TON DUC THANG UNIVERSITY
Faculty of Information Technology
Computer Science

Ensemble Learning

Lê Anh Cường

What is Ensemble Learning?

- Ensemble learning, in general, is a model that makes predictions based on a number of different models.
- By combining individual models, the ensemble model tends to be more flexible (less bias) and less data-sensitive (less variance).

Ensemble Models

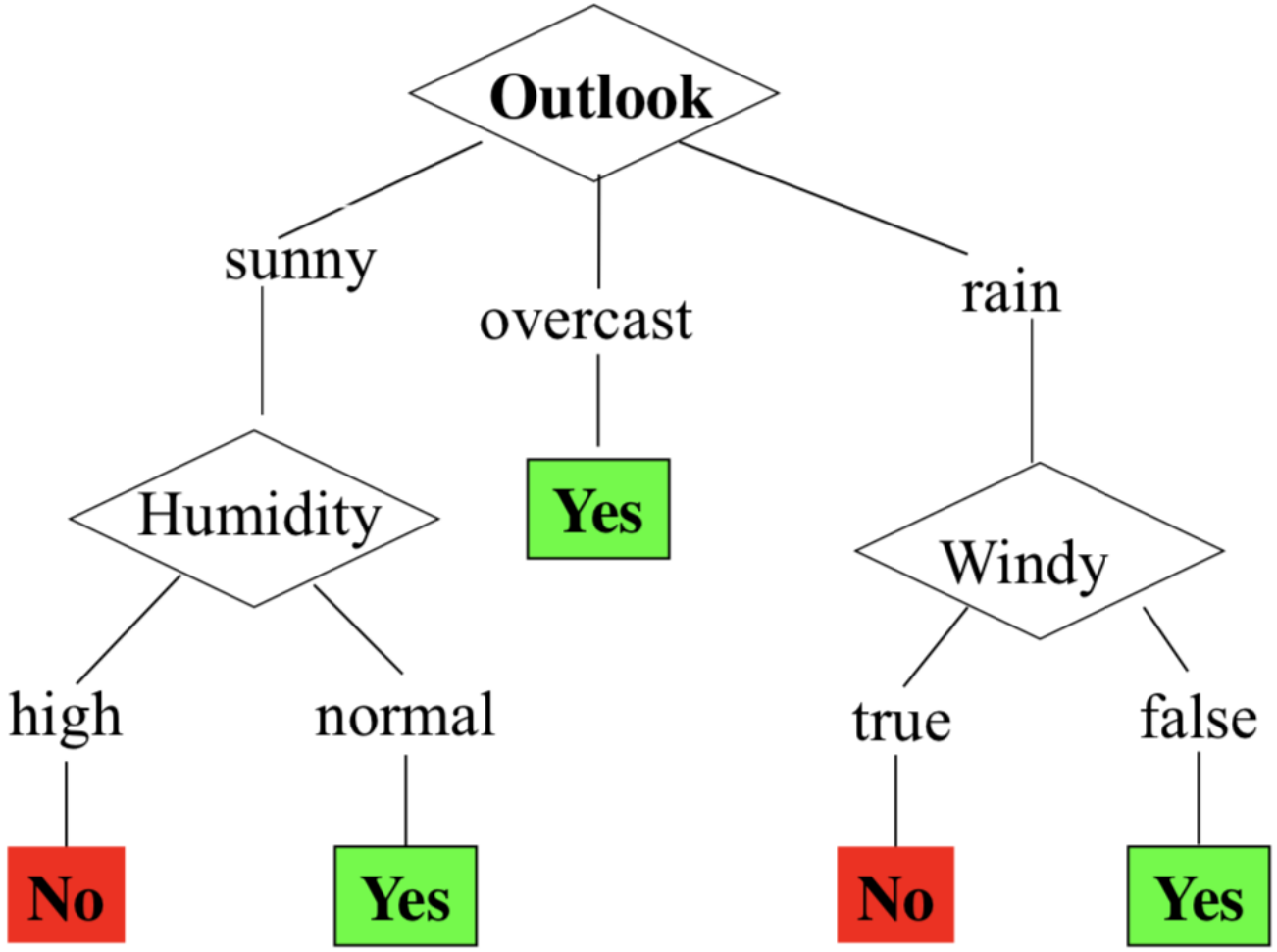
Two most popular ensemble methods
are **bagging** and **boosting**.

- **Bagging:** Training a bunch of individual models in a parallel way. Each model is trained by a random subset of the data.
- **Boosting:** Training a bunch of individual models in a sequential way. Each individual model learns from mistakes made by the previous model.

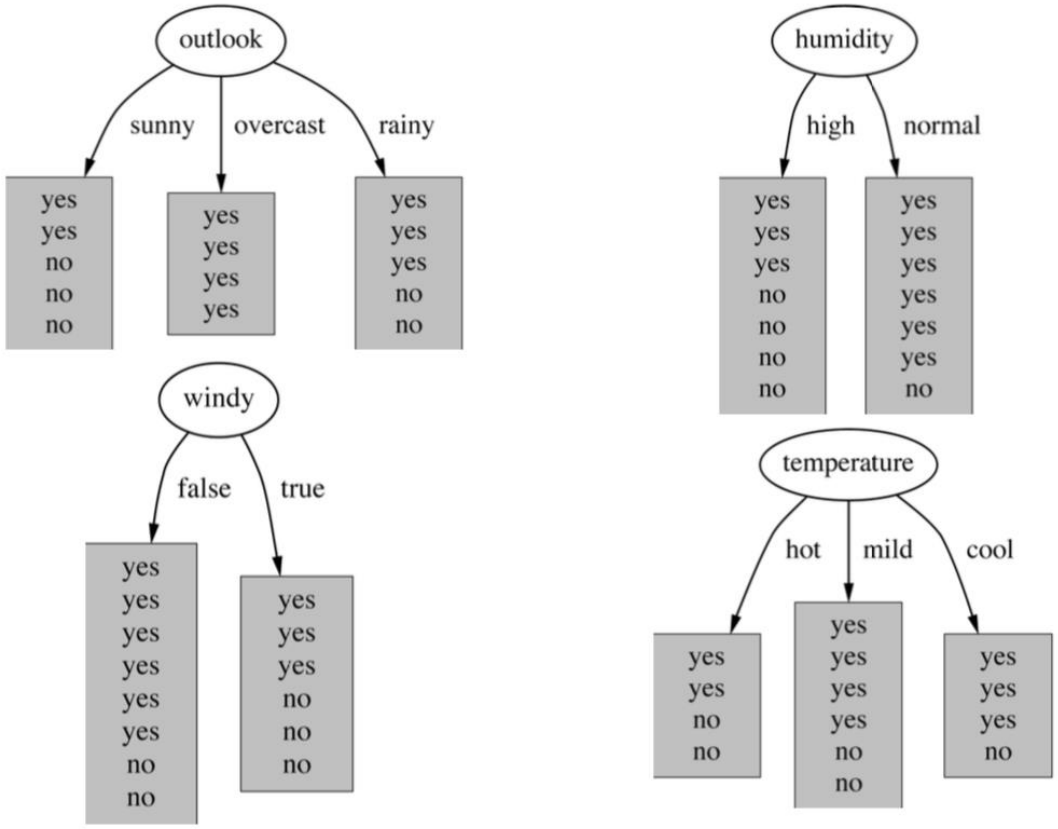
Decision Tree

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Decision Tree



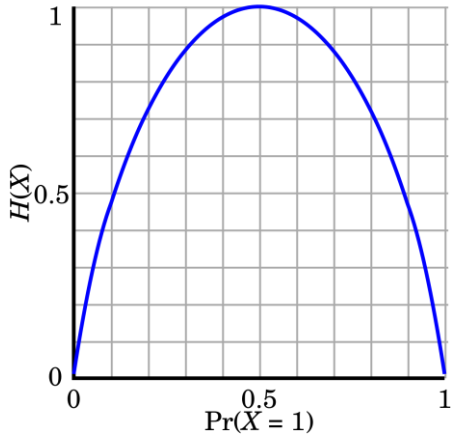
Choose attributes for partitioning



Entropy

Entropy is a **measure** of the unpredictability of the state, or equivalently, of its average **information** content.

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

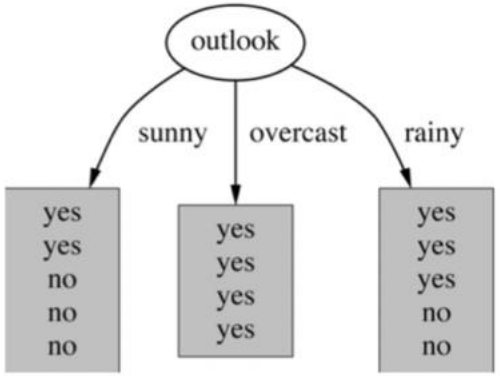


p=0.7, then

$$\begin{aligned}
\mathrm{H}(X) &= -p \log_2 (p) - q \log_2 (q) \\
&= -0.7 \log_2 (0.7) - 0.3 \log_2 (0.3) \\
&\approx -0.7 \cdot (-0.515) - 0.3 \cdot (-1.737) \\
&= 0.8816 < 1
\end{aligned}$$

$$\begin{aligned}
\mathrm{H}(X) &= -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i) \\
&= -\sum_{i=1}^{2} \frac{1}{2} \log_2 \frac{1}{2} \\
&= -\sum_{i=1}^{2} \frac{1}{2} \cdot (-1) = 1
\end{aligned}$$

Entropy of partition



- "Outlook" = "Sunny":

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -2/5\log(2/5) - 3/5\log(3/5) = 0.971 \text{ bits}$$

- "Outlook" = "Overcast":

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1\log(1) - 0\log(0) = 0 \text{ bits}$$

*chú ý : log(0) không xác định nhưng 0*log(0) là 0*

- "Outlook" = "Rainy":

$$\text{info}([3,2]) = \text{entropy}(3/5,2/5) = -3/5\log(3/5) - 2/5\log(2/5) = 0.971 \text{ bits}$$

- thông tin của thuộc tính outlook:

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$
$$= 0.693 \text{ bits}$$

Information gain of attributes

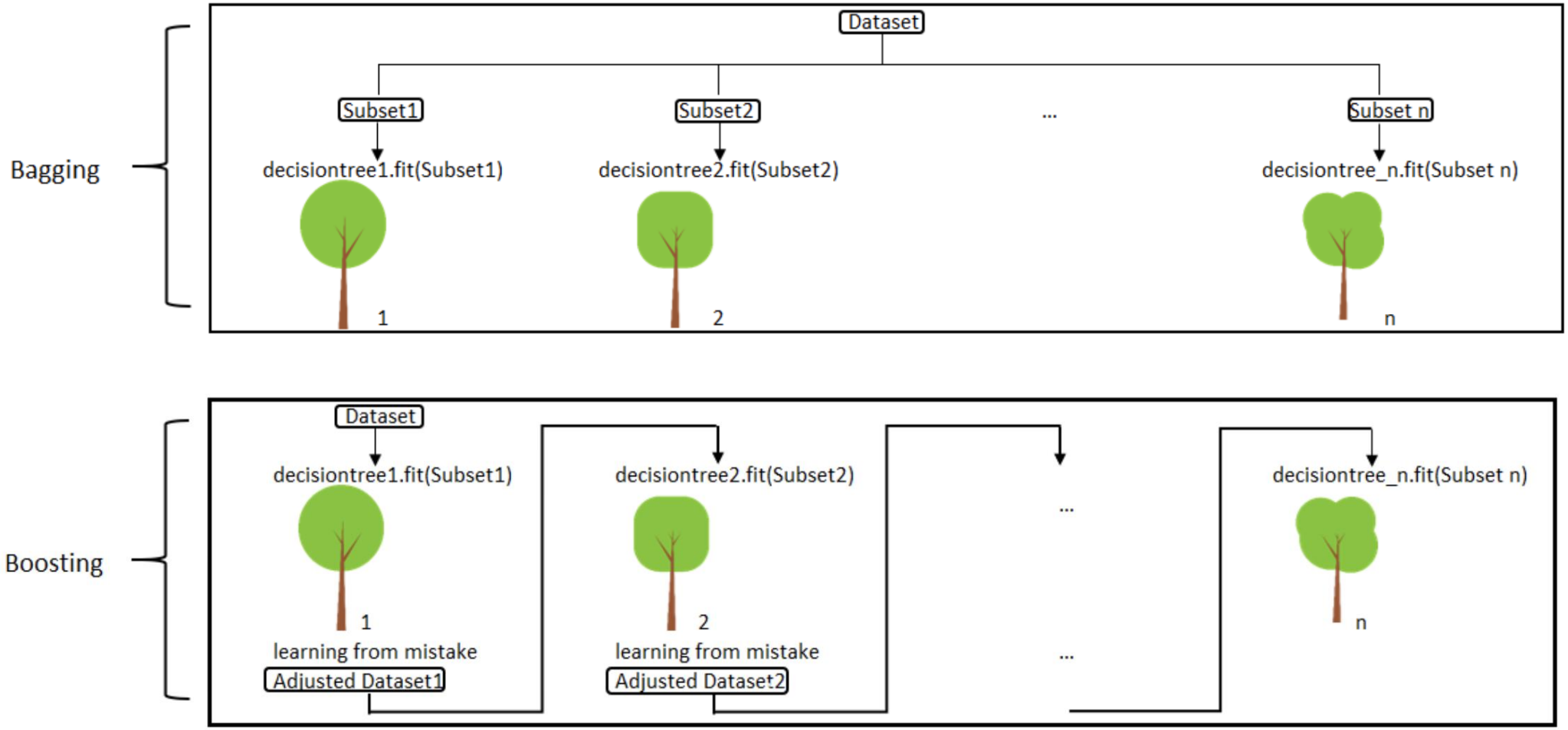
$$\text{gain}(\text{"Outlook"}) = 0.247 \text{ bits}$$

$$\text{gain}(\text{"Temperature"}) = 0.029 \text{ bits}$$

$$\text{gain}(\text{"Humidity"}) = 0.152 \text{ bits}$$

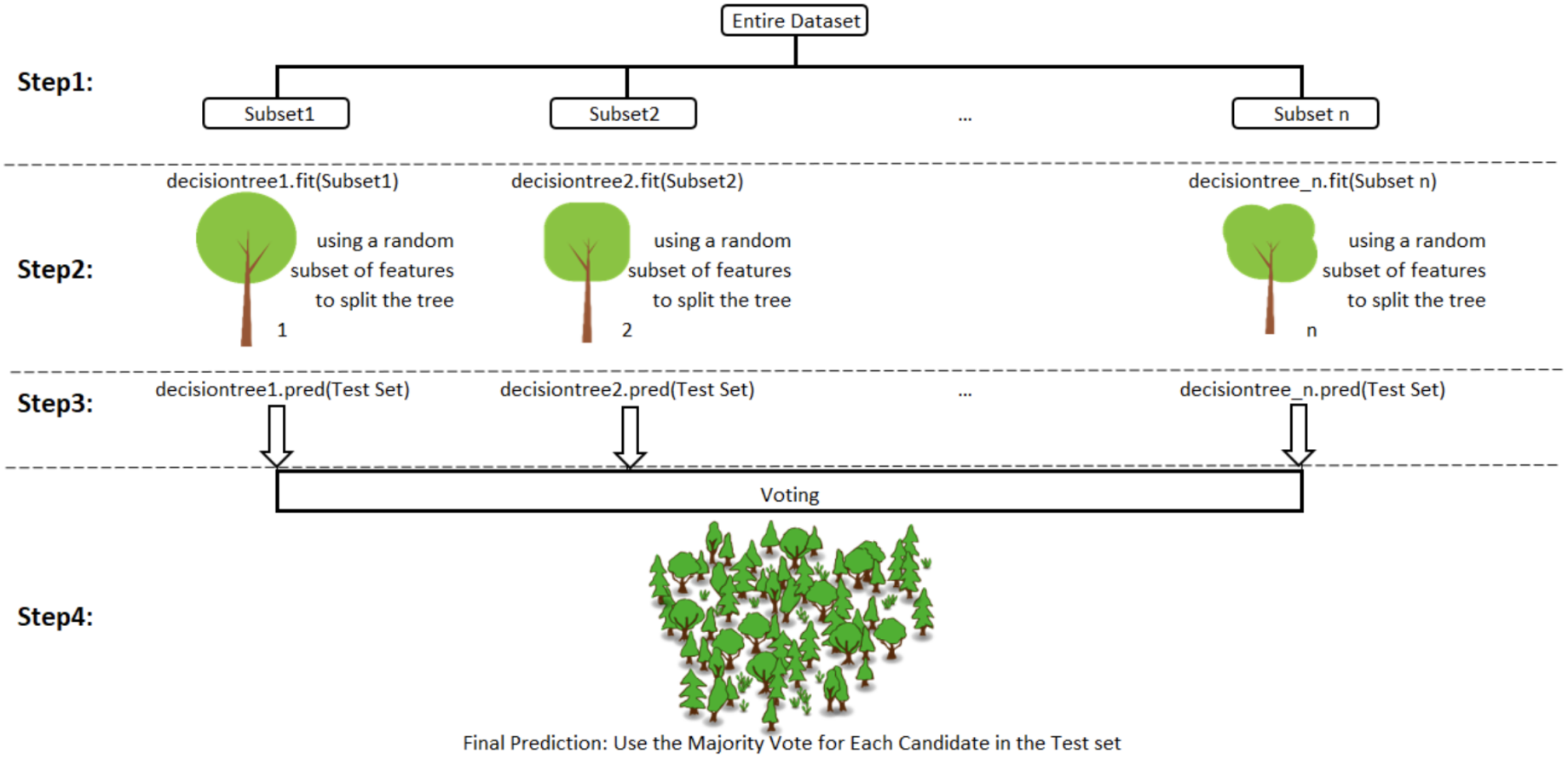
$$\text{gain}(\text{"Windy"}) = 0.048 \text{ bits}$$

Bagging vs Boosting



Random Forest

- Random forest is an ensemble model using bagging as the ensemble method and decision tree as the individual model.



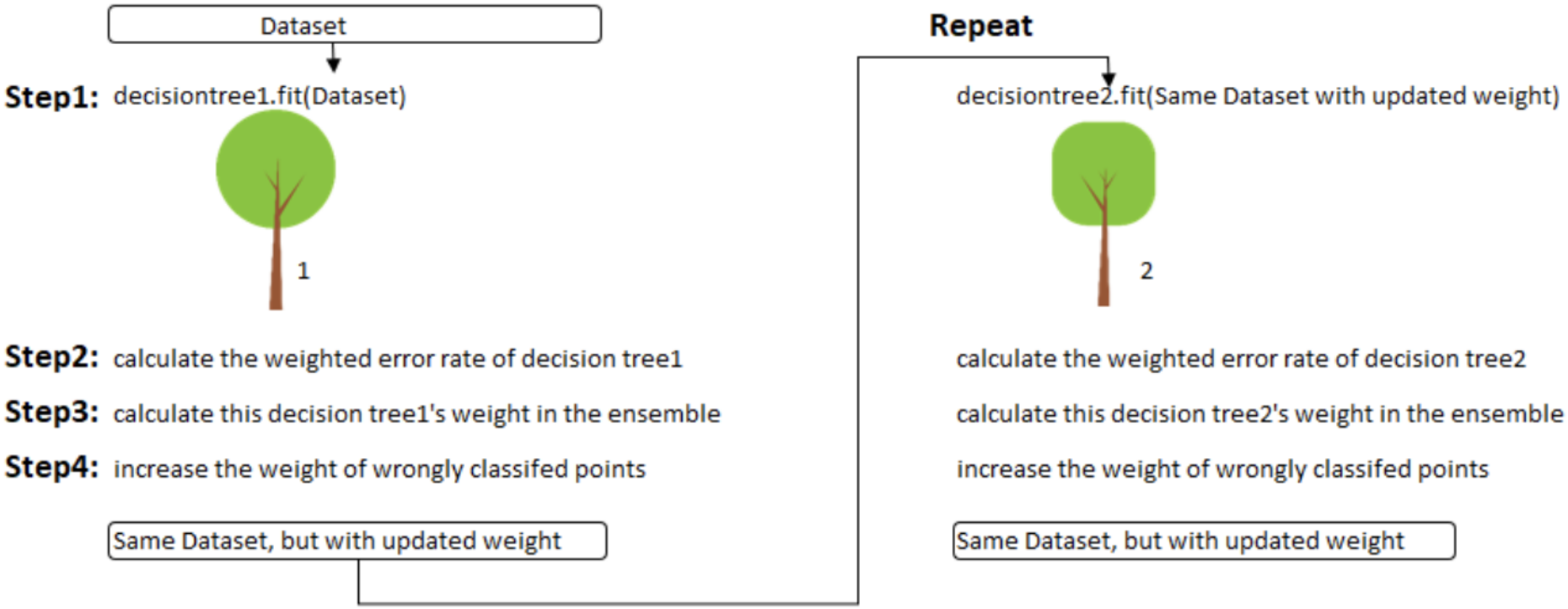
Random Forest

- Step 1: **Select n (e.g. 1000) random subsets** from the training set
- Step 2: **Train n (e.g. 1000) decision trees**
  - one random subset is used to train one decision tree
  - the optimal splits for each decision tree are based on a random subset of features (e.g. 10 features in total, randomly select 5 out of 10 features to split)
- Step 3: **Each individual tree predicts** the records/candidates in the test set, independently.
- Step 4: **Make the final prediction**

AdaBoost (Adaptive Boosting)

- AdaBoost is a boosting ensemble model and works especially well with the decision tree.
- Boosting model's key is learning from the previous mistakes, e.g. misclassification data points.

AdaBoost (Adaptive Boosting)



AdaBoost (Adaptive Boosting)

Initialization step: for each example x, set

$D(x)=\frac{1}{N}$, where N is the number of examples

Iteration step (for t=1…T):

1.Find best weak classifier $h_t(x)$ using weights $D_t(x)$

2. Compute the error rate ε_t as

$$\varepsilon_t = \sum_{i=1}^{N} D(x_i).I[y_i \neq h_t(x_i)]$$

3. assign weight α_t to classifier $h_t(x)$ in the final hypothesis

$$\alpha_t = \log((1-\varepsilon_t)/\varepsilon_t)$$

4. For each x_i , $D(x_i) = D(x_i).\exp(\alpha_t.I[y_i \neq h_t(x_i)])$

5.Normalize $D(x_i)$ so that $\sum_{i=1}^{N} D(x_i) = 1$

$$f_{final}(x) = sign\ [\sum \alpha_t\ h_t(x)]$$

AdaBoost (Adaptive Boosting)

- Step 0: **Initialize the weights** of data points. if the training set has 100 data points, then each point's initial weight should be 1/100 = 0.01.
- Step 1: **Train** a decision tree
- Step 2: **Calculate the weighted error rate (e)** of the decision tree. **The weighted error rate (e)** is just how many wrong predictions out of total and you treat the wrong predictions differently based on its data point's weight. **The higher the weight**, **the more the corresponding error will be weighted** during the calculation of the (e).

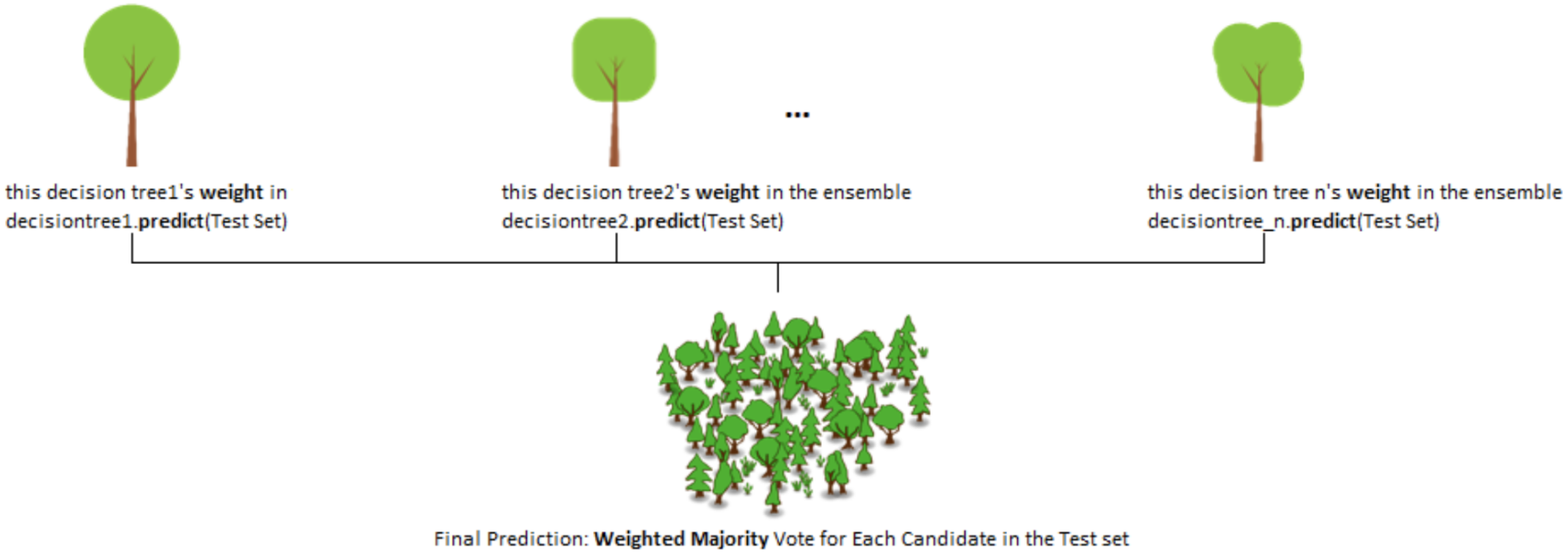
AdaBoost (Adaptive Boosting)

- Step 3: **Calculate this decision tree's weight** in the ensemble
  - the weight of this tree = learning rate * log((1 — e) / e)
  - the higher weighted error rate of a tree, the less decision power the tree will be given during the later voting
  - the lower weighted error rate of a tree, the higher decision power the tree will be given during the later voting
- Step 4: **Update weights** of wrongly classified points

  the weight of each data point =
  - if the model got this data point correct, the weight stays the same
  - if the model got this data point wrong, the new weight of this point = old weight * np.exp(weight of this tree)
- Step 5: **Repeat** Step 1(until the number of trees we set to train is reached)

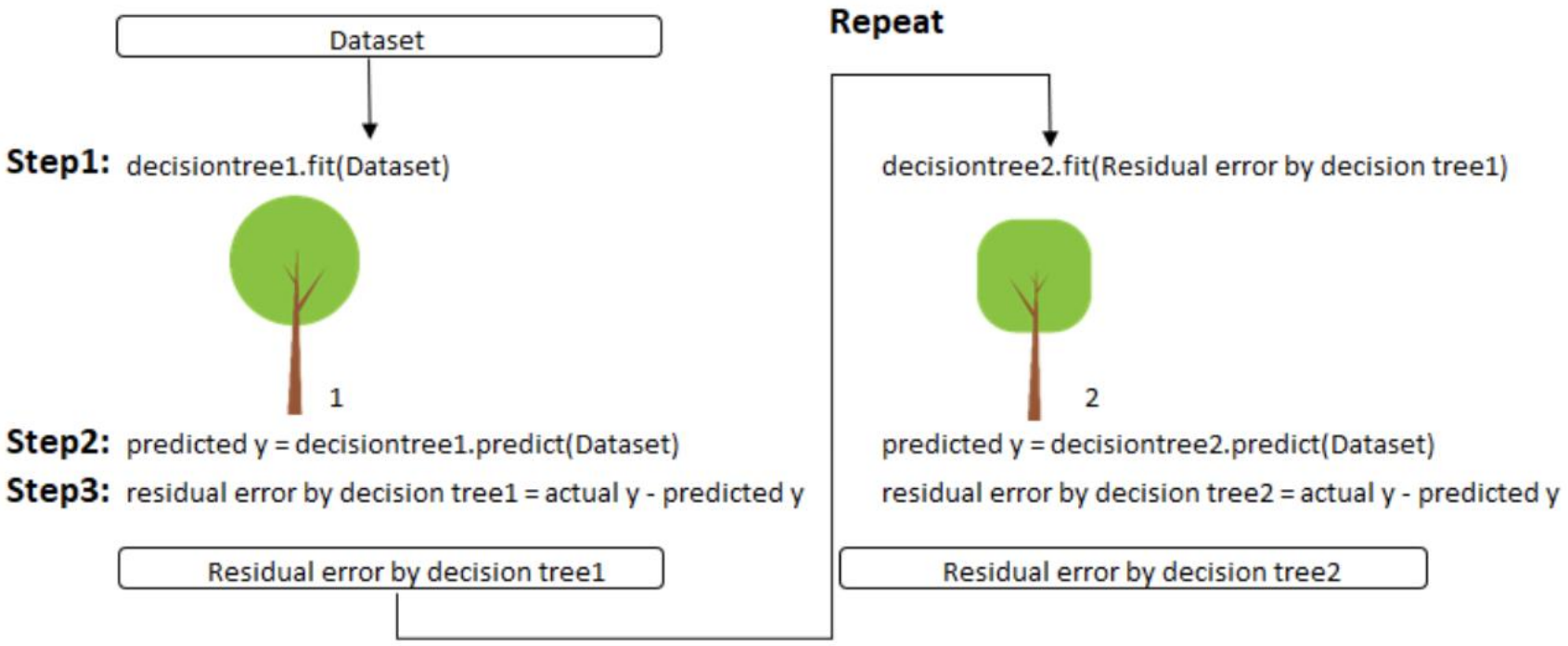
AdaBoost (Adaptive Boosting)

- Step 6: **Make the final prediction**
  - The AdaBoost makes a new prediction by adding up the weight (of each tree) multiply the prediction (of each tree). Obviously, the tree with higher weight will have more power of influence the final decision.



Gradient Boosting

- Gradient Boosting learns from the mistake — residual error directly, rather than update the weights of data points.



Gradient Boosting

- Step 1: T**rain** a decision tree
- Step 2: **Apply** the decision tree just trained to predict
- Step 3: **Calculate** the residual of this decision tree, Save residual errors as the new y
- Step 4: **Repeat** Step 1 (until the number of trees we set to train is reached)
- Step 5: **Make the final prediction**

Algorithm 1 Friedman's Gradient Boost algorithm

Inputs:

- input data $(x, y)_{i=1}^{N}$
- number of iterations M
- choice of the loss-function $\Psi(y, f)$
- choice of the base-learner model $h(x, \theta)$

$$\theta_n = \theta_{n-1} - \eta \frac{\partial}{\partial \theta} L(\theta_{n-1})$$

Algorithm:

1: initialize $\widehat{f}_0$ with a constant
2: **for** $t = 1$ to M **do**
3: compute the negative gradient $g_t(x)$
4: fit a new base-learner function $h(x, \theta_t)$
5: find the best gradient descent step-size ρ_t:

$$\rho_t = \arg\min_{\rho} \sum_{i=1}^{N} \Psi\left[y_i, \widehat{f}_{t-1}(x_i) + \rho h(x_i, \theta_t)\right]$$

6: update the function estimate:

$$\widehat{f}_t \leftarrow \widehat{f}_{t-1} + \rho_t h(x, \theta_t)$$

7: **end for**

Gradient Boost

- Initialize $F_0(x) = \arg\min_\rho \sum_{i-1}^{N} L(y_i, \rho)$
- For $m = 1$ to M do:
  - Step 1. Compute the negative gradient

$$\tilde{y}_i = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F_{x_i}}\right]$$

  - Step 2. Fit a model

$$\alpha_m = \arg\min_{\alpha,\beta} \sum_{i=1}^{N} [\tilde{y} - \beta h(x_i; \alpha_m)]^2$$

  - Step 3. Choose a gradient descent step size as

$$\rho_m = \arg\min_{\rho} \sum_{i-1}^{N} L(y_i, Fm - 1(x_i) + \rho h(x_i; \alpha))$$

  - Step 4. Update the estimation of $F(x)$

$$F_m(x) = F_{m-1}(x) + \rho_m h(x; \alpha_m)$$

- end for
- Output the final regression function $F_m(x)$

Gradient Descent

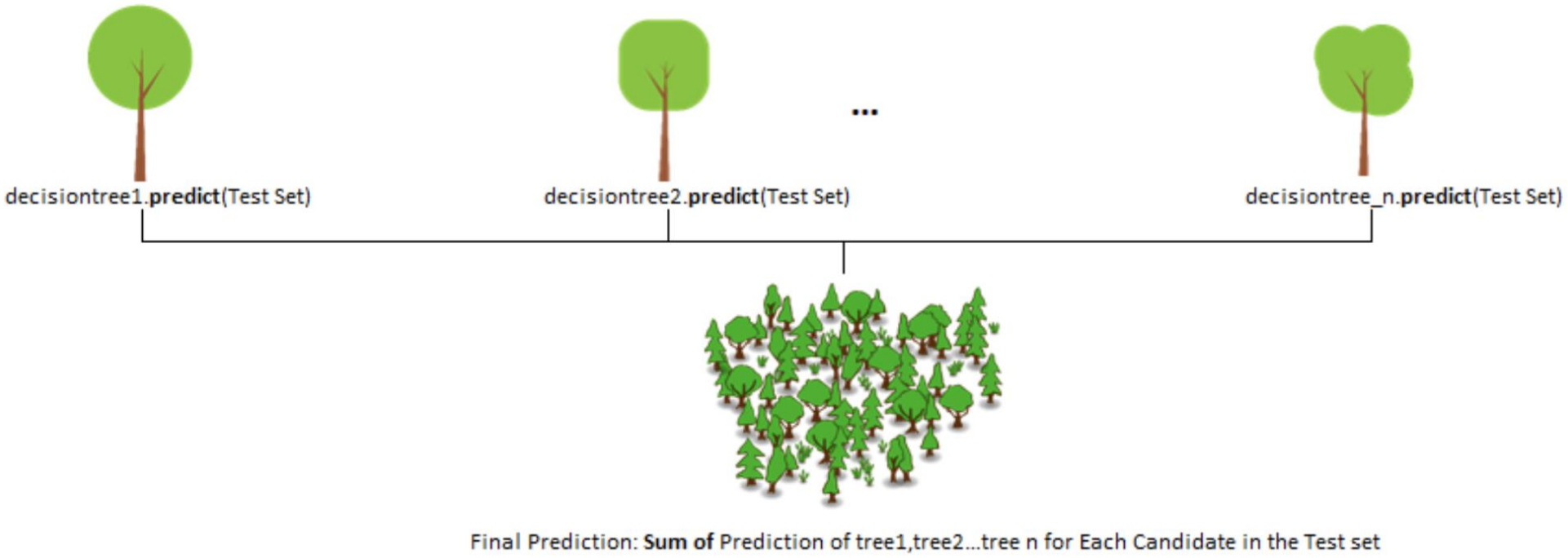
$$\theta_n = \theta_{n-1} - \eta \frac{\partial}{\partial \theta} L(\theta_{n-1})$$

Consider model boosting is a function of w, then each learner is a parameter *w*.

$$W_n = W_{n-1} - \eta \frac{\partial}{\partial w} L(W_{n-1})$$

Gradient Boosting

- The Gradient Boosting makes a new prediction by simply adding up the predictions (of all trees)



References

- https://towardsdatascience.com/basic-ensemble-learning-random-forest-adaboost-gradient-boosting-step-by-step-explained-95d49d1e2725
- https://viblo.asia/p/gradient-boosting-tat-tan-tat-ve-thuat-toan-manh-me-nhat-trong-machine-learning-YWOZrN7vZQ0