split the training set in a way that minimizes the MSE.
  - The MSE at a given node is also often referred to as intra-node variance, and the splitting criterion is thus called variance reduction.
  - Likewise to classification tasks, decision trees are prone to overfitting when dealing with regression tasks.

Overfiting(example)

Example: The Jeeves data set has 14 data points, this is an extremely small data set; we can still learn a reasonable decision tree for this data set.

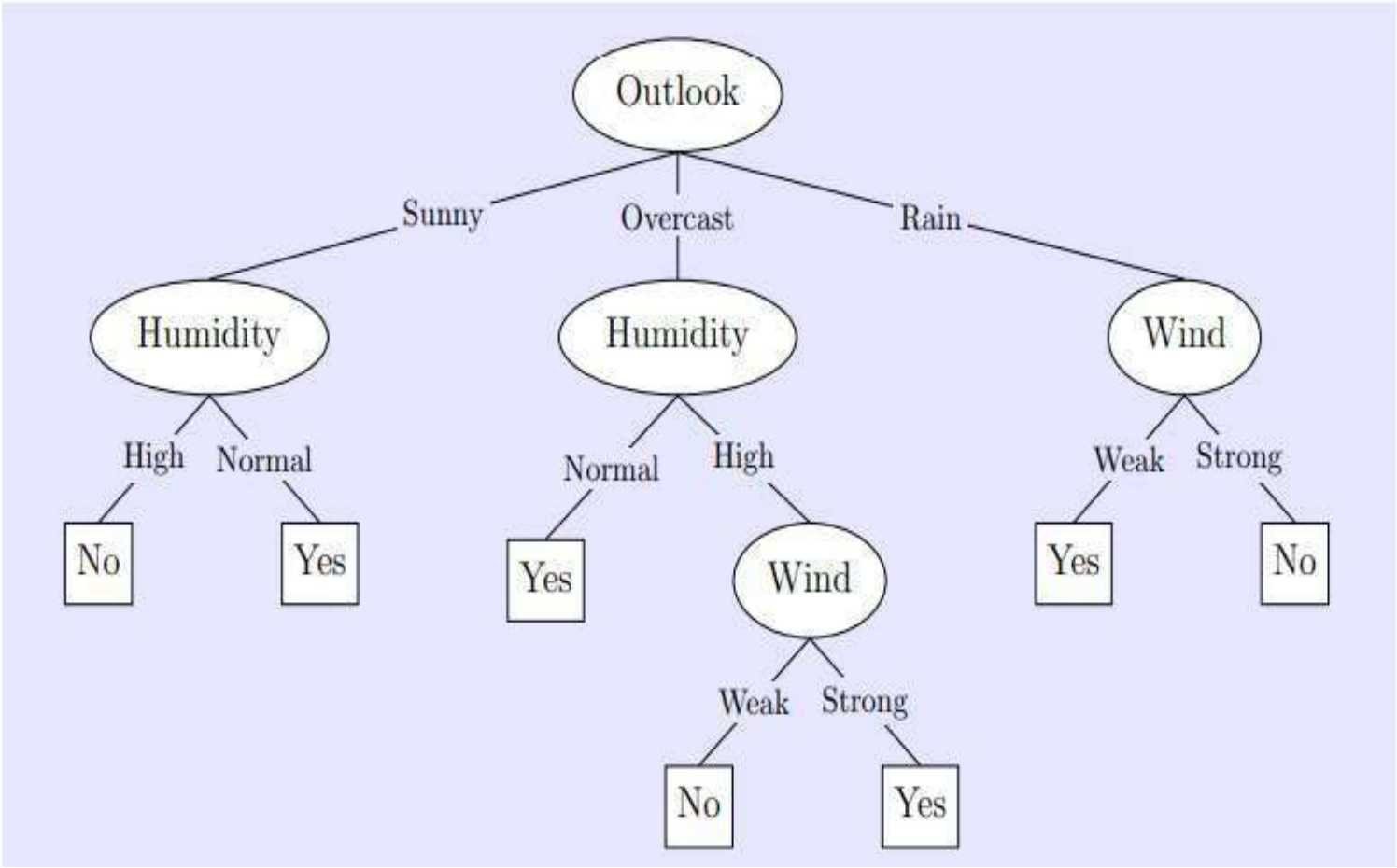
Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



Test error is 0 out of 14.

- Suppose that you sent the training set and the test set to your friend. For some reason the data set gets corrupted during the transmission. Your friend gets a corrupted training set where only one data point is different. For this third data, point, it changed from the label ”yes” to the label ”no”.

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	**No**
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



We grew an entire middle sub-tree to accommodate one corrupted data point, and this small corruption caused a dramatic change to the tree. This new tree is more complicated and it likely won't generalize well to unseen data.

Overfiting

- Decision Trees make very few assumptions about the training data!
- If left unconstrained, their structure will adapt itself to the training data, fitting it very closely, and most likely overfitting it!
- When tree is too complex and capture noise in the training samples the overfitting is occurred
- It would be better to grow a smaller and shallower tree.
- Smaller and shallower tree may not predict all of the training data points perfectly but it may generalize to test data better.
- have two options to prevent over-fitting when learning a decision tree:
  - Pre-pruning: stop growing the tree early.
  - Post-pruning: grow a full tree first and then trim it afterwards

Pre pruning

- decide not to split the examples at a node and stop growing the tree
- use the majority label as the decision for that leaf node
- What are the methods for pre pruning?

Post Pruning

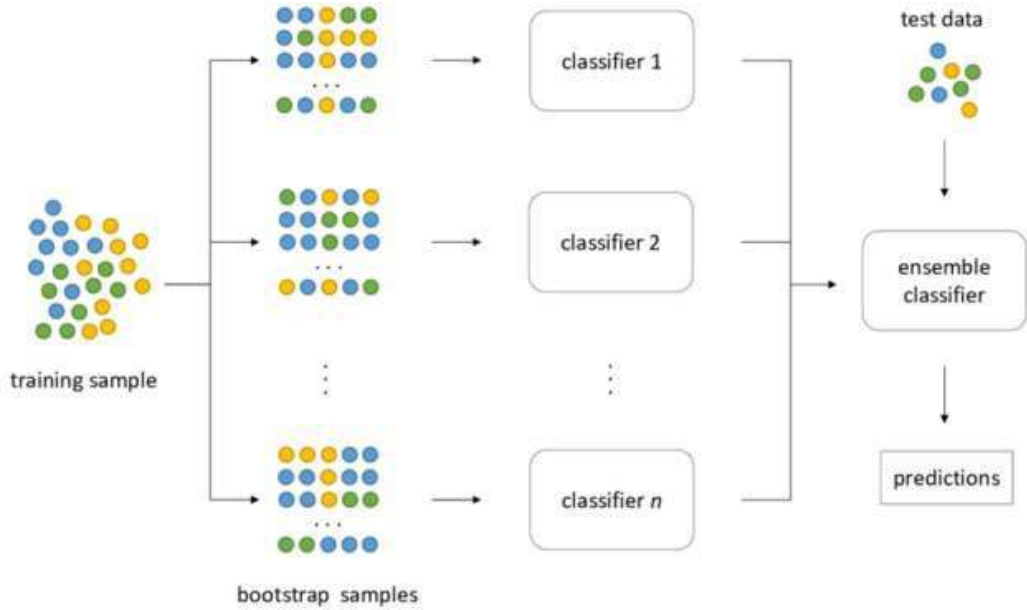
- Post-pruning: grow a full tree first and then trim it afterwards
- When we may use Post Pruning?

Idea of Ensemble Classifiers

- Using a single decision tree could be problematic
  - highly sensitive to small changes in data
- One solution :
  - instead of just using a single decision tree, train multiple decision trees to make it more robust (Bagging)
- Each decision tree is a different way of classifying the objects.
- The collection of the decision trees is called tree ensemble.

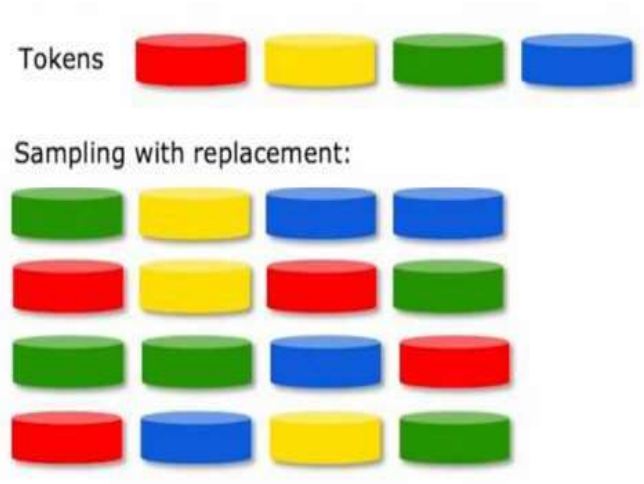
Bagging (bootstrap aggregation)

- learning method utilizing several different models
  - built by training on different sample subset
- The average, aggregation, or majority of the predictions from different models is calculated for prediction
- In the context of Decision Tree, tree ensemble and majority decision is used for bagging.



Sampling with replacement (bootstrap)

- to build tree ensemble, used sampling with replacement technique
- Sampling with replacement :
  - randomly select a sample from the dataset
  - after recording the sample that was picked, put in back to the dataset
  - pick another random sample from the dataset record it, and put it back again



Random Forest

- used for classification and regression
- Several decision trees are used in this ensemble learning technique to

increase accuracy and decrease overfitting.

- construct a huge number of decision trees, each of which is trained using a unique part of the training data and a random sample of features
- Each tree in the forest makes a forecast for an output given an input
- final prediction is formed by collecting the majority vote of all the trees in the forest
- Random feature selection and data sampling contribute to reducing overfitting.

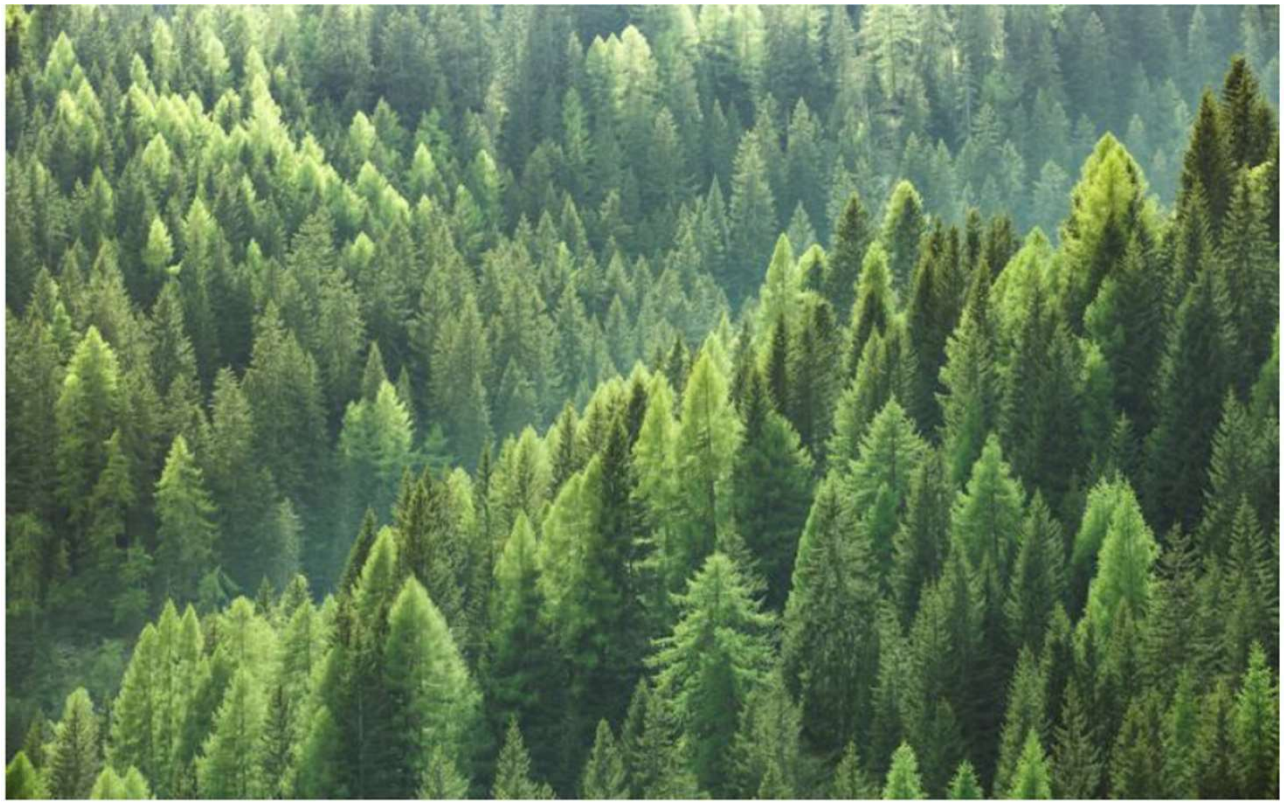
Random Forest

$$RandomForest = DecisionTree + Bagging + Boostrapping + RandomSplits$$

Given traing set of size m
For b = 1 to B:
Use sampling with replacemnt to create a new traing set of size m
Train a decision tree on the new dataset

B stands for bagging and denotes the number of decision tree we are going to train and is often recommended to be picked as 64 or 128.

- The key idea is that even with sampling with replacement, you always ended up with same root nodes and similar splits.
- Random Forest improve this by randomizing the feature choice.
- At each node, when choosing a feature to use to split, if n features are available, pick a random subset of k<n features and allow the algorithm to only choose from that subset of features



Random forests, adaptive boosting, gradient boosting