MYONGJI UNIVERSITY

Artificial Intelligence

Lecture 8. Decision Tree

Spring 2022

Prof. Jonghoon Chun, Ph.D.

E-mail: jchun@mju.ac.kr
Lecture Note: http://lms.mju.ac.kr

Agenda

- Basic Concept
- Bayes Classification Methods
- Decision Tree
- Ensemble Methods
- Random Forest

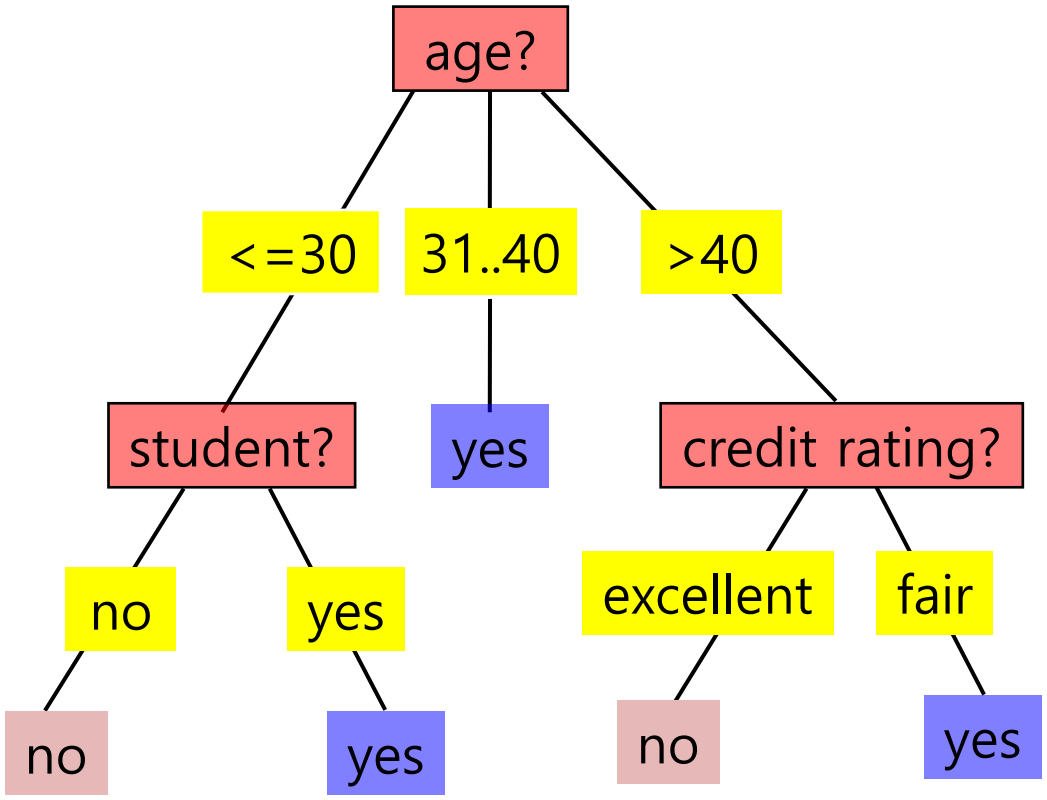
DECISION TREE

Classification by Decision Tree Induction

- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution (e.g., For class label *buys_computer = yes* or *buys_computer = no*)
- Decision tree generation consists of two phases
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree

Decision Tree Induction: An Example

- Training data set: Buys_computer
- Resulting tree:



age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

INFORMATION THEORY

Information Theory

- Motivation
  - 하나의 문서에 여러 단어가 포함되어 있는 경우를 상정
  - E.g. 총 10개의 단어 중, computer (5회), memory(2회), monitor(2회), keyboard(1회) 의 빈도수로 출현
  - 이 문서 (혹은 각각의 단어)가 제공하는 정보(information)을 정량적으로 측정한다면?

- Information theory : the information content of a message (a term) can be measured as an inverse function the probability of occurrence of the words in a given text.
  - the higher the probability of occurrence of a word, the less information it contains

$$INFORMATION = -log_2 (p)$$

where p is the probability of occurrence of the word

Information Theory

- $INFORMATION = -\log_2(.005)$
  $= -(-10)$
  $= 10$ if $p = .5\%$

문서에 덜 등장하는 단어가 가지는 정보가 더 많다는 의미로 해석될 수 있음

- $INFORMATION = -\log_2(.5)$
  $= -(-1)$
  $= 1$ if $p = 50\%$

문서에 자주 등장하는 단어가 가지는 정보가 더 적다는 의미로 해석될 수 있음

- When a document is characterized by n possible terms, each occurring with a specified probability p_k, the average, entropy, or expected information by using one of the terms is given by

$$AVE\_INFO = -\Sigma p_k \log_2 (p_k)$$

where Σ= summation from $k = 1$ to n

Information Theory

ex) computer, memory, monitor, keyboard

- expected to occur with the probabilities 0.5, 0.2, 0.2, and 0.1 respectively

$$AVE\_INFO = -[(0.5\log_2 0.5) + (0.2\log_2 0.2) + (0.2\log_2 0.2) + (0.1\log_2 0.1)]$$
$$= -[(-0.05) + (-0.46) + (-0.46) + (-0.33)]$$
$$= 1.3$$

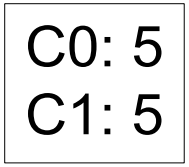
- *AVE_INFO* is maximized when the occurrence probabilities of the terms are all equal to *1/n* for *n* distinct terms (e.g., 4 terms are all expected to occur one-fourth of the time, the *AVE_INFO* value will be 2.)

SPLIT

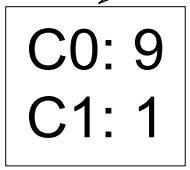
How to determine the Best Split

- Nodes with “purer” class distribution are preferred

- Need a measure of node impurity
  - Assume a binary split:



High degree of impurity



Low degree of impurity

Impurity degree가 낮음!
즉, 더 pure 하므로 이 split을 선호함!

Finding the Best Split

1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
   - Compute impurity measure of each child node
   - M is the weighted impurity of children
3. Choose the attribute test condition that produces the highest gain

$$Gain = P - M$$

or equivalently, lowest impurity measure after splitting (M)

Attribute Selection Measure

- Information gain
  - All attributes are assumed to be categorical
  - Can be modified for continuous-valued attributes
- Gini index
  - All attributes are assumed continuous-valued
  - Assume there exist several possible split values for each attribute
  - May need other tools, such as clustering, to get the possible split values
  - Can be modified for categorical attributes
- Gain Ratio

Information Gain

- Select the attribute with the highest information gain
- Assume there are two classes, *P* and *N*
  - Let the set of examples *S* contain *p* elements of class *P* and *n* elements of class *N*
  - The amount of information, needed to decide if an arbitrary example in *S* belongs to *P* or *N* is defined as

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Derived from Information Theory

Information Gain in Decision Tree Induction

- Assume that using attribute A, a set S will be partitioned into sets $\{S_1, S_2, \ldots, S_v\}$
  - If S_i contains p_i examples of P and n_i examples of N, the entropy, or the expected information needed to classify objects in all subtrees S_i is

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- The encoding information that would be gained by branching on A

$$Gain(A) = I(p, n) - E(A)$$

Attribute Selection by Information Gain Computation

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"
- I(p, n) = I(9, 5) =0.940
- Compute the entropy for *age* :

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
30…40	4	0	0
>40	3	2	0.971

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

Hence

$$Gain(age) = I(p,n) - E(age) = 0.246$$

Similarly

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

Gini Index

- If a data set *D* contains examples from *n* classes, gini index, *gini*(*D*) is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class *j* in *D*

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – P(C1)^2 – P(C2)^2 = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – (1/6)^2 – (5/6)^2 = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – (2/6)^2 – (4/6)^2 = 0.444

Gini Index for a Collection of Nodes

- When a node p is split into k partitions (children)

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,
n = number of records at parent node p.

- Choose the attribute that minimizes weighted average Gini index of the children
  - The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node
  - need to enumerate all the possible splitting points for each attribute

Computation of Gini Index

- D has 9 tuples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

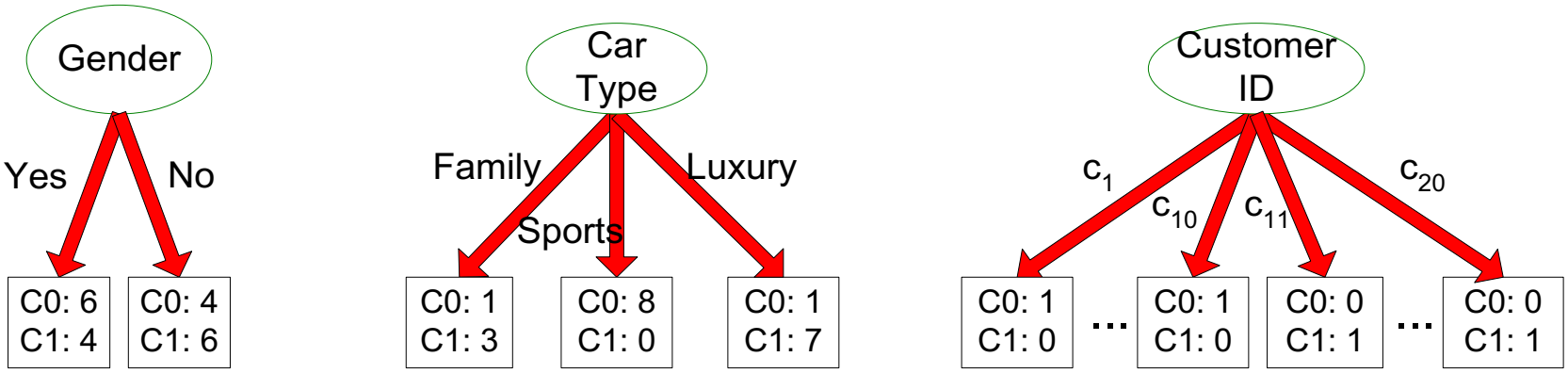
$$= Gini_{income \in \{high\}}(D).$$

$Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450.

- Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index!

Problem with large number of partitions

- Node impurity measures tend to prefer splits that result in large number of partitions, each being small but pure



  - E.g., Customer ID has highest information gain because entropy for all the children is zero

Gain Ratio

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \qquad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

  - Parent Node, p is split into k partitions
  - n_i is the number of records in partition i

  – Adjusts Information Gain by the entropy of the partitioning (SplitINFO).
    - Higher entropy partitioning (large number of small partitions) is penalized!
  – Designed to overcome the disadvantage of Information Gain

- The attribute with the maximum gain ratio is selected as the splitting attribute

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - **Information gain**:
    - biased towards multivalued attributes
  - **Gini index**:
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions
  - **Gain ratio**:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others

Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples

- Two approaches to avoid overfitting
  - Prepruning: *Halt tree construction early*—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: *Remove branches* from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Robust to noise (especially when methods to avoid overfitting are employed)
  - Can easily handle redundant or irrelevant attributes (unless the attributes are interacting)
- Disadvantages:
  - Space of possible decision trees is exponentially large. Greedy approaches are often unable to find the best tree.
  - Does not take into account interactions between attributes
  - Each decision boundary involves only a single attribute

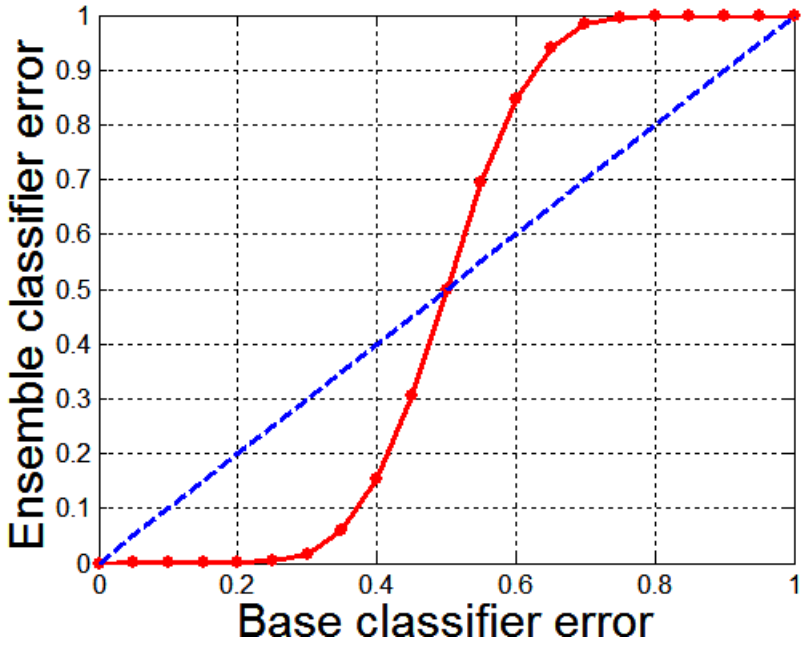
ENSEMBLE METHODS

Ensemble Methods – Basic Concepts

- Construct a set of classifiers from the training data
- Predict class label of test records by combining the predictions made by multiple classifiers
  - Perhaps by voting

Why Ensemble Methods work?

- Suppose there are 25 base (binary) classifiers
  - Each classifier has error rate, $\varepsilon = 0.35$
  - Assume errors made by classifiers are uncorrelated
  - Probability that the ensemble classifier makes a wrong prediction:



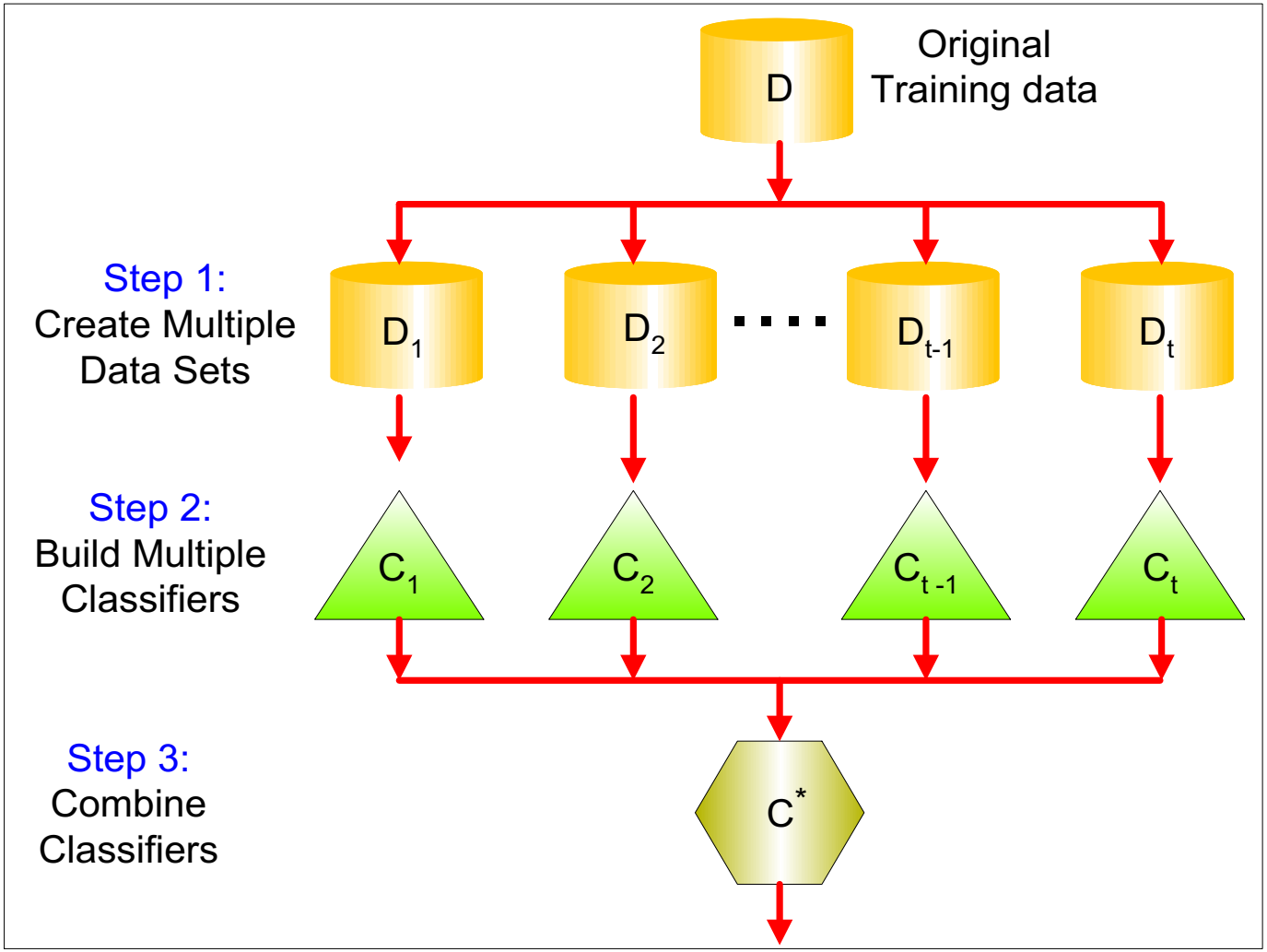
$$P(X \geq 13) = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Two necessary conditions

- The base classifiers should be independent of each other
  - Improvements in classification accuracies have been observed in ensemble methods in which the base classifiers are slightly correlated.

- The base classifiers should do better than a classifier that performs random guessing (better than 50% in case of binary classifier)

General Approach



Types of Ensemble Methods

- Manipulate data distribution (sampling)
  - Example: bagging, boosting (May be covered later!)
- Manipulate input features (choose subset of input features)
  - Example: Random Forests
- Manipulate class labels
  - Example: error-correcting output coding
- Manipulate learning algorithm
  - Example: use different network topology for Neural Net, choose different splitting procedure for decision tree

Comparison among Ensemble Methods

- The base classifiers used in each ensemble method consist of fifty decision trees.
- ten-fold cross-validation
- Ensemble classifiers generally outperform a single decision tree classifier!

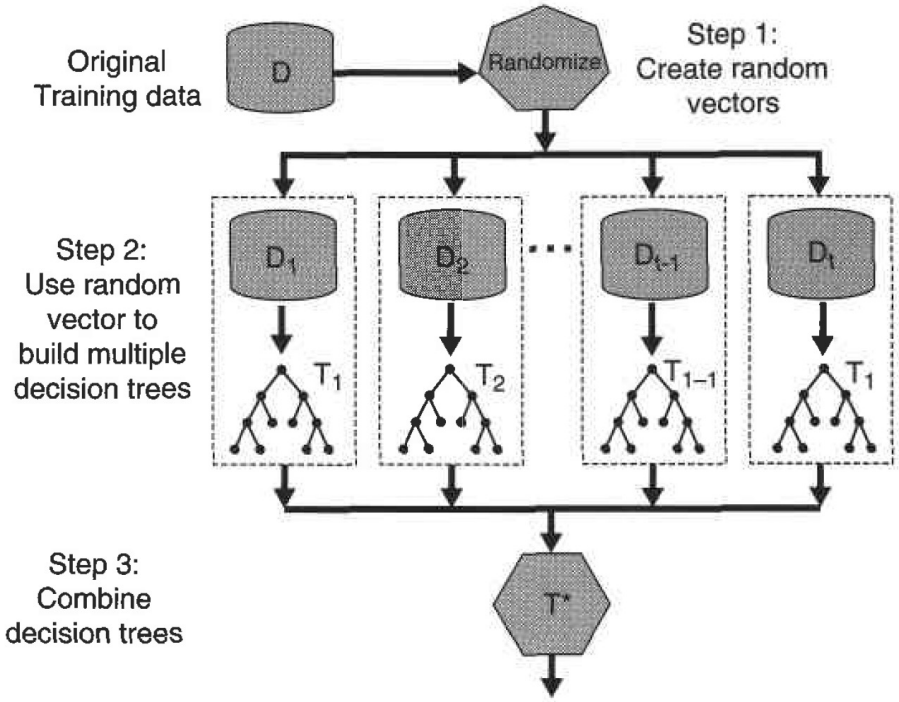
Table 5.5. Comparing the accuracy of a decision tree classifier against three ensemble methods.

Data Set	Number of (Attributes, Classes, Records)	Decision Tree (%)	Bagging (%)	Boosting (%)	RF (%)
Anneal	(39, 6, 898)	92.09	94.43	95.43	95.43
Australia	(15, 2, 690)	85.51	87.10	85.22	85.80
Auto	(26, 7, 205)	81.95	85.37	85.37	84.39
Breast	(11, 2, 699)	95.14	96.42	97.28	96.14
Cleve	(14, 2, 303)	76.24	81.52	82.18	82.18
Credit	(16, 2, 690)	85.8	86.23	86.09	85.8
Diabetes	(9, 2, 768)	72.40	76.30	73.18	75.13
German	(21, 2, 1000)	70.90	73.40	73.00	74.5
Glass	(10, 7, 214)	67.29	76.17	77.57	78.04
Heart	(14, 2, 270)	80.00	81.48	80.74	83.33
Hepatitis	(20, 2, 155)	81.94	81.29	83.87	83.23
Horse	(23, 2, 368)	85.33	85.87	81.25	85.33
Ionosphere	(35, 2, 351)	89.17	92.02	93.73	93.45
Iris	(5, 3, 150)	94.67	94.67	94.00	93.33
Labor	(17, 2, 57)	78.95	84.21	89.47	84.21
Led7	(8, 10, 3200)	73.34	73.66	73.34	73.06
Lymphography	(19, 4, 148)	77.03	79.05	85.14	82.43
Pima	(9, 2, 768)	74.35	76.69	73.44	77.60
Sonar	(61, 2, 208)	78.85	78.85	84.62	85.58
Tic-tac-toe	(10, 2, 958)	83.72	93.84	98.54	95.82
Vehicle	(19, 4, 846)	71.04	74.11	78.25	74.94
Waveform	(22, 3, 5000)	76.44	83.30	83.90	84.04
Wine	(14, 3, 178)	94.38	96.07	97.75	97.75
Zoo	(17, 7, 101)	93.07	93.07	95.05	97.03

RANDOM FORESTS

Random Forests

- Combines the predictions made by multiple decision trees, where each tree is generated based on the values of an independent set of random vectors
  - Random vectors are generated from a fixed probability distribution
  - Bagging* using decision trees is a special case of random forests



*Bagging (bootstrap aggregation) : sample with replacement

Random vector generation

- Forest-RI(Random Input Selection)
  - Randomly select F input features to split at each node of the decision tree
  - Decision to split a node is determined from these selected F features
  - No pruning
  - Predictions are combined using a majority voting scheme

- Trade off for F (number of features)
  - F: small then tree are less correlated
  - F: large then stronger classifiers
  - Common choice $F = \log_2 d + 1$ (d is the number of input features)

- Forest-RC(Random Combination)
  - Use it when d(number of original features) is too small
  - Create linear combination of input features

Random vector generation (cont'd)

- 3rd approach
  - Randomly select one of the F best splits at each node of the decision tree
  - Possibly more correlated than Forest-RI & -RC
  - Do not have runtime savings (must examine all the splitting features at each node of the decision tree)

- Classification accuracies of random forests are quite comparable to the AdaBoost algorithm
- More robust to noise and runs much faster than the AdaBoost algorithm

END