Unit 4. Classification

Disclaimer: Content taken from Han & Kamber slides, Data mining textbooks and Internet

Unit 4. Classification

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Accuracy and error measures
- Ensemble methods
- Summary

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

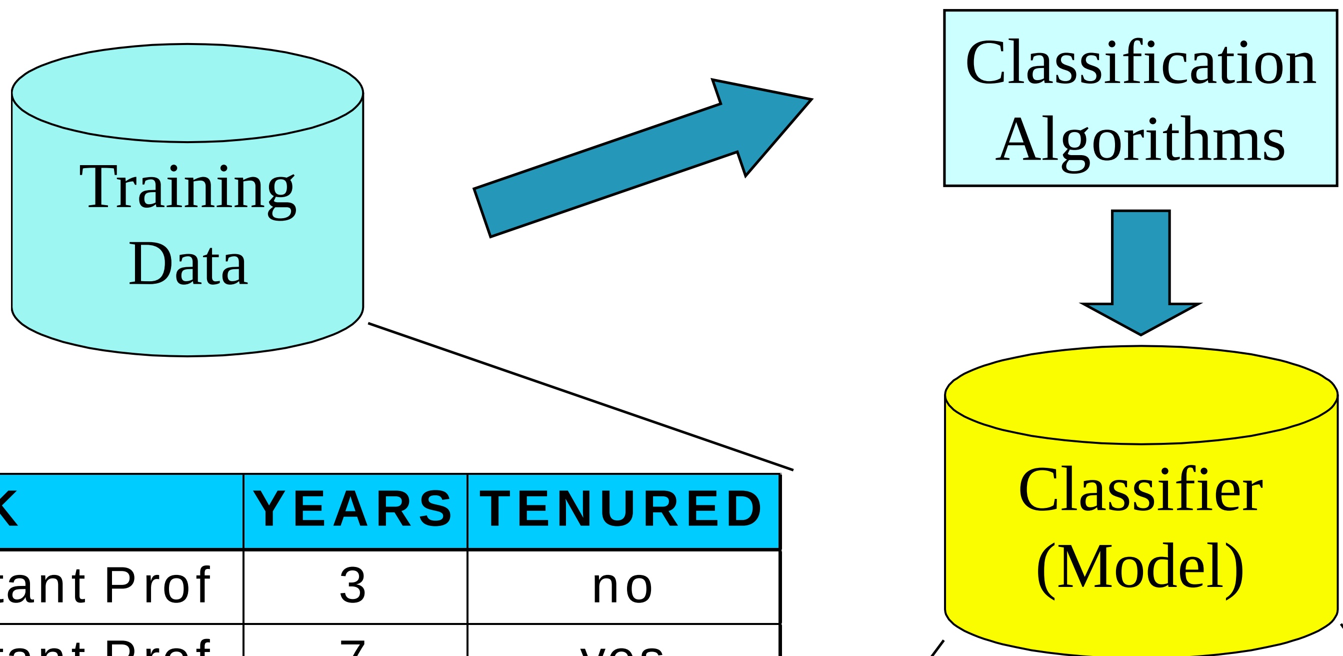
Classification vs. Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit approval
  - Target marketing
  - Medical diagnosis
  - Fraud detection

Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

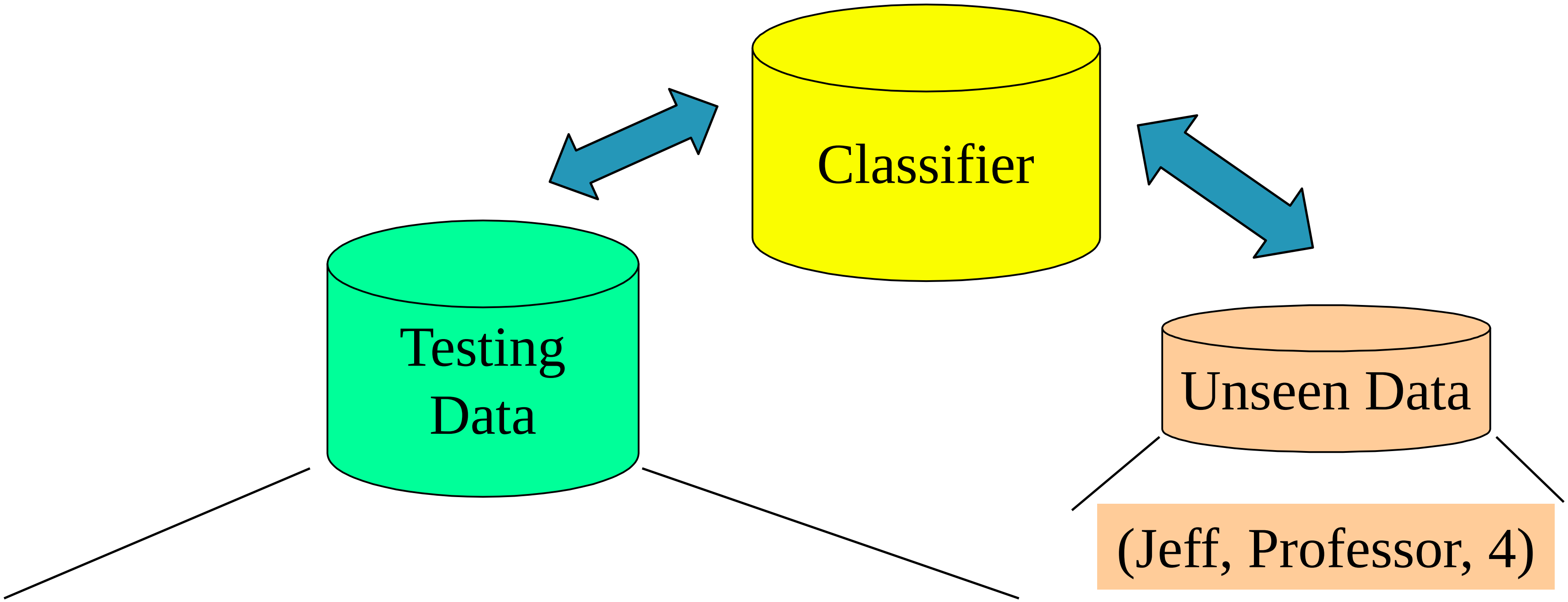
Process (1): Model Construction



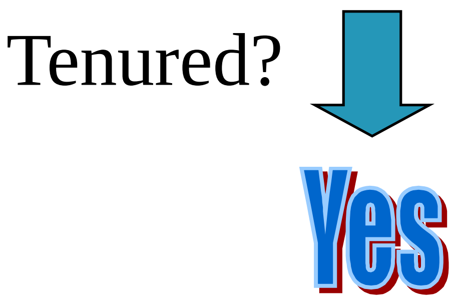
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Process (2): Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes



Unit 4. Classification

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Accuracy and error measures
- Ensemble methods
- Summary

Issues: Data Preparation

- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
- Data transformation
  - Generalize and/or normalize data

Issues: Evaluating Classification Methods

- Accuracy
  - classifier accuracy: predicting class label
  - predictor accuracy: guessing value of predicted attributes
- Speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Unit 4. Classification

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Bayesian classification
- Classification by decision tree induction
- Accuracy and error measures
- Ensemble methods
- Summary

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian Theorem: Basics

- Let **X** be a data sample ("*evidence*"): class label is unknown
- Let H be a *hypothesis* that X belongs to class C
- Classification is to determine P(H|**X**), the probability that the hypothesis holds given the observed data sample **X**
- P(H) (*prior probability*), the initial probability
  - E.g., **X** will buy computer, regardless of age, income, …
- P(**X**): probability that sample data is observed
- P(**X**|H) (*posteriori probability of X conditioned on H*), the probability of observing the sample **X**, given that the hypothesis holds
  - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Bayesian Theorem

- Given training data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes theorem

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})}$$

- Informally, this can be written as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to C_2 iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Towards Naïve Bayesian Classifier

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector **X** = $(x_1, x_2, \ldots, x_n)$
- Suppose there are *m* classes $C_1, C_2, \ldots, C_m$.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

$$where$$

$$P(X) = \sum P(\mathbf{X}|C_i)P(C_i)$$

- Since P(X) is constant for all classes, only needs to be maximized

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

Derivation of Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i, D}|$ (# of tuples of C_i in D)
- If A_k is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ and $P(x_k|C_i)$ is

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Naïve Bayes classifier - algorithm

The **naïve Bayesian** classifier, or **simple Bayesian** classifier, works as follows:

1. Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)$, depicting n measurements made on the tuple from n attributes, respectively, $A_1, A_2, \ldots, A_n$.
2. Suppose that there are m classes, $C_1, C_2, \ldots, C_m$. Given a tuple, $\boldsymbol{X}$, the classifier will predict that $\boldsymbol{X}$ belongs to the class having the highest posterior probability, conditioned on $\boldsymbol{X}$. That is, the naïve Bayesian classifier predicts that tuple $\boldsymbol{X}$ belongs to the class C_i if and only if

$$P(C_i|\boldsymbol{X}) > P(C_j|\boldsymbol{X}) \quad \text{for } 1 \le j \le m, j \ne i.$$

   Thus, we maximize $P(C_i|\boldsymbol{X})$. The class C_i for which $P(C_i|\boldsymbol{X})$ is maximized is called the *maximum posteriori hypothesis*. By Bayes' theorem (Eq. 8.10),

$$P(C_i|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|C_i)P(C_i)}{P(\boldsymbol{X})}. \tag{8.11}$$

3. As $P(\boldsymbol{X})$ is constant for all classes, only $P(\boldsymbol{X}|C_i)P(C_i)$ needs to be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, $P(C_1) = P(C_2) = \cdots = P(C_m)$, and we would therefore maximize $P(\boldsymbol{X}|C_i)$. Otherwise, we maximize $P(\boldsymbol{X}|C_i)P(C_i)$. Note that the class prior probabilities may be estimated by $P(C_i) = |C_{i,D}|/|D|$, where $|C_{i,D}|$ is the number of training tuples of class C_i in D.
4. Given data sets with many attributes, it would be extremely computationally expensive to compute $P(\boldsymbol{X}|C_i)$. To reduce computation in evaluating $P(\boldsymbol{X}|C_i)$, the naïve assumption of **class-conditional independence** is made. This presumes that the attributes' values are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus,

$$\begin{aligned} P(\boldsymbol{X}|C_i) &= \prod_{k=1}^{n} P(x_k|C_i) \qquad (8.12) \\ &= P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i). \end{aligned}$$

We can easily estimate the probabilities $P(x_1|C_i), P(x_2|C_i), \ldots, P(x_n|C_i)$ from the training tuples. Recall that here x_k refers to the value of attribute A_k for tuple $\boldsymbol{X}$. For each attribute, we look at whether the attribute is categorical or continuous-valued. For instance, to compute $P(\boldsymbol{X}|C_i)$, we consider the following:

(a) If A_k is categorical, then $P(x_k|C_i)$ is the number of tuples of class C_i in D having the value x_k for A_k, divided by $|C_{i,D}|$, the number of tuples of class C_i in D.

(b) If A_k is continuous-valued, then we need to do a bit more work, but the calculation is pretty straightforward. A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ, defined by

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{8.13}$$

so that

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}). \tag{8.14}$$

Naïve Bayesian Classifier: Training Dataset

Class:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

Data sample
X = (age <=30, Income = medium,
Student = yes, Credit_rating = Fair)

age	income	student	credit_rating	_comp
<=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

Naïve Bayesian Classifier: An Example

$P(C_i)$ – Prior probability of each class :

P(buys_computer = "yes") = 9/14 = 0.643
P(buys_computer = "no") = 5/14= 0.357

Compute likelihood $P(X|C_i)$ of each attribute value for each class

P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6

P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
P(income = "medium" | buys_computer = "no") = 2/5 = 0.4

P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
P(student = "yes" | buys_computer = "no") = 1/5 = 0.2

P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

Naïve Bayesian Classifier: An Example

- **For the unseen data, classify using bayes classifier**
- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

P(X|C$_i$) : P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

P(**C$_i$** |**X**) = **(P(X|C$_i$)*P(C$_i$)**

P(buys_computer = yes |X) = P(X|buys_computer = "yes") * P(buys_computer = "yes")
= 0.028

P(buys_computer = no |X) = P(X|buys_computer = "no") * P(buys_computer = "no")
= 0.007

P(buys_computer = yes |X) > P(buys_computer = no |X)

Therefore, X belongs to class ("buys_computer = yes")

Note: P(X) = 0.028+0.007 = 0.035

Avoiding the 0-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10),
- Use Laplacian correction (or Laplacian estimator)
  - Adding 1 to each case

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayesian Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.

      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

X = <Adam, M, 1.95m>

Name	Gender	Height	Output1
Kristina	F	1.6m	Short
Jim	M	2m	Tall
Maggie	F	1.9m	Medium
Martha	F	1.88m	Medium
Stephanie	F	1.7m	Short
Bob	M	1.85m	Medium
Kathy	F	1.6m	Short
Dave	M	1.7m	Short
Worth	M	2.2m	Tall
Steven	M	2.1m	Tall
Debbie	F	1.8m	Medium
Todd	M	1.95m	Medium
Kim	F	1.9m	Medium
Amy	F	1.8m	Medium
Wynette	F	1.75m	Medium

Prior Probabilities: P(short) = 4/15 = 0.267; P(medium) = 8/15 = 0.533; P(tall) = 3/15 = 0.2

For continuous data -> assume Gaussian distribution

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

So that $\mathrm{P}(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$

So we need to calculate mean and standard deviation of the values for the attribute for class C
From training set, μ = 1.9 and σ = 0.3 then

1. P(height = 1.95|short): Range (1.6m – 1.7m) , μ = 1.65 and σ = 0.05 then

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\,0.05} e^{\frac{-(1.95-1.65)^2}{2*0.05^2}}$$

$g(1.95, \mu_{short}, \sigma_{short}) = 1.2154 \times 10^{-7}$

2. P(height = 1.95|medium): Range (1.75m – 1.95m) , μ = 1.85 and σ = 0.1 then

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\,0.1} e^{\frac{-(1.95-1.85)^2}{2*0.1^2}}$$

$g(1.95, \mu_{medium}, \sigma_{medium}) = 2.419$

3. P(height = 1.95|tall): Range (2.0 m – 2.2m) , $\mu = 2.1$ and $\sigma = 0.1$ then

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\,0.1} e^{\frac{-(1.95-2.1)^2}{2*0.1^2}}$$

$g(1.95, \mu_{tall}, \sigma_{tall}) = 1.29$

Posteriori probability of M conditioned on C
P(M|short) = 0 = 0
P(M|medium) = 2/8
P(M|tall) = 3/3

Posteriori probability of C conditioned on X

P(short|X) = P(M|short) x P(1.95|short) x P(short) = 0

P(medium|X) = 2/8 x 2.419 x 8/15= 0.3

P(tall|X) = 1 x 1.29 x 3/15 = 0.25

Since the probability of Tall is highest we classify X as Medium.

Prior Probabilities: P(short) = 4/15 = 0.267; P(medium) = 8/15 = 0.533; P(tall) = 3/15 = 0.2

Since height is continuous data -> split it into ranges
Height Range: (0,1.6] , (1.6,1.7] , (1.7, 1.8] , (1.8, 1.9] , (1.9, 2.0] , (2.0, ∞)

Posteriori probability of X conditioned on C
P(X|short) = 1/4 x 0 = 0
P(X|medium) = 2/8 x 1/8 = 0.031
P(X|tall) = 3/3 x 1/3 = 0.333

Posteriori probability of C conditioned on X
P(short|X) = 0 x 0.267 = 0
P(medium|X) = 0.031 x 0.533 = 0.016
P(tall|X) = 0.333 x 0.2 = 0.066

Since the probability of Tall is highest we classify X as Tall.

Example for Naïve Bayes Classification

Name	Hair	Height	Weight	Dublin	Result
Katie	Brown	Short	Light	Yes	None
Annie	Blonde	Short	Average	No	Sunburned
Emily	Red	Average	Heavy	No	Sunburned
Sarah	Blonde	Average	Light	No	Sunburned
Pete	Brown	Average	Heavy	No	Sunburned
Sam	Brown	Short	Average	Yes	Sunburned
Alex	Brown	Short	Average	No	None
John	Brown	Average	Heavy	No	None
Dana	Blonde	Tall	Average	No	None
Max	Red	Tall	Light	No	Sunburned

X = <brown, tall, average, no>

Unit 4. Classification

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Bayesian classification
- Classification by decision tree induction
- Accuracy and error measures
- Ensemble methods
- Summary

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

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i, D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

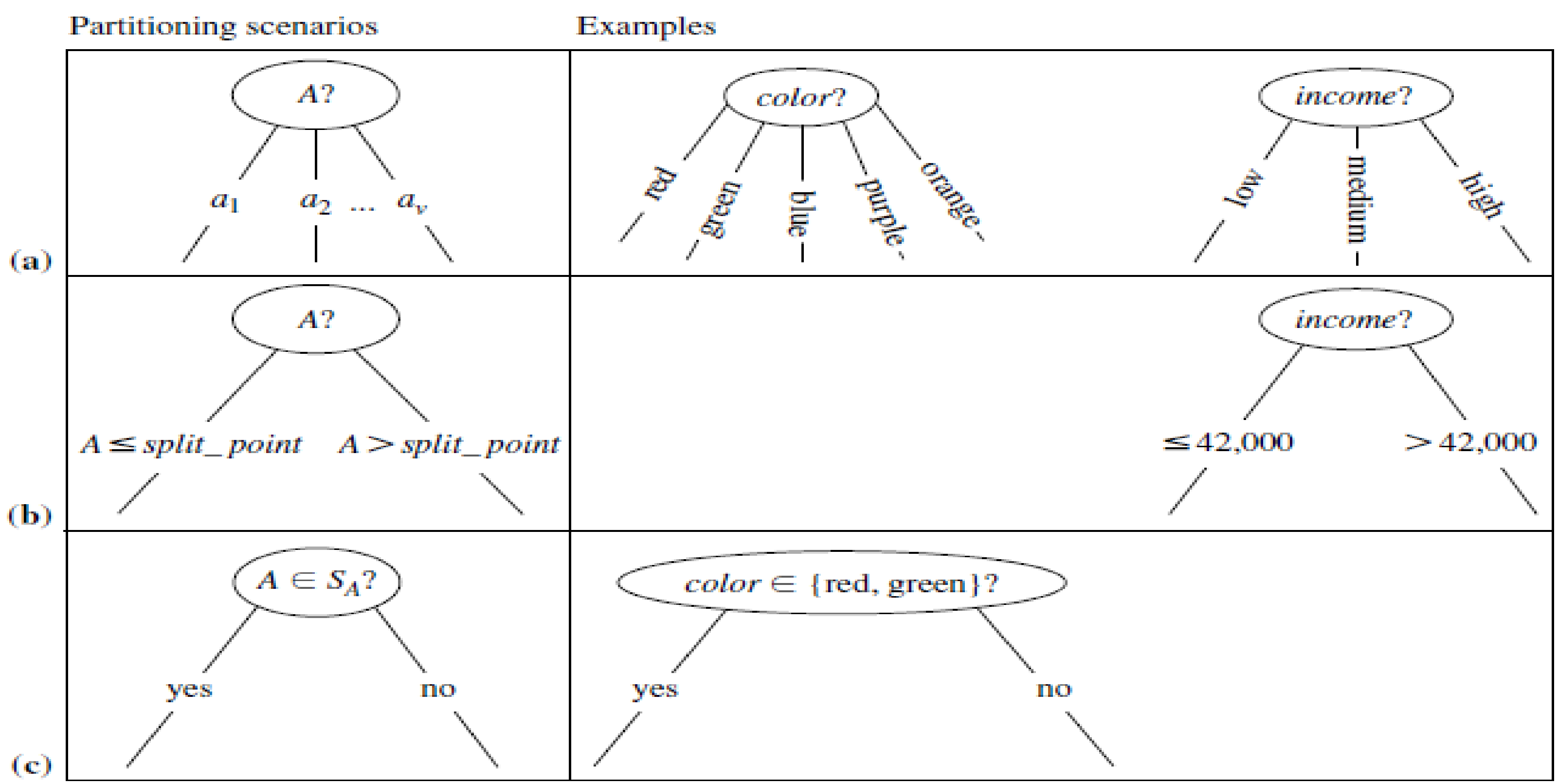
$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$


Partitioning scenarios
Examples
(a)
A?
a_1
a_2
...
a_v
color?
red
green
blue
purple
orange
income?
low
medium
high
(b)
A?
$A \leq split\_point$
$A > split\_point$
income?
$\leq 42{,}000$
$> 42{,}000$
(c)
$A \in S_A$?
yes
no
$color \in$ {red, green}?
yes
no

Algorithm for Decision Tree Induction

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, D.

Input:

- Data partition, D, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset.*

Output: A decision tree.

Method:

(1) create a node N;
(2) **if** tuples in D are all of the same class, C, **then**
(3)   return N as a leaf node labeled with the class C;
(4) **if** *attribute_list* is empty **then**
(5)   return N as a leaf node labeled with the majority class in D; // majority voting
(6) apply **Attribute_selection_method**(D, *attribute_list*) to **find** the "best" *splitting_criterion*;
(7) label node N with *splitting_criterion*;
(8) **if** *splitting_attribute* is discrete-valued **and**
    multiway splits allowed **then** // not restricted to binary trees
(9)   *attribute_list* $\leftarrow$ *attribute_list* $-$ *splitting_attribute*; // remove *splitting_attribute*
(10) **for each** outcome j of *splitting_criterion*
    // partition the tuples and grow subtrees for each partition
(11)   let D_j be the set of data tuples in D satisfying outcome j; // a partition
(12)   **if** D_j is empty **then**
(13)   attach a leaf labeled with the majority class in D to node N;
(14)   **else** attach the node returned by **Generate_decision_tree**(D_j, *attribute_list*) to node N;
    endfor
(15) return N;

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

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

Entropy of whole dataset

$$Info(D) = I(9,5)$$

$$Info(D) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14})$$

$$Info(D) = 0.940$$

Entropy of attribute age

Age	buys_computer = "yes"	buys_computer = "no"
<=30	2	3
31…40	4	0
>40	3	2

Attribute Selection: Information Gain

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

$$Info_{age}(D) = 0.694$$

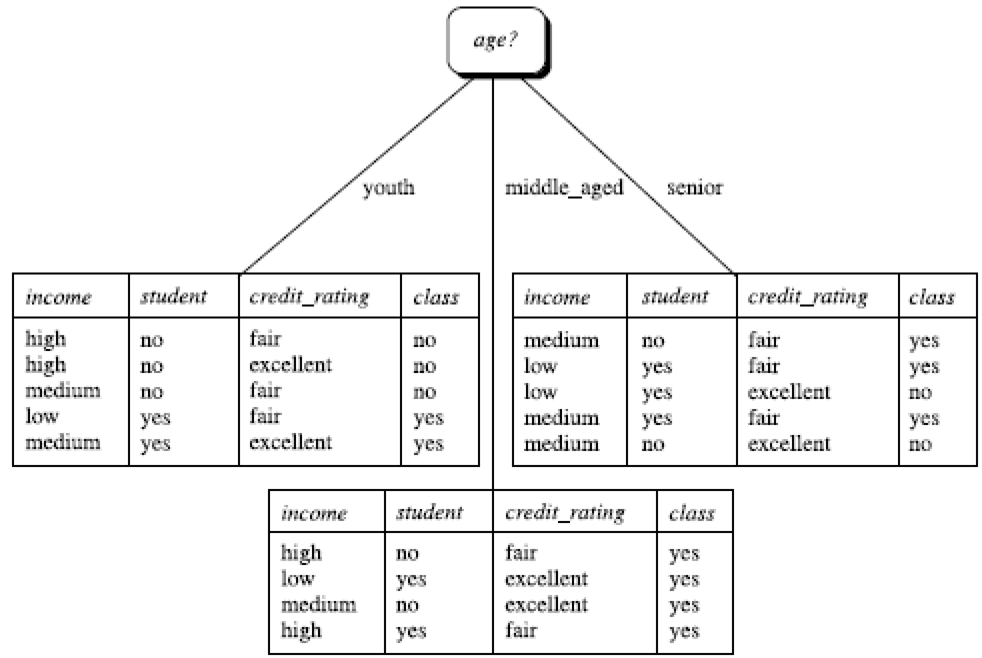
$\frac{5}{14} I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

$$Info_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4}\right) + \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right)$$
$$= 0.694$$

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Attribute Selection: Information Gain

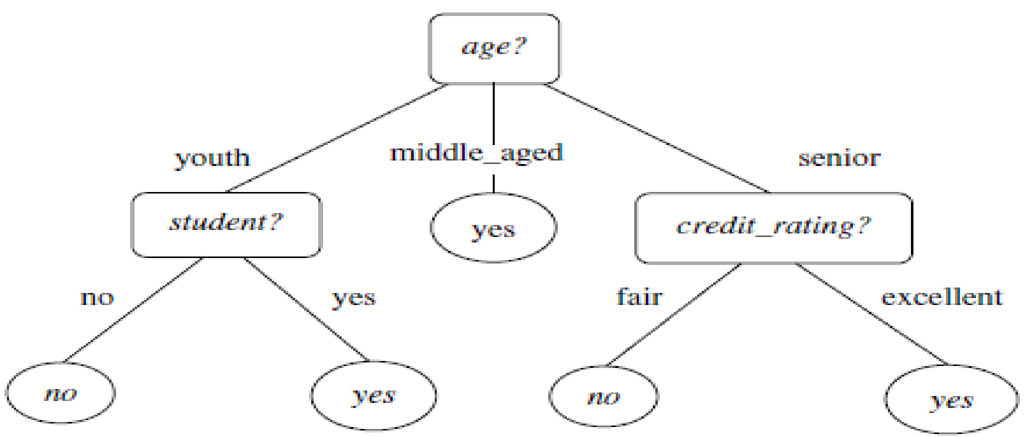
$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$



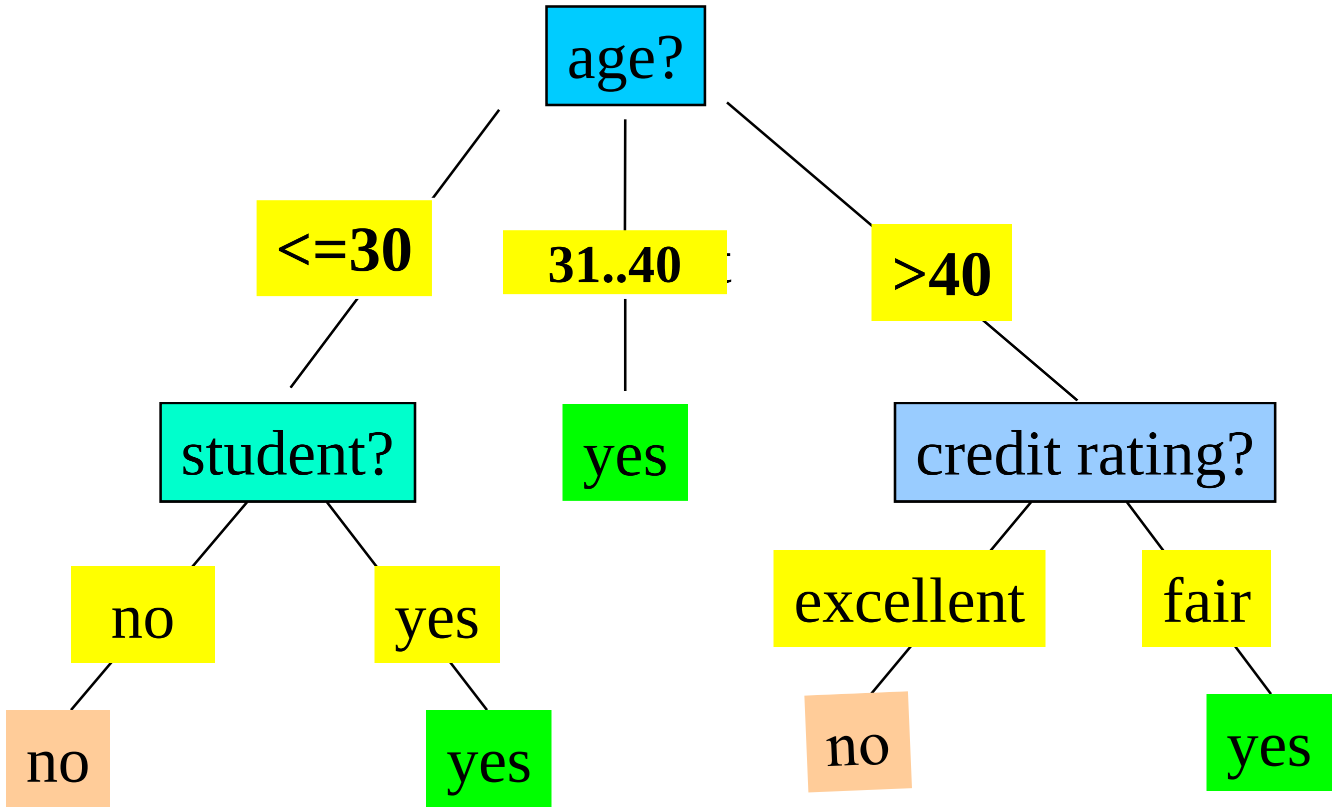
income	student	credit_rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	student	credit_rating	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

income	student	credit_rating	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no



Output: A Decision Tree for "*buys_computer*"

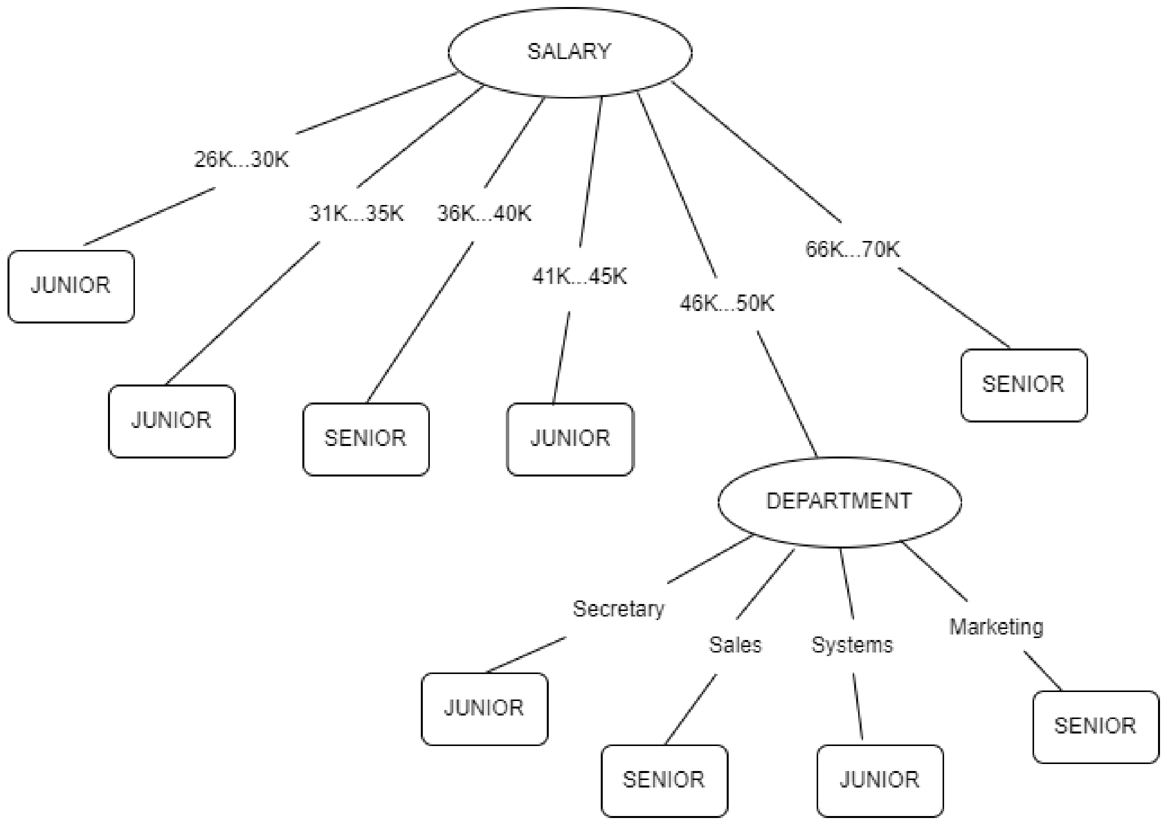


Example Decision Tree

- The following table consists of training data from an employee database. The data have been generalized. Let status be the class label attribute. Using ID3 algorithm construct a decision tree. Classify X = <systems, 26 . . . 30, 46–50K> using your constructed DT.

department	*status*	*age*	*salary*
sales	senior	31 . . . 35	46K . . . 50K
sales	junior	26 . . . 30	26K . . . 30K
sales	junior	31 . . . 35	31K . . . 35K
systems	junior	21 . . . 25	46K . . . 50K
systems	senior	31 . . . 35	66K . . . 70K
systems	junior	26 . . . 30	46K . . . 50K
systems	senior	41 . . . 45	66K . . . 70K
marketing	senior	36 . . . 40	46K . . . 50K
marketing	junior	31 . . . 35	41K . . . 45K
secretary	senior	46 . . . 50	36K . . . 40K
secretary	junior	26 . . . 30	26K . . . 30K

Example Decision Tree



X = <systems, 26 . . . 30, 46–50K> is classified as Junior

Computing Information-Gain for Continuous-Value Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

- Example: let Age be

Age	40	48	60	72	80	90
Buys	No	No	Yes	Yes	Yes	No

- Determine the threshold where a change in the class is observed : (48+60)/2 = 54 and (80+90)/2 = 85
- Evaluate the information gain for Age > 54 and Age > 85
- Select threshold on information gain.

Information gain of Age (>54, <54)

D = [3 Y, 3 N] Info (D)= $-3/6 \log_2 3/6 - 3/6 \log_2 3/6 = 1.0$

D (<54) = [0 Y, 2 N] Info (<54) = $0 - 2/2 \log_2 2/2 = 0$

D (>54) = [3 Y, 1 N] Info (>54)= $-3/4 \log_2 3/4 - 1/4 \log_2 1/4 = 0.81$

Gain (D, Age>54) = Info(D) - $Info_A(D)$ = $Info(D) - \sum_{v \in \{<54, >54\}} \frac{|D_v|}{|D|} Info(D_v)$ = 1 – 2/6 Info(D<54) – 4/6 Info(D>54)
= 1 – 0 – 4/6*0.81 = 0.45

Information gain of Age (> 85, <85)

D = [3 Y, 3 N] Info (D)= $-3/6 \log_2 3/6 - 3/6 \log_2 3/6 = 1.0$

D (<85) = [3 Y, 2 N] Info (<54) = $-3/5 \log_2 3/5 - 2/5 \log_2 2/5 = 0.971$

D (>85) = [0 Y, 1 N] Info (>54)= $0 - 1/1 \log_2 1/1 = 0$

Gain (D, Age>85) = Info(D) - $Info_A(D)$ = $Info(D) - \sum_{v \in \{<85, >85\}} \frac{|D_v|}{|D|} Info(D_v)$ = 1 – 5/6 Info(D<85) – 1/6 Info(D>85)
= 1 – 5/6 * 0.971 - 0 = 0.19

Computing Information-Gain for Continuous-Value Attributes

Gain (D, Age>54) = 0.45

Gain (D, Age>85) = 0.19

- Age >54 has higher gain, hence selected as splitting attribute value
- Now with this discretization we have 2 groups Age < 54 and Age > 54 instead of 6 different values

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

  - GainRatio(A) = Gain(A)/SplitInfo(A)

- Ex.

$$SplitInfo_A(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 0.926$$

  - gain_ratio(income) = 0.029/0.926 = 0.031

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

Suppose you have a dataset about whether to play tennis based on certain weather conditions:

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Using gain ratio

$$\text{Entropy}(S) = -\left(\frac{9}{14} \times \log_2\left(\frac{9}{14}\right)\right) - \left(\frac{5}{14} \times \log_2\left(\frac{5}{14}\right)\right) = 0.940$$

Outlook has three possible values: Sunny, Overcast, and Rain.

- **For Sunny:**

$$\text{Entropy}(Sunny) = -\left(\frac{2}{5} \times \log_2\left(\frac{2}{5}\right)\right) - \left(\frac{3}{5} \times \log_2\left(\frac{3}{5}\right)\right) = 0.971$$

- **For Overcast:**

$$\text{Entropy}(Overcast) = 0 \quad \text{(all are positive examples)}$$

- **For Rain:**

$$\text{Entropy}(Rain) = -\left(\frac{3}{5} \times \log_2\left(\frac{3}{5}\right)\right) - \left(\frac{2}{5} \times \log_2\left(\frac{2}{5}\right)\right) = 0.971$$

Now, calculate the gain:

$$\text{Gain}(S, \text{Outlook}) = 0.940 - \left(\frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971\right) = 0.246$$

Using gain ratio

- For Temperature:
  - Hot: $\text{Entropy}(Hot) = 1$
  - Mild: $\text{Entropy}(Mild) = 0.918$
  - Cool: $\text{Entropy}(Cool) = 0.811$

$$\text{Gain}(S, \text{Temperature}) = 0.940 - \left(\frac{4}{14} \times 1 + \frac{6}{14} \times 0.918 + \frac{4}{14} \times 0.811\right) = 0.029$$

- For Humidity:
  - High: $\text{Entropy}(High) = 0.985$
  - Normal: $\text{Entropy}(Normal) = 0.591$

$$\text{Gain}(S, \text{Humidity}) = 0.940 - \left(\frac{7}{14} \times 0.985 + \frac{7}{14} \times 0.591\right) = 0.151$$

- For Wind:
  - Weak: $\text{Entropy}(Weak) = 0.811$
  - Strong: $\text{Entropy}(Strong) = 1$

$$\text{Gain}(S, \text{Wind}) = 0.940 - \left(\frac{8}{14} \times 0.811 + \frac{6}{14} \times 1\right) = 0.048$$

Using gain ratio

2. Calculate the Split Information for Each Attribute

The Split Information is calculated as follows:

$$\text{Split Information}(A) = -\sum_{v \in \text{Values of A}} \frac{|S_v|}{|S|} \times \log_2\left(\frac{|S_v|}{|S|}\right)$$

- **For Outlook:**

$$\text{Split Information}(\text{Outlook}) = -\left(\frac{5}{14} \times \log_2\left(\frac{5}{14}\right) + \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) + \frac{5}{14} \times \log$$

- **For Temperature:**

$$\text{Split Information}(\text{Temperature}) = -\left(\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) + \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) + \frac{4}{14}\right.$$

- **For Humidity:**

$$\text{Split Information}(\text{Humidity}) = -\left(\frac{7}{14} \times \log_2\left(\frac{7}{14}\right) + \frac{7}{14} \times \log_2\left(\frac{7}{14}\right)\right) = 1.000$$

- **For Wind:**

$$\text{Split Information}(\text{Wind}) = -\left(\frac{8}{14} \times \log_2\left(\frac{8}{14}\right) + \frac{6}{14} \times \log_2\left(\frac{6}{14}\right)\right) = 0.985$$

Using gain ratio

3. **Calculate the Gain Ratio for Each Attribute**

The Gain Ratio is calculated as:

$$\text{Gain Ratio}(A) = \frac{\text{Gain}(A)}{\text{Split Information}(A)}$$

- **For Outlook:**

$$\text{Gain Ratio}(\text{Outlook}) = \frac{0.246}{1.577} = 0.156$$

- **For Temperature:**

$$\text{Gain Ratio}(\text{Temperature}) = \frac{0.029}{1.557} = 0.019$$

- **For Humidity:**

$$\text{Gain Ratio}(\text{Humidity}) = \frac{0.151}{1.000} = 0.151$$

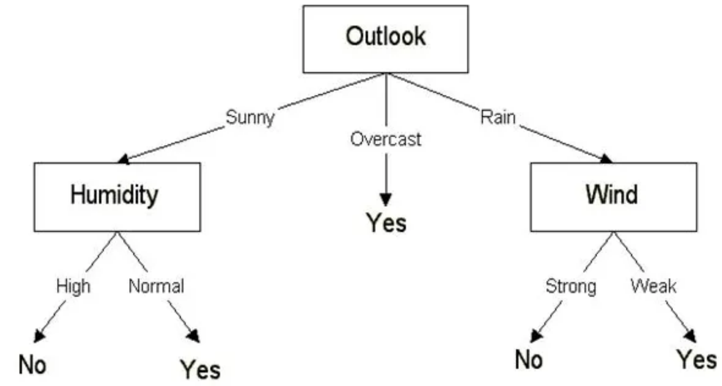
- **For Wind:**

$$\text{Gain Ratio}(\text{Wind}) = \frac{0.048}{0.985} = 0.049$$

4. **Select the Attribute with the Highest Gain Ratio**

Comparing the Gain Ratios, we have:

- **Outlook:** 0.156
- **Temperature:** 0.019
- **Humidity:** 0.151
- **Wind:** 0.049



Gini index (CART, IBM IntelligentMiner)

- 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

- If a data set D is split on A into two subsets D_1 and D_2, the *gini* index $gini(D)$ is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- 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*)

Gini Index [CART] - Example

Building a Decision Tree [CART]

- Consider the following dataset, we want to decide whether the customer is likely to buys_computer or not for 14 records, where
  - Class **P** = 9: buys_computer = "**yes**"
  - Class **N** = 5: buys_computer = "**no**"

1. Compute the ***Gini index*** for the overall collection of training examples.
2. Select the best split (among ***age, income, student,*** and ***credit_rating***) for the root. List/show all the splits you considered together with their corresponding values of the ***Gini index***. Justify your selection for the root split condition.
3. Find all the remaining splits to construct a full decision tree where all leaves contain only a single class. List/show all the splits that you considered, include Gini index for each one. Assign a class to each leaf.
4. Use the final tree to classify the record (***youth, low, no, excellent***).

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

Gini Index [CART] - Example

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- Compute the impurity of D:
- or Calculate ***Gini index*** of Class attribute
  - Total tuples: 14
  - Class P = 9: buys_computer = "yes"
  - Class N = 5: buys_computer = "no"

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

- We, need to compute the ***Gini Index*** of each attribute (age, income, student, credit_rating)

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

		Class		
		Yes	No	
age	Youth	2	3	5
	Middle aged	4	0	4
	senior	3	2	5
		9	5	14

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

- Lets now consider: **age:** {youth, middle_aged, senior}
  - Now consider each possible splitting subsets

{{youth, middle_aged}, {youth, senior}, {middle_aged, senior}, {youth},{middle_aged}, {senior}}

$$gini_{age \in \{youth, middle\_aged\}}(D) = \left(\frac{D_1}{14}\right) gini(D_1) + \left(\frac{D_2}{14}\right) gini(D_2)$$

$$= \frac{9}{14}\left(1 - \left(\frac{6}{9}\right)^2 - \left(\frac{3}{9}\right)^2\right) + \frac{5}{14}\left(1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2\right) = \mathbf{0.4571}$$

$$= \boldsymbol{gini}_{age \in \{senior\}}(D)$$

$$gini_{age \in \{youth, senior\}}(D) = \left(\frac{D_1}{14}\right) gini(D_1) + \left(\frac{D_2}{14}\right) gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{5}{10}\right)^2 - \left(\frac{5}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{4}{4}\right)^2 - \left(\frac{0}{4}\right)^2\right) = \mathbf{0.3571}$$

$$= \boldsymbol{gini}_{age \in \{middle\_aged\}}(D)$$

$$gini_{age \in \{middle\_aged, senior\}}(D) = \left(\frac{D_1}{14}\right) gini(D_1) + \left(\frac{D_2}{14}\right) gini(D_2)$$

$$= \frac{9}{14}\left(1 - \left(\frac{7}{9}\right)^2 - \left(\frac{2}{9}\right)^2\right) + \frac{5}{14}\left(1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2\right) = \mathbf{0.3936} = \boldsymbol{gini}_{age \in \{youth\}}(D)$$

		Class		
		yes	no	
income	low	3	1	4
	medium	4	2	6 ✓
	high	2 ✓	2 ✓	4 ✓
		9	5	14

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

$$\Delta gini(A) = gini(D) - gini_A(D)$$

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

- Lets first consider: **income:** {low, medium, high}
    - Now consider each possible splitting subsets

{{low, medium}, {low, high}, {medium, high}, {low}, {medium}, {high}}

$$\boldsymbol{gini}_{income \in \{low,medium\}}(D) = \left(\frac{D_1}{14}\right) gini(D_1) + \left(\frac{D_2}{14}\right) gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) = \mathbf{0.4428}$$

$$= \boldsymbol{gini}_{income \in \{high\}}(D)$$

$$\boldsymbol{gini}_{income \in \{low,high\}}(D) = \left(\frac{D_1}{14}\right) gini(D_1) + \left(\frac{D_2}{14}\right) gini(D_2)$$

$$= \frac{8}{14}\left(1 - \left(\frac{5}{8}\right)^2 - \left(\frac{3}{8}\right)^2\right) + \frac{6}{14}\left(1 - \left(\frac{4}{6}\right)^2 - \left(\frac{2}{6}\right)^2\right) = \underline{\mathbf{0.4583}} ✓$$

$$= \boldsymbol{gini}_{income \in \{medium\}}(D)$$

$$\boldsymbol{gini}_{income \in \{medium,high\}}(D) = \left(\frac{D_1}{14}\right) gini(D_1) + \left(\frac{D_2}{14}\right) gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2\right) = \mathbf{0.45} = \boldsymbol{gini}_{income \in \{low\}}(D)$$

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- Lets now consider: **student**
  - It is a binary attribute

		Class		
		yes	no	
student	yes	6	1	7 ✓
	no	3	4	7
				14

$$gini_{student}(D) = \left(\frac{D_1}{14}\right) gini(D_1) + \left(\frac{D_2}{14}\right) gini(D_2)$$

$$= \frac{7}{14}\left(1 - \left(\frac{6}{7}\right)^2 - \left(\frac{1}{7}\right)^2\right) + \frac{7}{14}\left(1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2\right) = \mathbf{0.3673}$$

- Lets now consider: **credit_rating**
  - It is a binary attribute

		Class		
		yes	no	
credit_rating	fair	6	2	8
	excellent	3	3	6
				14

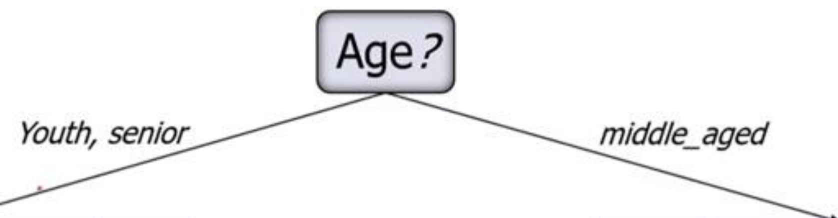
$$gini_{credit-rating}(D) = \left(\frac{D_1}{14}\right) gini(D_1) + \left(\frac{D_2}{14}\right) gini(D_2)$$

$$= \frac{8}{14}\left(1 - \left(\frac{6}{8}\right)^2 - \left(\frac{2}{8}\right)^2\right) + \frac{6}{14}\left(1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2\right) = \mathbf{0.4285}$$

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

Attribute	Split	Gini index	Reduction in impurity $\Delta gini = gini(D) - gini_A(D)$
age	{youth, senior} or {middle_aged}	0.3571	0.459 - 0.3571 = 0.1019
income	{medium, high} or {low}	0.4428	0.459 - 0.4428 = 0.0162
student	Binary	0.3673	0.459 - 0.3673 = 0.0917
credit_rating	Binary	0.4285	0.459 - 0.4285 = 0.0305

- **age** is selected with minimum Gini index & highest reduction in impurity



income	student	credit_rating	class
high	no	Fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	Yes
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	student	credit_rating	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - Information gain:
    - biased towards multivalued attributes
  - Gain ratio:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - 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

Other Attribute Selection Measures

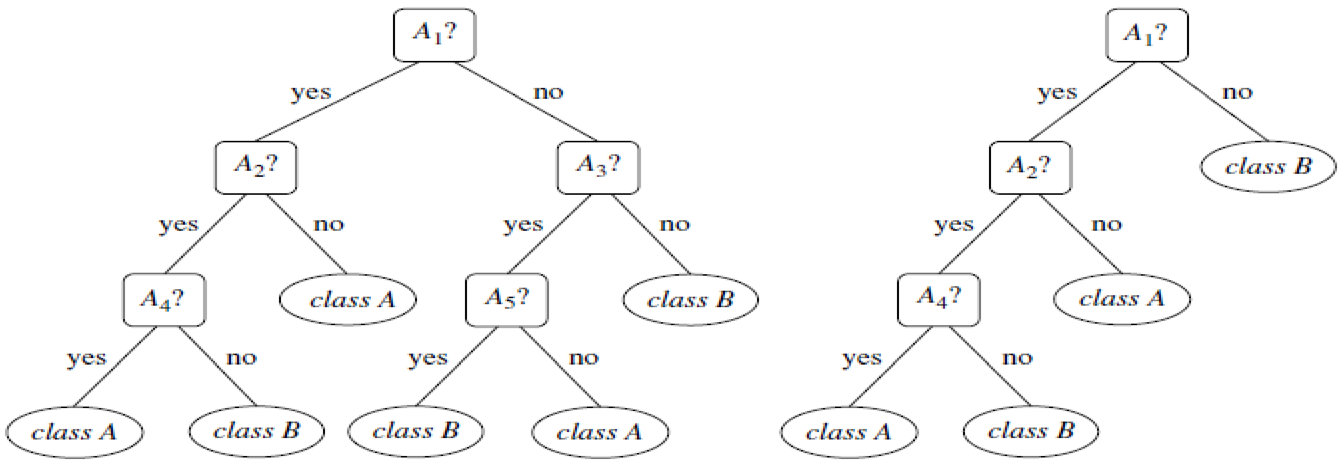
- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistics: has a close approximation to χ^2 distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than 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
    - Statistical significance, information gain, Gini index to be used for setting threshold
    - Difficult to choose an appropriate threshold
    - High Threshold – over simplified trees
    - Low Threshold – more splits, large trees with very little simplification

Overfitting and Tree Pruning

- Postpruning: Remove branches from a “fully grown”
  - Replace branches by a leaf node labeled with most frequent class among the subtree
  -



Overfitting and Tree Pruning

- PostPruning technique

  1. Cost complexity pruning: used by CART

     - Cost complexity of a tree to be a function of the number of leaves in the tree and the error rate of the tree (where the error rate is the percentage of tuples misclassified by the tree).
     - It starts from the bottom of the tree.
     - For each internal node, N, it computes the cost complexity of the subtree at N, and the cost complexity of the subtree at N if it were to be pruned (i.e., replaced by a leaf node).
     - The two values are compared.
     - If pruning the subtree at node N would result in a smaller cost complexity, then the subtree is pruned.
     - Otherwise, it is kept.
     - A separate pruning set is used to compute the cost complexity.

2. Pessimistic pruning: used by C4.5 similar to cost complexity

   - It also uses error rate estimates to make decisions regarding subtree pruning.
   - However, does not require the use of a prune set.
   - Instead, it uses the training set to estimate error rates.
   - Adjusts the error rates obtained from the training set by adding a penalty, so as to counter the bias incurred by using the training set as the prune set.

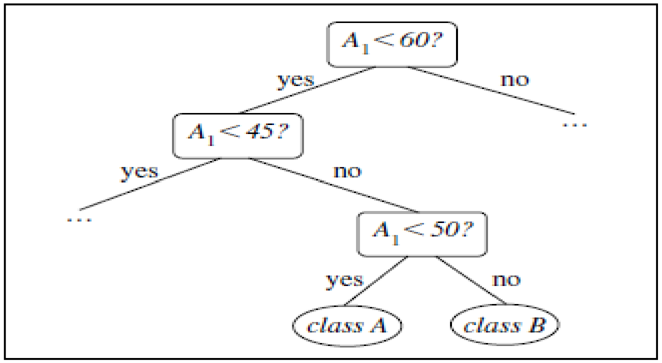
Overfitting and Tree Pruning

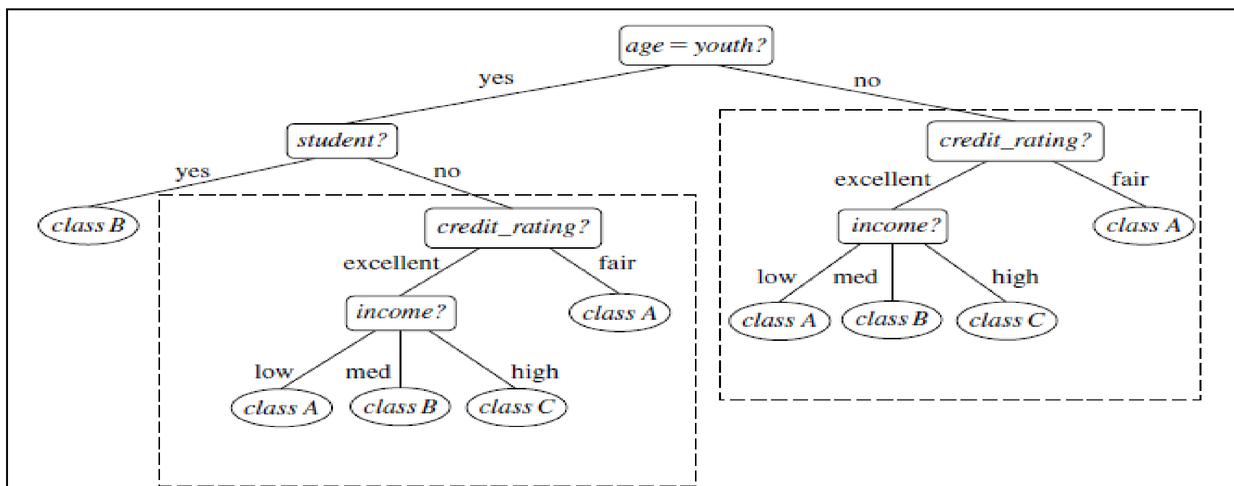
- Prune trees based on the number of bits required to encode them.
  - The “best” pruned tree is the one that minimizes the number of encoding bits.
  - This method adopts the MDL principle

- Which pruning is most suitable???
  - Postpruning requires more computation than prepruning
  - Interleaved pre and post pruning can be done
  - No one technique is best

Problems with Decision Trees

- Repetition and replication





Enhancements to Basic Decision Tree Induction

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

Unit 4. Classification

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Bayesian classification
- Classification by decision tree induction
- Accuracy and error measures
- Ensemble methods
- Summary

Metrics for evaluating classifier

- Training data – overoptimistic
- Testing data - √
- Positive tuples – tuples of main class of interest
- Negative tuples – other tuples
- True Positives (TP)
- True Negatives (TN)
- False Positives (FP)
- False Negatives (FN)

Confusion matrix

- Tool for analysing how well your classifier can recognize tuples of different classes
- TP & TN – indicates when classifier is getting things right
- FP & FN – indicates when classifier is getting things wrong

		Predicted class		
		Yes	No	Total
Actual class	Yes	True positive (TP)	False negative (FN)	P
	No	False positive (FP)	True negative (TN)	N
	Total	P′	N′	P+N

- Given m classes ($m \geq 2$), confusion matrix is m x m table
- Entry CMi,j   no. of tuples of class i labeled as class j
- Ideal confusion matrix – diagonal entries have more values and others are zero or close to zero

Confusion matrix

		Predicted class		
		Yes	No	Total
Actual class	Yes	True positive (TP)	False negative (FN)	P
	No	False positive (FP)	True negative (TN)	N
	Total	P′	N′	P+N

P = positive class tuples (TP + FN)

N = negative class tuples (FP + TN)

P′= tuples classified as positive (TP + FP)

N′= tuples classified as negative (FN + TN)

Total = TP+TN+FP+FN = P+N = P′+ N′

classes	buy_computer = yes	buy_computer = no	total	recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
total	7366	2634	10000	95.42

Accuracy = 6954+2588/10000 = 95.42
Sensitivity = 6954/7000 = 99.34
Specificity = 2588/3000 = 86.27
Precision = 6954/7366 = 94.40
Recall = 6954/7000 = 99.34

Confusion Matrix for information retrieval	
Ret , Rel	Rel, Not Ret
Ret, Not Rel	Not Rel, Not Ret

Evaluation Measures

- Accuracy of a classifier M, acc(M): percentage of test set tuples that are correctly classified by the model M

$$\text{accuracy} = \frac{TP+TN}{P+N}$$

- Error rate (misclassification rate) of M = 1 – acc(M)
- $error\,rate = 1 - accuracy = \frac{FP+FN}{P+N}$
- Accuracy measure works well when the distribution of data is balanced
- For imbalanced data other measures are used e.g. (cancer diagnosis)

Alternative Evaluation Measures

- Sensitivity = TP/P
  - true positive (recognition) rate
  - Proportion of positive tuples that are correctly identified
- Specificity = TN/N
  - true negative rate
  - Proportion of negative tuples that are correctly identified
- accuracy = sensitivity * P/(P + N) + specificity * N/(P + N)
- Precision = TP/(TP + FP)
  - Measure of exactness
  - What percentage of tuples labeled as positive are actually positive
- Recall = TP / (TP + FN)
  - Measure of completeness
  - What percentage of positive tuples are labeled as positive

Alternative Evaluation Measures

- F-measure:
  - Harmonic mean of precision and recall
  - Gives equal weight to precision and recall
  - $F = \frac{2 \times precision \times recall}{precision + recall}$

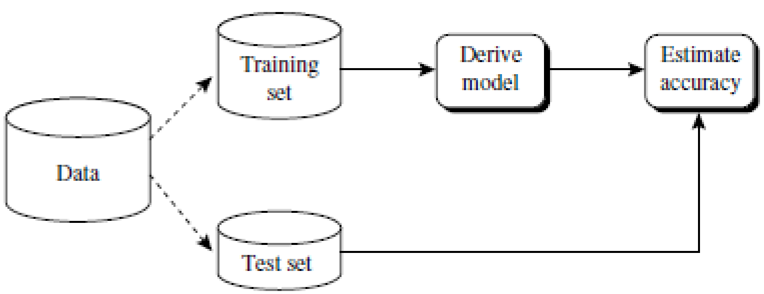
- F_β : weighted measure of precision and recall
  - $F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$
  - F_2 – which weights recall twice as much as precision
  - $F_{0.5}$ – which weights precision twice as much as recall

Other aspects to compare classifiers

- Speed – computational costs involved in generating and using the given classifier
- Robustness – ability of classifier to make correct predictions give noisy data or data with missing values
- Scalability – ability to construct the classifier efficiently given large amount of data
- Interpretability – level of understanding and insight provided by classifier
  - E.g. decision tree and classification rules are easy to interpret

Evaluating the Accuracy of a Classifier or Predictor (I)

- Holdout method
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
- Random subsampling: a variation of holdout
  - Repeat holdout k times,
  - accuracy = avg. of the accuracies obtained from each iteration



Evaluating the Accuracy of a Classifier or Predictor (I)

- <u>Cross-validation</u> (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets D_1, D_2, … D_k, each approximately equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - Each sample is used the same number of times for training and once for testing
-

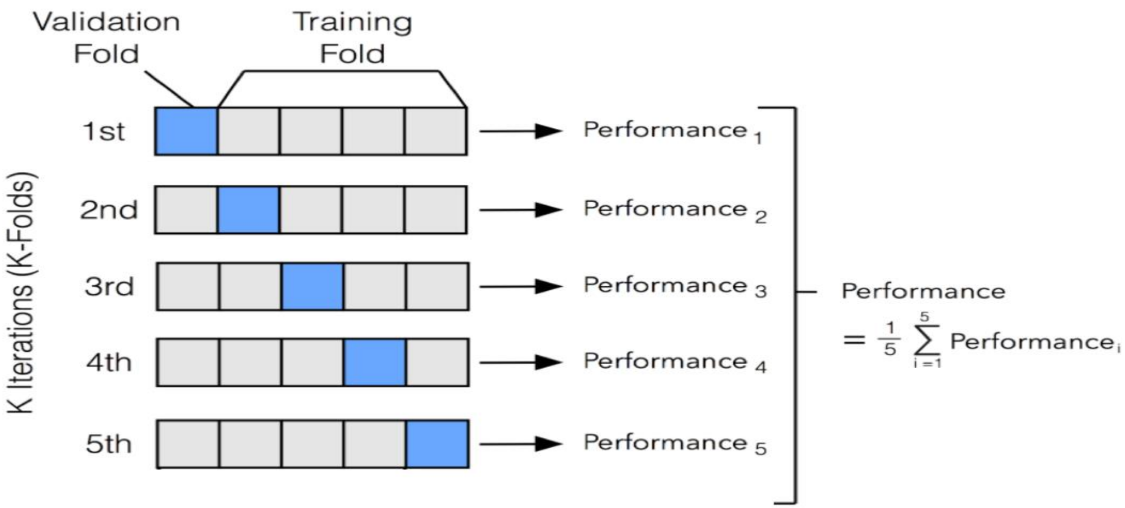


Image source: Kaggle.com

Evaluating the Accuracy of a Classifier or Predictor (I)

Leave-one-out:

- k folds where k = # of initial tuples, for small sized data
- One sample is left out at a time for test set

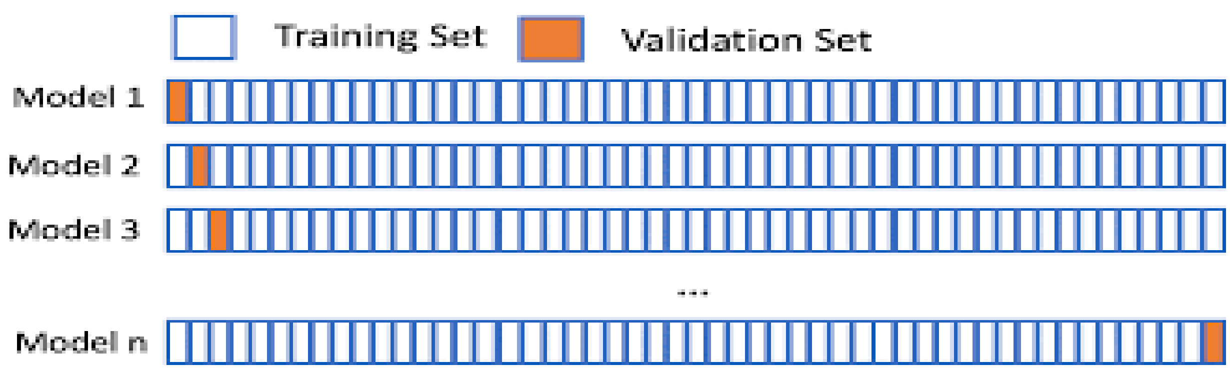


Image source: campus.datacamp.com

Evaluating the Accuracy of a Classifier or Predictor (I)

- Stratified cross-validation:
  - folds are stratified so that class distribution in each fold is approx. the same as that in the initial data
  - Stratified 10-fold cross-validation is recommended

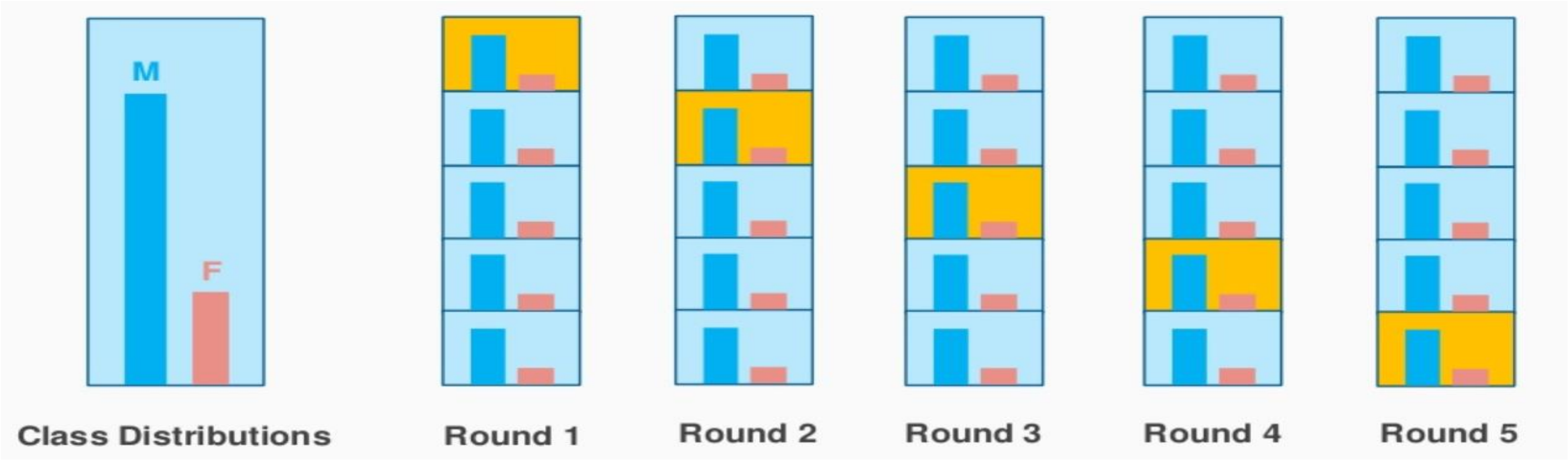


Image source: stats.stockexchange.com

Different cross validations

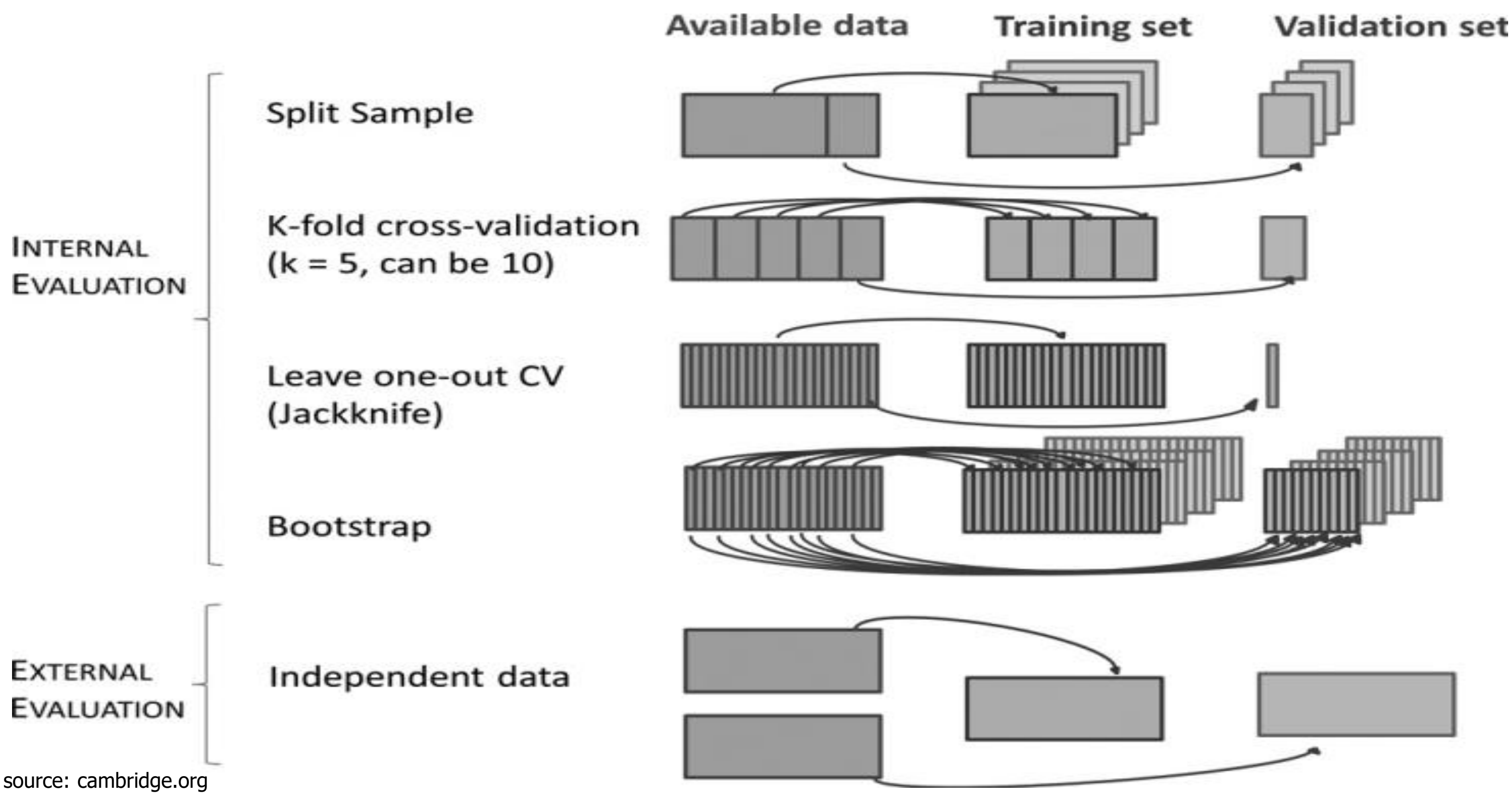


Image source: cambridge.org

Evaluating the Accuracy of a Classifier or Predictor (II)

- Bootstrap
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set

Evaluating the Accuracy of a Classifier or Predictor (II)

- Bootstrap
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 bootstrap**
  - Suppose we are given a data set of d tuples.
  - The data set is sampled d times, with replacement, resulting in a training set of d samples.
  - The data tuples that did not make it into the training set end up forming the test set.
  - About 63.2% of the original data will end up in the bootstrap, and the remaining 36.8% will form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedure k times, overall accuracy of the model:

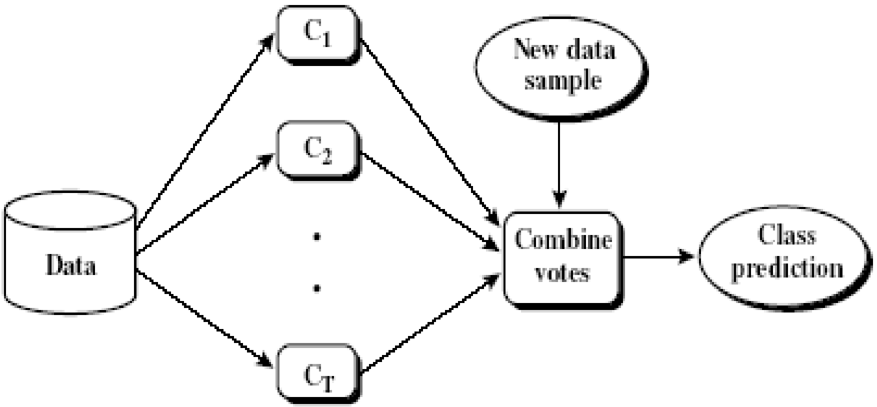
$$acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times acc(M_i)_{test\_set} + 0.368 \times acc(M_i)_{train\_set})$$

Unit 4. Classification

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Bayesian classification
- Classification by decision tree induction
- Accuracy and error measures
- Ensemble methods
- Summary

Techniques to improve classifier accuracy

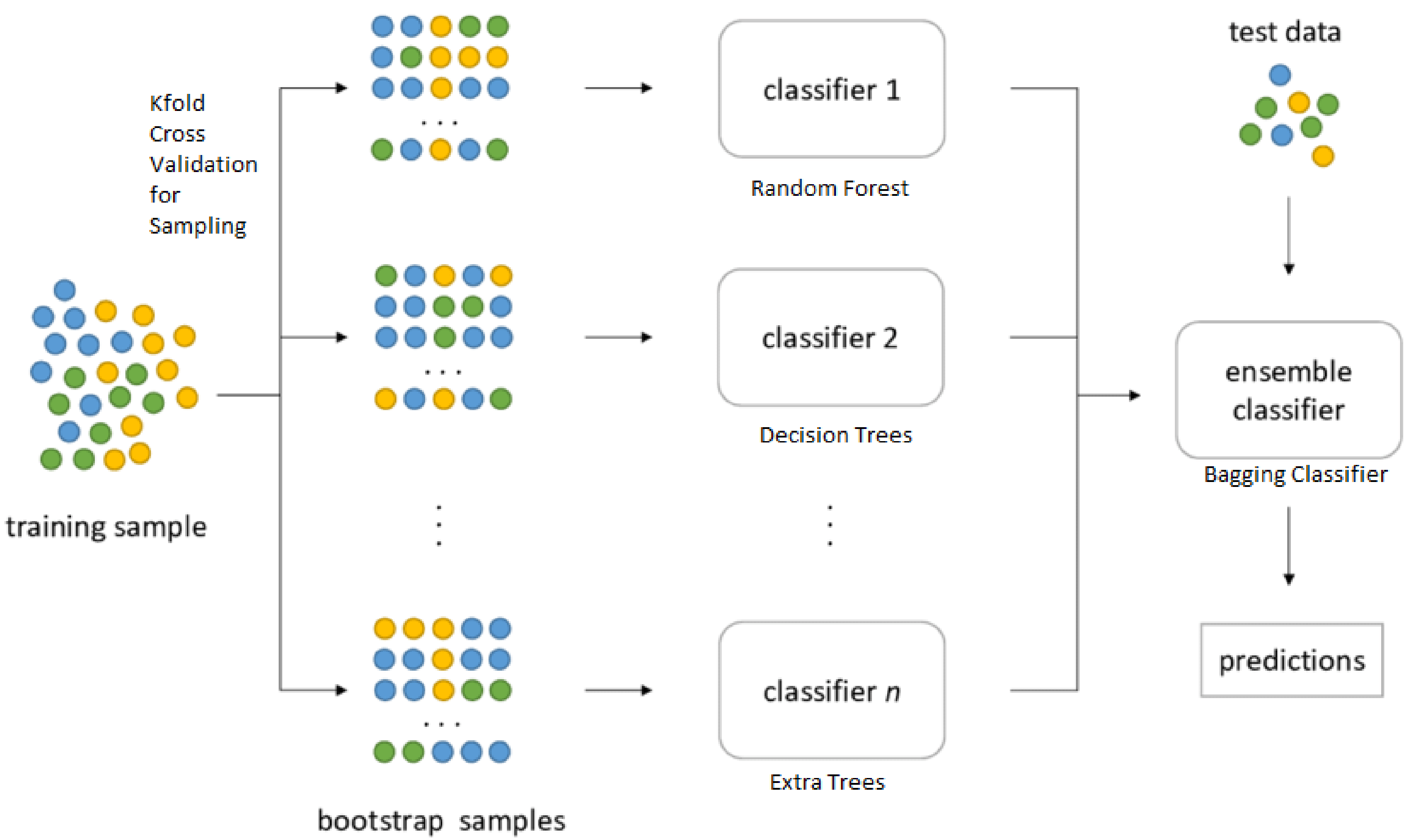
Ensemble Methods



- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

Bagging: Bootstrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significant better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction



Bagging Classifier Process Flow

Image Source: Internet

Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - Weights are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to pay more attention to the training tuples that were misclassified by M_i
  - The final M* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- The boosting algorithm can be extended for the prediction of continuous values
- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data

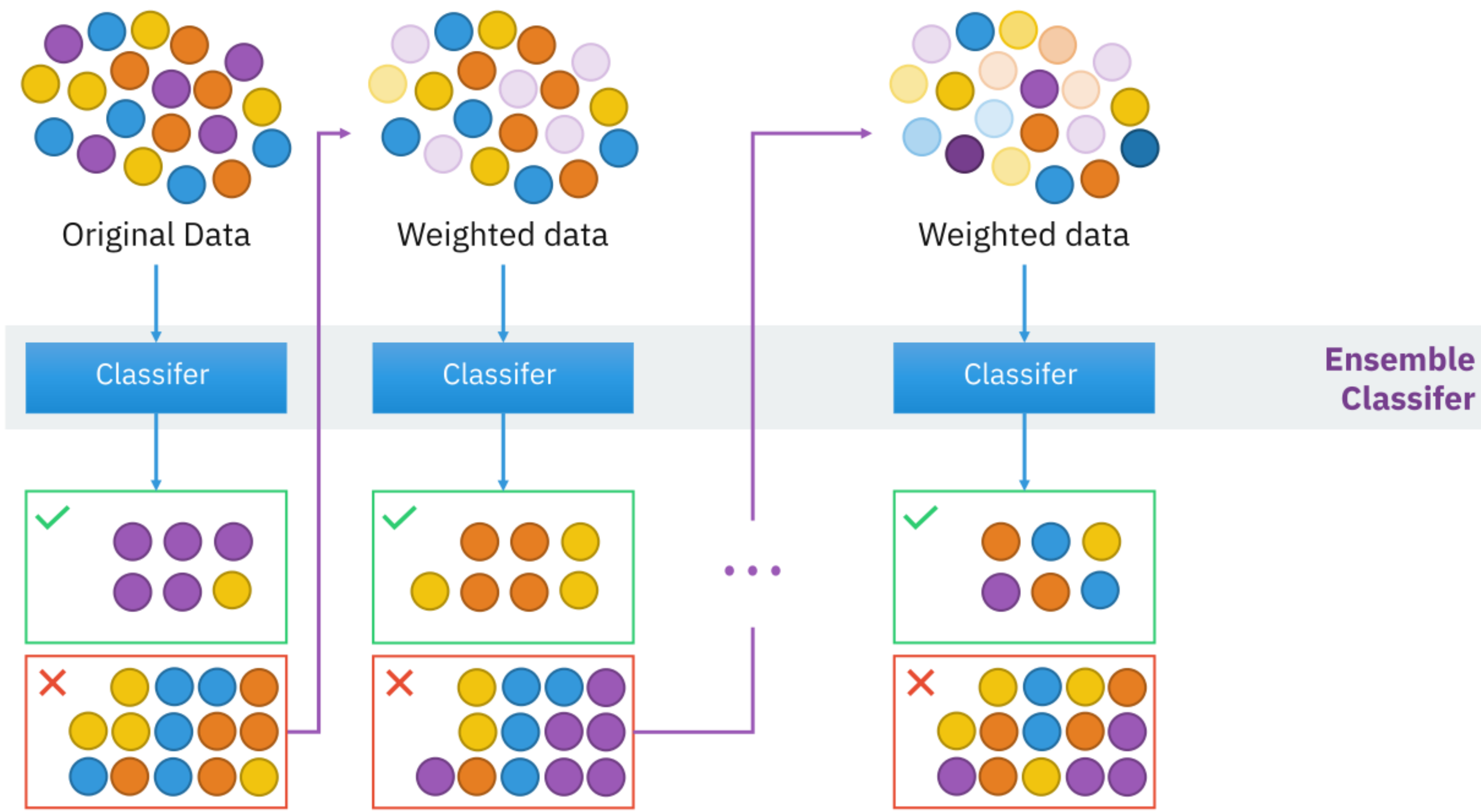


Image Source: Internet

Adaboost (Freund and Schapire, 1997)

- Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, y_1), …, ($\mathbf{X_d}$, y_d)
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model M_i is derived from D_i
  - Its error rate is calculated using D_i as a test set
  - If a tuple is misclassified, its weight is increased, else it is decreased

- Error rate: err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$.
  - Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_{j}^{d} w_j \times err(\mathbf{X_j})$$

Adaboost (Freund and Schapire, 1997)

- The weights of the tuples are updated as follows:
- Tuples that are correctly classified, the weight is multiplied by error(Mi)/(1-error(Mi))
- Once the weights of correctly classified tuples are updated, the weights for all tuples are normalized as
  - Weight x sum (old weights) / sum (updated weights)
- This will increase weights of misclassified tuple and decrease those of correctly classified tuples
- Classifying unseen tuples
  - Weight of each classifier's vote

$$\log \frac{1 - error(M_i)}{error(M_i)}$$

- Image source: [Internet]()