

# Machine Learning Principles

Class13 : October 20

Decision Tree & Random Forest

Instructor: Diana Kim



# Today's Lecture

1. Decision Tree Model as a Combining Model
2. Training a Decision Tree (greedy divide & conquer)
  - feature selection : mutual information / Gini index
  - split / stop
3. Complexity Control: stopping criterion & pruning
4. Pros & Cons for tree methods
5. Ensemble method, Random Forest
  - bagging
  - random features

# [1] Combining Models

We have explored a wide range of models to solve regression and classification problems.

It is often found that improved performance can be obtained **by combining multiple models together in some way**, instead of just using a single model in isolation.

## [2] Combining Models (homework2-example)

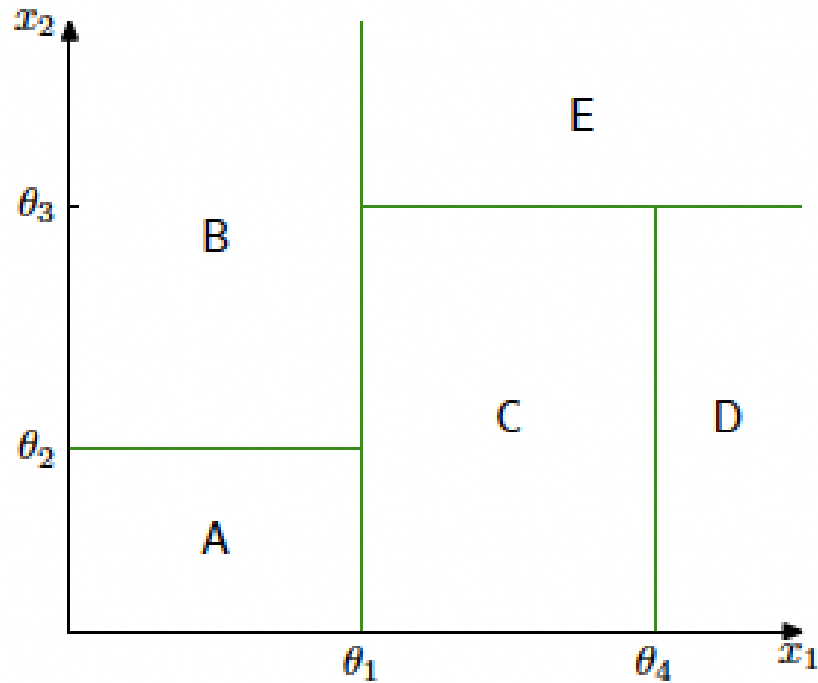
**3. [Learning Sinusoidal Functions]** You will recover a sinusoidal function  $f(x)$  from noisy data by using MMSE regression. Data samples and required specifications are given below.

- train and test data files: train.npz, test.npz, and readme.txt
- use ten polynomial basis functions:  $1, x, x^2, \dots, x^8, x^9$
- use MSE (Mean Square Error) for the performance metric:  $\frac{1}{N} \sum_{i=1}^N (\bar{w}^t \phi(x_i) - y_i)^2$
- use five-fold cross-validation for the selection of the parameter of ridge regression  $\lambda$

**3.3 Plot the two models:  $w(\lambda = 0)$  and  $w(\lambda^*)$  over the range  $0 \leq x \leq 1$ . The five models can be averaged: `w=np.mean(w, axis=0)`.**

❖ The five models are averaged; they are called committee.

### [3] Combining Models (tree-based model)



[Bishop Figure 14.5]

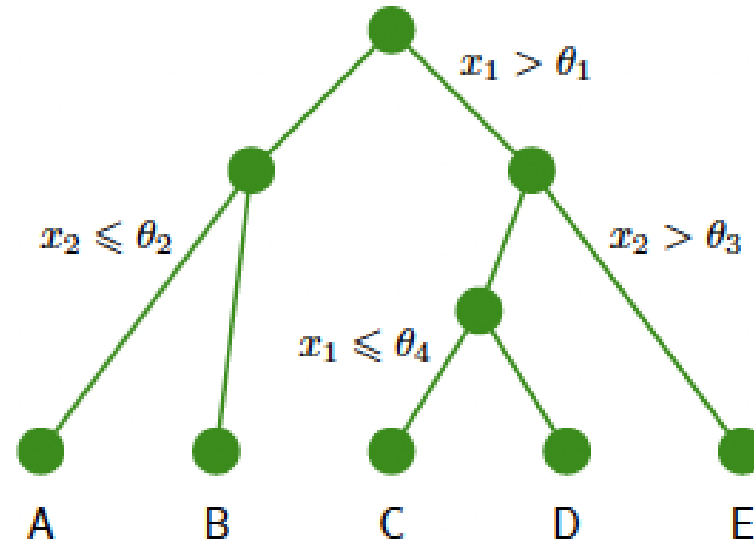
- Today we learn a combining model, called “tree-based model”.
- It partitions input space into cuboid regions, whose edges are aligned with the axes, and assigns a classification/ regression rule to each region.

Q: how many classification rules?

## [4] Combining Models (tree-based model)

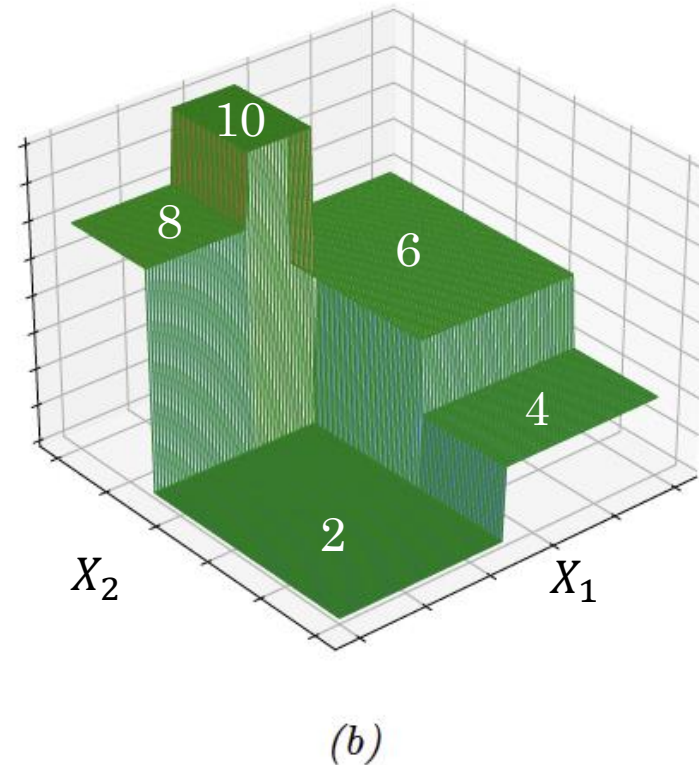
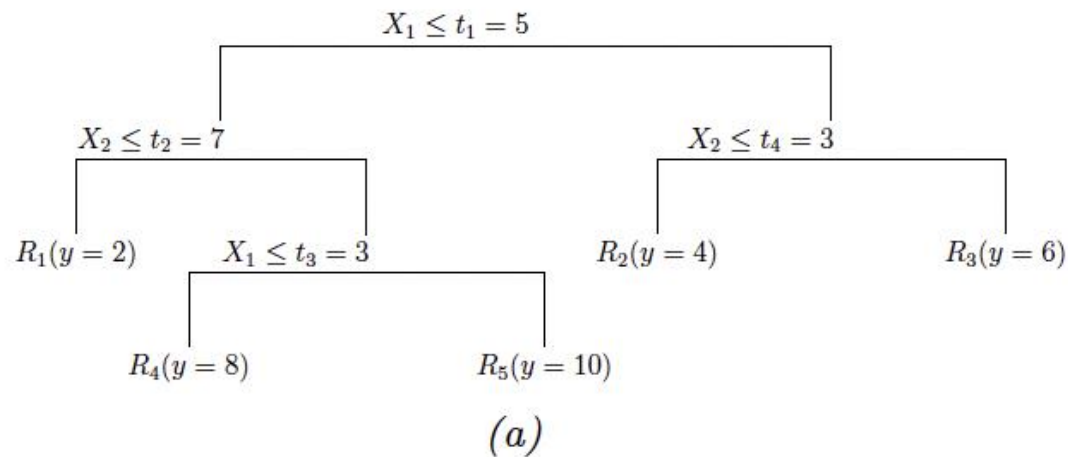
Binary tree corresponding to the partitioning of input space shown in Figure 14.5.

[Bishop Figure 14.6]



- The process of selecting a specific model given a new input  $x$ , can be described by a sequential decision: the traversal of a binary tree describing nested decision rules.

## [5] Combining Models (tree-based model, regression example)



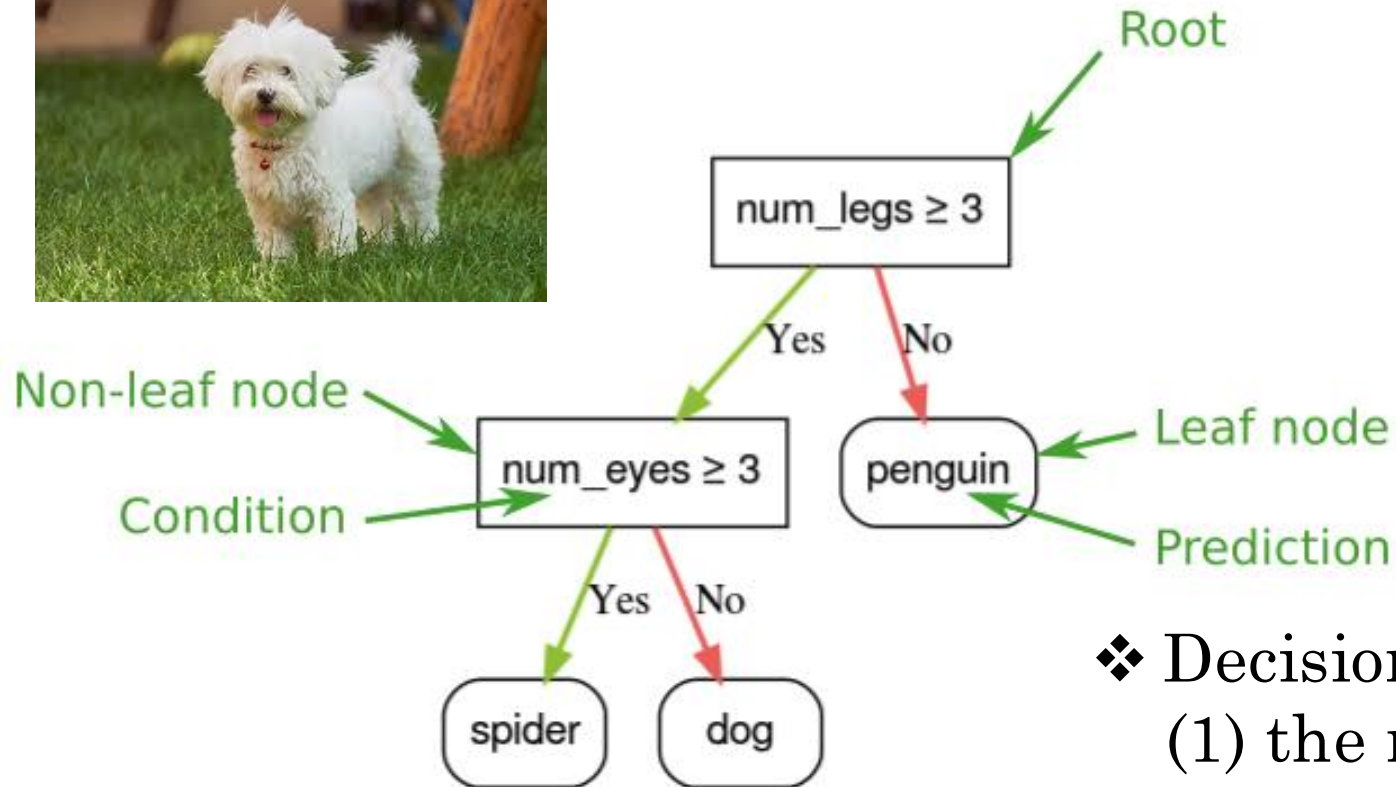
Q: how many regression model here?

Q: how the model selection is performed?

[Figure 18.1 Murphy]

## [6] Combining Models (decision tree)

<https://developers.google.com/machine-learning/decision-forests/decision-trees>



- ❖ Decision Tree models presents
  - (1) the nested rules to select a model
  - (2) the combining models.



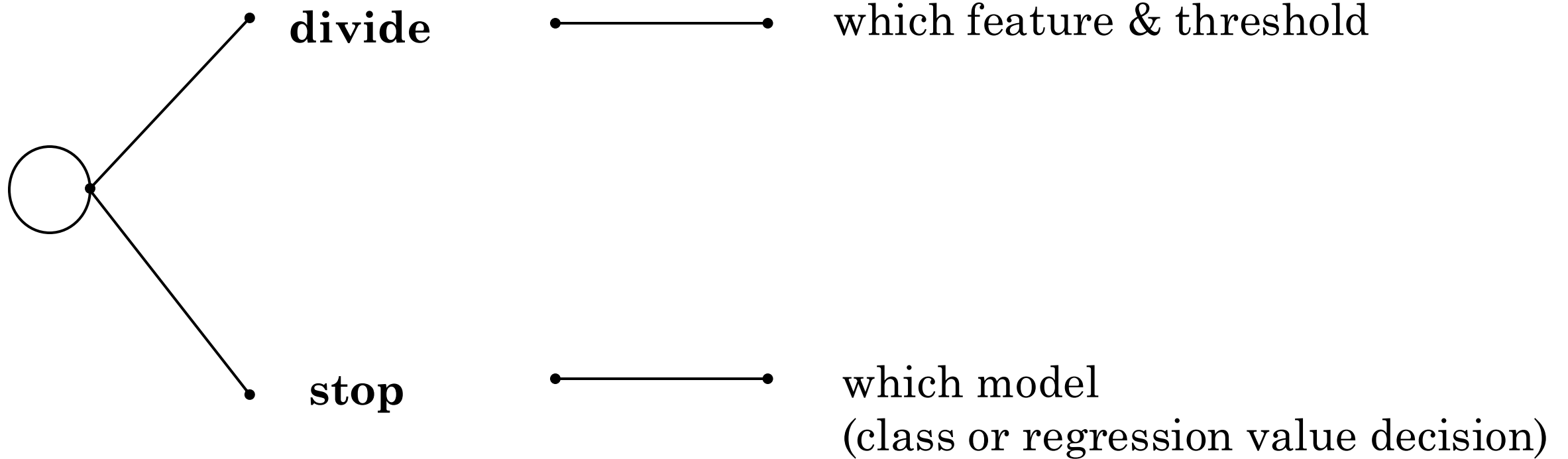
- Training Decision Tree:  
(focus on binary decision tree)  
(recursive division of the data regions into two subregions at a time)

# [1] Learning Decision Tree

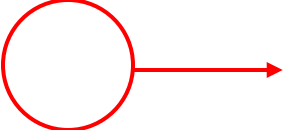
❖ Learning a Decision Tree is to learn the recursive rules for division.

## [2] Learning Decision Tree

❖ Learning a Decision Tree is to learn the recursive rules for division.



### [3] Learning Decision Tree (pseudo code)

def **Tree** ( $X$ ):  a set of data points!

    if divide ( $X$ ):

        assign a majority class

    break

$f$  = feature selection to split ( $X$ )

$\eta$  = threshold selection to split the feature ( $X, f$ )

$X_1, X_2$  = divide ( $X, f, \eta$ )

**Tree**( $X_1$ )

**Tree**( $X_2$ )

## [4] Learning Decision Tree (Restaurant Waiting Prediction Example)

Q: building a structured/nested rule to answer the question: **wait or not in the restaurant?**

Example	Input Attributes										Goal
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
<b>x<sub>1</sub></b>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>0–10</i>	<i>y<sub>1</sub> = Yes</i>
<b>x<sub>2</sub></b>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>30–60</i>	<i>y<sub>2</sub> = No</i>
<b>x<sub>3</sub></b>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Some</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>y<sub>3</sub> = Yes</i>
<b>x<sub>4</sub></b>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Thai</i>	<i>10–30</i>	<i>y<sub>4</sub> = Yes</i>
<b>x<sub>5</sub></b>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>&gt;60</i>	<i>y<sub>5</sub> = No</i>
<b>x<sub>6</sub></b>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Italian</i>	<i>0–10</i>	<i>y<sub>6</sub> = Yes</i>
<b>x<sub>7</sub></b>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>y<sub>7</sub> = No</i>
<b>x<sub>8</sub></b>	<i>No</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Thai</i>	<i>0–10</i>	<i>y<sub>8</sub> = Yes</i>
<b>x<sub>9</sub></b>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>&gt;60</i>	<i>y<sub>9</sub> = No</i>
<b>x<sub>10</sub></b>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>Italian</i>	<i>10–30</i>	<i>y<sub>10</sub> = No</i>
<b>x<sub>11</sub></b>	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>0–10</i>	<i>y<sub>11</sub> = No</i>
<b>x<sub>12</sub></b>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>30–60</i>	<i>y<sub>12</sub> = Yes</i>

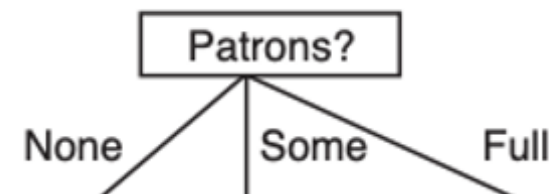
**Figure 18.3** Examples for the restaurant domain.

book: AI, modern approach

## [5] Learning Decision Tree (Restaurant Waiting Prediction Example)

Q: building a structured/nested rule to answer the question: **wait or not in the restaurant?**

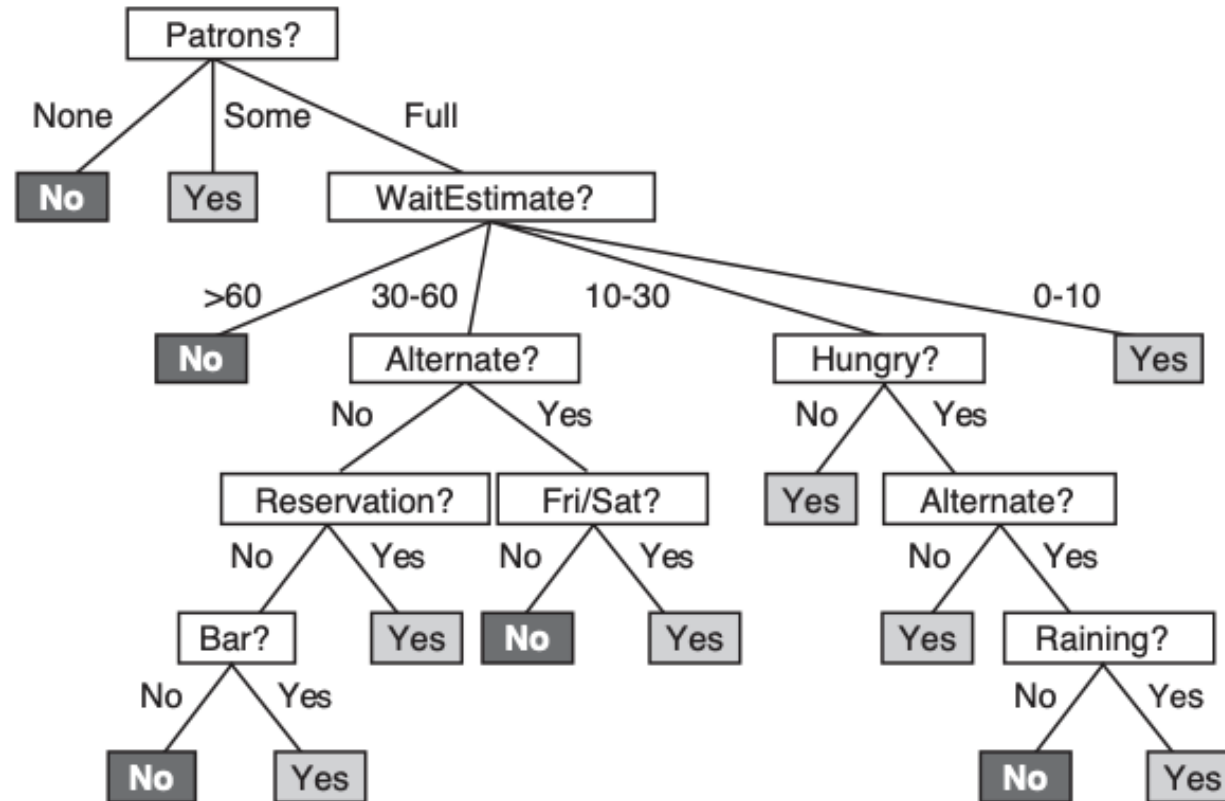
Example	Input Attributes										Goal
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
<b>x<sub>1</sub></b>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>0–10</i>	<i>y<sub>1</sub> = Yes</i>
<b>x<sub>2</sub></b>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>30–60</i>	<i>y<sub>2</sub> = No</i>
<b>x<sub>3</sub></b>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Some</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>y<sub>3</sub> = Yes</i>
<b>x<sub>4</sub></b>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Thai</i>	<i>10–30</i>	<i>y<sub>4</sub> = Yes</i>
<b>x<sub>5</sub></b>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>&gt;60</i>	
<b>x<sub>6</sub></b>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Italian</i>	<i>0–1</i>	
<b>x<sub>7</sub></b>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>0–1</i>	
<b>x<sub>8</sub></b>	<i>No</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Thai</i>	<i>0–1</i>	
<b>x<sub>9</sub></b>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>&gt;60</i>	<i>y<sub>9</sub> = No</i>
<b>x<sub>10</sub></b>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>Italian</i>	<i>10–30</i>	<i>y<sub>10</sub> = No</i>
<b>x<sub>11</sub></b>	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>0–10</i>	<i>y<sub>11</sub> = No</i>
<b>x<sub>12</sub></b>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>30–60</i>	<i>y<sub>12</sub> = Yes</i>



**Figure 18.3** Examples for the restaurant domain.

book: AI, modern approach

## [6] Learning Decision Tree (Restaurant Waiting Prediction Example)



**Figure 18.2** A decision tree for deciding whether to wait for a table.

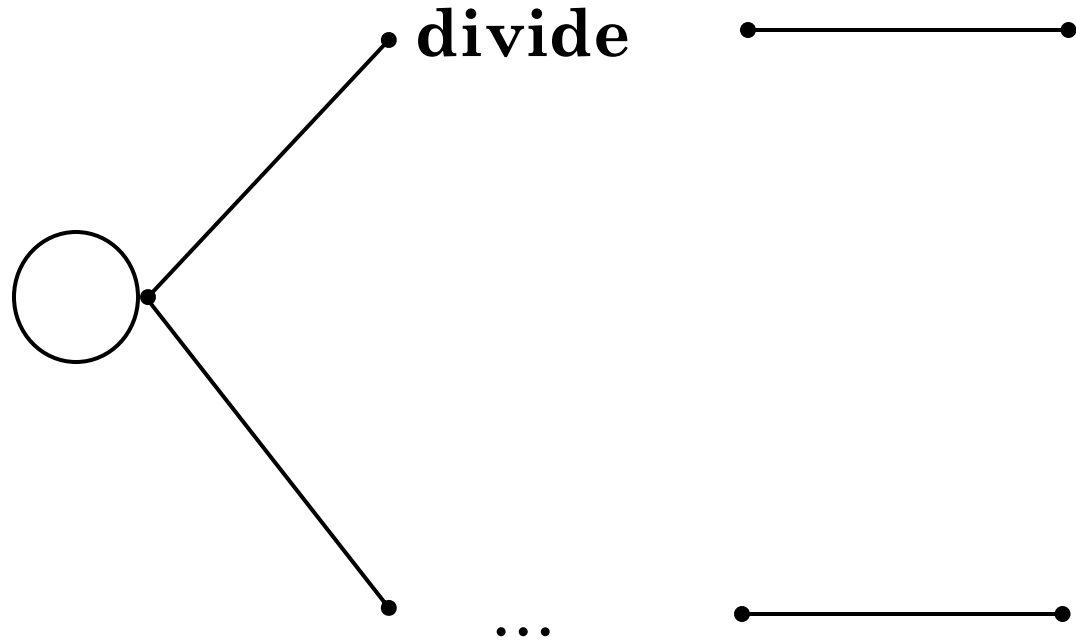
book: AI, modern approach

## [7] Learning Decision Tree

Q: How do we select a feature?



## [8] Learning Decision Tree (Feature Selection)



Hungry?	Price?	Wait
no	\$\$\$	yes
yes	\$\$\$	no
yes	\$	no
no	\$\$	yes

[Restaurant Waiting Decision]

- Feature Selection  
based on **mutual information**  $I(X_k; Y) = H(Y) - H(Y | X_k)$

## [1] Feature Selection (example)

$X_1$	$X_2$	$Y$
T	T	+
F	T	+
T	T	+
F	T	+
T	F	-
F	T	-
T	F	-
F	T	-

Q: if you can select either feature  $X_1$  or  $X_2$  then which feature would you select to predict  $Y$ ?

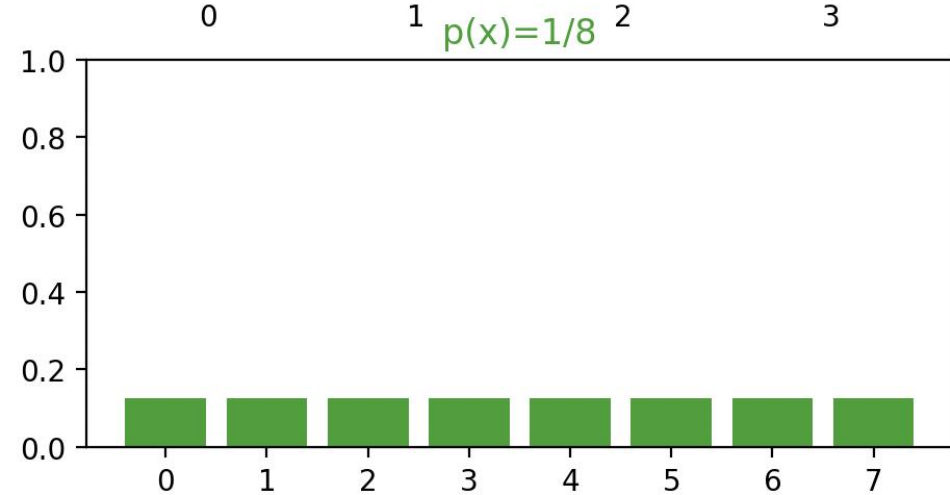
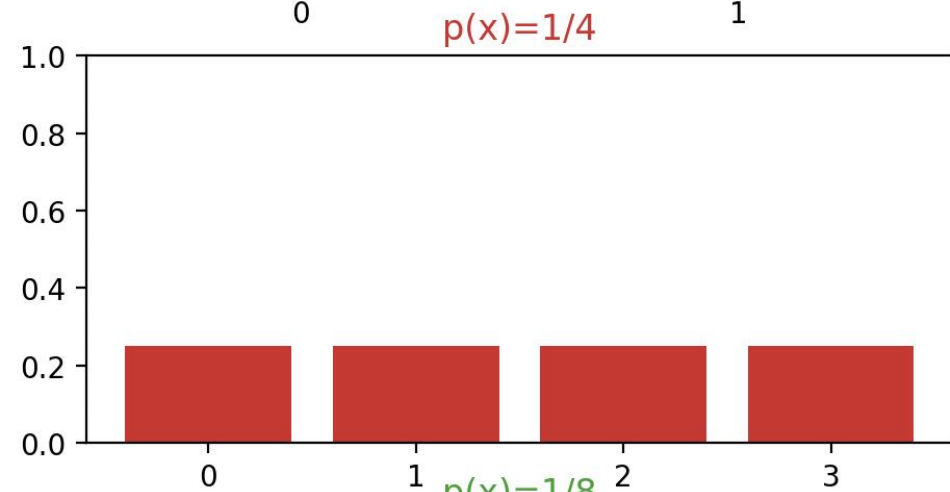
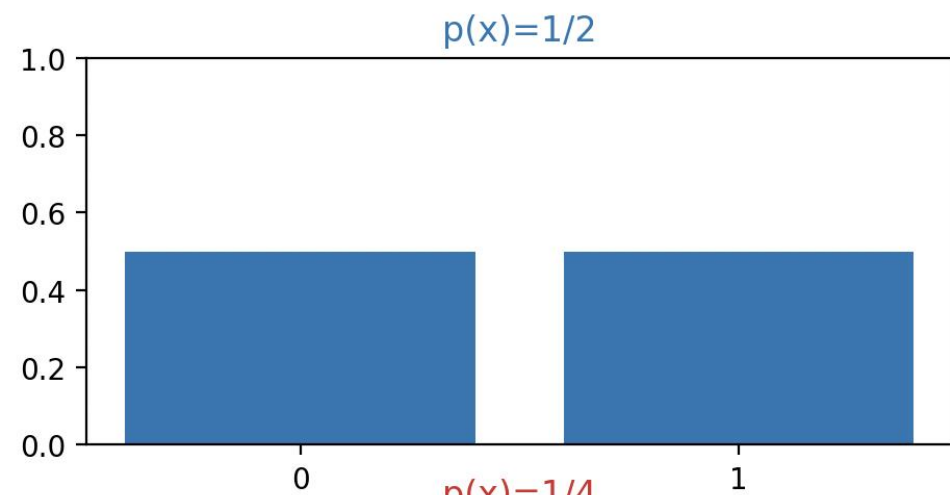
## [2] Feature Selection (based on mutual information)

- ❖ mutual information  $I(X; Y)$  quantifies the information that a feature  $X$  provide about  $Y$ .

$$I(X; Y) = \sum_x \sum_y P(x, y) \log \frac{P(x, y)}{P(x)P(y)}$$



$$I(X; Y) = H(Y) - H(Y|X)$$

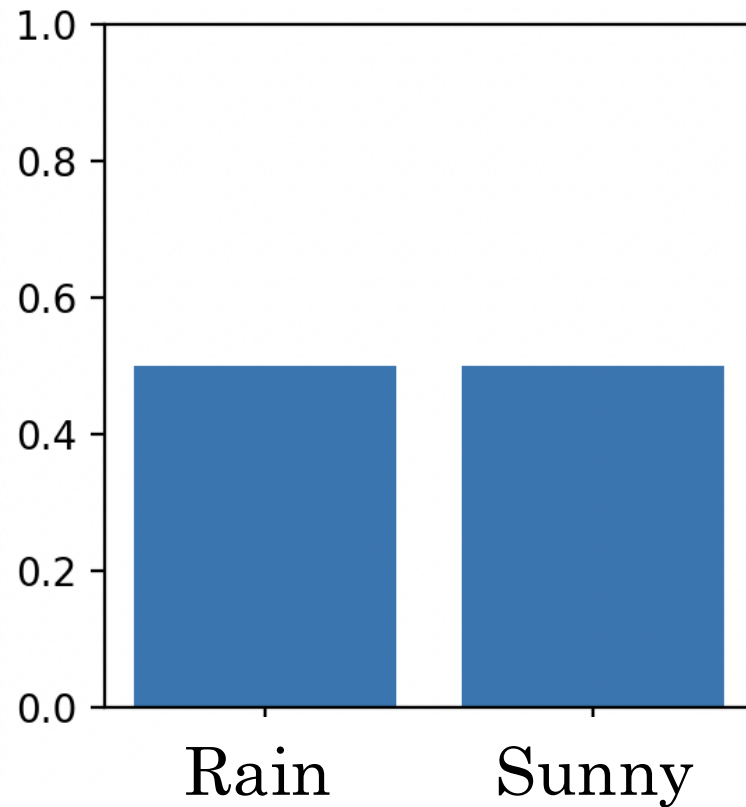


**\*\* entropy (meaning I)**

Q: how many bits are required to represent the information?

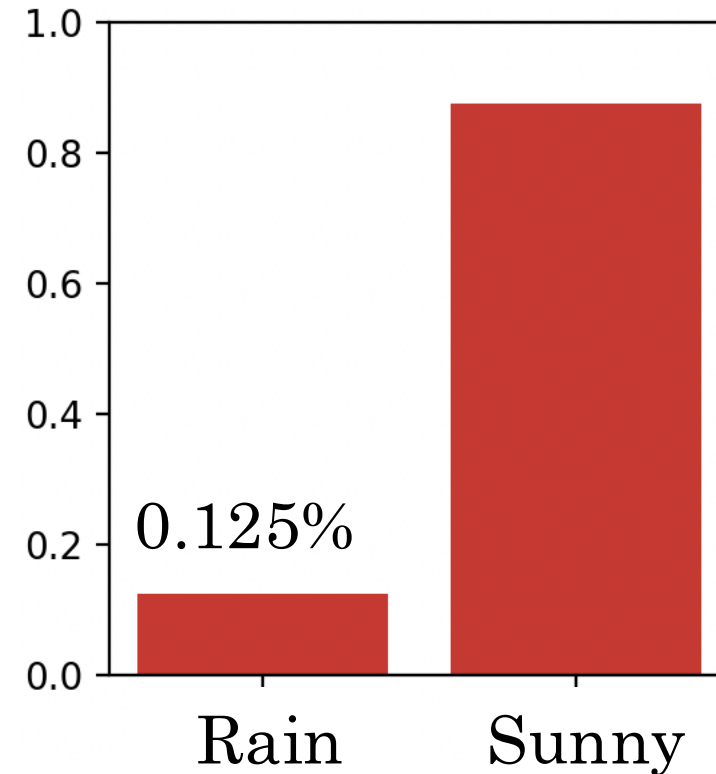
## \*\* entropy (meaning II)

Q: entropy measures the uncertainty of a random variable.  
in which case, we are hard to predict the weather tomorrow?



$$H(p) = 1$$

vs.



$$H(p) = 0.543$$

## \*\* recall: Entropy / Cross Entropy / KL Divergence (Kullback-Leibler)

$$\begin{aligned} D(p||q) &= \sum_x p(x) \log \frac{p(x)}{q(x)} \\ &= \sum_x p(x) \log \frac{1}{q(x)} - \sum_x p(x) \log \frac{1}{p(x)} \end{aligned}$$

- **[KL divergence]:**  
how many bits more needed when using  $q(x)$  instead of the original density  $p(x)$ ?
- this measures  
the distance between  $p(x)$  and  $q(x)$

## \*\* conditional entropy

$$\begin{aligned} H(Y|X) &= \sum_x p(x) H(Y|x = x) \\ &= \sum_x p(x) \sum_y p(y|x) \log \frac{1}{p(y|x)} \\ &= \sum_x \sum_y p(x, y) \log \frac{1}{p(y|x)} \end{aligned}$$

- conditional entropy  $H(Y|X)$  is an averaged entropy of  $H(Y|X = x)$  over  $p(x)$ .



### [3] Feature Selection (mutual information)

$$I(X;Y) = \sum_x \sum_y P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$
$$= H(Y) - H(Y|X) = \sum_x \sum_y P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$

## [4] Feature Selection (mutual information & its three interpretations)

[1] the shared information between  $X$  and  $Y$

$$I(X; Y) = \sum_x \sum_y P(x, y) \log \frac{P(x, y)}{P(x)P(y)}$$

[2] the reduction of the uncertainty of  $Y$  due to the knowledge of  $X$

$$I(X; Y) = H(Y) - H(Y|X)$$

[3] the distance between  $p(x, y)$  &  $p(x)p(y)$

$$I(X; Y) = KL(p(x, y) || p(x)p(y))$$

## [5] Feature Selection (mutual information example I)

Q: compute mutual information for the joint PMF below?

$$I(X; Y) = \sum_x \sum_y P(x, y) \log \frac{P(x, y)}{P(x)P(y)}$$

$P(x, y)$	$X = 0$	$X = 1$
$Y = 0$	1/4	1/2
$Y = 1$	1/8	1/8

## [6] Feature Selection (mutual information example II)

$X_1$	$X_2$	$Y$
T	T	+
F	T	+
T	T	+
F	T	+
T	F	−
F	T	−
T	F	−
F	T	−

$$\begin{aligned} I(X_2; Y) &= H(Y) - H(Y|X_2) \\ &= H(Y) - \{p(X_2 = T)H(Y|X_2 = T) + p(X_2 = F)H(Y|X_2 = F)\} \\ &= 1 - \{3/4 \cdot (2/3 \cdot \log 3/2 + 1/3 \cdot \log 3) + 1/4 \cdot 0\} \end{aligned}$$

## [7] Feature Selection (mutual information and Gini index)

$$i = \arg \max_i I(X_i; Y)$$

$$i = \arg \min_i H(Y|X_i) = \sum_{x_i} p(x_i) H(Y|x_i)$$

$$i = \arg \min_i \text{Gini-Index}(X_i) = \sum_{x_i} p(x_i) \cdot \text{Gini-Index}(x_i) \quad \text{where } \text{Gini-Index}(x_i) = 1 - \sum_y p(y|x_i)^2$$

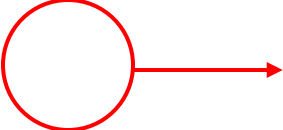
measuring impurity  
↓

CART (**C**lassification and **R**egression Tree Algorithm): use gini-index  
ID3, C4.5 (Iterative Dichotomiser 3): use mutual information

## [8] Feature Selection (Greedy Divide & Conquer )

- ❖ The nature of tree learning is “greedy” and “irreversible”.
- ❖ Divide & Conquer strategy

## [9] Learning Decision Tree (pseudo code)

def **Tree** ( $X$ ):  a set of data points!

    if divide ( $X$ ):

        assign a majority class

    break

$f$  = feature selection to split ( $X$ )

$\eta$  = threshold selection to split the feature ( $X, f$ )

$X_1, X_2$  = divide ( $X, f, \eta$ )

**Tree**( $X_1$ )

**Tree**( $X_2$ )

- selecting one optimal feature at a time (**local optimal**)
- selecting one optimal threshold at a time (**local optimal**)

## [10] Feature Selection (Greedy Divide & Conquer )

- ❖ The nature of tree learning is “irreversible” and “greedy”
- ❖ (Divide & Conquer strategy)
  - once a feature is selected for a split, other possible features are not reconsidered even when they could lead to better splits (shorter/ purer) in the long term. (not a global optimum)
  - divisions in a decision tree will be different depending on which question comes first / later. (unstable)



## \*\*Intractability of Exhausted Search

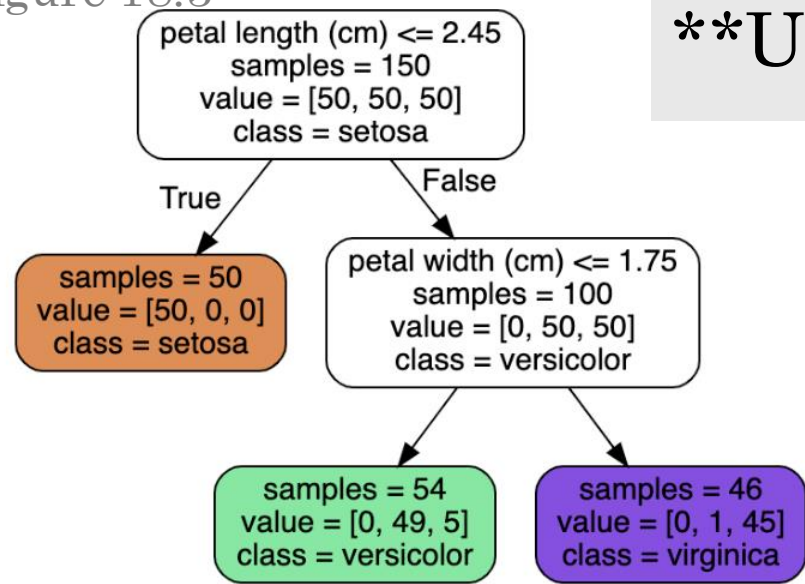
Q: The greedy divide & conquer method is not globally optimal.  
how could we search (train) an optimal division given a training set?

# possible  $N$  divisions  $\approx$  # possible trees with  $N$  nodes

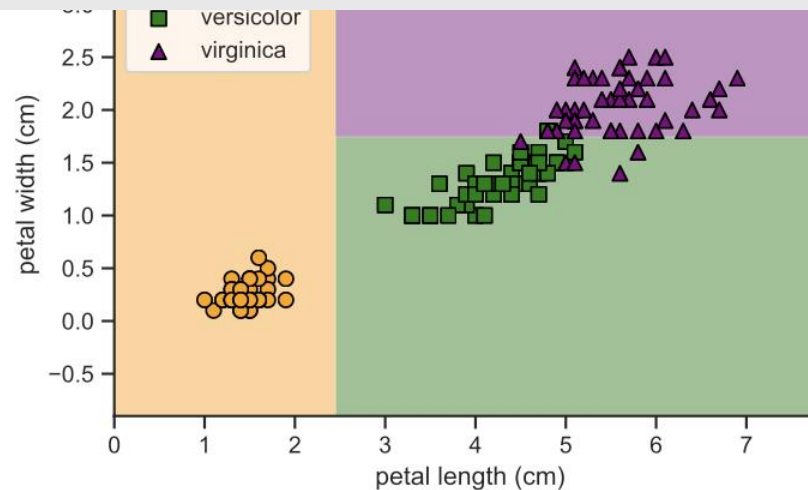
$$C_{N+1} = \sum_{k=0}^N C_k \cdot C_{N-k}$$

Catalan Recurrence  $C_n \sim \frac{4^n}{n^{3/2} \sqrt{\pi}}$

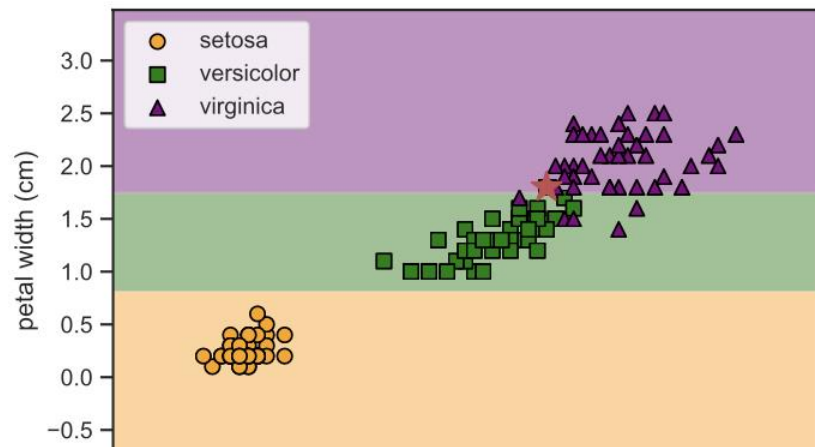
**\*\*Unstable; small change but large effect**



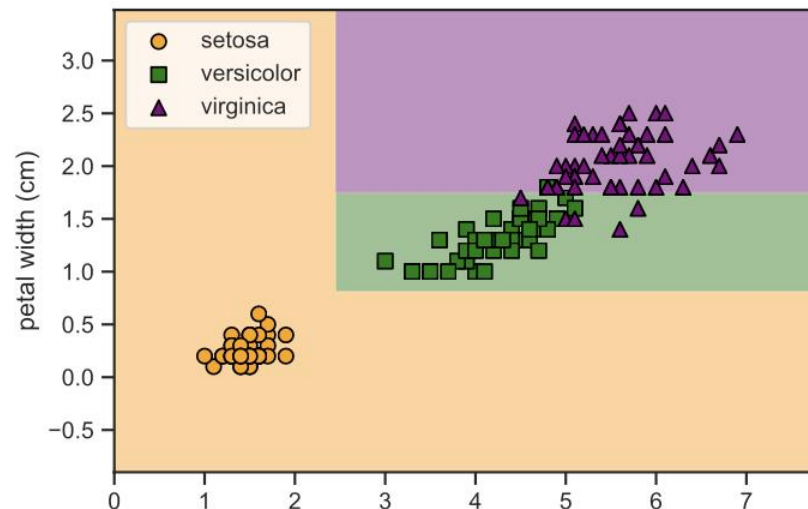
[a] decision tree



[b] decision surface by the tree [a]



[c] one missing data results in very different decision surface.



[d] Ensemble Method

## [11] Feature Selection (Greedy Divide & Conquer )

- ❖ no guarantees that the local optimal selection to be a global optimal.
- ❖ no broad perspectives

Q: when do we need to stop? How do we know it?

we often empirically find that none of available splits produces a significant error reduction but after several more splits a substantial error reduction is found. Then what should we do?

- Tree Complexity Control
  - use the stopping criterion
  - pruning the trees

## [1] Tree Complexity Control (use a preset stopping criterion)

- Use the preset stopping rule:

$$\max_{X_i} I(Y; X_i) < \eta \quad (\text{no feature gain is greater than the criterion})$$

- too low  $\eta$ : too many splits and tree becomes too large.
- too large  $\eta$  (early stopping):

however, it is found empirically often that a small reduction for now but after several more splits a substantial impurity reduction is found.  
(for greedy algorithm)

## [2] Tree Complexity Control (tree pruning)



- ❖ The best strategy is to set a low stopping criterion  $\eta$ , so (1) grow a very large tree and (2) prune the tree later.

### [3] Tree Complexity Control (tree pruning)

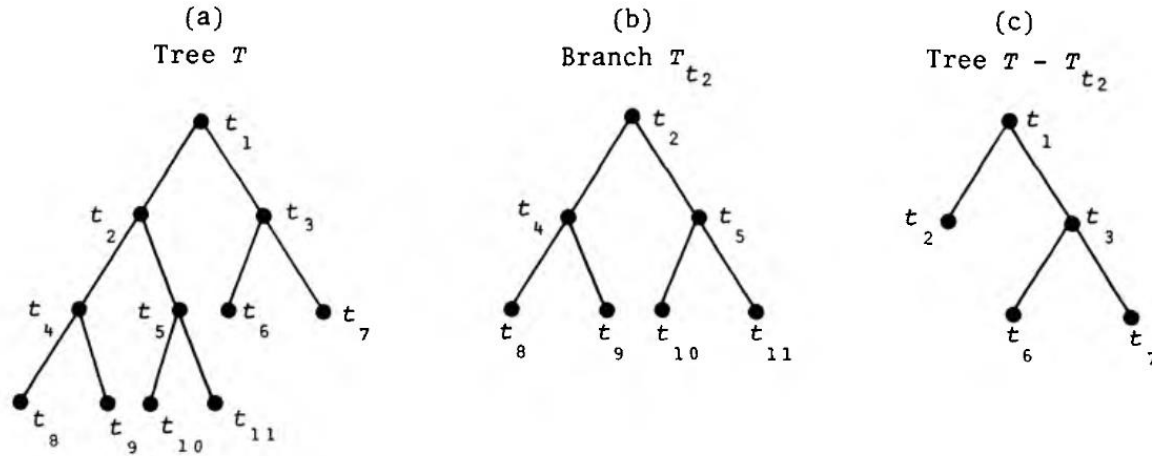


FIGURE 3.1

- **Branch (subtree)**

A branch  $T_t$  of  $T$  with root node  $t$  consists of the node  $t$  and all descendants of  $t$  in  $T$ .

- **Pruning (result)**

Pruning  $T_t$  from a tree  $T$  consists of deleting all descendants of  $t$  except for the root node  $t$ .

## [4] Tree Complexity Control (tree pruning)

(1)  $T, T_1, T_2, \dots$ , *root node*

(a subtree whose # of terminal node reduced by 1)

(2) pick a subtree that minimizes the pruning criterion.

$$\text{Pruning Criterion} = \sum_{i=1}^{\tilde{T}} Q(i) + \alpha |\tilde{T}|$$

- $Q(i)$  is a performance measurement such as impurity, entropy, etc.
- $\tilde{T}$  : *of terminal node of a subtree*
- $\alpha$ : *control model complexity*



- Pros & Cons for a Decision Tree

## [1] Decision Tree (why decision Tree: pros)

- simple to train
- efficient in handling non-homogeneous data
  - requires little data pre-processing
- robust to outliers
- naturally handle and interpret semantic features
  - one at a time
  - tree classifier provides a natural way of understanding the structure of the problem.

## [2] Decision Tree (sequential decision process)

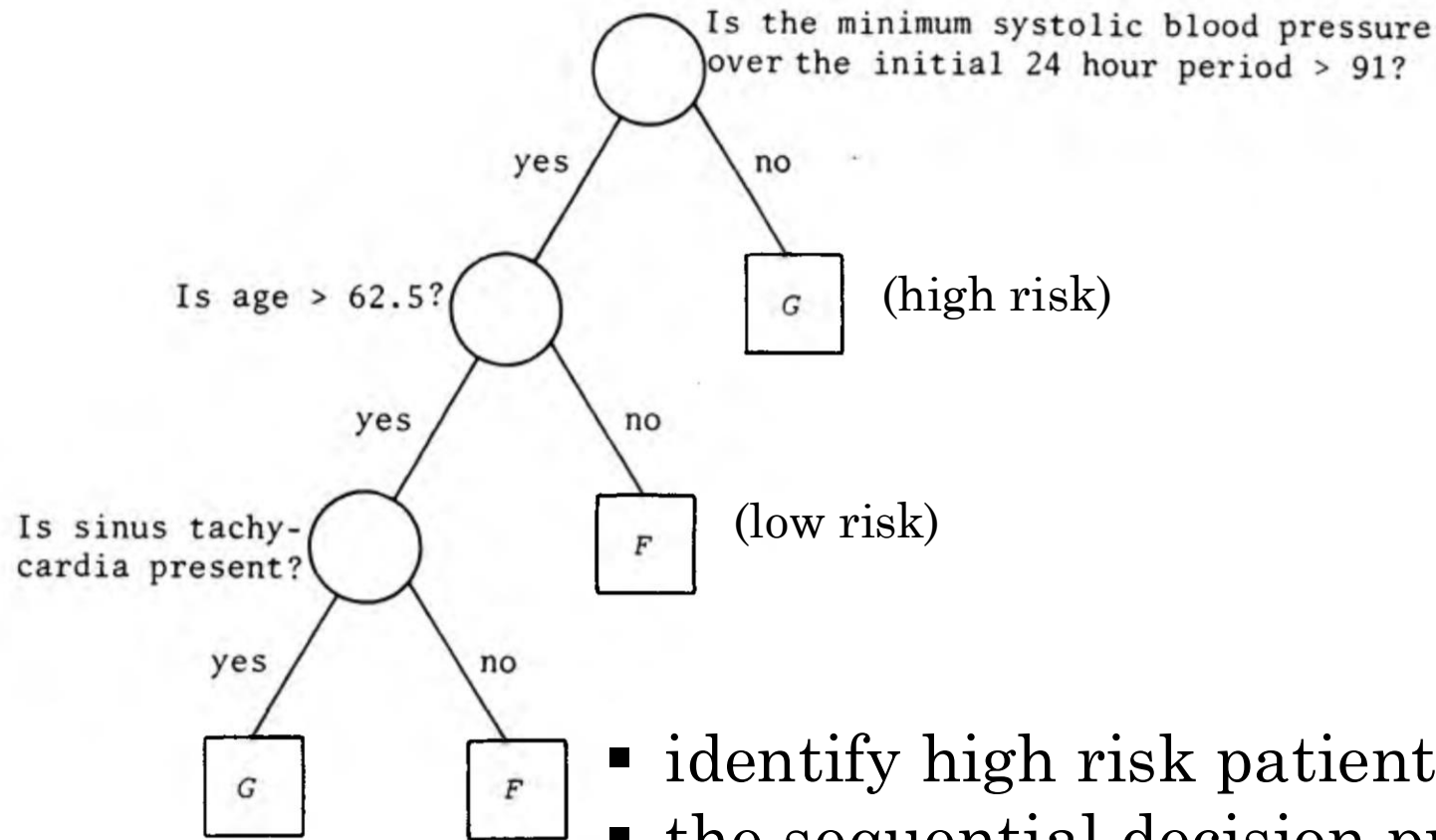


FIGURE 1.1

- identify high risk patient
- the sequential decision process can be shown clearly.

From the book “classification and regression trees” by Breiman, Leo,

### [3] Decision Tree (Cons)

1. However, tree decision boundaries (perpendicular to the coordinates axes ) poorly uncovers the original data structure.

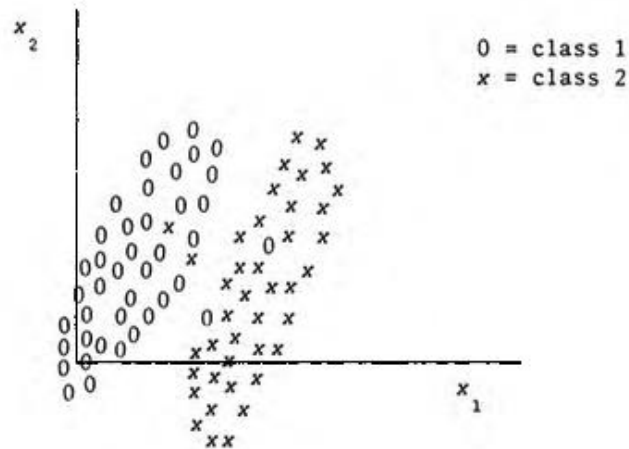


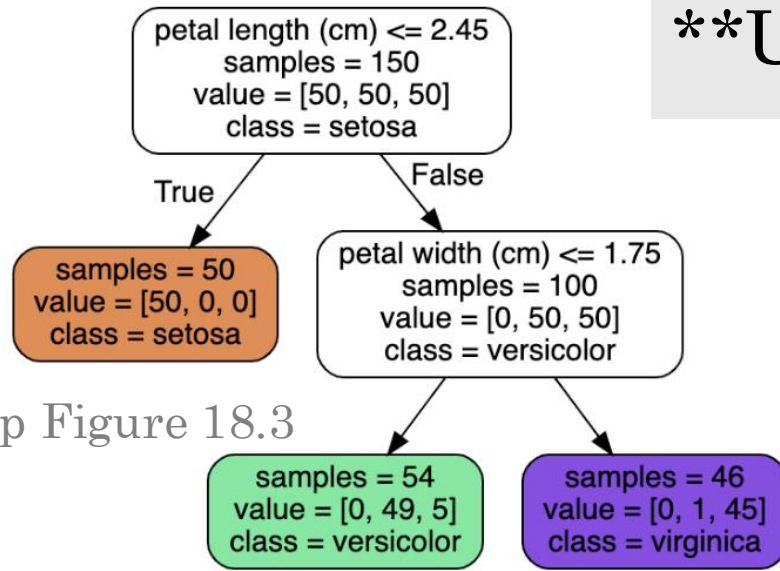
FIGURE 2.10

From the book “classification and regression trees” by Breiman, Leo,

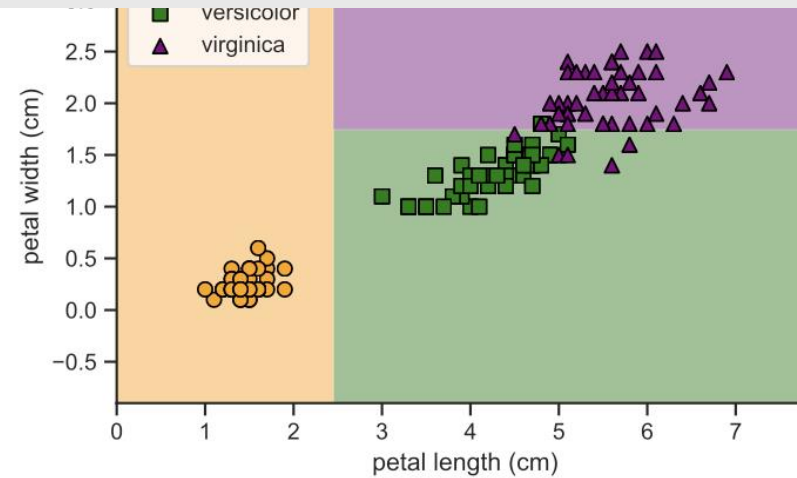
## [4] Decision Tree (Cons)

2. Tree is unstable: small changes to the input data can have large effects on the structure of the tree, due to hierarchical nature of the tree growing process, causing errors at the top to affect the rest of the tree.  
(high variance for different data sets)

**\*\*Unstable; small change but large effect**

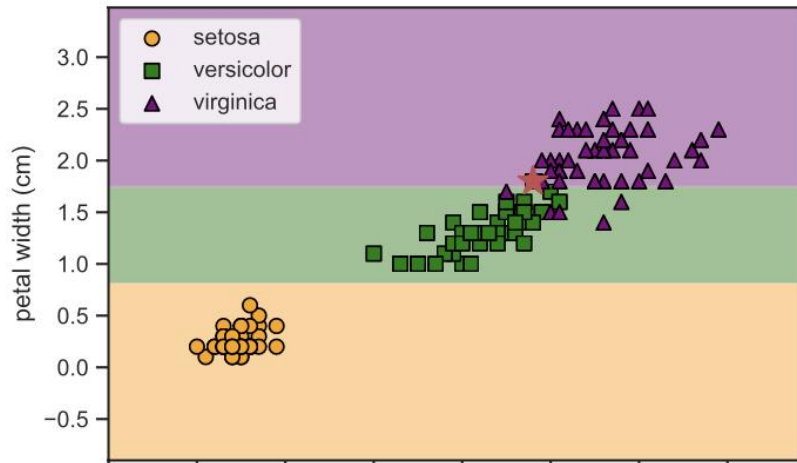


From bishop Figure 18.3

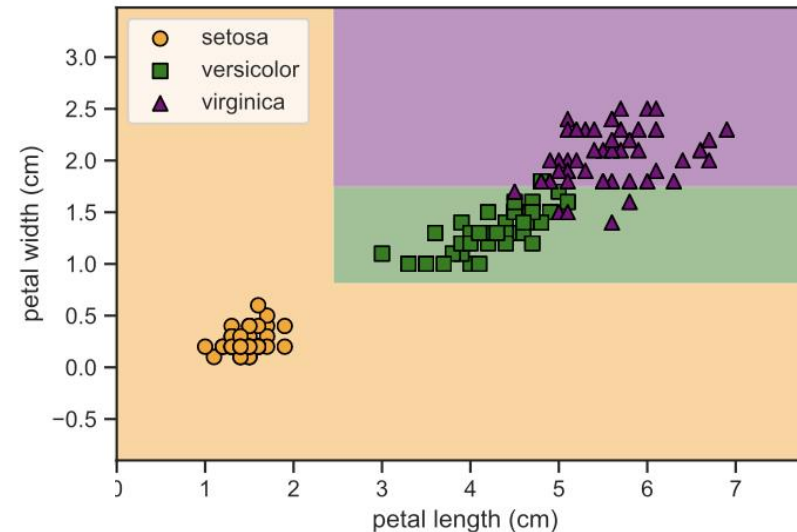


[a] decision tree

[b] decision surface by the tree [a]



[c] one missing data results  
in very different decision surface.



[d] ensemble method

## [5] Decision Tree (cons)

Q: in the previous figure, a simple structure tree is sensitive to the small perturbation of train data.  
how can we handle the high variance issue?

- Random Forest (ensemble method)
  - (1) bagging (acronym: bootstrap aggregating)
  - (2) random vector



## \*\* averaging models reduces errors (combining models)

- averaging combination model

$$t = \frac{1}{M} \cdot \sum_{m=1}^M f_m(x) + \epsilon_m \quad \forall m = 1, \dots, M$$

- MSE by a single model

$$\text{MSE} = E[|\epsilon_m|^2]$$

- ❖ The MSE of a single model can be reduced by a factor of  $M$

- MSE by averaging multiples

$$\text{MSE} = E\left[\left|\frac{1}{M} \sum_{m=1}^M f_m(x) - t\right|^2\right]$$

$$= E\left[\left|\frac{1}{M} \sum_{m=1}^M (f_m(x) - t)\right|^2\right]$$

$$= E\left[\left|\frac{1}{M} \sum_{m=1}^M (\epsilon_m)\right|^2\right]$$

$$= \frac{1}{M^2} \cdot M \cdot E[|\epsilon_m|^2]$$

$$= \frac{1}{M} E[|\epsilon_m|^2]$$

$$E[\epsilon_m \cdot \epsilon_l] = 0 \quad \forall m \neq l$$

(uncorrelated)

**\*\* averaging models reduces errors (combining models)**

❖ Decorrelation among the errors, in other words,  
is the key to the success of ensemble method.

# [1] Random Forest (growing diverse trees)

Q: The ensemble trees need to be diverse  
to reduce the correlation among their errors and improve overall performance.  
How can we grow diverse trees?



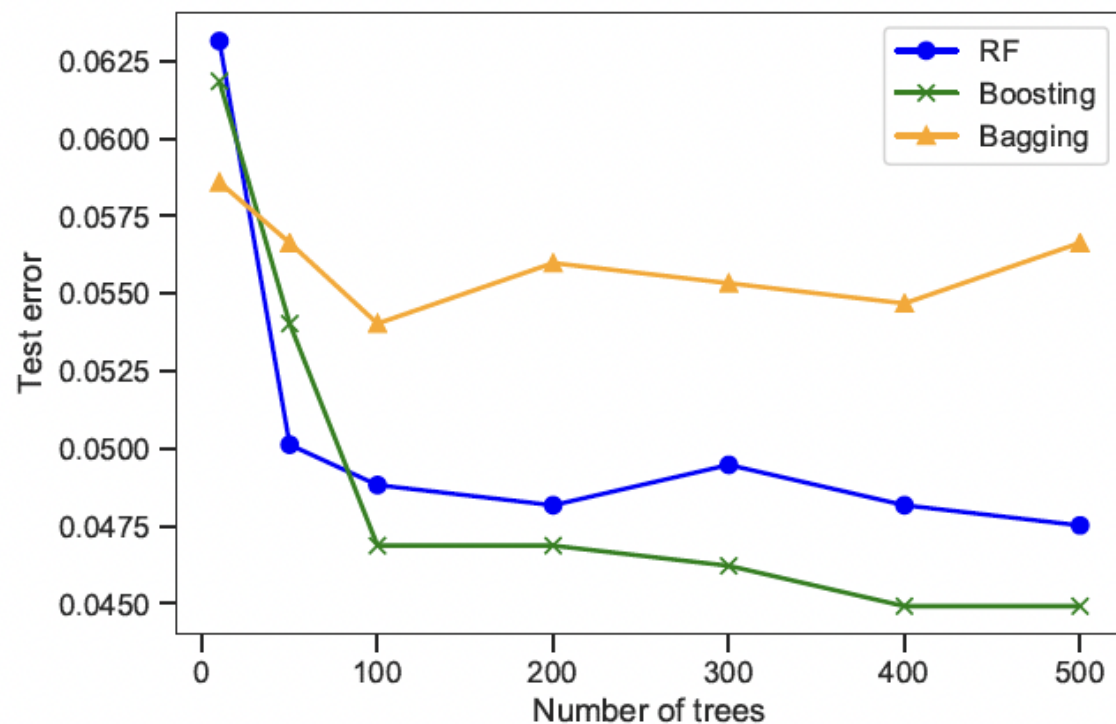
## [2] Random Forest (two random source)

- Random forest introduces the “two source of randomness”.
  - **bagging**: each tree is fully grown using a bootstrap sample of training data.  
(with replacement but the same number of the original training set)  
(63.8% included vs. 36.8% not included)
  - **random feature**: at each node, best split is chosen from a random sample of  $K$  features instead of taking all the features as a candidate.

### [3] Random Forest (two random source)

- Random forest introduces the two source of randomness.
  - **bagging**: each tree is fully grown using a bootstrap sample of training data.  
(with replacement but the same number of the original training set)  
(63.8% included vs. 36.8% not included)
  - **random vector** : at each node, best split is chosen from a random sample of  $K$  features instead of taking all the features as a candidate.
- ❖ The bias is scarified in random forest a bit to leverage the advantage of diverse models and reduce variance more effectively.
- ❖ How about bagging only?

## [4] Random Forest (comparison of bagging only & random forest)



❖ Random forest decorrelates the base learner even further than Bagging.