Classification - A Two step process

(1) Model construction: describing a set of predetermined classes

→ Each of tuple/sample is assumed to belong to a predefined class, as determined by the class label attribu

→ The set of tuples used for model construction is training set:

→ The model is represented as classification rules, decision trees, or mathematical formulae.

(2) Model usage: for classifying future or unknown objects.

→ Estimate accuracy of the model.

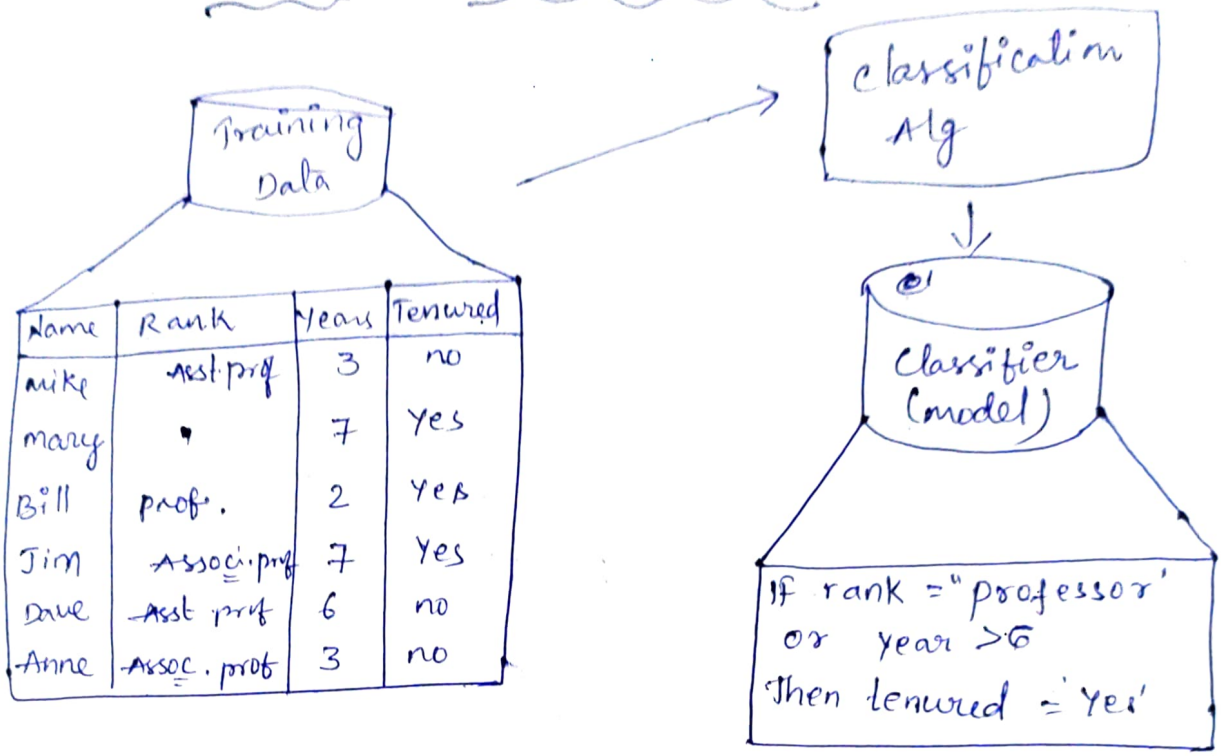
(1) The known label of the test sample is compared with the classified result from the model.

(2) Accuracy rate is the percentage of test set samples that are correctly classified by the model.

(3) Test set is independent of training set, otherwise over-fitting will occur

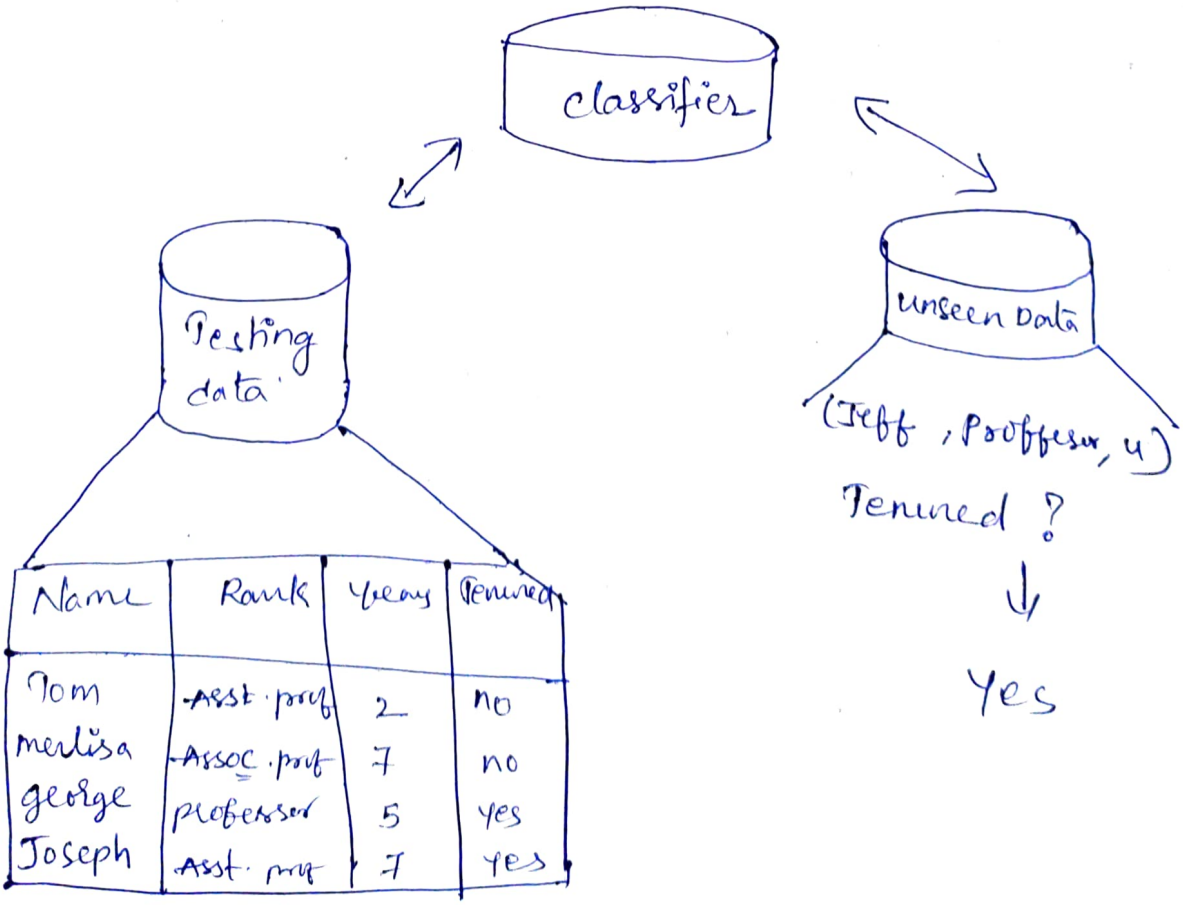
→ If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known.

Process ① : Model Construction



Name	Rank	Years	Tenured
mike	Asst. prof	3	no
mary		7	yes
Bill	prof.	2	yes
Jim	Assoc. prof	7	yes
Dave	Asst prof	6	no
Anne	Assoc. prof	3	no

Process ② : Using the model in prediction

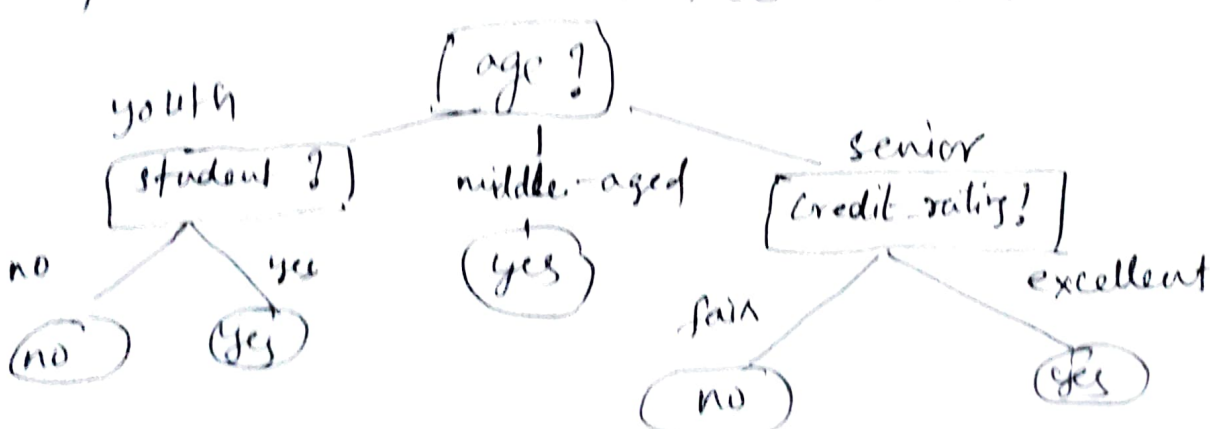


Name	Rank	Years	Tenured
Tom	Asst. prof	2	no
merlisa	Assoc. prof	7	no
george	professor	5	yes
Joseph	Asst. prof	7	yes

(Classification) by Decision Tree Induction

→ Decision tree induction is the learning of decision trees from class-labeled training tuples.

→ A decision tree is a flow-chart like tree structure, where each internal node (nonleaf node) denotes a test on an attribute, each branch repr an out come of the test, and each leaf node (or terminal node) holds a class label. the top-most node is a tree is the root node.



Decision Tree Induction

Alg: 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 ← attri_list − splitting_attribute; // remove splitting_attribute
⑩ for each outcome j of splitting_criterion.
// partition the tuples and grow subtrees for each partition

let D_j be the set of data tuples in D satisfying outcome j; // a partition

if D_j is empty then

attach a leaf labeled with the majority class in D to node N;

else attach the node returned by Generate_decision_tree (D_j, attribute_list) to node N;

end for

return N;

A is discrete valued
A is continuous - valued
A is discrete - valued and a binary tree

fig: Basic Alg for inducing a decision tree from training tuples.

→ The alg is called with 3 parameters : D, attribute_list and Attribute_selection_method

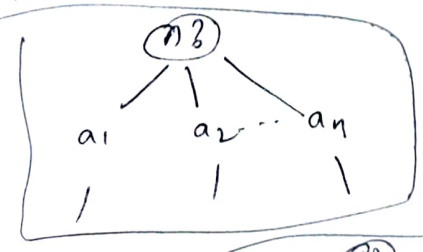
where D — data partition, Initially, it is the complete set of training tuples and their associated class labels.

→ The parameter attribute_list is a list of attributes describing the tuples.

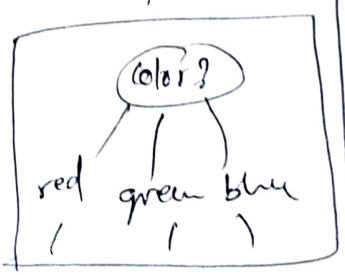
→ Attribute_selection_method specifies a heuristic procedure for selecting the attribute that "best" discriminates the given tuples according to class.

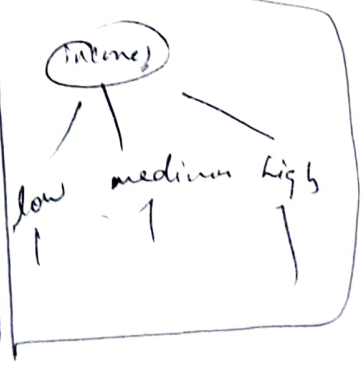
This procedure employs an attribute selection measure, such as information gain or the Gini index.

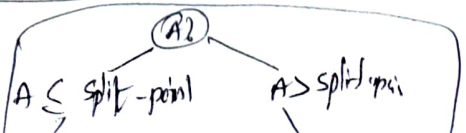
partitioning scenarios



Examples







Splitting Criterion

Attribute selection Measures

→ An attribute selection measure is a heuristic for selecting the splitting criterion that "best" separates a given data partition, D, of class-labeled training tuples into individual classes.

Information gain

→ ID3 (Iterative Dichotomizer) uses info gain as its attribute selection measure. This measure is based on pioneering work by Claude Shannon on info theory, which studied the value or "info content" of messages.

Let node N rep or hold the tuples of partition D. The attr with the highest info gain is chosen as the splitting attr for the node N. This attr minimizes the info needed to classify the tuples in the resulting partitions and reflects the least randomness or "impurity" in these partitions.

→ Such an approach minimizes the expected no. of tests needed to classify a given tuple and guarantees that a simple (but not necessarily the simplest) tree is found.

The expected info needed to classify a tuple in D is given by

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

→ where P_i is the probability that an arbitrary tuple in D belongs to class C_i and is estimated by $|C_{i,D}| / |D|$

→ A log function to the base is used, because the information is encoded in bits.

→ Info(D) is just the average amount of information needed to identify the class label of a tuple in D.

→ Note that, at this point, the Info we have is based solely on the proportions of tuples of each class.

→ Info(D) is also known as the entropy of D.

Now, Suppose we were to partition the tuples in D on some attribute A having V distinct values, $\{a_1, a_2 \ldots a_v\}$ as observed from the training data

→ If A is discrete-valued, these values correspond directly to the V outcomes of a test on A.

→ Attribute A can be used to split D into v partitions or subsets, $\{D_1, D_2 \ldots D_v\}$, where D_j contains those tuples in D that have outcome a_j of A.

→ These partitions would correspond to the branches grown from node N.

→ Ideally, we would like this partitioning to produce an exact classification of the tuples.

→ That is, we would like for each partition to be pure.

→ However, it is quite likely that the partitions will be impure (e.g where a partition may contain a collection of tuples from diff classes rather than from a single class)

→ How much more info would we still need (after the partitioning) in order to arrive at an exact classification?

→ This amount is measured by

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

→ The term $\frac{|D_j|}{|D|}$ acts as the weight of the jth partition.

→ $Info_A(D)$ is the expected info required to classify a tuple from D based on the partitioning by A.

→ Information gain is defined as the difference b/w the original info requirement (i.e based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on A).

→ That is

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

→ Gain(A) tells us how much would be gained by branching on A.

$$\text{Info}(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.409 + 0.530$$

$$= 0.940 \text{ bits}$$

$$\text{Info}_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)\right)$$

$$+ \frac{4}{14} \times \left(-\frac{4}{4}\log\frac{4}{5} - \frac{0}{4}\log_2\frac{0}{4}\right)$$

$$+ \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right)$$

$$= 0.694 \text{ bits}.$$

Hence, the gain in information from such a partitioning would be

$$\text{Gain}(age) = \text{Info}(D) - \text{Info}_A(D)$$

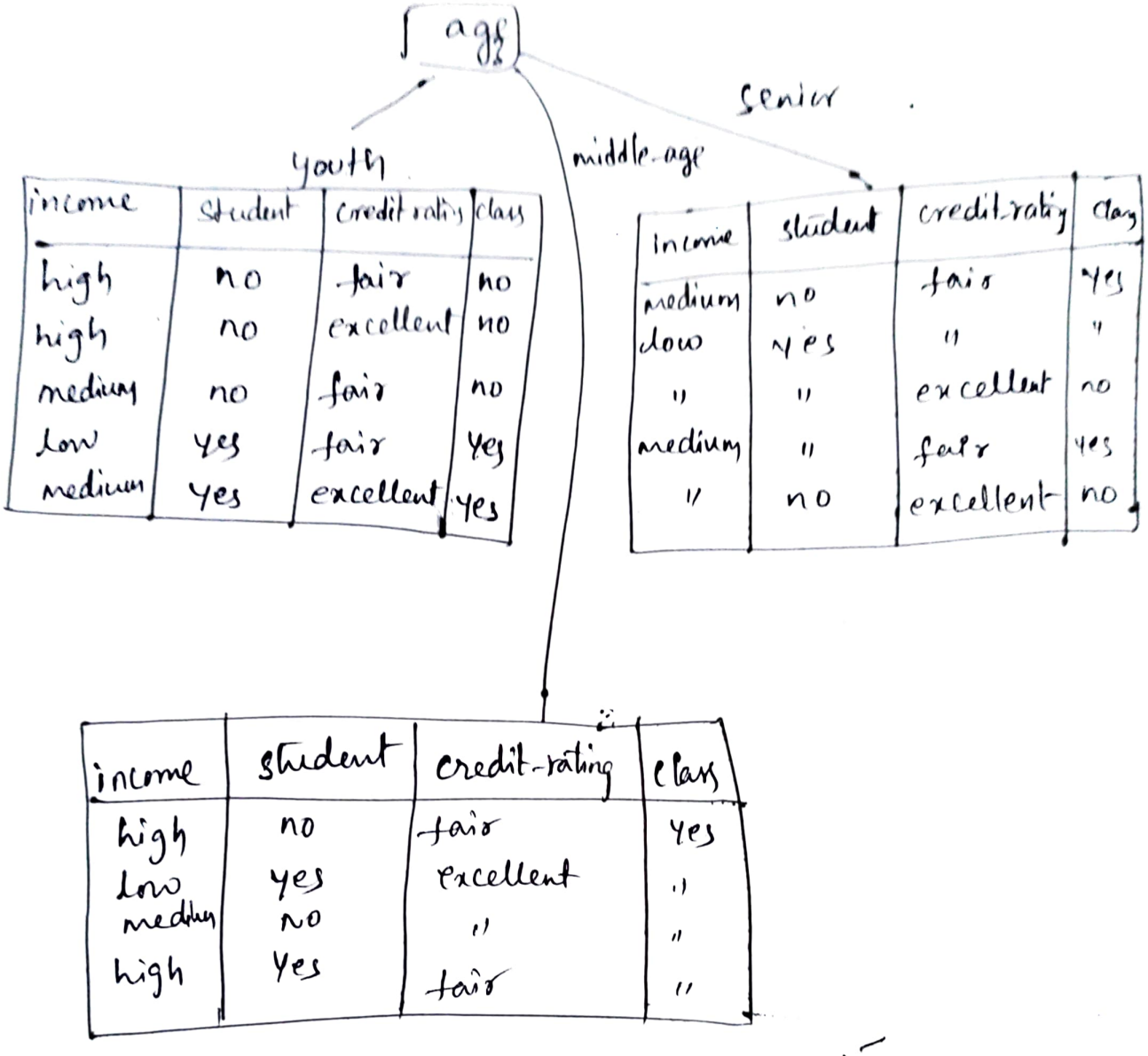
$$= 0.940 - 0.694 = 0.246 \text{ bits}$$

similarly, we compute Gain (income) = 0.029 bits

Gain (student) = 0.151 bits

Gain (credit_rating) = 0.048 bits

Because age has the highest info gain among the attributes, it is selected as the splitting attribute.



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
medium	no	fair	yes
low	yes	"	"
"	"	excellent	no
medium	"	fair	yes
"	no	excellent	no

income	student	credit-rating	class
high	no	fair	yes
low	yes	excellent	"
medium	no	"	"
high	yes	fair	"

entropy. $S = [9+, 5-] = 0.94$ — impurity

$S_{youth} = [2+, 2-] = 1$

$S_{middle} = [4+, 2-] = 0.91$

$S_{senior} = [4+, 0] = 0$ — pure

purity

$\frac{2+}{-}, \frac{2-}{-}$

Gain ratio

→ The info gain measure is biased toward tests with many outcomes.

→ That is, it prefers to select attributes having a large no. of values

For ex: consider an attribute that acts as a unique identifier, such as product_ID. A split on product_ID would result in a large no. of partitions (as many as there are values), each one containing just one tuple.

→ Because each partition is pure, the info required to classify data set D based on this partitioning would be $\boxed{Info_{Product\_ID}(D) = 0}$

Therefore, the info gained by partitioning on this attribute is maximal.

C4.5, a successor of ID3, uses an extension to info gain known as gain ratio, which attempts to overcome this bias. It applies a kind of normalization to info gain using a "Split information" value defined analogously with Info(D) as

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

→ This value represents the potential info generated by splitting the training data set, D, into v partitions, corresponding to the v outcomes of a test on attribute A.

Note that, for each outcome, it considers the no. of tuples having that outcome with respect to the total no. of tuples in D.

→ It differs from info gain, which measures the info w.r.t. classification that is acquired based on the same partitioning.

The gain ratio is defined as

$$\text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}(A)}$$

→ The attribute with the maximum gain ratio is selected as the splitting attribute.

Ex: In above example, A test on income splits the data into three partitions, namely low, medium & high containing four, six and four tuples, resp.

→ To compute the gain ratio of income, we first use eqn (1) to obtain

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

$$= 0.926.$$

we have $Gain_A(income) = 0.029$

$\therefore$ Gain Ratio (income) $= 0.029 / 0.926 = 0.031.$

Gini Index

→ The Gini Index is used in CART. Using the Gini Index measures the impurity of D, a data partition or set of training tuples, as

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

where p_i is the probability that a tuple in D belongs to class C_i and is estimated by $|C_i, D| / |D|$. The sum is computed over m classes

Classification & Regression Trees

When considering a binary split, we compute a weighted sum of the impurity of each resulting partition

Ex: if a binary split on A partitions D into D_1 the gini index of D given that partitioning is

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \quad \text{— (1)}$$

$$\Delta Gini(A) = Gini(D) - Gini_A(D) \quad \text{— (2)}$$

Ex: let D be the training data of above table, where there are nine tuples belonging to the class buys-computer = yes and the remaining five tuples belong to the class buys-computer = no. A (root) node N is created for the tuples in D. We use eqn (1) for GINI index to compute the impurity of D:

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

To find the splitting criteria for the tuples in D, we need to compute the gini index for each attribute.

$$Gini_{income \in \{low, medium\}}(D)$$

value = {low, medium, high}

Subsets = $2^3 = 8$ = {l,m,h} {l,m} {l,h} {m,h} {low} {m} {h} {}

$$= \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2)$$

low, medium = yes → 7 tuples; high = no → 3 tuples

$$= \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right)$$

yes: 1, no: 3

$$= 0.450$$

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

Select the best attribute which gives the minimum gini index overall.

Bayesian classification

Bayesian classifiers are statistical classifiers. They can predict class membership probabilities, such as the prob that a given tuple belongs to a particular class.

Bayesian — to be comparable in performance with decision tree and selected neural n/w classifier.
also exhibited high accuracy & speed when applied to large db's.

Naive Bayesian classifiers — independent of the values of the other attributes. This assumption is called class conditional independence.

Bayes theorem

Let 'x' be a data tuple.
X is considered "evidence.
Let H be some hypothesis, such as that the data tuple X belongs to a specified class C.
P(H/X) is the posterior prob of H conditioned on X.
X is 35-yr old customer with an income of $40,000.
Suppose that H is the hypothesis that our customer will buy a computer.

$$P(H/X) = \frac{P(X/H)\,P(H)}{P(X)}$$

Naive Bayesian classification or simple Bayesian classifier

(1) Let D be a training set of tuples and their associated class labels. $X = (x_1, x_2, \ldots x_n)$ $A_1, A_2 \ldots A_n$ n attributes
(2) m classes $C_1, C_2 \ldots C_m$, Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X.
$P(C_i/X) > P(C_j/X)$ for $1 \le j \le m, j \ne i$
Thus we maximize $P(C_i/X)$ is called maximum posteriori hypothesis.

$$P(C_i/X) = \frac{P(X/C_i)\,P(C_i)}{P(X)}$$

(3) As $P(X)$ is constant for all classes, only $P(X/C_i)P(C_i)$ need be maximized.

(4) In order to reduce computation in evaluating $P(X/C_i)$ the naive assumption of class conditional independence is made.

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

$$= P(x_1/C_i) \times P(x_2/C_i) \times \ldots \times P(x_n/C_i)$$

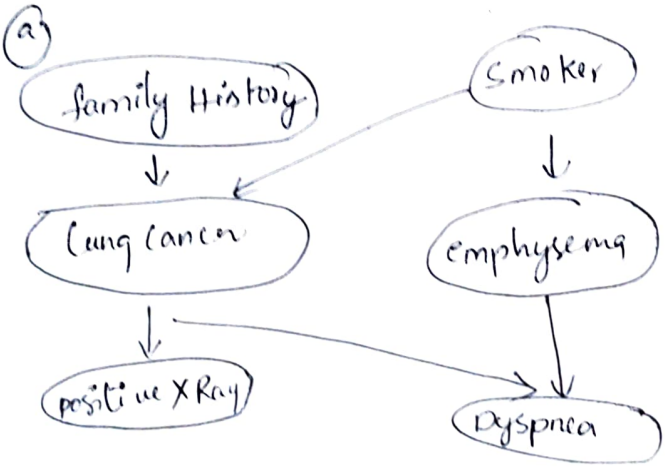
(5) In order to predict the class label of X, $P(X/C_i)\,P(C_i)$ is evaluated for each class C_i.

$$P(X/C_i)P(C_i) > P(X/C_j)\,P(C_j) \text{ for } 1 \le j \le m,\ j \neq i$$

(3) Bayesian Belief n/wks :- is defined by 2 components

(1) a directed acyclic graph
(2) a set of conditional probability table.

(a)



(b)

	FH, S	FH, ~S	~FH, S	~FH
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

fig: the conditional probability table for the values of the variables LungCancer (LC) showing each possible combination of the values of its parent nodes, FamilyHistory (FH) & Smoker (S).

(2)

fig: A proposed causal model, represented by a directed acyclic graph.

Example : X = (age = youth, income = medium, student = yes, credit-rating = fair), predict the class label using naive bayesian classification

RID	age	income	student	credit-rating	class: Buys_computer
1	Youth	high	no	fair	no
2	"	"	"	excellent	no
3	middle-aged	"	"	fair	yes — (1)
4	senior	medium	"	"	" — (2)
5	"	low	yes	"	" — (3)
6	"	"	"	Excellent	no
7	middle-aged	"	"	"	yes — (4)
8	Youth	medium	no	fair	no
9	"	low	yes	"	yes — (5)
10	Senior	medium	"	"	" — (6)
11	Youth	"	"	Excellent	" — (7)
12	middle-aged	"	no	"	" — (8)
13	"	high	yes	fair	" — (9)
14	Senior	medium	no	Excellent	no

Prior probabilities

$P(yes) = 9/14 = 0.643$

$P(no) = 5/14 = 0.357$

Probability of likelihood

$P(Youth/yes) = 2/9 = 0.222$

$P(Youth/no) = 3/5 = 0.600$

P(credit_rating = fair | yes) = 6/9 = 0.667

P(credit_rating = fair | no) = 2/5 = 0.400

P(income = medium | yes) = 4/9 = 0.444

P(income = medium | no) = 2/5 = 0.600

P(student = yes | yes) = 6/9 = 0.667

P(student = no | no) = 1/5 = 0.200

X => (age = youth, income = medium, student = yes, credit_rating = fair)

P(yes / youth, medium, yes, fair)

P = [P(youth | yes) * P(medium | yes) * P(yes | yes) * P(fair | yes)] * P(yes)

= (0.222 * 0.444 * 0.667 * 0.667) * 0.643

= 0.028.

P(no | youth, medium, yes, fair)

= [P(youth | no) * P(medium | no) * P(yes | no) * P(fair | no)] * P(no)

= (0.600 * 0.400 * 0.200 * 0.400) * 0.357

= 0.007.

∴ The naive bayesian classifier predicts buys_computer = yes for tuple X.

Bayesian Belief Networks

→ The naive Bayesian classifier makes the assumption of class conditional independence, that is, given the class label of a tuple, the values of the attributes are assumed to be conditionally independent of one another.

→ This simplifies computation.

→ Bayesian belief n/w's specify joint conditional probability distributions. They allow class conditional independencies to be defined b/w subsets of variables.

→ They provide a graphical model of causal relationships, on which learning can be performed.

→ Bayesian belief n/w's are also known as belief n/w's, Bayesian n/w's and probabilistic n/w's.

→ A belief n/w is defined by two components
  (1) A directed acyclic graph.
  (2) A set of conditional probability tables.

→ Each node in the directed acyclic graph represents a random variable.

→ The variables may be discrete or continous-valued.

→ They may correspond to actual ~~relation~~ attributes given in the data or to "hidden variables" believed to form a relationship

Ex: in the case of medical data, a hidden variable may indicate a syndrome, representing a no. of symptoms that, together, characterize a specific disease).

→ In fig allow a representation of causal knowledge.

Ex: having lung cancer is influenced by a person's family history of lung cancer, as well as whether or not the person is a smoker.

Note: that the variable Positive XRay is independent of whether the patient has a family history of lung cancer or is a smoker, given that we know the patient has lungcancer.

→ In otherwoods, once we know the outcome of the variable LungCancer, then the var's FamilyHistory & Smoker do not provide any additional info regarding Positive XRay.

→ The arcs also show that the var LungCancer is conditionally independent of Emphysema, given its parents, familyHistory & Smoker.

→ A belief nlw has one conditional probability table (CPT) for each var. The CPT for a var Y specifies the conditional distribution P(Y/parents(Y)), where parents(Y) are the parents of Y.

→ Fig (b) shows a CPT for the variable LungCancer.

→ The conditional probability for each known value of LungCancer is given for each possible combination of values of its parents.

→ For instance, from the upper leftmost and bottom rightmost entries, resp, we see that-

$$P(\text{LungCancer} = \text{yes} \mid \text{FamilyHistory} = \text{Yes}, \text{Smoker} = \text{Yes}) = 0.8$$

$$P(\text{LungCancer} = \text{no} \mid \text{familyHistory} = \text{no}, \text{smoker} = \text{no}) = 0.9$$

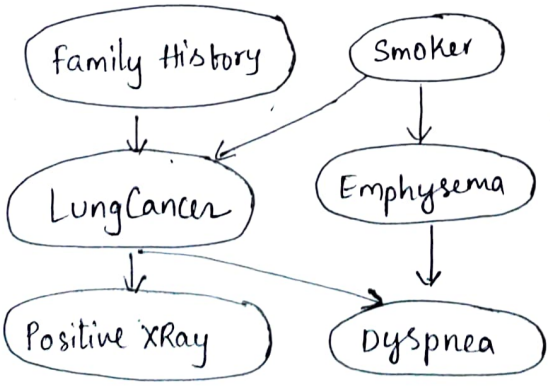


fig: A proposed causal model, represented by a directed acyclic graph.

	FH, S	FH, ~S	~FH, S	~FH, ~S
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

fig: The conditional probability table for the values of the var LungCancer (LC) showing each possible combination of the values of its parent nodes, FamilyHistory (FH) and Smoker (S).