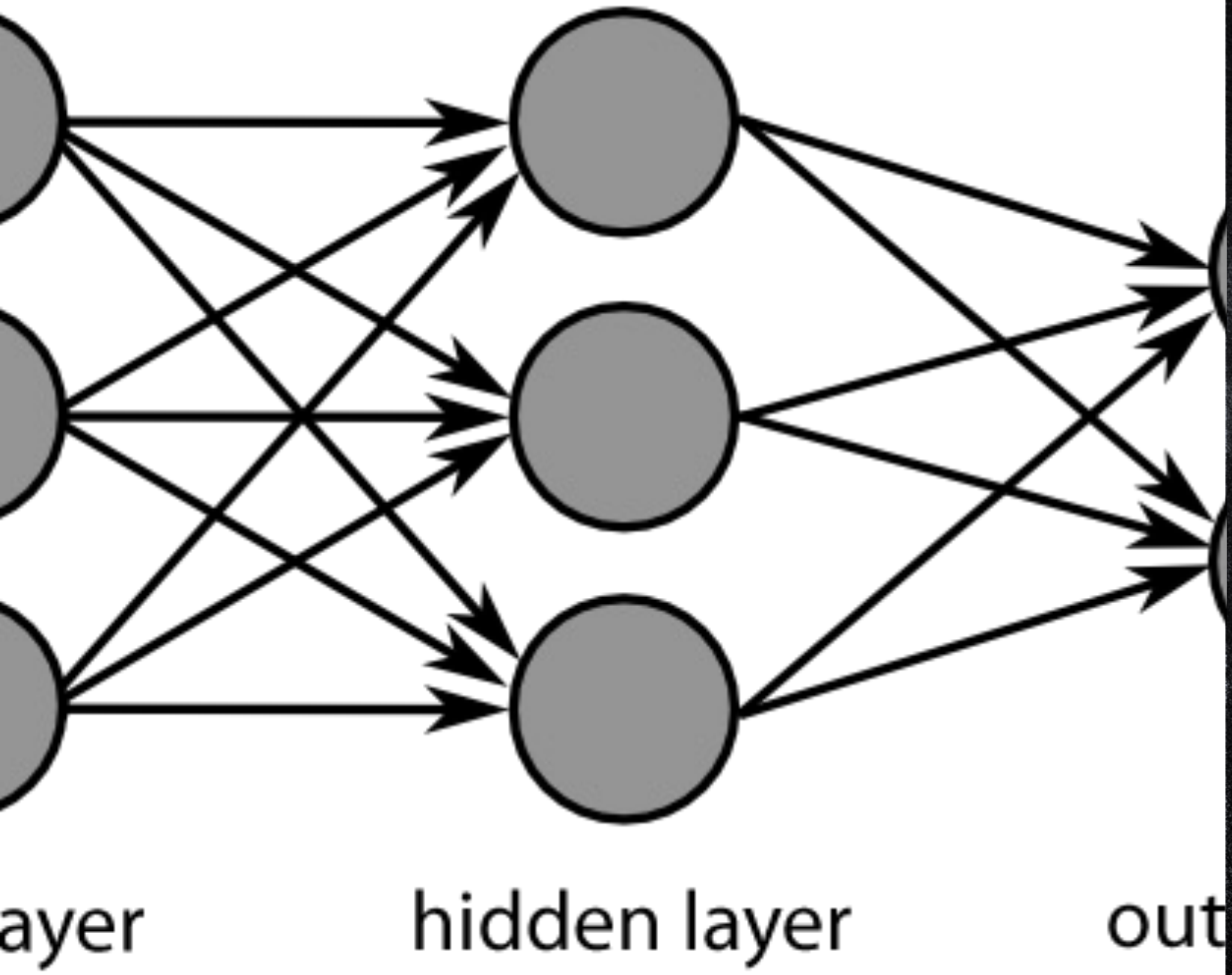


Aprendizaje Supervisado

TC3006C

Supervised Learning

Definition

- Supervised learning is the process of discovering a function h (**hypothesis**) that approximates a real unknown function f
- Training set
  - $(x_1, y_1), (x_2, y_2), \cdots (x_n, y_n)$
  - Input-output pairs generated by an unknown $y = f(x)$
- Called "*Supervised Learning*" due to the fact that for every $\mathbf{x}$, its corresponding output $\mathbf{y}$ is known.
- Learning is a search through the space of possible hypotheses until:
  - A function h is found
  - Function h **generalizes** well: performs well on a **training set** and on unseen elements (**test set**)

Supervised Learning

Classification

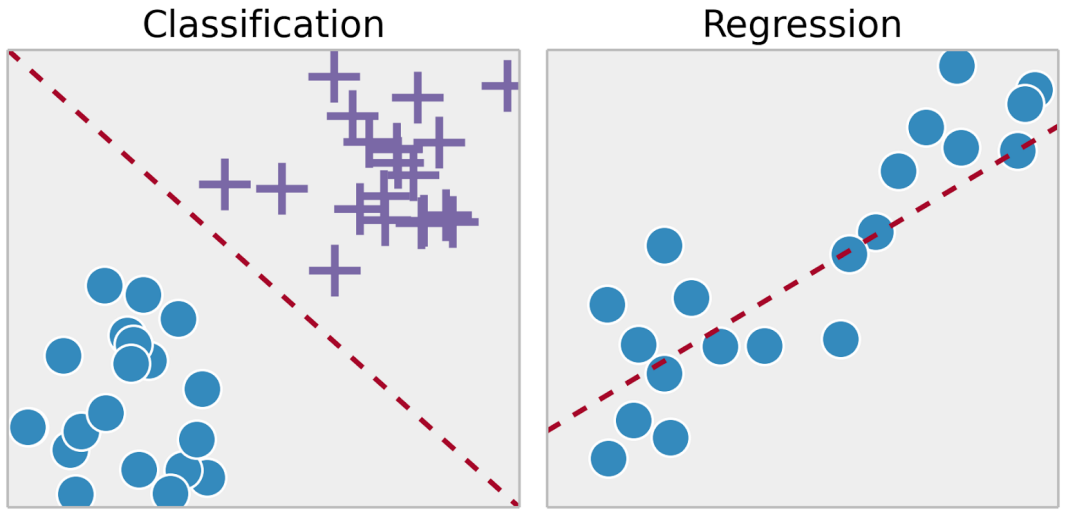
- Output y is one of a finite set of values
  - Example of sets:
    - Red, blue, yellow, green
    - Hot, mild, cold

- If set has only two elements is called Boolean (Binary) Classification

Regression

- Output y is a number
  - Examples:
    - Temperature
    - Velocity

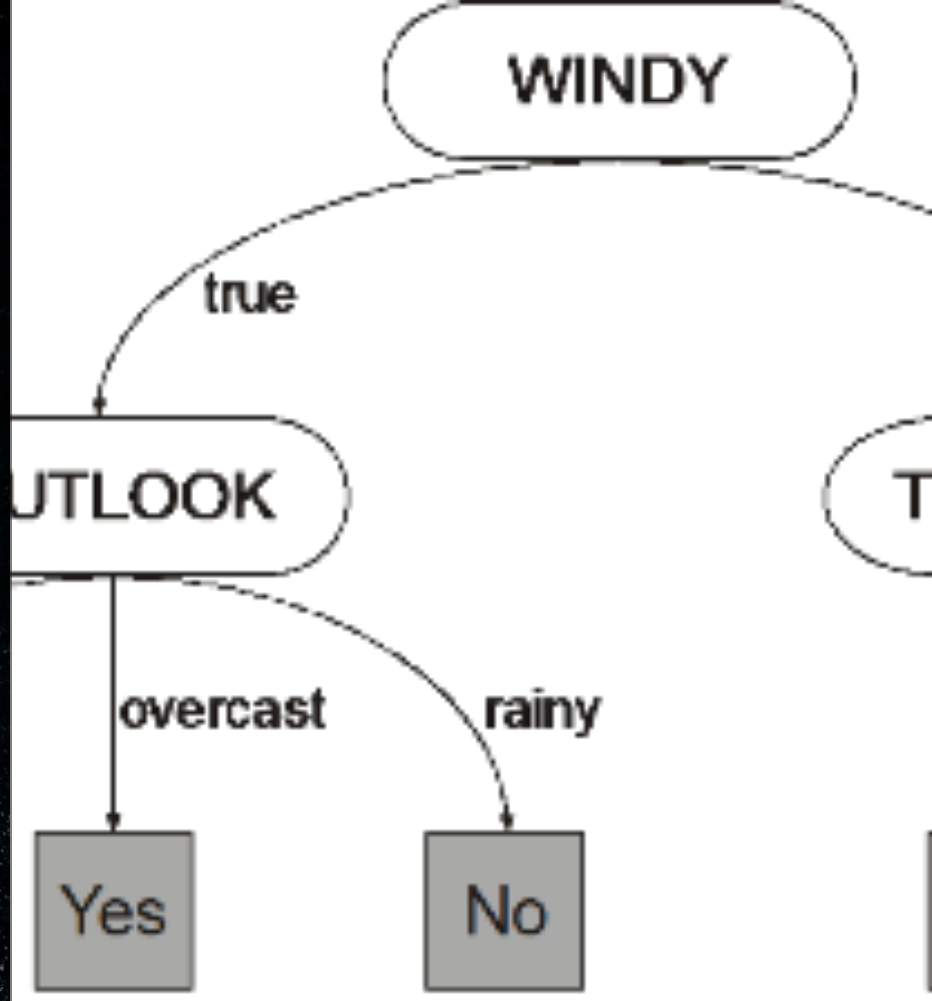
- **Consistent hypothesis:** function h agrees with all the data



https://miro.medium.com/max/3200/1*ASYpFfDh7XnreU-ygqXonw.png

Decision Trees

Supervised Learning



juresti@tec.mx 2022AD

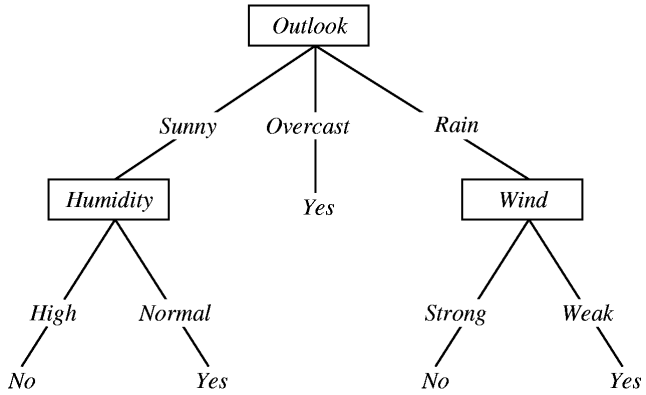
Decision Trees

Definitions

- Represents a function
  - Input: vector of attributes
  - Output: single value (“decision”)
- Values can be discrete or continuous
- Boolean classification
  - Input: discrete values
  - Output: True or False

Function

- Reaches a decision by performing a series of tests
- Each internal node in a tree corresponds to a test
- Each leaf node corresponds to the output to be returned

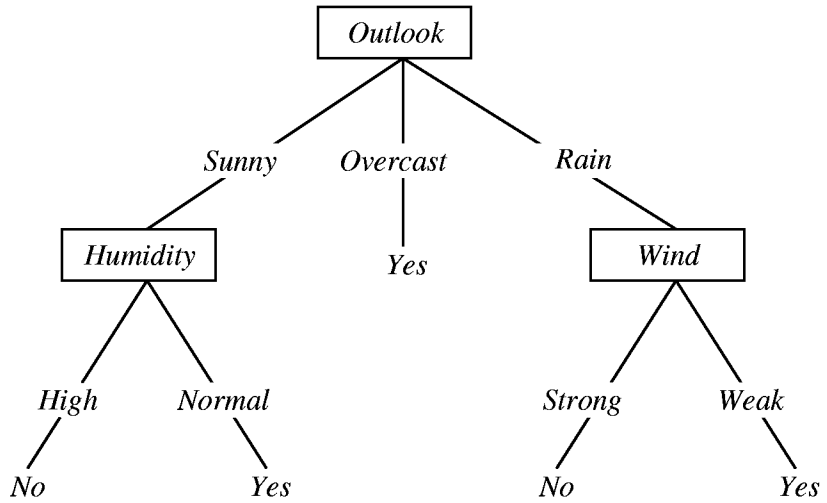


https://dzone.com/articles/machine-learning-with-decision-trees

Example of typical data

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

Expressiveness of decision trees



- Boolean decision tree is logically equivalent to a Boolean expression

$$(Outlook = Sunny \wedge Humidity = Normal)$$
$$\vee \quad (Outlook = Overcast)$$
$$\vee \quad (Outlook = Rain \wedge Wind = Weak)$$

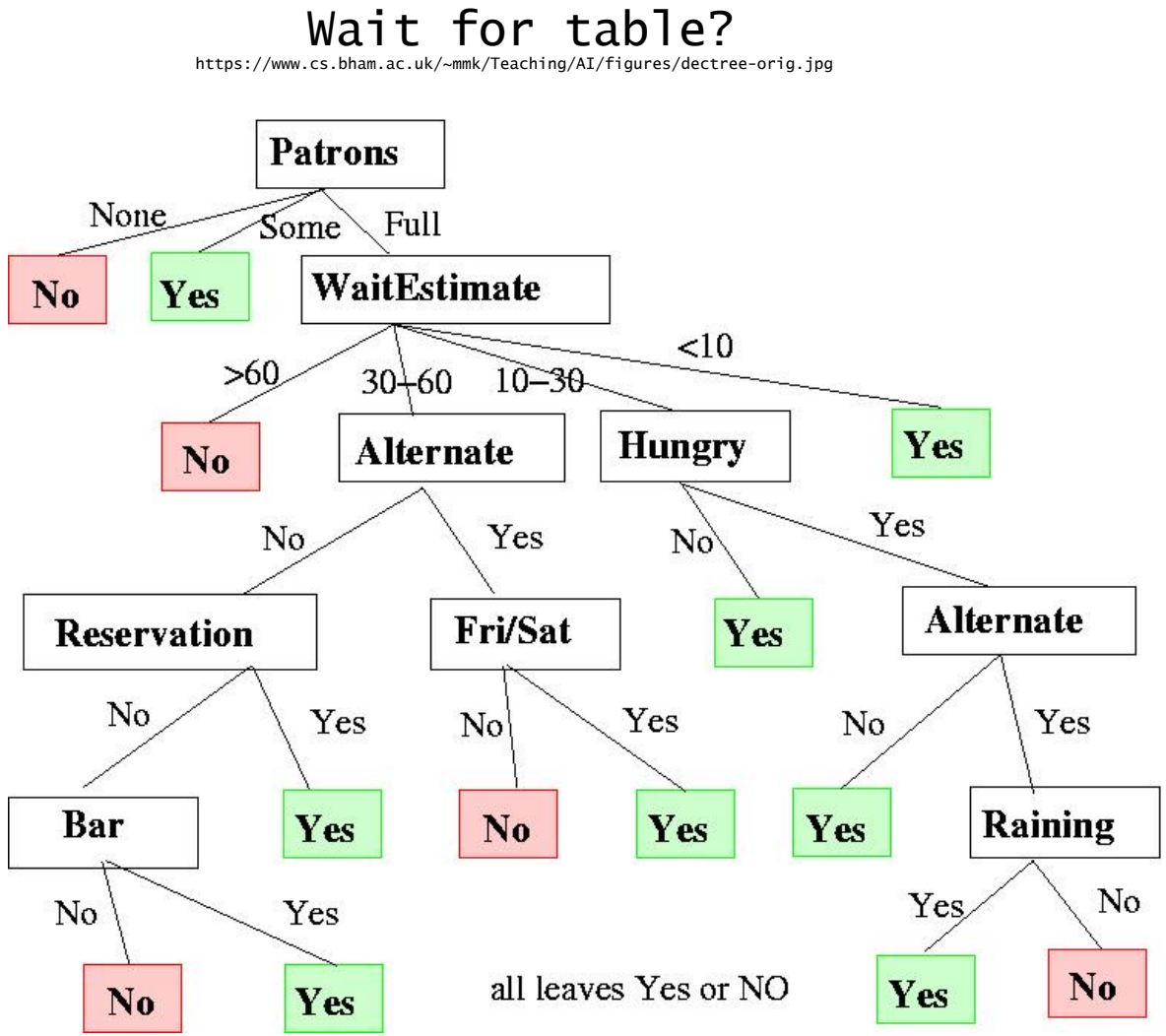
- The expression is obtained from the branches that have a True value leaf node

- How would this instance be classified?

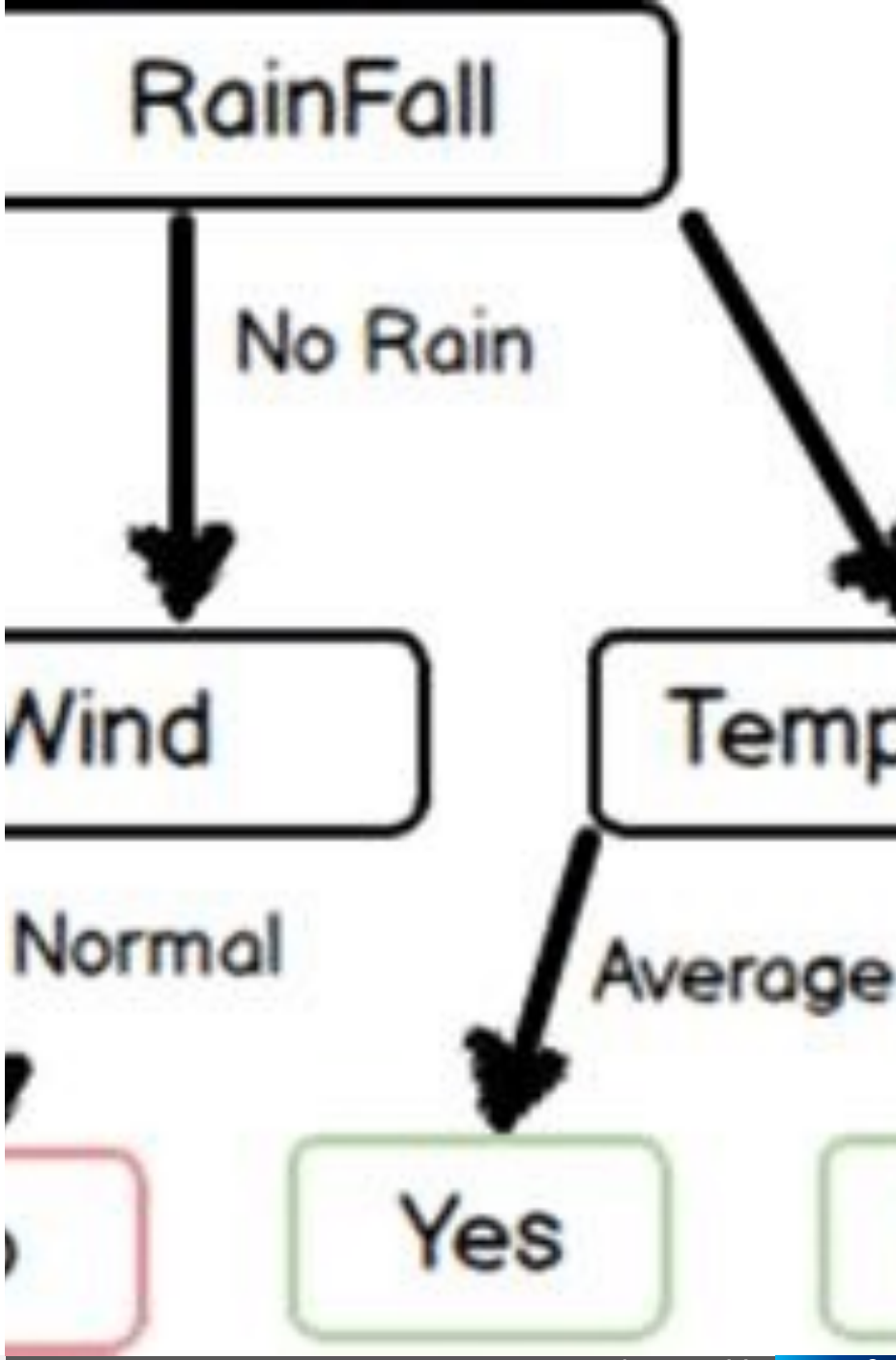
$$\langle Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong \rangle$$

Appropriate problems

- Instances represented by attribute-value pairs.
- Target function has discrete output values.
- Disjunctive descriptions may be required.
- Training data may contain errors.
- Training data may contain missing attribute values.



ID3



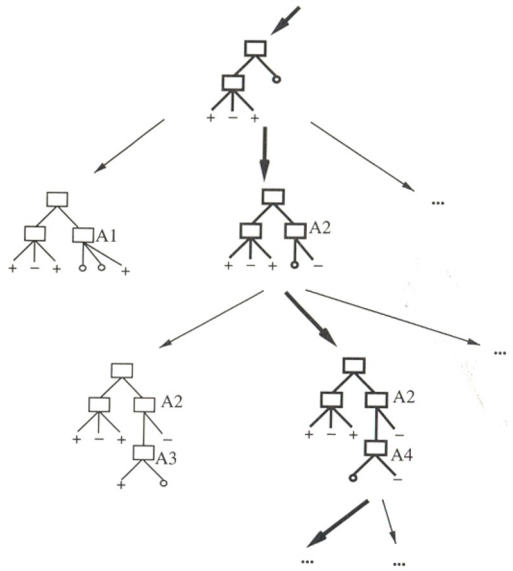
Play tennis data

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

ID3 – Basic decision tree learning algorithm

Introduction

- Top-down, greedy search through the space of possible decision trees (Quinlan, 1986).
  - Succesor C4.5 (Quinlan, 1993)



- Basic question: "*Which attribute should be tested at the root of the tree?*"
  1. Each instance attribute is evaluated to determine how well it alone classifies training examples.
  2. Best attribute selected as the root node.
  3. Descendants of the root node are created.
  4. Training examples are sorted to the appropriate descendant.
  5. Entire process repeated for the rest of the attributes.

ID3(Examples, Target attribute, Attributes)

Examples are the training examples. Target attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a *Root* node for the tree

- If all *Examples* are positive, Return the single-node tree *Root*, with label = +

- If all *Examples* are negative, Return the single-node tree *Root*, with label = -

- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*

- Otherwise Begin
  - A ← the attribute from *Attributes* that best classifies *Examples*
  - The decision attribute for *Root* ← *A*
  - For each possible value, v_i, of *A*,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{vi}$ be the subset of *Examples* that have value v_i for *A*
    - If $Examples_{vi}$ is empty
      - Then below this new branch add a leaf node with label = most coomon value of *Target Attribute* in *Examples*
      - Else below this new branch add the subtree ID3($Examples_{vi}$, *Target_attribute*, *Attributes* – {*A*}))

- End
- Return *Root*

Which attribute is the best classifier?

- ***Information gain***
  - Measures how well a given attribute separates training examples according to their classification.

- ***Entropy***
  - Characterizes (im)purity of an arbitrary collection of examples.
  - Given a collection of examples (*S*), containing + and – examples:

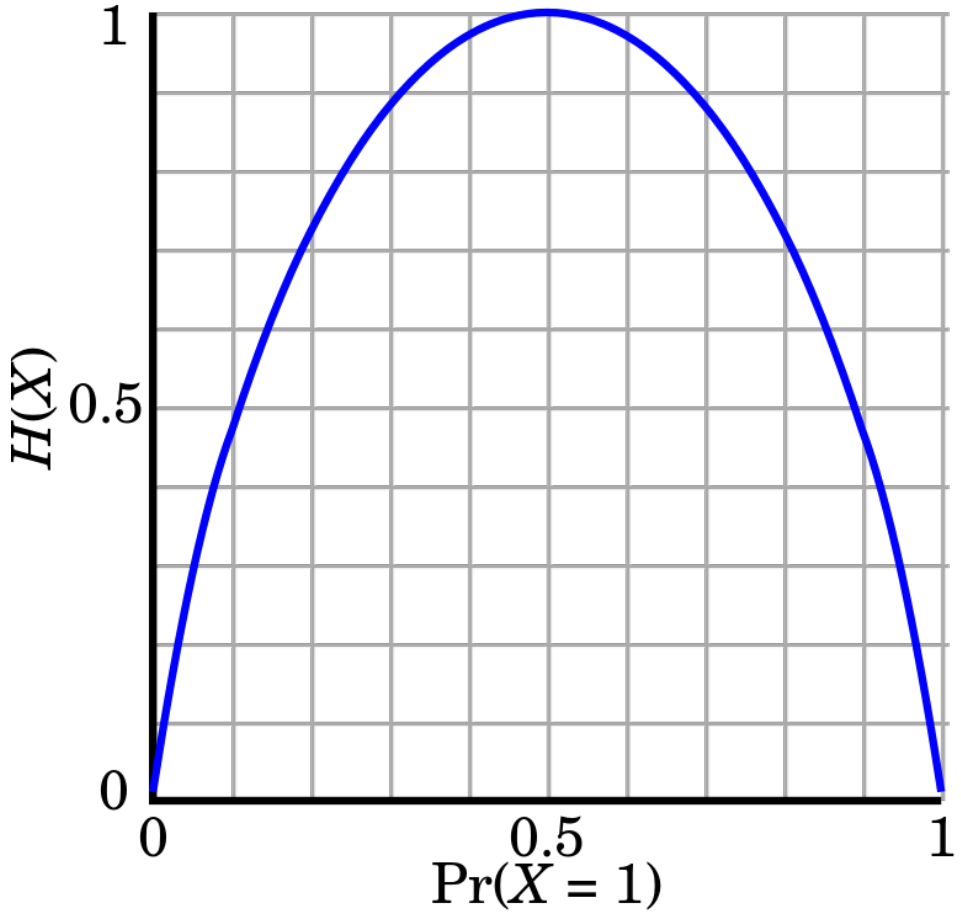
$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Best? …

- Notice that:
  - *Entropy (S) = 0* means that all members of S belong to the same class.
  - *Entropy (S) = 1* means that members of S are evenly distributed.
  - $0 \log_2 0 = 0$

$$Entropy(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

  - c different values.
  - p_i is the proportion of S belonging to class *i*.
  - In the graph, H(X) is the entropy of S



Information gain

- Expected reduction in entropy caused by partitioning the examples according to an attribute *A*.

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

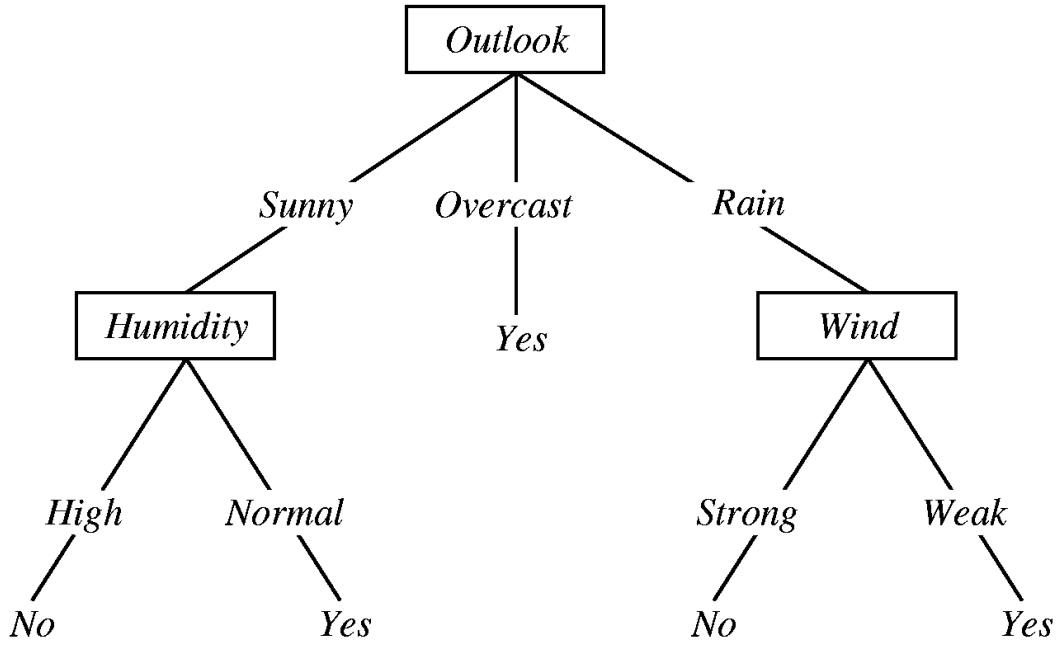
- In other words, is the information provided about the *target function value* given the value of some other attribute *A*.

ID3 Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

ID3 Example …

- Information gain values are:
  - Gain(S,Outlook) = 0.246
  - Gain(S, Humidity) = 0.151
  - Gain(S, Wind) = 0.048
  - Gain(S, Temperature) = 0.029



ID3 Example …

Class Example

- Tree Worksheet.xlsx

Class Exercise

- Tree Exercise Empty.xlsx
- Check Github

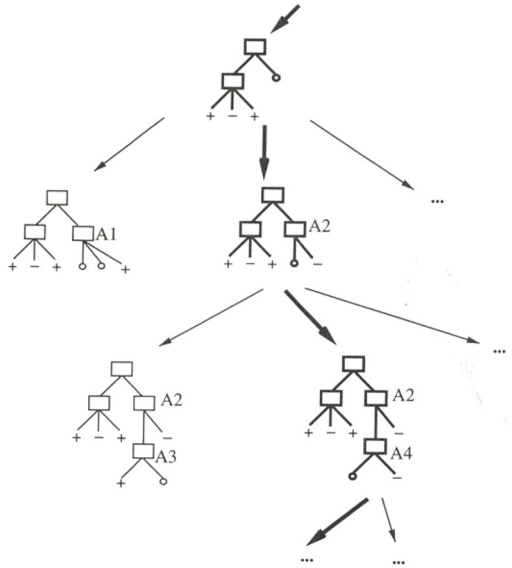
Characteristics of ID3

Decision Tree Learning

Hypothesis space search in ID3

- ID3 searches a space of hypotheses (set of possible decision trees) for one that fits training examples.

- ID3 performs a hill-climbing search
  - Begins with the empty tree.
  - Uses information gain as the evaluation function.



- Advantages and disadvantages
  - Hypotheses space is complete
    - Every finite discrete-valued function can be represented by a decision tree).
  - Maintains only a single current hypothesis.
    - Does not have the ability to determine alternative solutions
  - In its pure form, does not have backtracking, pruning a possible solution.
  - Uses all training examples at each step instead of using only the current training example (incrementally).
  - Information gain (statistical data) makes it less sensible to errors in individual examples.

Inductive bias in ID3

- Policy by which ID3 generalizes from observed training examples to classify unseen instances.

- Which decision tree selects that is consistent with data?
  - Selects in favor of shorter trees over longer ones
  - Selects trees that place attributes with highest information gain closest to the root.

Overfitting and bias

Restriction Biases and Preference Biases

- ID3 searches a *complete* hypothesis space.
  - Set of all decision trees of a finite discrete-value function
  - Searches the space incompletely – pruning using information gain.
  - Inductive bias consequence of ordering hypotheses by search strategy.
  - **Preference bias (search bias)**
    - Preference of certain hypotheses over others.
    - No restrictions on hypotheses space.

Overfitting

- More likely to occur with flexible models (nonparametric and non linear)
- Decision trees are nonparametric, therefore, prone to overfitting
- Pruning a tree may help with overfitting

Decision Trees

Using scikit-learn

- Decision Trees - IrisDS.ipynb

- Check Github

Random Forests

Ensemble Learning Algorithms

Ensemble Learning Algorithms

Definition

- Meta approach
- Combines predictions from multiple models
- Usually gets better predictions
- Many methods exist, but usually are a variation of the 3 main ensemble classes

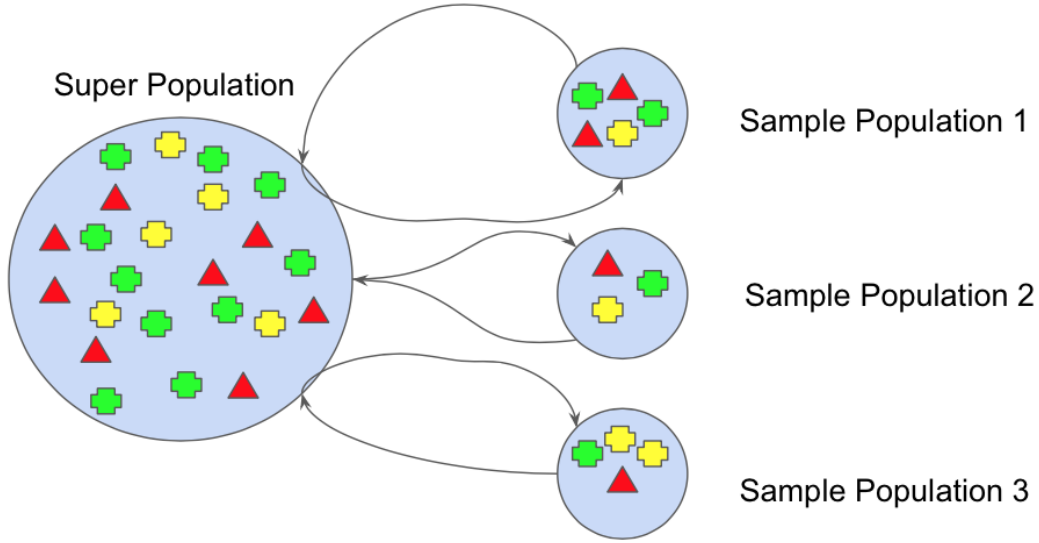
Main classes of ensemble methods

- Bagging
- Stacking
- Boosting

Bagging (Bootstrap AGGregatING)

Bootstrap aggregation

- Generates a diverse group of ensemble members by varying training data
- Generally, it only uses one ML algorithm
- Trains each model on a different sample of the same training data
  - Each model has its own UNIQUE sample
- Predictions made by ensemble members are combined (usually voting or averaging)



https://medium.com/time-to-work/bagging-classifier-6261138b92d4

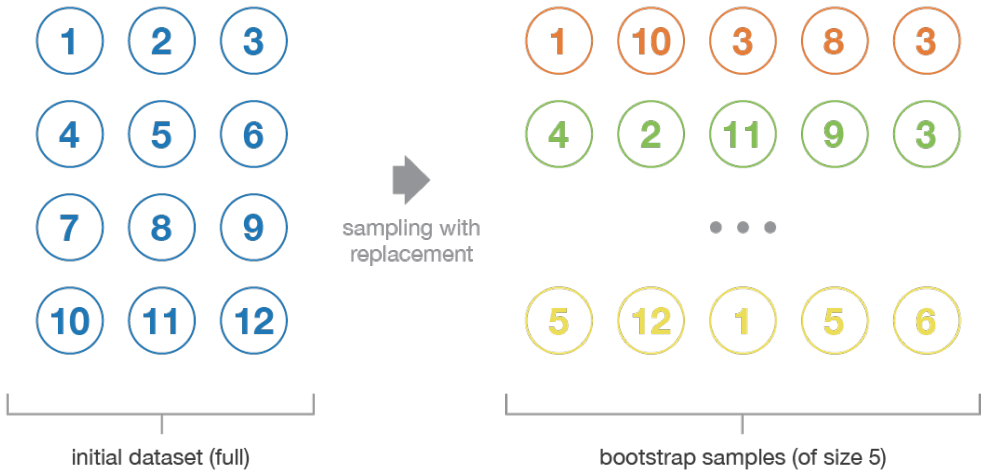
Bagging…

Procedure:

- Randomly select an instance of the complete dataset (row)
- Add it to the new sample dataset for training
- Return selected instance to complete dataset
- Continue selecting instances from the dataset until the new sample dataset has the desired number of elements

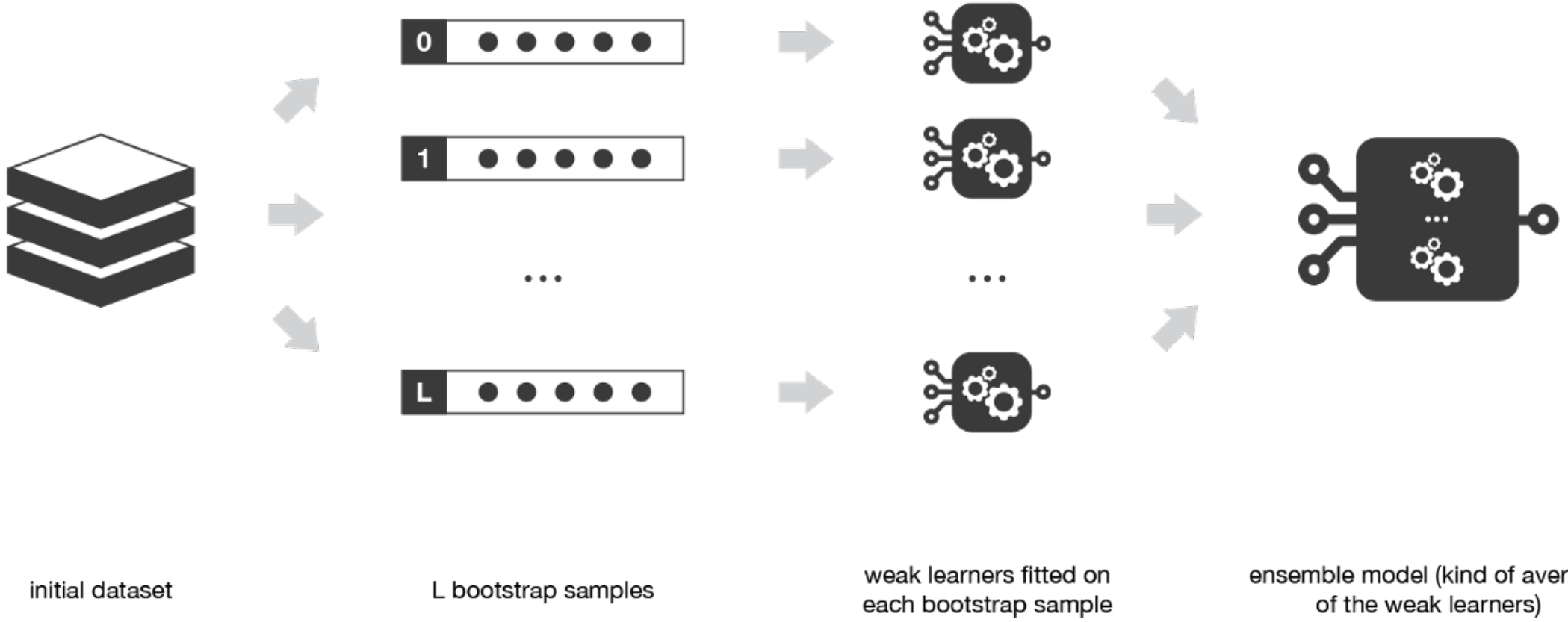
Characteristics:

- An instance can be selected several times for a new sample dataset
- Some instances may have never been selected for a sample dataset



https://miro.medium.com/max/1400/1*lWnm3eJVe3uo95OcSg5jUA@2x.png

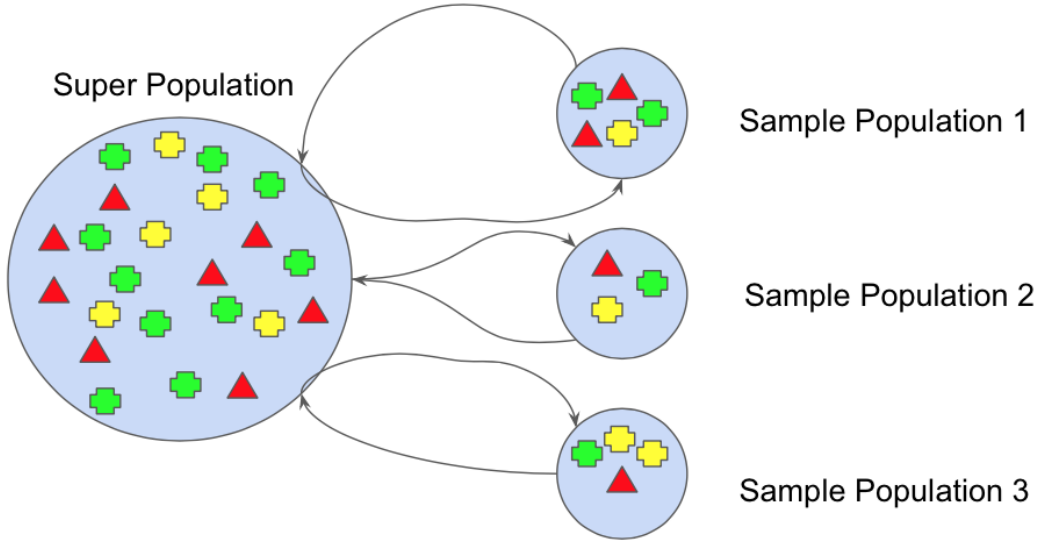
Bagging…



https://miro.medium.com/max/1400/1*zAMhmZ78a6V9W878zfk5eA@2x.png

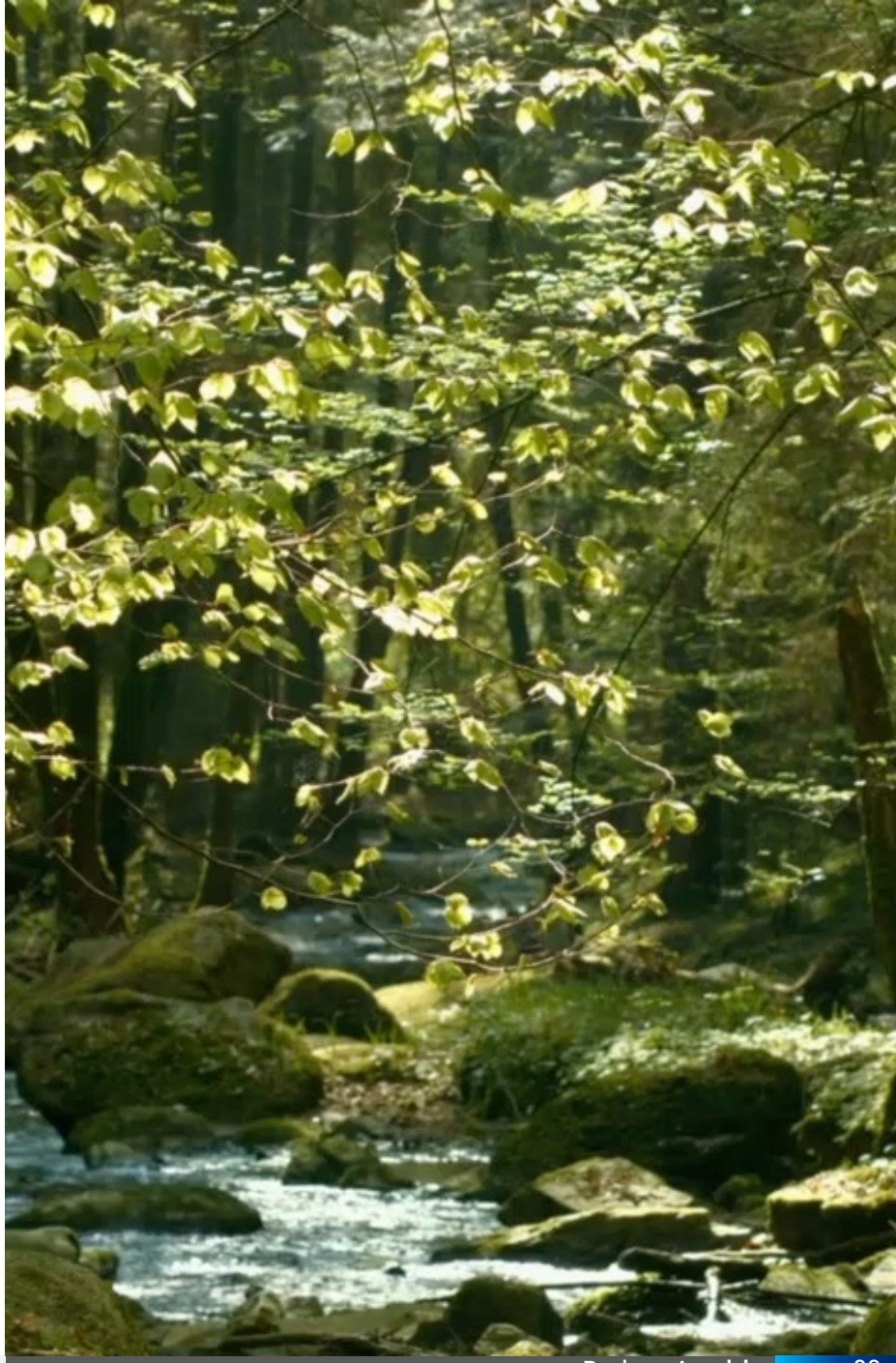
Bagging

Class Example



- Code using scikit-learn
  - Bagging Ensemble Method.ipynb
  - Check Github

Random Forests



Random Forests

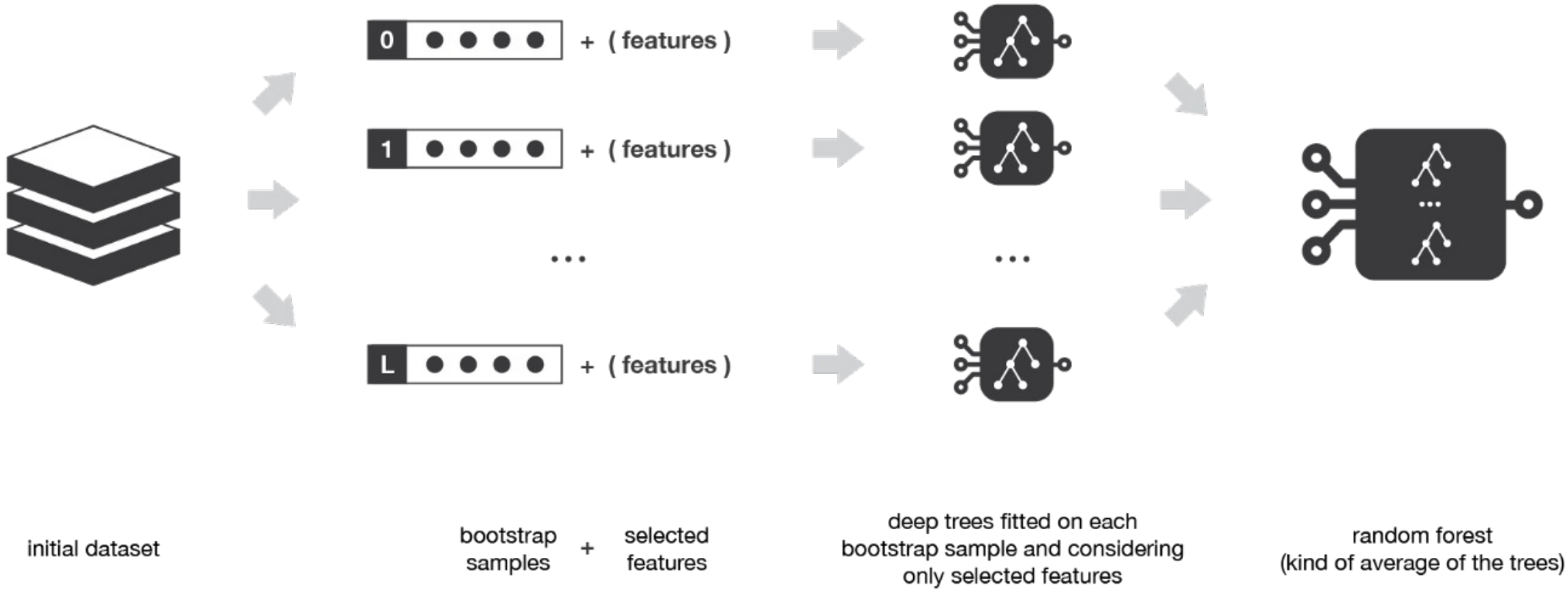
Definition

- Ensemble method using Decision Trees
- Uses Bagging ensemble
- Creates several small trees that can be trained in parallel (instead of just one big tree)
- Helps reduce overfitting and inaccuracy in Decision Trees
  - Decision trees do not work very well with new data

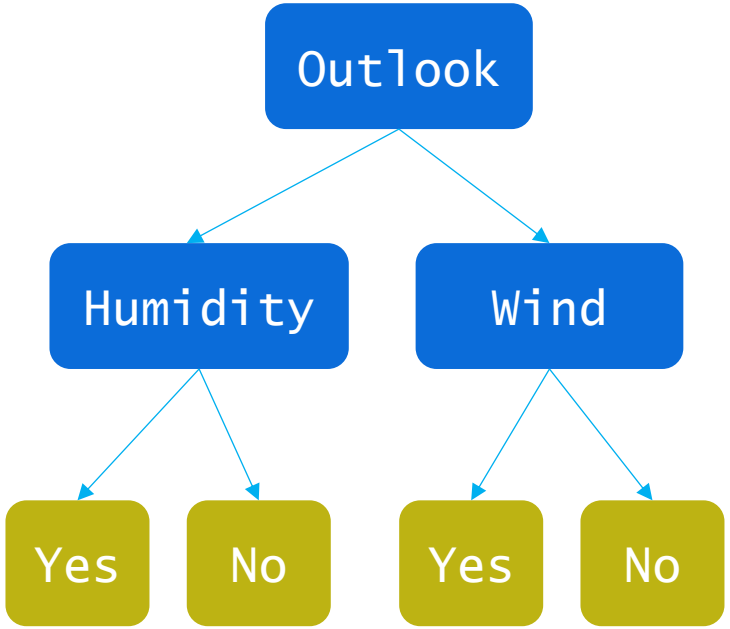
Procedure

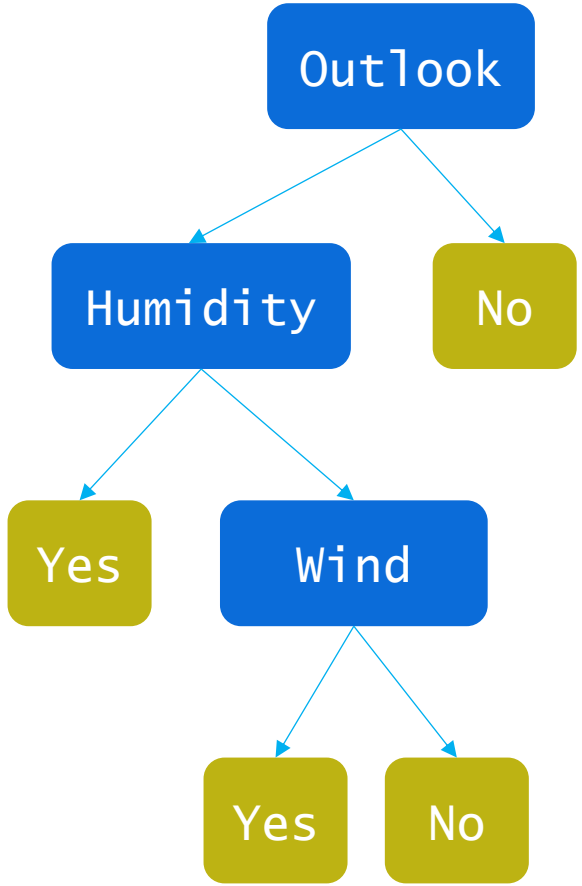
1. Create a bagging dataset
2. Create a decision tree using the bagging dataset
   1. Use a random subset of features (columns) AT EACH STEP
      1. Features not selected are treated as if they had been already processed (for this step)
      2. In a new step, all features are considered to select a new subset of features
3. Repeat from step 1 until the desired number of trees has been reached

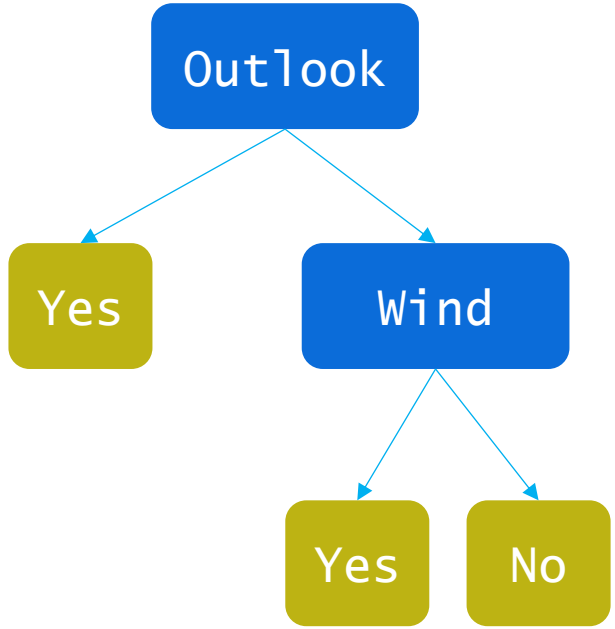
Random Forests…



https://miro.medium.com/max/1400/1*_B5HX2whbTs3DS8M6YBD_w@2x.png


Outlook
Humidity
Wind
Yes
No
Yes
No


Outlook
Humidity
No
Yes
Wind
Yes
No


Outlook
Yes
Wind
Yes
No

Random Forests…

Prediction

1. Get new data

Day	Outlook	Temperature	Humidity	Wind
13	Rain	Hot	Low	Mild

2. Present it to every tree in the forest
    1. Take note of the decision of every tree (Yes/No)

3. The decision with more votes is returned as the decision of the random forest

Play Tennis?

Yes	No
3,947	1,053

Random forests…

General issues

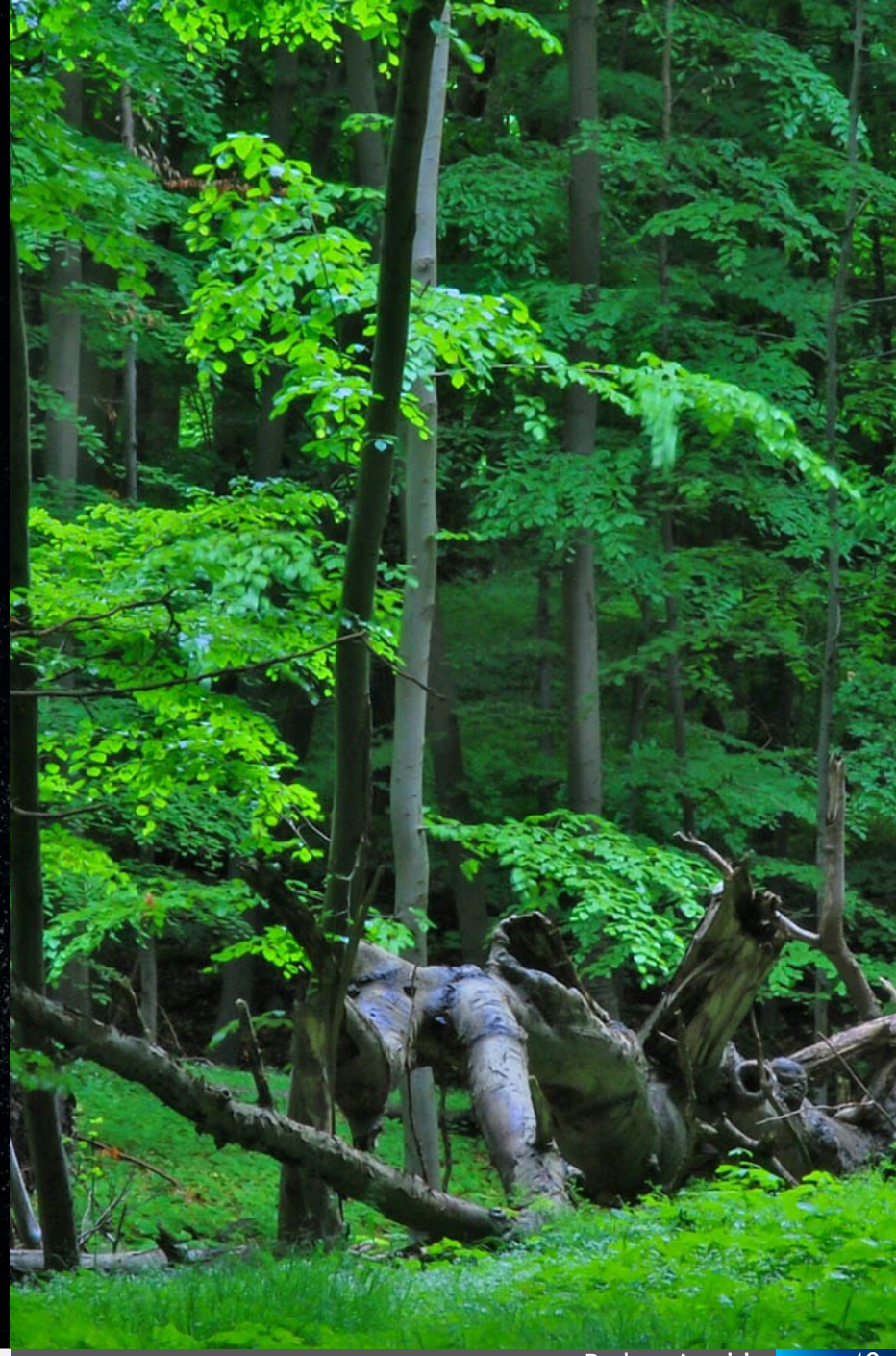
- Approximately, 1/3 of original data is not selected to create trees
  - Bagging uses replacement when selecting data
- This data is called “Out-of-Bag Dataset”
- Out-of-Bag Dataset is used as the Test Partition
- Accuracy = proportion of Out-of-Bag instances correctly classified
- Out-of-Bag Error = proportion of Out-of-Bag instances incorrectly classified

Random forests…

Class Example

- Code in scikit-learn
  - Ramdom Forest.ipynb
  - Check Github

Improving Trees and Random Forests



juresti@tec.mx 2022AD

Hyper parameters of a Decision Tree

Hyper parameters of a Decision Tree

Criterion

- **Gini** – measure of impurity
  - A pure node contains only elements of the same classs
  - Value of 0 means no impurity (no randomness)
  - Value of 1 means randomness
  - A leaf node has Gini = 0
- **Entropy** – measure of randomness
  - The more randomness, the higher the entropy
- A tree with all pure leaf nodes, usually overfits (too specific tree)
  - *min_impurity_decrease* can be set to a value higher than 0 (default) so tree does not specializes

Maximum depth

- Specifies the maximum depth of the tree
- Helps to reduce overfitting
- Usually *min_impurity_decrease* works better as it can prune the tree before it reaches the *max_depth*

Hyper parameters of a Decision Tree…

Maximum number of leaves

- Works best-first using Gini or Entropy
- Grows tree until number of leaves has been reached

Maximum number of features

- Number of features used for the next split
- If not defined, uses all features
- Features are selected randomly
- Good to prevent overfitting when combining with *max_depth*
  - If not combined, a very deep tree can use all features

Hyper parameters of a Random forest

Hyper parameters of a Random forest

Fixed number of features per step

- When creating trees, instead of using a random number of features per step, fix the number of features per step
  - Compare accuracy from different forests
  - For instances, forest using only 2 features per step vs. forest using 3 features per step
- A useful heuristic is to use the square root of the number of features to start
  - Then try some numbers below and above that initial number
- Usually known as maximum number of features (*max_features*)

Number of trees in forest

- Defines the number of trees present in the forest
- Is better to keep this value low
- Can reduce processing time

Hyper parameters of a Random forest…

Minimum samples split

- Minimum number of samples (instances) to split a nonleaf node
- Nodes with 2 or more data are usually split
- Using a larger number of samples can help generalization and reduce overfitting
  - By restricting learning very specific details
  - If the value is too large, it can lead to underfitting

Minimum number of instances in a leaf

- Defines the minimum number of samples in a node to be considered a leaf
- Can help with overfitting
- Sometimes may encourage a split that is not necessary

Hyper parameters of a Random forest…

Minimum impurity decrease

- Normally set to 0

- Can be modified to a larger value to try minimize overfitting

References

- Russell, S. & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Third Edition. Prentice Hall.

- Mitchell, T. (1997). Machine Learning. McGraw-Hill.

- Creator of the presentation design: Pedro Armijo. MS Office theme.