上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Single Table Learning

Teacher: Zheng Wang

Social Network Analysis (NIS8023)

Shanghai Jiao Tong University

Outline

- **Shallow Methods**
- Deep Learning Methods

Outline

- **Shallow Methods**
  - **Decision Tree**
  - Ensemble Learning
- Deep Learning Methods

Decision Tree

- Decision tree is the most well-known "classical" method for tabular data.

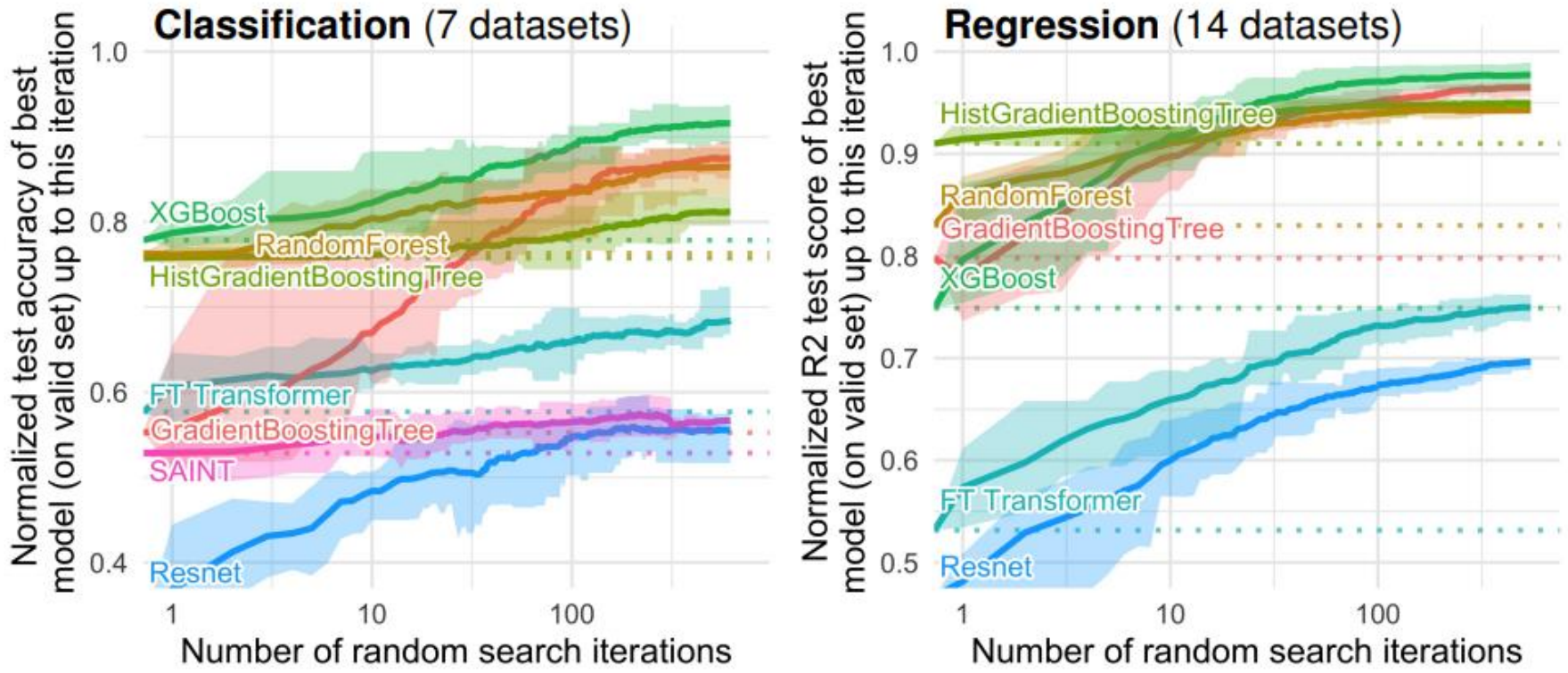


Figure 2: **Benchmark on medium-sized datasets, with both numerical and categorical features.**

Image from "Why do tree-based models still outperform deep learning on typical tabular data?." *Nips 22*.

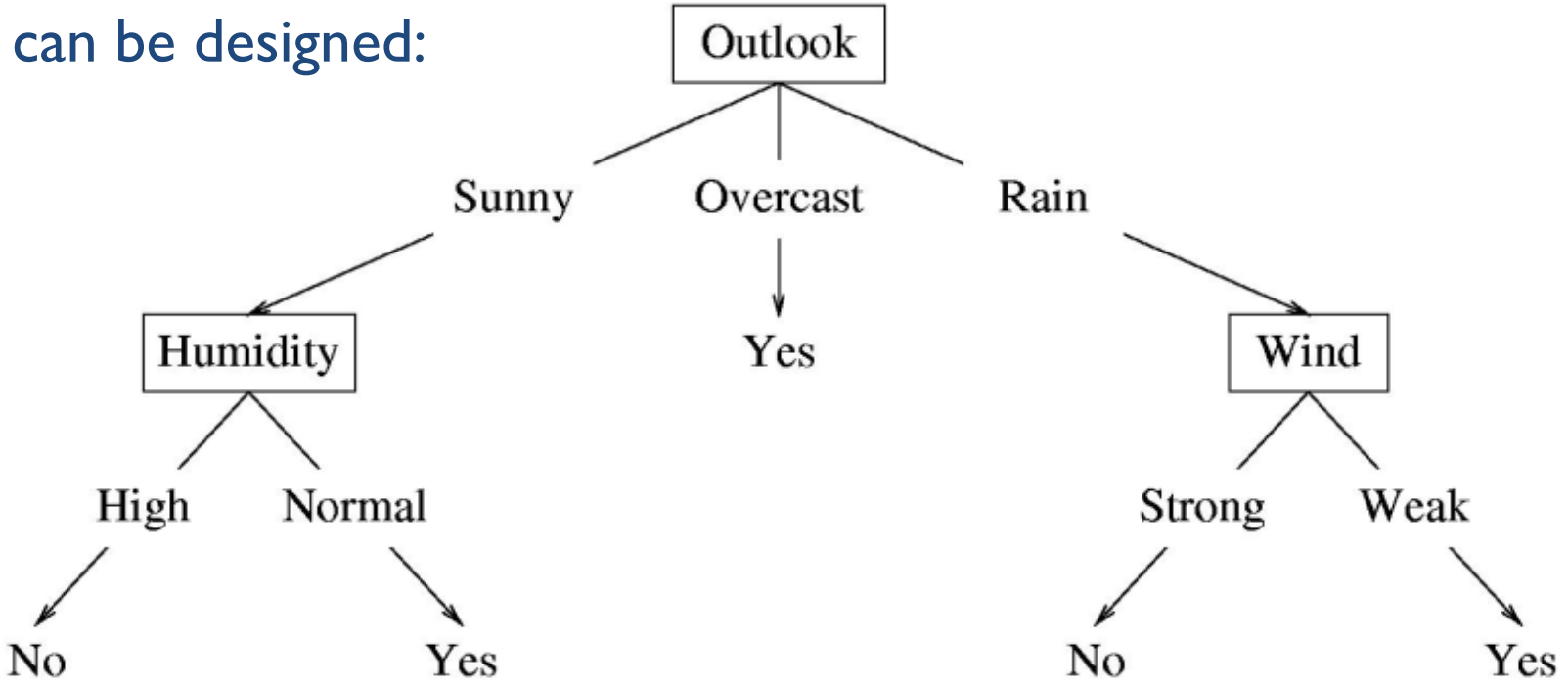
Decision Tree: An example

- Given a history record table, we want to infer whether a tennis game will be played.

Day	Outlook	Temperature	Humidity	Wind	Decision
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	Week	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

Decision Tree

- A possible decision tree can be designed:

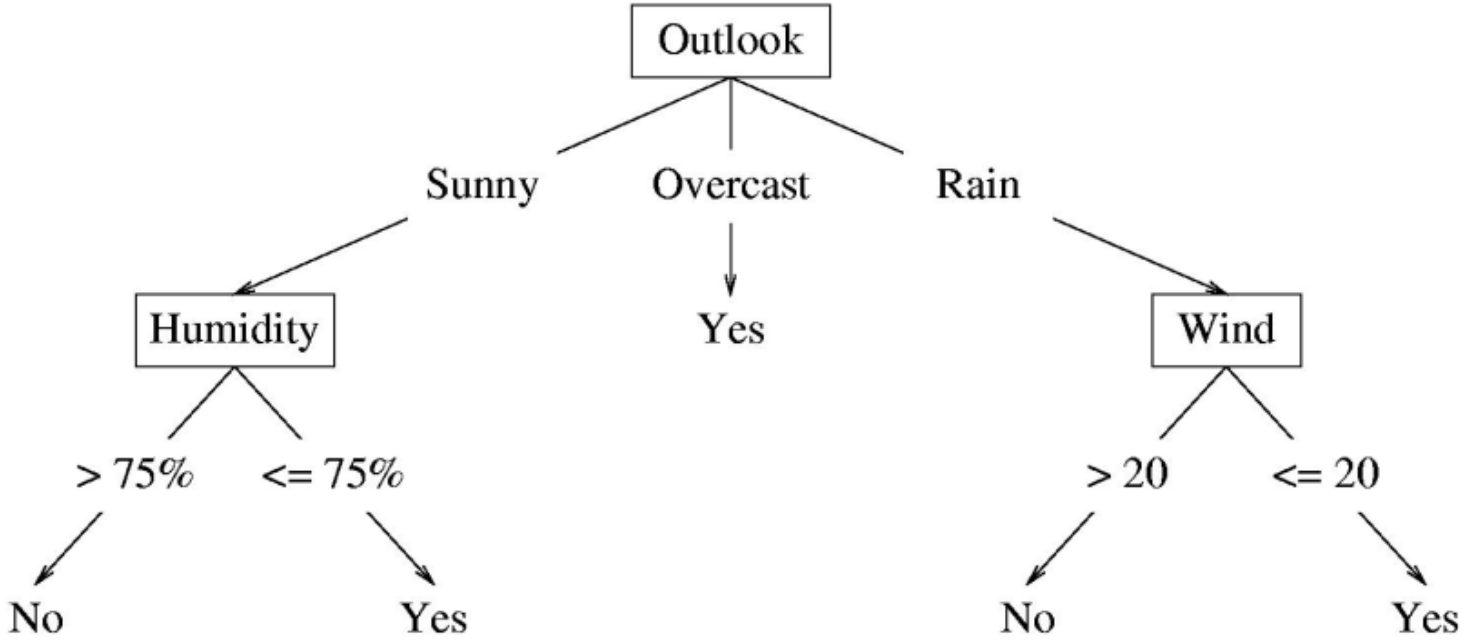


- The above tree represents the logical formula:

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

Decision Tree

- If features are continuous, internal nodes can be seen as a threshold

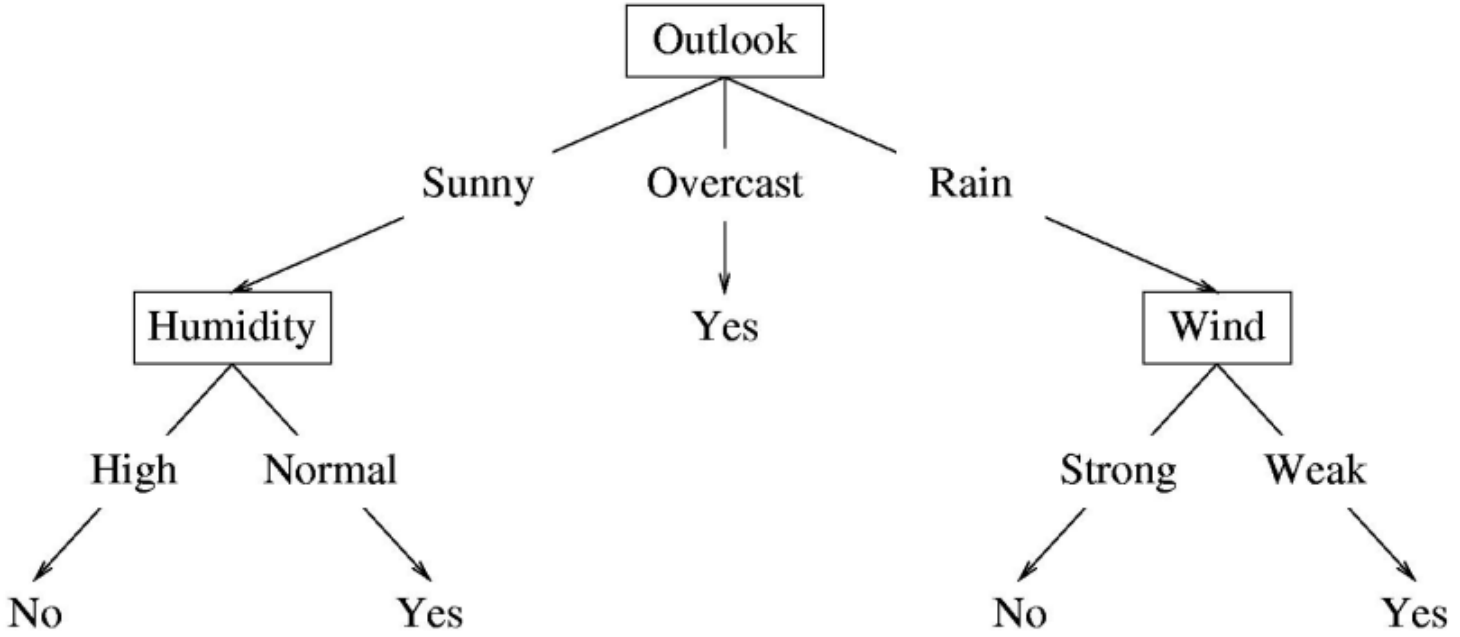


How to construct a decision tree

- A decision tree includes three types of nodes:
  - Root Node: The first decision point.
  - Non-leaf (branch) Nodes: The intermediate process.
  - Leaf Nodes: The final decision results.

Decision Tree: Generation Rules

- When generating a decision tree, a recursive process is mainly adopted:
  - Starting from the root node, it branches into several subtrees;
  - From each subtree, root nodes and subtrees are continuously generated;
  - Each subtree continues to recursively generate new subtrees until reaching leaf nodes.



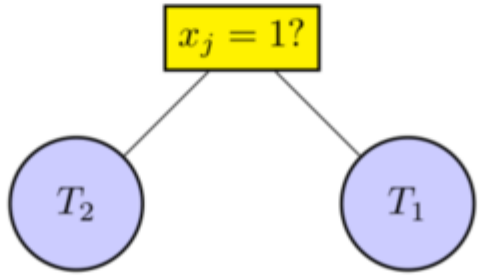
Decision Tree

- Key problem: choosing which an attribute to split a given set of examples
  - CLS: Randomly selects an attribute.
  - ID3: Selects the attribute with the highest Information Gain
  - C4.5: Uses the attribute with the highest Gain Ratio
  - CART:
    - For classification, uses the Gini Index (an impurity measure) to determine the best split.
    - For regression, uses Mean Squared Error (MSE) as the splitting criterion.

The ID3 algorithm

$\texttt{ID3}(S, A)$

INPUT: training set S, feature subset $A \subseteq [d]$
if all examples in S are labeled by 1, return a leaf 1
if all examples in S are labeled by 0, return a leaf 0
if $A = \emptyset$, return a leaf whose value = majority of labels in S
else :
  Let $j = \text{argmax}_{i \in A} \texttt{Gain}(S, i)$
  if all examples in S have the same label
    Return a leaf whose value = majority of labels in S
  else
    Let T_1 be the tree returned by $\texttt{ID3}(\{(\mathbf{x}, y) \in S : x_j = 1\}, A \setminus \{j\})$.
    Let T_2 be the tree returned by $\texttt{ID3}(\{(\mathbf{x}, y) \in S : x_j = 0\}, A \setminus \{j\})$.
    Return the tree:



Shai Shalev-Shwartz and Shai Ben-David. Understanding machine learning: From theory to algorithms, 2014.

Outline

- **Shallow Methods**
  - Decision Tree
  - **Ensemble Learning**
    - Bagging
    - Boosting
- Deep Learning Methods

Ensemble Learning

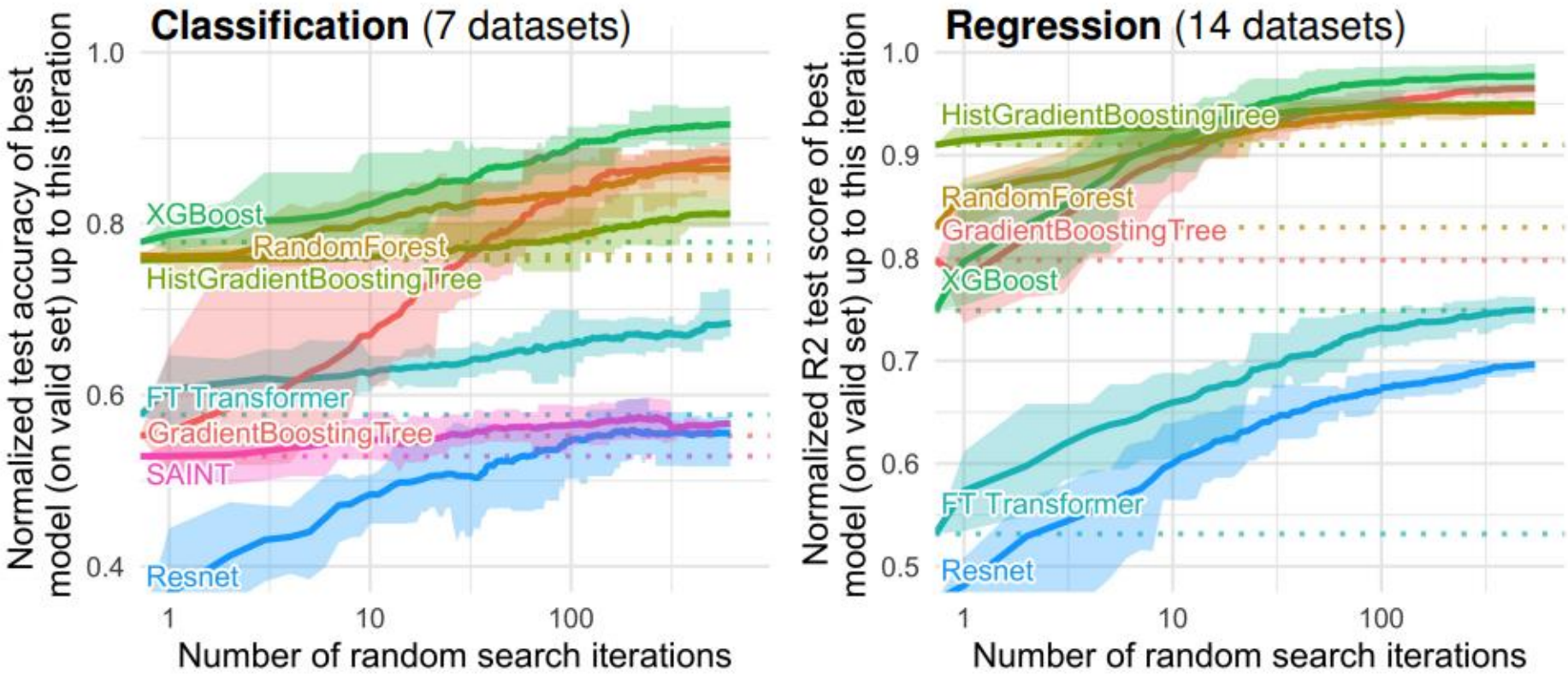


Figure 2: **Benchmark on medium-sized datasets, with both numerical and categorical features.**

Ensemble learning methods are SOTA in lots of table learning tasks!

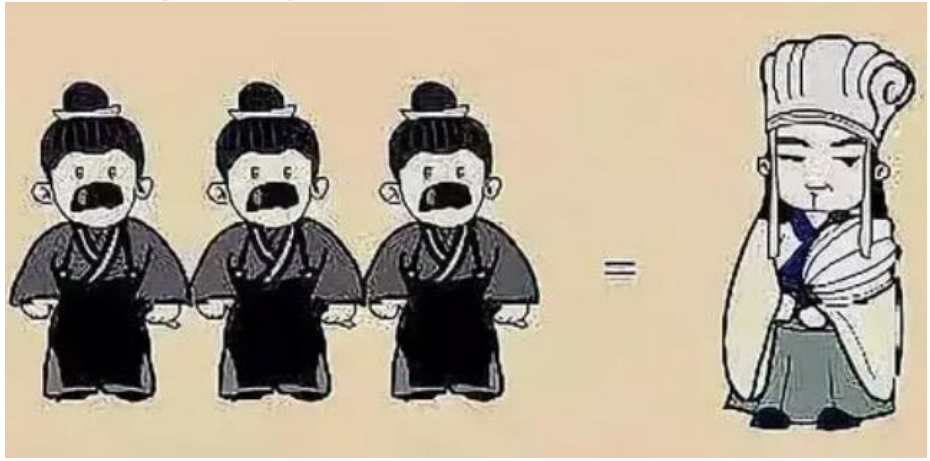
Image from "Why do tree-based models still outperform deep learning on typical tabular data?." *Nips 22*.

Why ensemble learning are powerful?

- A simple theoretical explanation: Assume there are N base classifiers, each with an error rate of ε. If we assume these classifiers are independent and apply the majority voting rule, then the error rate of the ensemble classifier is:

$$\sum_{i=N/2+1}^{N} \binom{N}{i} \varepsilon^{i} (1-\varepsilon)^{N-i}$$

which is significantly lower than the error rate ε of any single classifier.



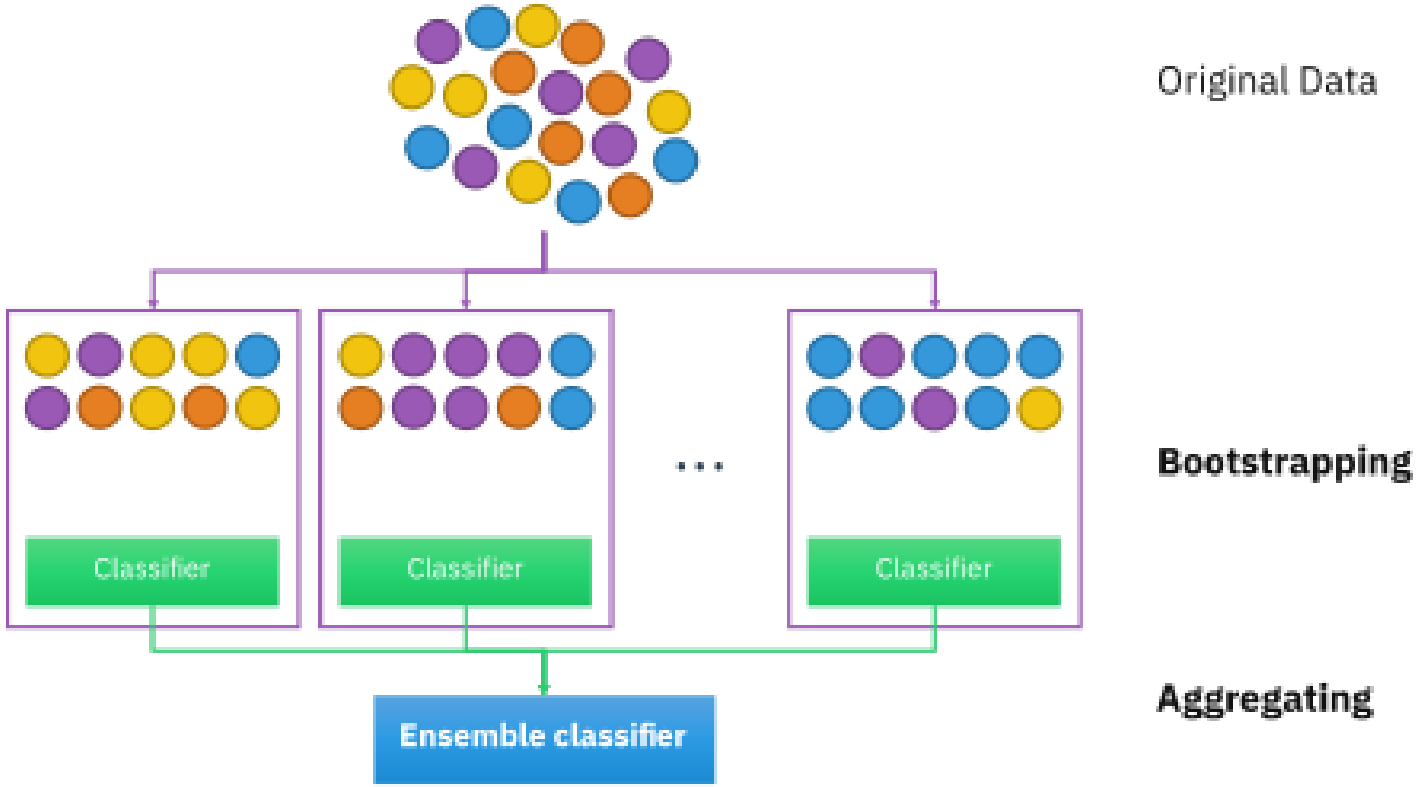
Two heads are better than one

Ensemble of decision trees

- Bagging (**B**ootstrap **agg**regat**ing**): Train multiple decision trees independently on different bootstrapped subsets of the data and average their predictions to reduce variance and improve stability.
- Boosting: Train decision trees sequentially, where each tree corrects the errors of the previous one, improving accuracy by focusing on difficult cases.

Bagging (Bootstrap aggregating)

- Learns multiple trees in parallel over independent samples of the training data
  - Bootstrapping - selecting random subsets of observations to train decision trees
  - Aggregating - using the learned decision trees to vote on the final results



Pseudocode code of Bagging

Given a standard training set D of size n

For i = 1 .. M

- Draw a sample of size $n^* < n$ from D uniformly and with replacement
- Learn classifier C_i

Final classifier is a vote of C_1 .. C_M

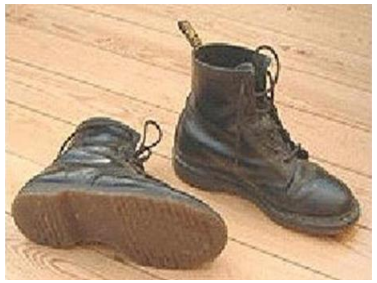
Increases classifier stability/reduces variance

Typical Bagging methods

- Bagged Decision Trees: Each decision tree is trained on a different bootstrapped subset of the data, and their predictions are aggregated (usually by majority vote for classification or averaging for regression).
- Random Forests: An extension of bagged decision trees that not only uses bootstrapped samples but also selects a random subset of features at each split, which further reduces correlation among trees.

How to understand Bagging

- The term 'Bootstrap' originally refers to the strap or loop at the back of a boot, used to help pull the boot on. The name for this algorithm comes from the phrase 'pull oneself up by one's bootstraps,' which originally described an impossible task. Over time, the meaning evolved to metaphorically signify 'improving oneself using only one's own resources, without external help.



- In the 18th-century novel Adventures of Baron Munchausen (also known as The Ultimate Warrior) by German writer Rudolf Erich Raspe, there is a story where the Baron, after falling into a lake and sinking to the bottom, manages to pull himself out by lifting himself up using the straps of his own boots in a moment of desperation.

Boosting

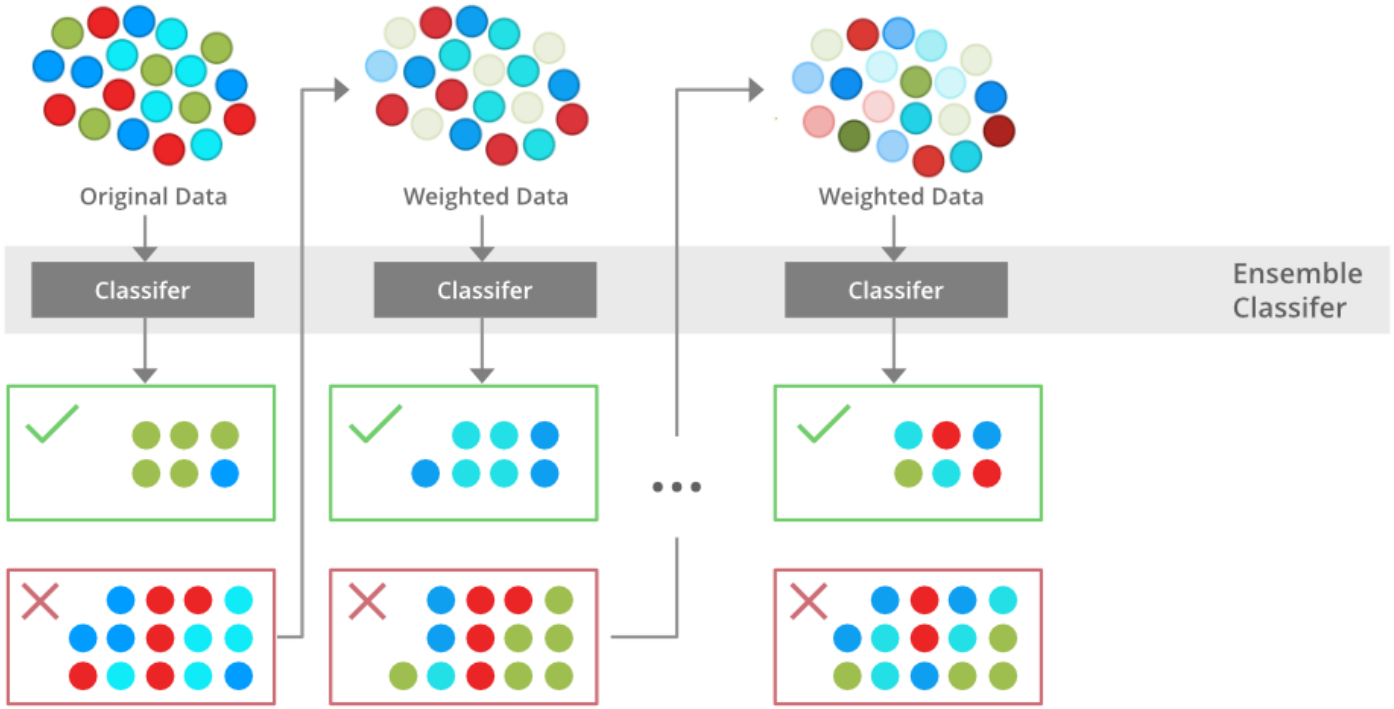
- Strong Learner
  - Take labeled data for training
  - Produce a classifier which can be very accurate
- Weak Learner
  - Take labeled data for training
  - Produce a classifier which is more accurate than random guessing

Questions: Can a set of weak learners create a single strong learner ?

Answer: Yes. Boost weak classifiers to a strong learner.

How Boosting works

- Combines the outputs of many *sequential* "weak" classifiers to produce a strong one
  1. Learns multiple weak learner sequentially, each trying to improve upon its predecessor
  2. Final classifier is weighted sum of these weak learners



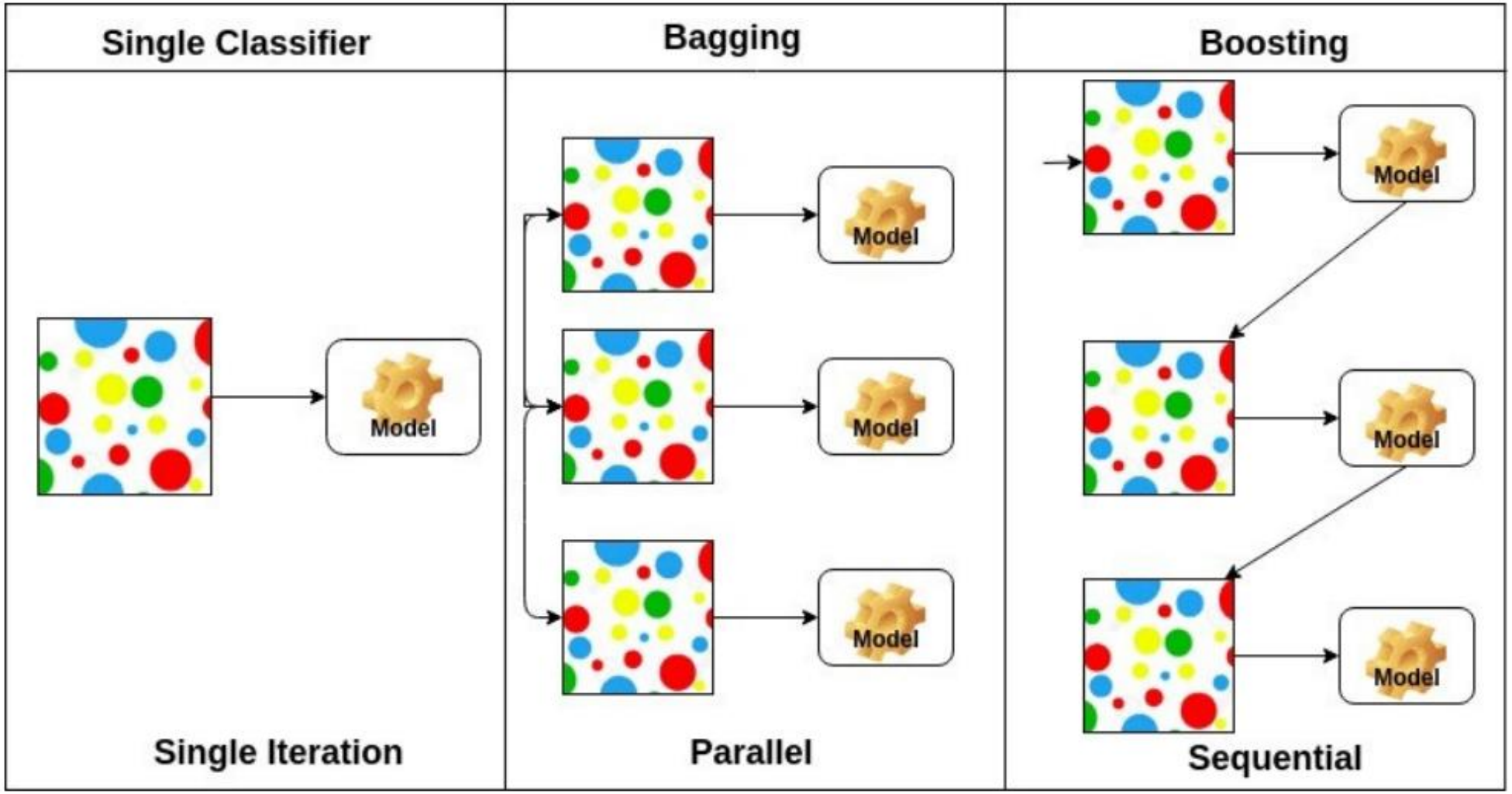
Pseudocode code of Boosting

C =0; /* counter*/
M = m; /* number of hypotheses to generate*/

1 Set same weight for all the examples (typically each example has weight = 1);

2 While (C < M)
    2.1 Increase counter C by 1.
    2.2 Generate hypothesis h_C .
    2.3 Increase the weight of the misclassified examples in hypothesis h_C
3 Weighted majority combination of all M hypotheses (weights according to how well it performed on the training set).

Comparison of Bagging and Boosting



Typical Boosting Methods

- AdaBoost: Adjusts weights of misclassified data iteratively to minimize error, optimizing a sequence of weak learners into a strong predictor.
- Gradient Boosting: Sequentially trains on residuals of previous models using gradient descent, focusing on correcting errors rather than adjusting data weights.
- XGBoost: An optimized version of gradient boosting. It leverages parallel processing for fast, scalable learning, making it effective for large datasets.

Thanks for your time.
QA.