Classification

classification is the process of finding a model that predicts the class label like categorical values.

Example :- (1) A bank loan officer needs analysis of her data to learn which loan applicants are safe (or) risky

(2) A marketing manager at All-Electronics needs data analysis to help whether customers with a given profile will buy a new computer. class labels like Yes (or) No.

(3) A medical researcher wants to analyze breast cancer data to predict which one of three specific treatments a patient should receive. class labels like Treatment A Treatment B, Treatment C for medical data.

In Each of these Example Data Analysis task is classification, where a model (or) classifier is constructed to predict class label.

Suppose that the marketing manager wants to predict the how much of a given customer will spend during a sale at All-Electronics. The data Analysis task is Numeric prediction. this is called Regression Analysis

classification & Regression is two Popular terms for Prediction.

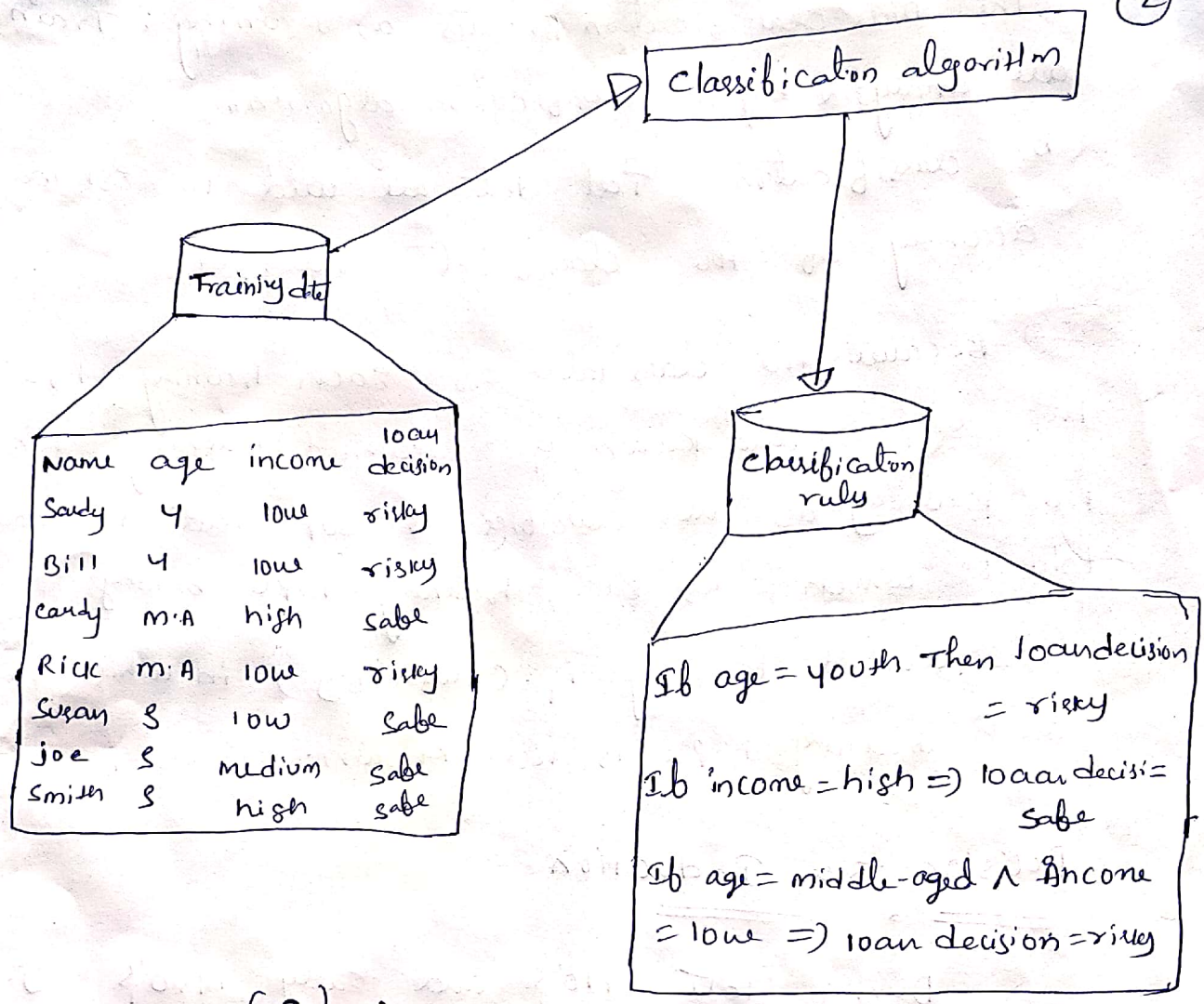
General Approach to classification

Data classification is two-step process

① learning step :- where a classification model is constructed

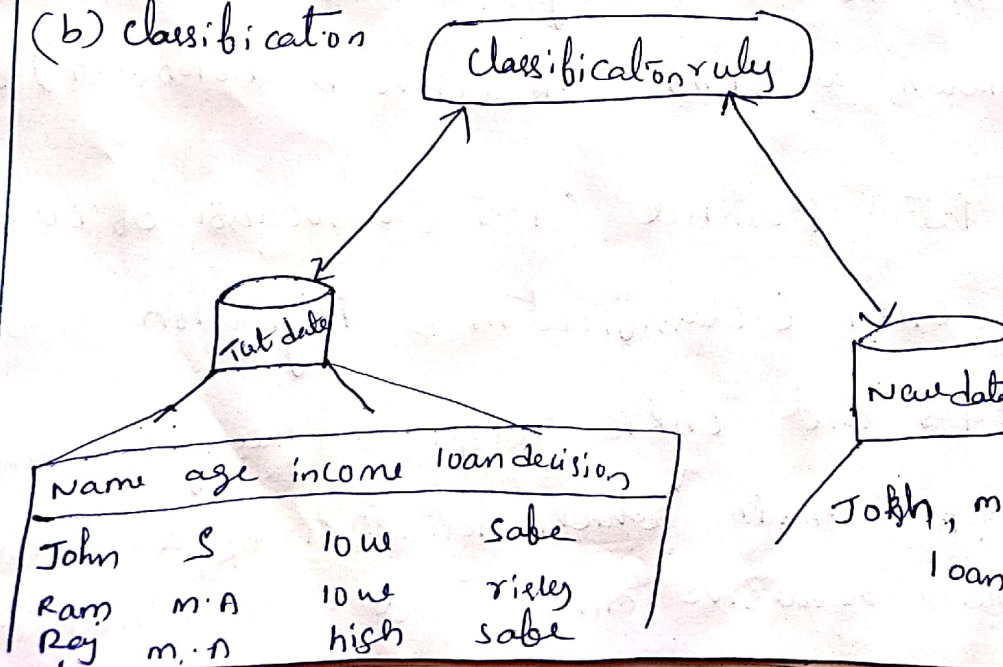
In first step, a classifier is built describing a predetermined set of data classes (or) concepts. This is learning step (Training phase), where a classification algorithm builds classifier by analyzing (or) learning from a training set made up of database tuples & their class labels

② classification step. Test data are used to Estimate the accuracy of the classification rules. If accuracy is considered acceptable, the rules can be applied to the classification of new data tuples.



(a) Learning step

(b) classification



The Data classification process (a) learning : Training data are analyzed by a classification algorithm

(b) classification : Test data are used to estimate the accuracy of the classification rules.

→ Because the class label of each training tuple is provided this step is also known as supervised learning that is the learning of the classifier is supervised in that it is told to which class each training tuple belongs

unsupervised learning is (Clustering) in which class label of each training tuple is not known.

Decision Tree Induction

→ During late 1970's and early 1980's J. Ross Quinlan, researches in machine learning, developed a decision Tree algorithm known as ID3 (Iterative Dichotomiser).

→ Quinlan later presented C4.5 (a successor of ID3) and then CART (classification & Regression Trees.

ID3, C4.5 and CART adopt a greedy (ie Non back Tracking) approach in which decision Trees are constructed in a topdown recursive divide and conquer manner.

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 denotes a test on an attribute, branch represents an outcome of the test, and each leaf node holds a class label. The top most node in a tree is the root node.

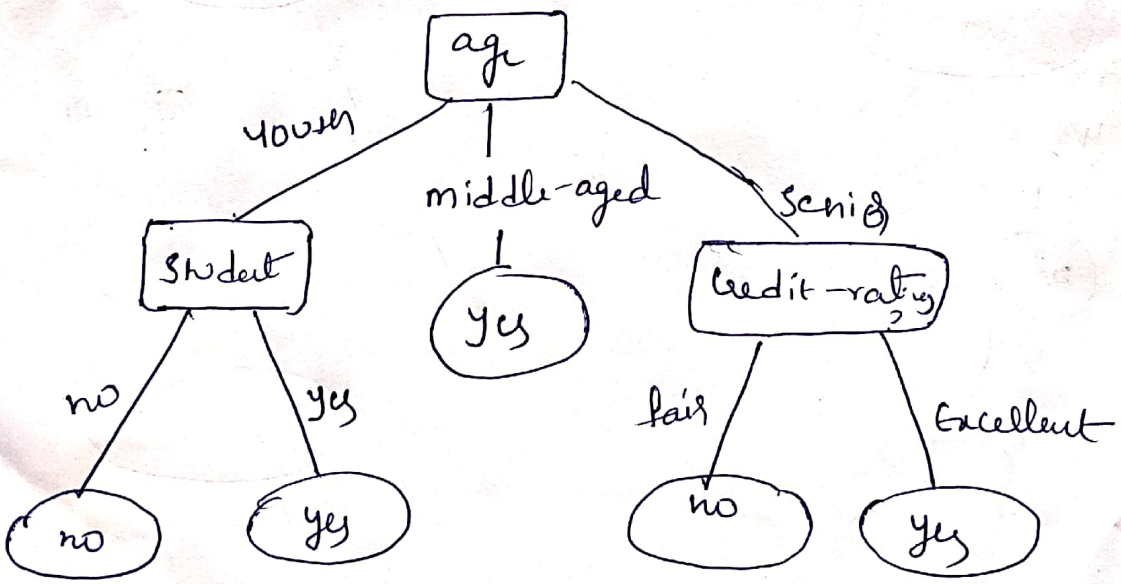


fig Decision Tree for the concept buys-comput

Selection measures

most decision Tree algorithms follow a top-down approach, which starts with a Training set of tuples and their associated class labels.

Algorithm: Generate decision-Tree. Generate a decision Tree from the training tuples of data Partition, D.

Input:

- ✓ Data Partition, D, which is 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.

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;

(6) apply Attribute_selection_method (D, attribute_list) to find the "best" splitting criterion

(7) labeled node N with splitting_criterion;

(8) if splitting_attribute is discrete_valued and multiway splits allowed then

(9) attribute_list ← attribute list - splitting attribute;

(10) for each outcome j of splitting_criterion

(11) let Dj be the set of data tuples in D satisfying outcome j;

(12) if Dj 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 to node N;

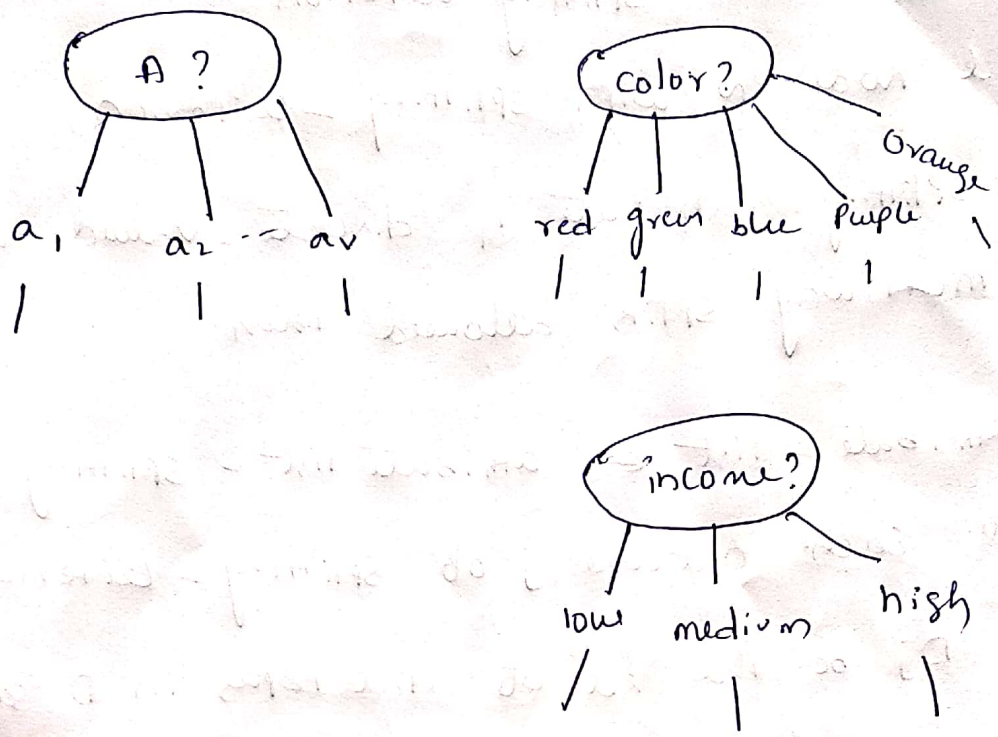
End for

(15) return N;

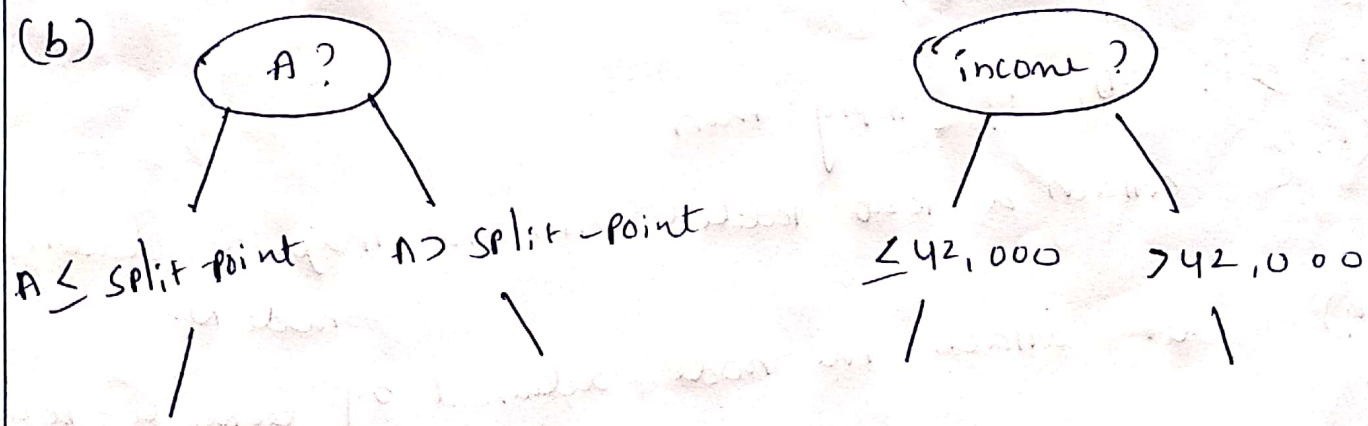
Splitting criterion

Examples

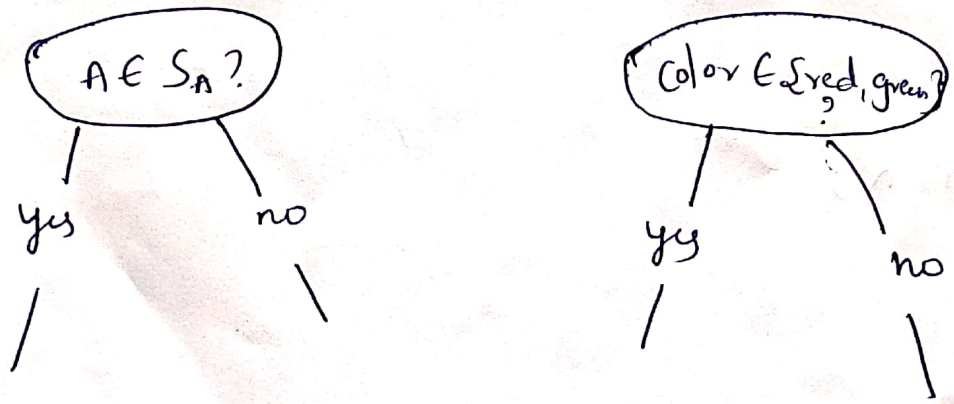
(a)



(b)



(c)



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. Conceptually, the "best" splitting criterion is the one that most closely results in such a scenario.

→ Attribute selection measures are also known as splitting rules because they determine how the tuples at a given node are to be split.

→ The attribute selection measure provides a ranking for each attribute describing the given training tuples

→ The attribute having the best score for the measure is chosen as the splitting attribute for the given tuples.

→ Attribute selection measures describes 3 types

1. Information gain
2. Gain ratio
3. Gini index.

Q1

Information Gain:- ID3 uses Information gain as its attribute selection measure

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

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

The term $\frac{|D_j|}{|D|}$ acts as the weight of the j^{th} partition.

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

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

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

Example

RID	age	income	Student	credit-rating	class: buys-comp
1	Youth	high	no	fair	no
2	y	h	no	Excellent	yes
3	middle-aged	h	no	fair	yes
4	Senior	medium	no	fair	yes
5	s				
6	s	low	yes	fair	no
7	m	l	yes	Excellent	yes
8	y	l	yes	e	
9	y	medium	no	f	no
10	s	low	yes	f	yes
11	y	medium	yes	f	yes
12	m	m	yes	e	yes
13	m	m	no	e	yes
14	s	high	yes	f	yes
		medium	no	e	no

Q4 distinct classes (i.e; m=2) {yes, no}

$$\text{Info}(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits}$$

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

$$\frac{4}{14} \times \left[-\frac{4}{4}\log_2\frac{4}{4} - 0\right] + \frac{5}{14} \times \left[-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right]$$

$$= 0.694 \text{ bits}$$

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

$$\text{Gain(age)} = \text{Info}(D) - \text{Info}_{age}(D) = 0.940 - 0.694$$

$$= 0.264 \text{ bits.}$$

Similarly Gain(income) = 0.029 bits, Gain(credit-rating) = 0.048 bits

Gain(student) = 0.151 bits.

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

Node N is labled with age and branches are grown for each of the attribute's values.

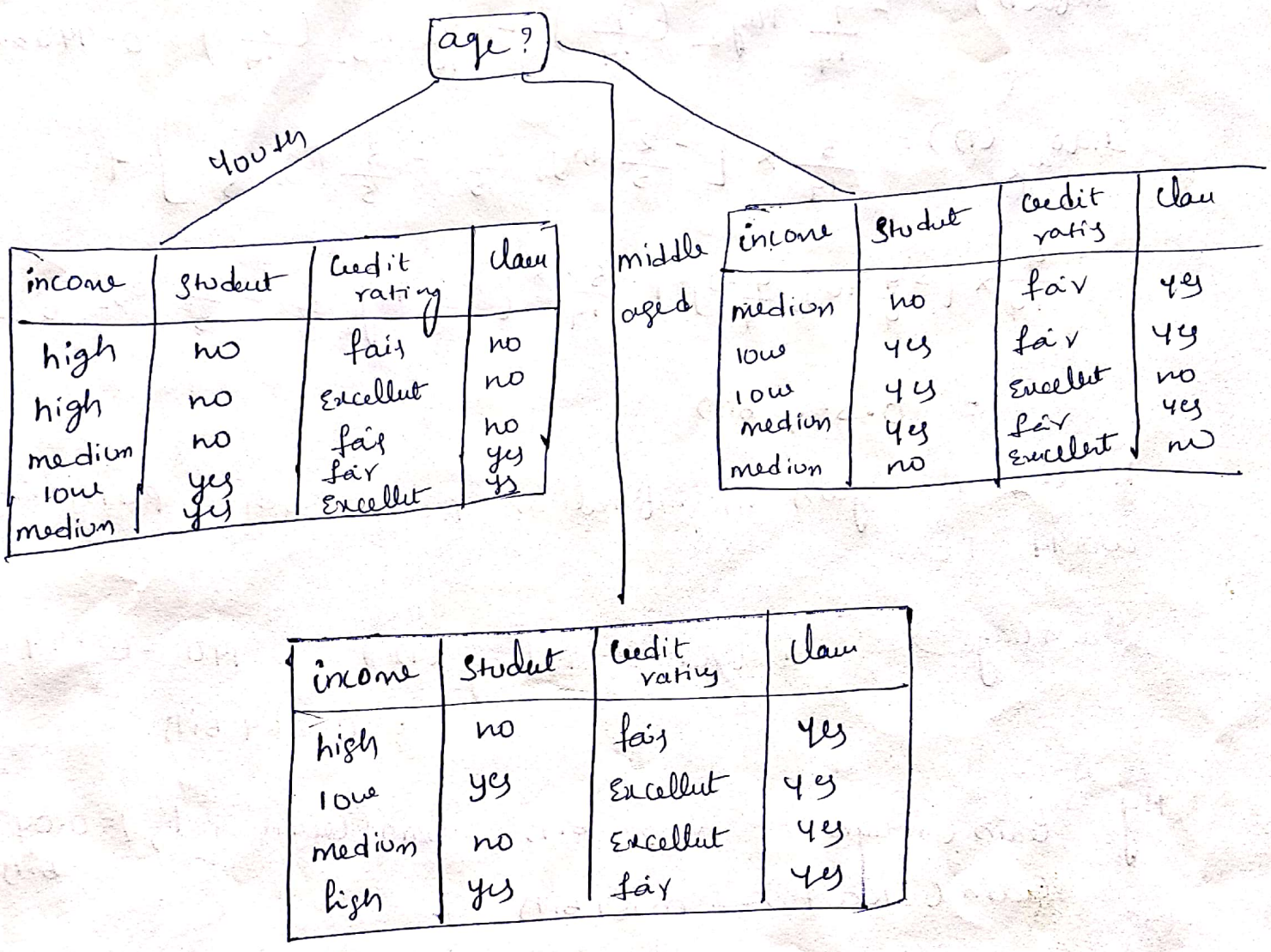


fig: decision Tree.

Other Attribute Selection measures

① MDL (minimum Description length) principle have the least bias toward multivalued attributes.

→ MDL - based measure use encoding techniques to define the "best" decision tree as the one that requires the fewest number of bits to both

(1) encode the Tree

(2) encode the Exceptions to the tree

② multivariate splits : where the partitioning of tuples is based on a combination of attributes rather than a single attribute.

→ CART system, for example, can find multivariate splits based on a linear combination of Attributes

→ multivariate splits are a form of attribute construction (feature) where new attributes are added to the existing ones.

Gain Ratio :- The information gain measure is biased tests with many outcomes.

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

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

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

Computation of gain ratio for the attribute income

A test on income splits the data of Table (Previous te) into three partitions, namely low, medium & high ; containing four, six and four tuples. To compute the gain ratio of income

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

$$= 1.557$$

we have $Gain(income) = 0.029$ $\therefore$ $GainRatio(income) = \frac{0.029}{1.557} = 0.019$

Gini Index

The Gini index is used in CART. using the notation described above. the Gini index measures the impurity of D, a data partition (or) set of Training tuples as

$$Gini(D) = 1 - \sum_{i=1}^{m} P_i^2$$ (It is measure of the impurity (inequality) of D

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 Gini index considers a binary split for each attribute. Lets first consider the case where A is a discrete-valued attribute having v distinct values $\{a_1, a_2 - a_v\}$, occuring in D. TO determine binary split on A, we examine all of possible subsets that test for attribute A of the form "$A \in S_A$?"

Eg: if income has three possible values $^{(v)}$ {low, medium, high} then possible subsets are $2^v \Rightarrow 2^3 = 28$ {low, medium, high}, {low, medium} {low, high}, {medium, high}, {low} {medium} {high} and {}

$$Gini_A(D) = \frac{D_1}{D} Gini(D_1) + \frac{|D_2|}{D} Gini(D_2)$$

If a binary split on income, Partitions D in to D_1 & D_2 the gini index of D given that Partitioning is

For continous-valued attributes

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

Example Let D be the Training data, where there are 9 tuples belonging to the class buys-computer = yes and the remaining five tuples belong to buys-computer = NO.

Gini index

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

Consider subset {low, medium}. This would result in 10 tuples in partition D_1 satisfying the condition "income $\in$ {low, medium}". The remaining four tuples of D would be assigned to Partition D_2.

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

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

$$= 0.450$$

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

My Gini index values for splits on the remaining subsets are 0.315 for subsets {low, high} & {medium} and 0.300 for

the subsets {medium, high} and low. Therefore the best binary split for attribute income is on {medium, high} & {low} because it minimizes Gini index.

Evaluating the attribute, we obtain {youth, senior} & {middle-aged} as the best split for age with Gini index of 0.375; the attributes {student} and {credit-rating} are both binary, with Gini index values of 0.367 & 0.429.

Find reduction in impurity using.

income $Gini(A) = Gini(D) - Gini_A(D)$

$= 0.459 - 0.300 = 0.159$

Step1: compute the impurity of D

age $Gini(age) = 0.459 - 0.315 = .0084$

$Gini_{(student)}$ Binary $= 0.459 - 0.367 = 0.92$

$Gini_{(credit\text{-}rating)}$ Binary $= 0.459 - 0.429 = 0.03$

Income is selected with minimum gini index & highest reduction in impurity.

Four Steps:- of Gini index

(1) Find the impurity of D_1 using formula 1

(2) Find the impurity of each resulting partition using

$$Gini_A(D) = \frac{|D_1|}{D} Gini(D_1) + \frac{|D_2|}{D} Gini(D_2)$$

(3) Find reduction in impurity using formula

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

(4) Now select the best attribute which gives the minimum gini index

Bayes classification methods

→ Bayesian classification (or) Bayesian classifiers are statistical classifiers. They can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.

→ Baysian classification is based on Baye's theorem, Baysian classifier is also known as Navie Bayesian classifier

→ In Baysian terms, X is considered "evidence". As usual, it is described by measurements made on a set of n attributes.

→ Let H be some hypothesis such as that the data tuple X belongs to a specified class C. For classification problems we want determine $P(H|X)$

$P(H|X)$ is the posterior probability. Suppose our world of data tuples is confined to customers described by the attributes age & income respectively and that X is 35 years old customer with an income of $40,000

H is the hypothesis that our customer will buy a 'comp' Given age & income.

P(H) is the prior probability. this is the probability that any given customer will buy a computer, regardless of age & income, (or) any other information.

Ny P(X|H) is the posterior probability of X conditioned on H. i.e probability that a customer X, is 35 years & Earns 40,000 given we know the customer will buy a computer.

(12)

Baysian classification

Baysian classification is based on Baye's theorem. Baysian classification is also known as naive Bayesian classifier

Bayes Theorem:

$P(H|x)$ is the posterior probability (or) a posteriori probability, of H conditioned on X.

$$P(H|x) = \frac{P(x|H)\,P(H)}{P(x)}$$

$$P(C_i|x) = \frac{P(x|C_i)\,P(C_i)}{P(x)}$$

$P(x|C_i)\,P(C_i) > P(x|C_j)\,P(C_j)$ for $1 \le j \le m$, $j \neq i$

$X = (age = youth,\ income = medium,\ Student = yes,\ Creditrating = fair)$

$P(x|C_i)\,P(C_i)$ for $i = 1, 2$, $P(C_i)$ the Prior probability of Each class.

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

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

To compute P(x|Ci) for i = 1, 2

P(age = youth | buys_comp = yes) = 2/9 = 0.222

P(age = youth | buys_comp = no) = 3/5 = 0.600

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

P(income = medium | buys_comp = no) = 2/5 = 0.400

P(Student = yes | buys_comp = no) = 1/5 = 0.200

P(Creditrating = fair | buys_comp = yes) = 6/9 = 0.667

P(Creditrating = fair | buys_comp = no) = 2/5 = 0.400

P(X | buys_comp = yes) = P(age = youth | buys_comp = yes) x

P(income = medium | buys_comp = yes) x

P(Student = yes | buys_comp = yes) x

P(Creditrating = fair | buys_comp = yes)

= 0.222 x 0.444 x 0.667 x 0.667 = 0.044.

lly

$P(x \mid buys\text{-}comp = no) = 0.660 \times 0.400 \times 0.200 \times 0.400 = 0.019$.

$P(x \mid buys\text{-}comp = yes)\ P(buys\text{-}comp = yes) = 0.044 \times 0.643$

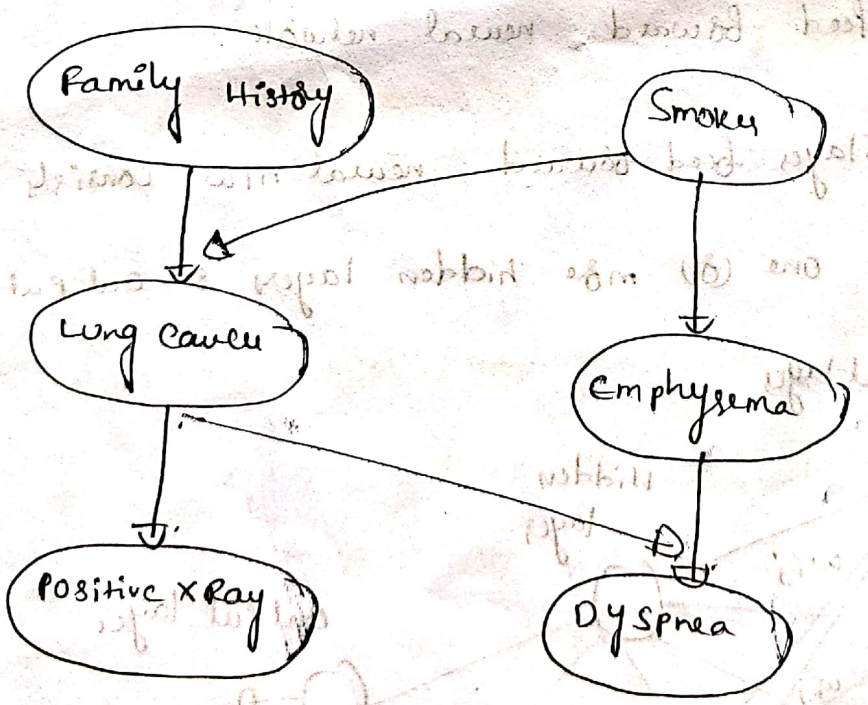
$= 0.028$

$P(x \mid buys\text{-}comp = no)\ P(buys\text{-}comp = no) = 0.019 \times 0.357$

$= 0.007$

∴ navie Bayesian classifier predicts buys-comp = yes to tuple x.

A Simple Bayesian belief network.



Classification By Backpropagation

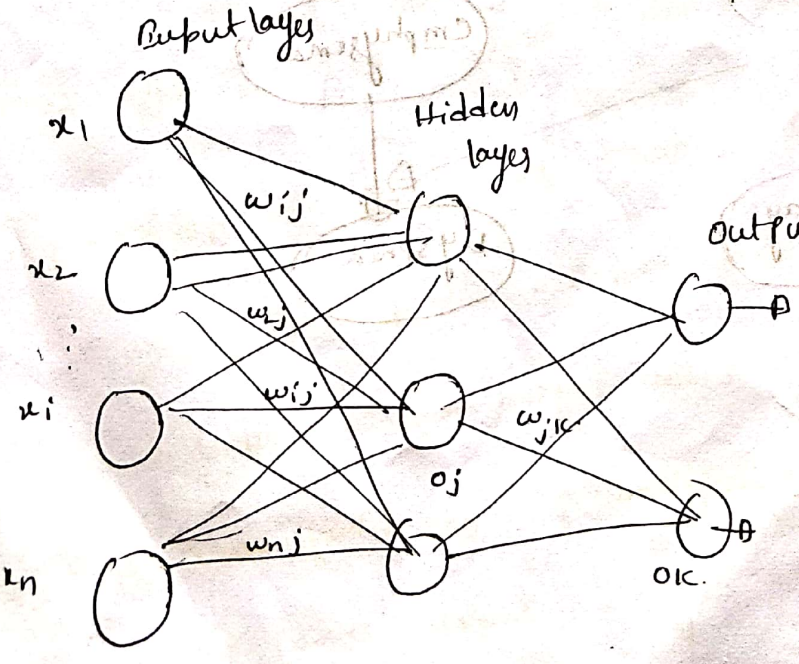
Backpropagation is a neural network learning algorithm.

"Neural Network is a set of connected input/output units in which each connection has weight associated with it".

A multilayer feed-forward Neural Network

* the backpropagation algorithm performs learning on a multilayer feed-forward neural network.

* A multilayer feed forward neural n/w consists of an input layer, one (or) more hidden layers & output layer.



Tree Pruning

→ when a decision Tree is built, many of the branches will reflect anomalies in the training data due to noise (or) outliers.

→ Tree Pruning methods address this problem of overfitting the data.

→ unpruned Tree tend to larger & complex but Pruned trees tend to smaller and less complex Easier to comprehend.

There are two common approaches to Tree Pruning

1. Pre Pruning
2. Post Pruning

Prepruning :- → A Tree is "Pruned" by halting its construction Early (Ex: by deciding not to further split (or) partition the subset of training tuples at a given node) upon halting, the node becomes a leaf. The leaf may hold the most frequent class among the subset tuples

→ when constructing a tree measures such as Information gain, Gini index etc. can be used to assess the goodness of split.

If Partitioning the tuples at a node would result in a split that falls below a prespecified threshold, then further partitioning of the given subset is halted.

→ High thresholds could result in oversimplified trees, while low thresholds could result in very little simplification.

Post pruning: which removes subtrees from a "fully grown" tree. A subtree at a given node is pruned by removing its branches & replacing with its leaf.

→ The leaf is labeled with the most frequent class among the subtree being replaced. For example A_3 in the unpruned tree of below figure.

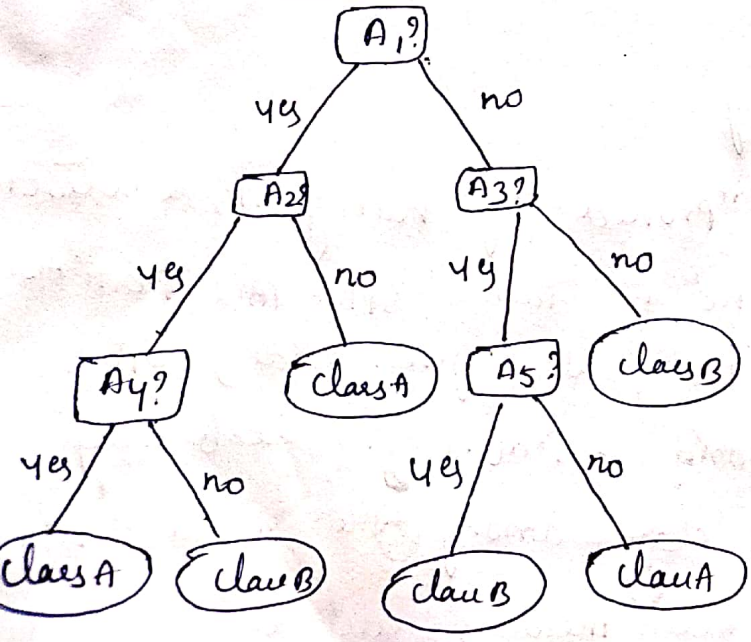


fig (a) unpruned decision tree

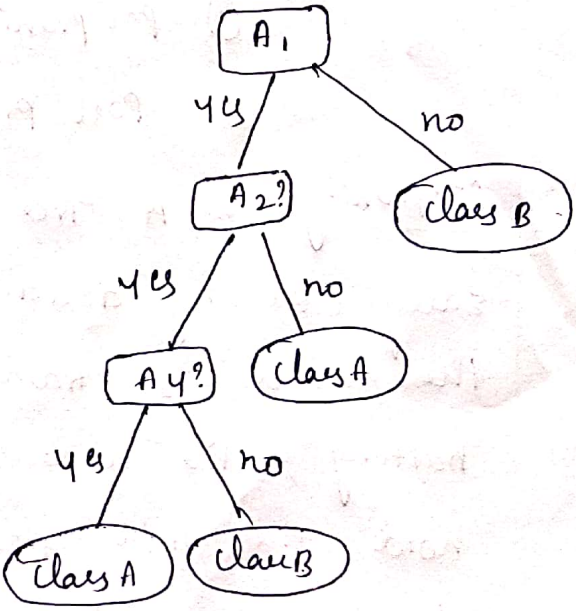
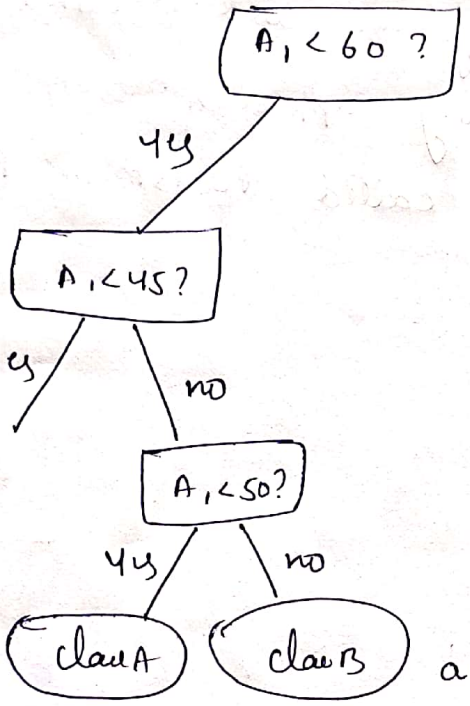
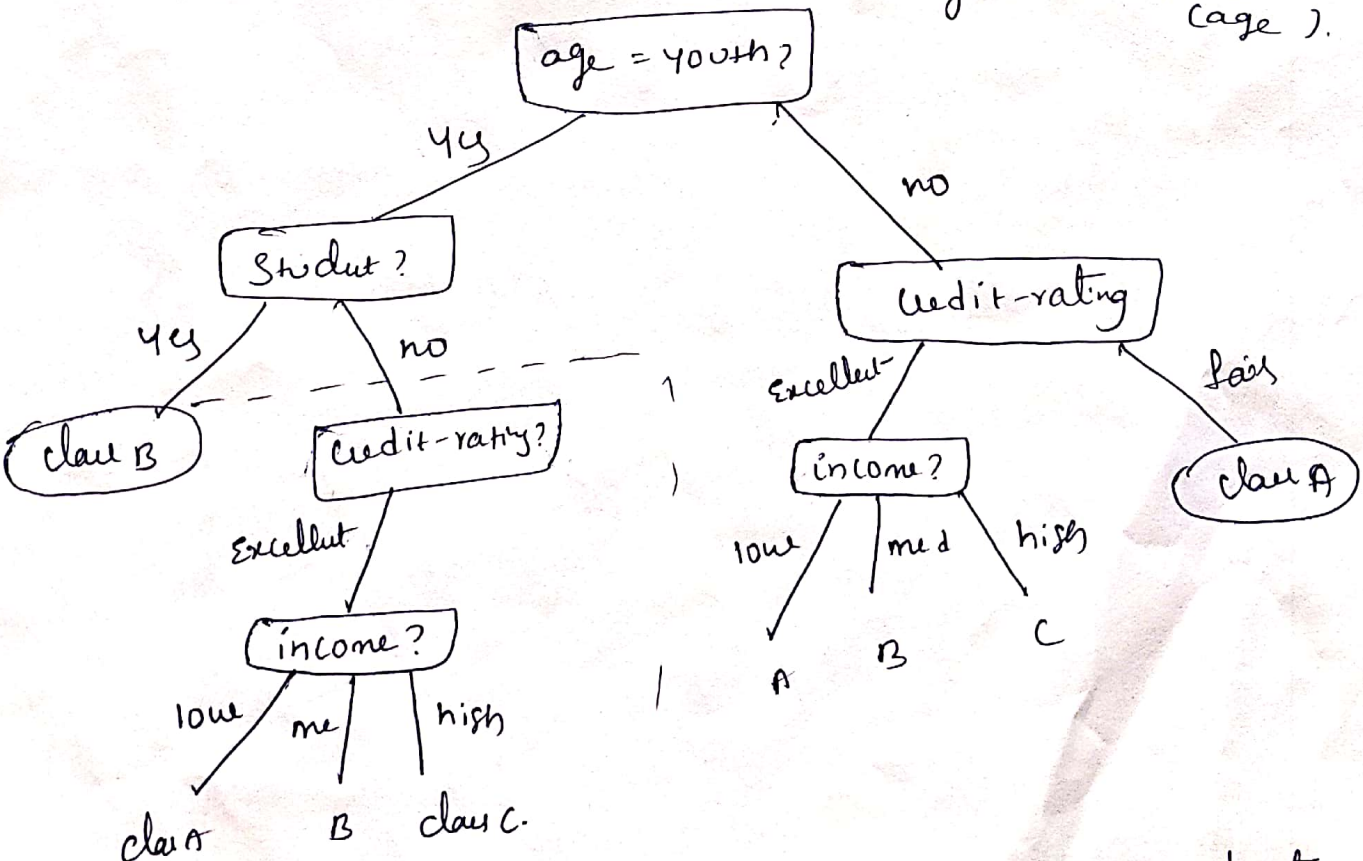


fig (b): Pruned decision tree



(a) Subtree repetition, where an attribute is repeatedly tested along a given branch of Tree (age).



(b) Subtree replication, where duplicate Subtree exist within a Tree (C.R)

→ Cost complexity pruning algorithm used in CART is an example of the post pruning approach.

→ C4.5 uses a method called Pessimistic Pruning.

Alternatively

Scalability and Decision Tree Induction

"How scalable is decision tree induction"? The efficiency of existing decision tree algorithms, such as ID3, C4.5 & CART has been well established for small data sets.

→ Efficiency becomes an issue of concern when these algorithms are applied to the mining of very large real world data bases.

→ In Data mining applications, very large training set of millions of tuples are common. the Training data will not fit in memory. Therefore decision Tree construction becomes inefficient due to swapping of the Training tuples in and out of main & cache memory.

→ Several Scalable decision tree induction methods have been introduced in recent studies. Rainforest, for Example adapts to the amount of main memory available and applies to any decision tree induction algorithm.

→ This method maintains a AVC-set (Attribute value class label) for each attribute

age	buys-computer	
	yes	NO
youth	2	3
middle-aged	4	0
Senio	3	2

income	buy-computer	
	yes	NO
low	3	1
medium	4	2
high	2	2

Student	buys-computer	
	yes	no
yes	6	1
no	3	4

Credit-rating	buys-computer	
	yes	no
fair	6	2
Excellent	3	3

fig: The use of data structures to hold aggregate information regarding the Training data (These AVC-sets describing previous Training data set) are one approach to improving scalability of decision Tree induction.

→ "Boat" (BootStrapped optimistic Algorithm for Tree construction) is a decision Tree Algorithm that takes a completely different Approach to scalability

→ It is not based on the use of any special data structure Instead, it uses a Statistical technique known as "bootstrapping to create several smaller samples of the given training data, each of which fits in memory

→ BOAT can use any attribute selection measure that selects binary splits and that is based on the notation of purity of partitions such as the Gini index.

→ BOAT was found to be two to three times faster than Rain Forest. Advantage of BOAT is that it can be used for incremental updates. That is BOAT can take new insertions and deletions for the training data and update the decision tree to reflect these changes, without having to reconstruct the tree from scratch.

Visual mining for Decision Tree Induction

→ How can visualize the Decision Trees means one of the method like Perception-based classification (PBC).

→ PBC is an interactive approach based on multidimensional visualization technique and allows the user to incorporate background knowledge about the data when building a decision Tree.

→ The Resuting Trees tend to be smaller than that built using Traditional decision Tree induction methods

" How can the data be visualized to support interactive decision tree construction ?

→ PBC uses a pixel-oriented approach to view multidimensional data with its class label information.

→ The circle segments approach is adapted, which maps d-dimensional data objects to a circle that is partioned into d segments, each representing one attribute. Here an attribute value of an object is mapped to one colred pixel. Each data object.

→ PBC System displays a split screen, consisting of Data Interaction window and a Knowledge Interaction window

Data Interaction windows: displays the circle segments of the data under examination

Knowledge Interaction window :- displays the decision tree constructed.

Initially the complete training set is visualized in the Data Interaction window, while the Knowledge Interaction window displays an Empty decision tree

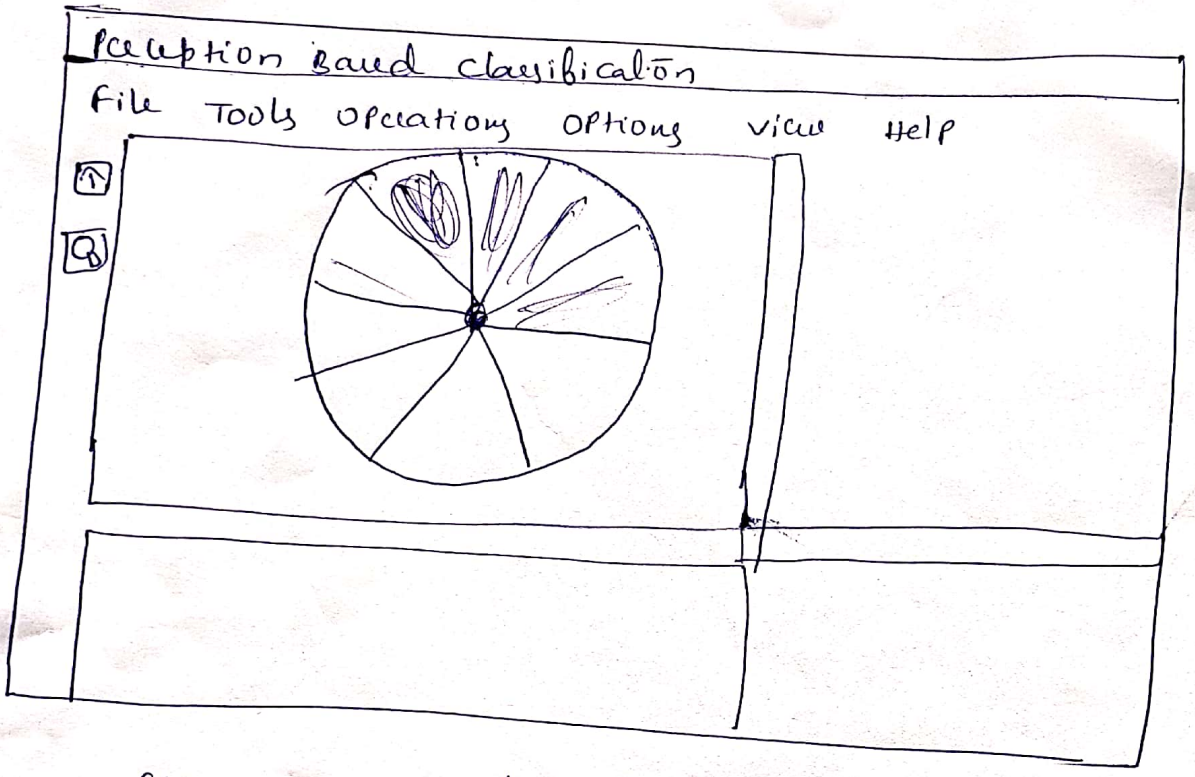


Fig : A Screenshot of PBC. for interactive decision tree construction

Model Evaluation and Selection

model selection: choosing one classifier over another

metrics for evaluating classifier performance

This section presents measures for assessing how good or how "accurate" your classifier is at predicting the class label of tuples.

measure	formula
Accuracy, Recognition rate	$\frac{TP + TN}{P + N}$
Error rate, misclassification rate	$\frac{FP + FN}{P + N}$
Sensitivity, True Positive rate	$\frac{TP}{P}$
Specificity	$\frac{TN}{N}$
Precision	$\frac{TP}{TP + FP}$
F, F_1, F-score harmonic mean of Precision & Recall	$\frac{2 \times Precision \times recall}{Precision + recall}$
F_β, where β is a non negative real number	$\frac{(1+\beta)^2 \times Precision \times recall}{\beta^2 \times Precision + Recall}$

There are the evaluation measures. TP, TN, FP, P, N refers to the number of true positive, true negative, false positive, positive, and negative samples.

There are four additional terms we need to know that are the "building blocks" used in computing many evaluation measures.

True positives (TP):- These refers to the positive tuples that were correctly labeled by the classifier. Let TP be the number of true positives.

True negatives:- (TN) These are the negative tuples that were correctly labeled by the classifier. Let TN be the number of true negatives.

False positives: (FP): These are the negative tuples that were incorrectly labeled as positive. Let FP be the number of false positives.

False negatives: (FN): These are the positive tuples that were mislabeled as negative. Let FN be the number of false negatives.

These terms are summarized in the confusion matrix

- based classification :-

Predicted class

Actual class	yes	no	Total
yes	TP	FN	P
no	FP	TN	N
Total	P'	N'	P+N

confusion matrix, with totals for positive & negative tuples.

Classes	Buys-computer=yes	buys-comp=no	Total	Recognition
buy-com=yes	6954	46	7000	99.34
buys-com=no	412	2588	3000	86.27
Total	7366	2634	10,000	95.42

Fig: confusion matrix for the classes buys-computer=yes & buys computer=NO.

Example: Sensitivity and specificity :- confusion matrix for medical data where the class values are yes and no for a class label attribute cancer.

classes	yes	no	Total	Recognition (%)
	TP	FN		
yes	90	210	300	30.00
no	140	9560	9700	98.56
	FP	TN		

The sensivity of the classifier is $\frac{90}{300} = 30.00\%$.

The specificity is $\frac{9560}{9700} = 98.56$

The classifier overall accuracy $= \frac{9650}{10,000} = 96.50\%$.

The Precision & recall measures are also widely used in classification. Precision can be thought of as a measure of Exactness (i.e what percentage of tuples labeled as positive are actually), recall is a measure of completeness.

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{TP}{P}$$

Rule - Based classification: -

Example

$$\frac{TP}{TP+FP} = \frac{90}{90+140}$$

$$\text{Precision} = \frac{90}{230} = 39.13\%.$$

$$\frac{TP}{P} \qquad \text{Recall} = \frac{90}{300} = 30.00\%.$$

In addition to accuracy - based measures, classifiers can also compared with respect to the following additional aspects.

Speed:

Example

Actual class | Predicted class	cancer = yes	cancer = no	Total
cancer = yes	90	210	300
cancer = no	140	9560	9700
Total	230	9770	10000

Sensitivity = 30.0 Recall = 30.00%.

Specificity = 98.56

Accuracy = 96.40 based on above formulas.

precision = 39.13

Methods for evaluating classifier accuracy

1. Hold out
2. Random sampling
3. Cross validation
4. Bootstrap

Hold out :-→ In this method, the given data are randomly partitioned into two independent sets, like a Training dataset and test set.

→ Two-thirds (2/3) of data are collected to the Training set, and the remaining one-third (1/3) is allocated to the test set.

→ The Training data set is used to derive the model.

→ The model's accuracy is estimated with the test set.

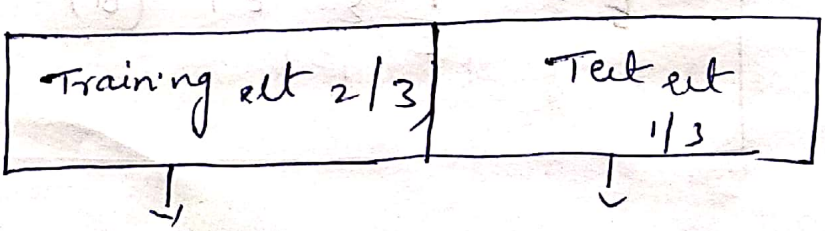
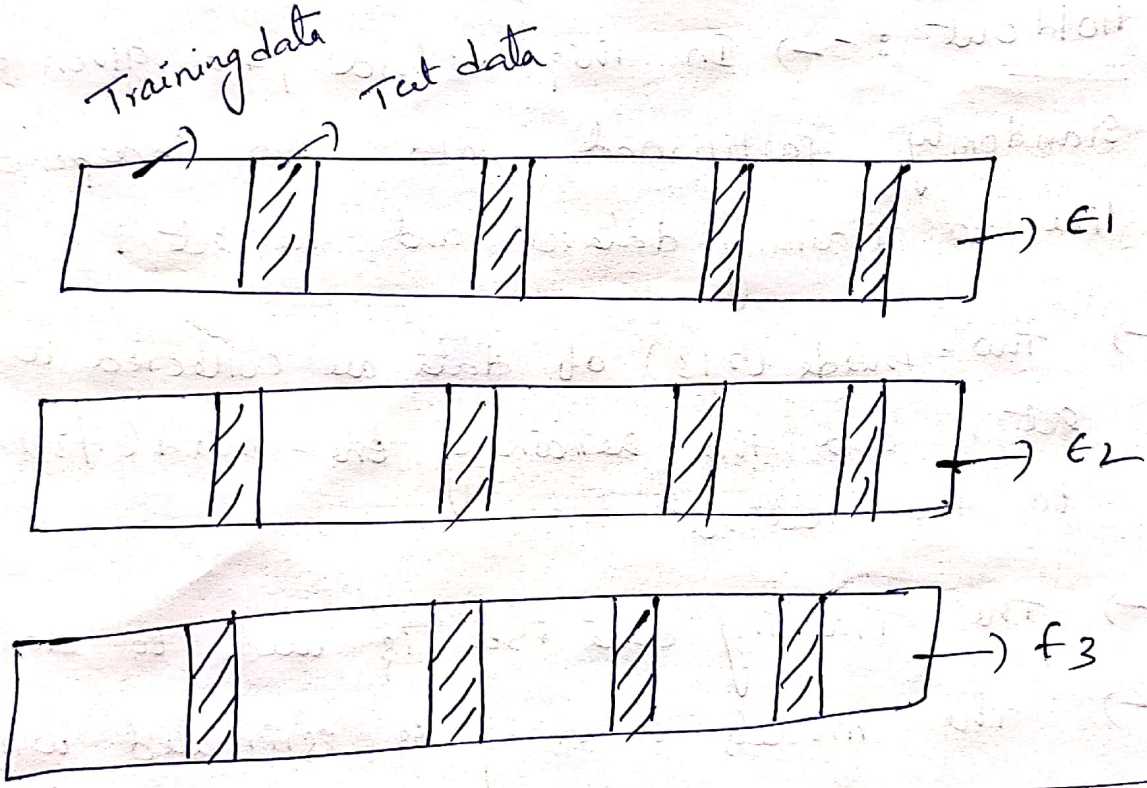


fig: Hold out method.

Random Sampling :- (ii) Repeated Hold out method

→ Random sampling is a variation of the holdout method in which the hold out method is repeated K times.

→ The overall accuracy estimate is taken as the advantage of the accuracies obtained from each iteration.



$$E = \frac{1}{K} \sum_{i=1}^{K} E_i \quad (\text{or}) \quad \frac{E_1 + E_2 + E_3}{3}$$

Cross validation

→ In K-fold cross-validation, the initial data are randomly partitioned into K mutually exclusive (or) folds $D_1, D_2 \cdots D_K$. each of approximately same size.

→ Training and Testing is performed K times In iteration i, partition D_i is reserved as the test set, and the remaining partitions are collectively used to train the model.

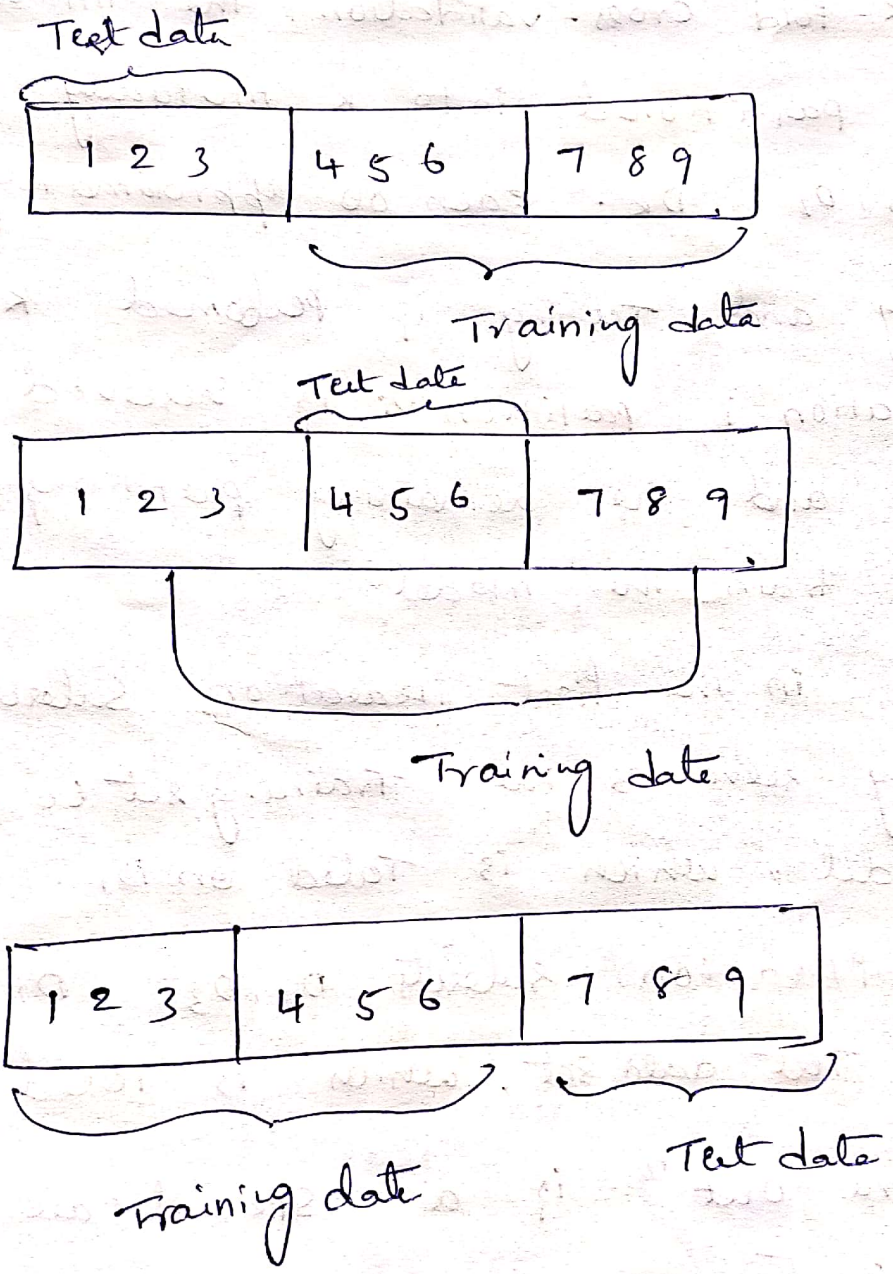
→ That is, in the first iteration, subsets $D_2 D_3 \cdots D_K$ collectively serve as the Training set to obtain a first model, which is Tested on D_1.

→ Second iteration, subsets $D_1, D_3 \cdots D_K$ collectively serve as Test data set, which is Tested on D_2

→ "leave one-out" is a special case of K-fold cross validation where K is set to the number of initial tuples. That is only one sample is "left out" at a time for the test set.

→ "Stratified Cross-validation, the folds are stratified so, that the class distribution of tuples in each fold is approximately the same as in the initial data.

3 - Fold Cross validation



Bootstrap:-

→ works well with small data sets.

→ Samples the given training tuples uniformly with replacement.

→ Each time tuple is selected, it is equally likely to be selected again and re-added to the Training set.

→ Several Bootstrap methods like .632 bootstrap.

A dataset with d tuples is sampled d times, with replacement, resulting in Training set of d samples. The data tuple that did not make it in to the Training set end up forming the test set.

→ About 63.2% of the original data end up with bootstrap, and Remaining 36.8% from the test set $(1 - 1/d)^d = e^{-1} \simeq 0.368$.

→ Repeat this Procedure k times, Overall accuracy of the model.

$$Acc(M) = 1/k \sum_{i=1}^{k} (0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{Train\_set})$$

Rule - Based classification :-

x Rule - Based classifier (or) classification is represented as a set of IF - THEN rules.

x we will see how such rules are used for classification

Using If - Then rules are one of the rule based classification method (or) learning method.

Syntax:

If condition THEN conclusion.

↓ ↓

Rule antecedent Rule consequent

(or)

Pre condition

Ex:-

R_1 : If age = youth AND income = high AND student = yes then buys comp = yes

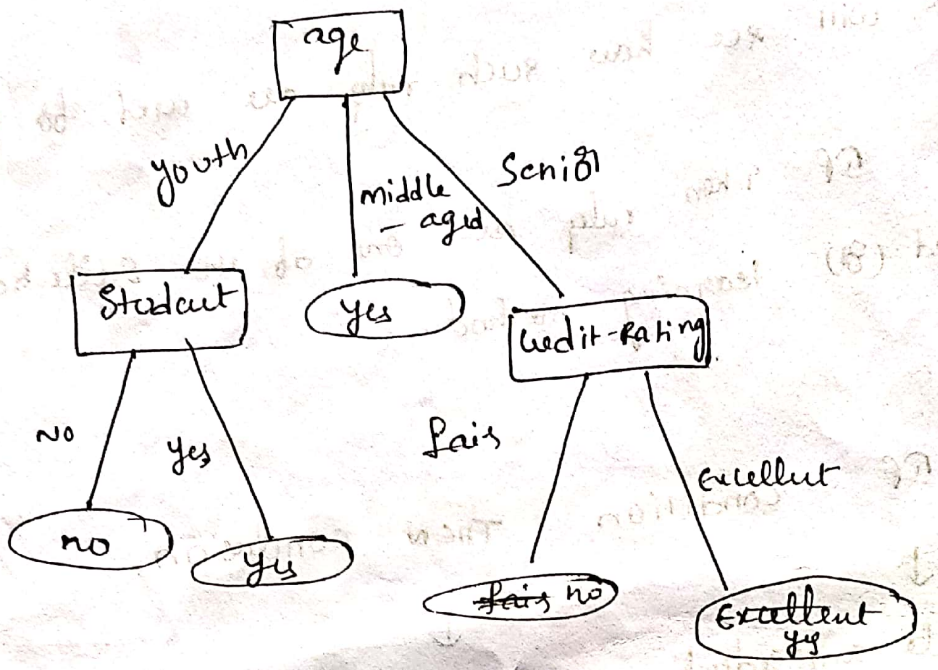
Another way rep If - THEN Rule.

(age = youth) ∧ (income = High) ∧ (student = yes) ⇒ buys comp = yes.

Coverage(R) = $\frac{n_{covers}}{|D|}$ = 2/14 = 14.28%, Accuracy(R) = $\frac{n_{correct}}{n_{covers}}$ = 2/2 = 100%

How to extract the Rules means: ie Rules are build(8). Extracted from the Decision Tree. Decision Tree is Popular method to extract the Rules.

Ex:



Here condition is taken from the path. Path is taken from root node & conclusion is taken from the leaf node.

R_1: age = youth AND Student = no then buys-comp = no

R_2: age = youth AND Student = yes then buys-comp = yes.

R_3: age = middle-aged then buys-comp = yes.

R_4: age = Senior AND Creditrating = Fair then buys comp = no

R_5: age = Senior AND Creditraty = Excellent then buys-comp = yes.

age	income	student	creditrating	buys comp
y	h	NO	fair	yes
y	h	NO	excellent	NO
m	h	NO	fair	yes
s	medium	NO	"	NO
s	medium	yes	"	yes
y	low	NO	"	NO

If (age = youth then) then
buys comp = yes

If age = youth
then buys comp = yes

If age = middle-aged
then buys-comp
= yes

If age = youth
THEN buys comp = yes

If age = income = high
THEN buys comp = yes

If income = high
AND creditrating = fair
THEN buys-comp = yes

If age = middleaged
then buy comp = yes

Rule Induction using a Sequential covering Algorithm.

* IF-THEN rules can be extracted directly from the Training data (without having to generate a decision tree) using a sequential covering algo.

* Sequential covering algo's are the most widely used approach to mine classification Rules.

Algo:-

Input:

- D, a data set class-labeled tuples;
- Att_vals, the set of all attributes & their possible values.

Output: A set of IF-THEN rules.

* In this Algo first rules learn sequentially ie 1st learn first rules next learn second rule. until all rules are completed;

Method
(1) Rule-set = {};
(2) for each class c do
(3) repeat
(4) Rule = learn-one-Rule(D, Att-vals, c);
(5) Remove tuples covered by Rule from D;
(6) Rule-set = Rule-set + Rule;
(7)
(8) until terminating condition;
(9) Return Rule-set;
End for

How are rules learned?

Rules are grown in a general-to-specific manner. first start with empty set. and then gradually keep appending attribute tests to it. suppose our Training data set, D consists of loan application data. Attributes regarding each applicant include age, income, loan term, residence, Credit rating. The classifying attribute is loan-decision, which indicates that whether a loan is accepted (or) Rejected.

TO learn a Rule for the class "Accept"

IF THEN loan-decision = accept.

age	loan term	income	creditrating	loan decision
Youth	short	high	Excellent	Accept
~~Youth~~ middle-age	long	high	Excellent	Accept
Youth	medium	high	fair	Accept
Youth	short	low	E	Reject

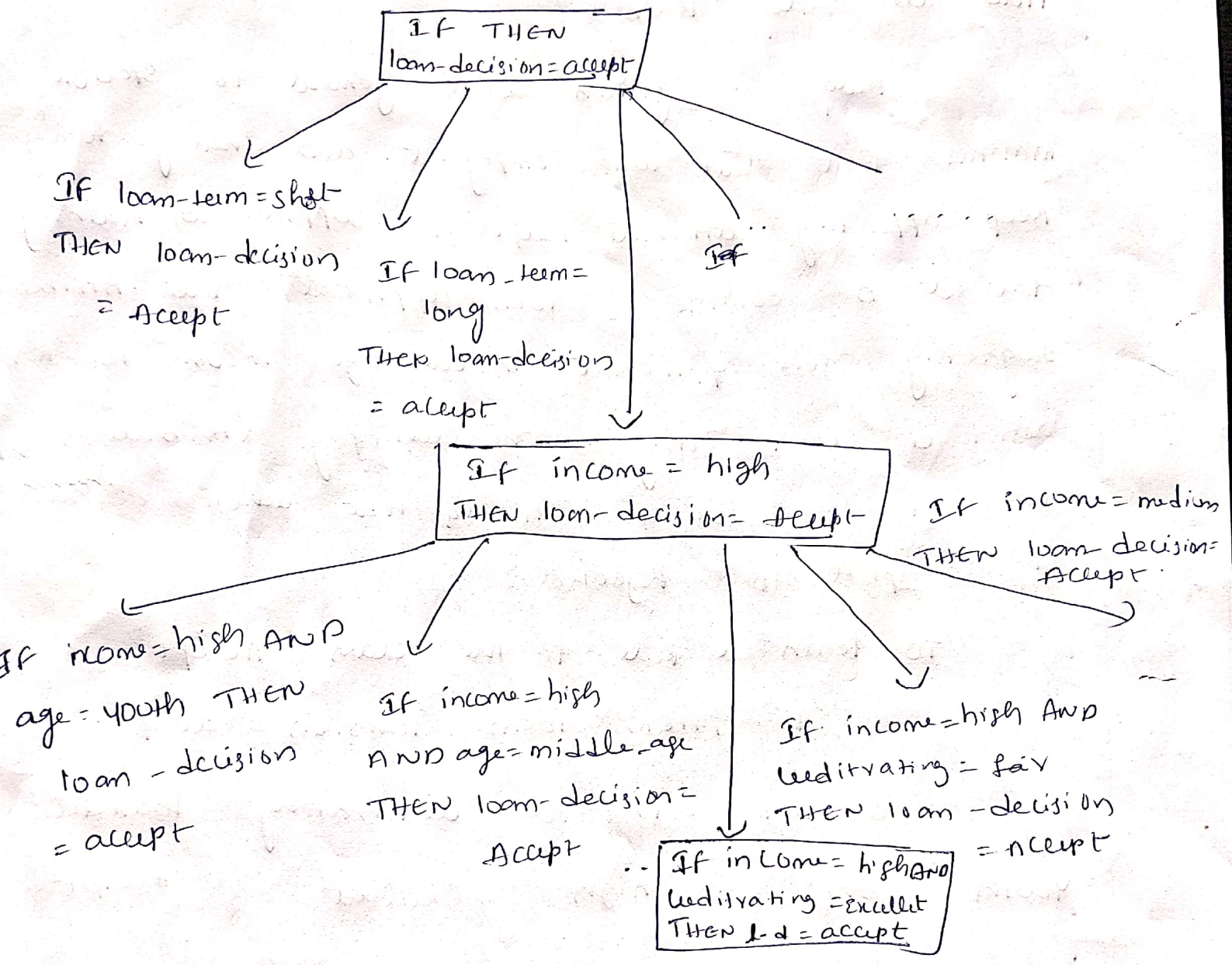


Fig: A general-to-specific search through rule space

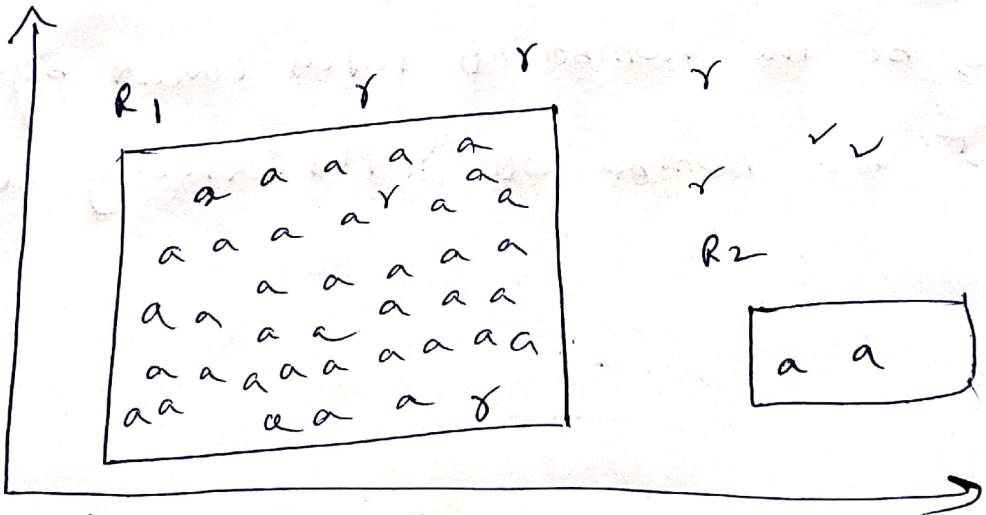
Training data will contain many attributes, each of which may have several possible values. Finding an optimal rule set.

Rule Quality measures

Learn-one-Rule needs a measure of rule quality. Every time it considers an attribute-test, it must check to see if appending such a test to the current values condition will result in an improved rule.

Ex Choosing b/w two rules based on Accuracy

Consider the two rules as illustrated below. Both are for the class loan decision = accept. we use 'a' to rep the tuples of class "accept", 'r' for the tuples of class reject. Rule R_1 correctly classifies 38 of the 40 tuples it covers. Rule R_2 covers only two tuples, which it correctly classifies. Their accuracies are 95% & 100%, R_2 greater accuracy than R_1.



Rules for the class loan decision = accept

Another measure is FOIL (First order Inductive Learner)

$$\text{FOIL- Gain} = pos' \times \left(\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg}\right)$$

Rule Pruning

$$\text{FOIL- Prune}(R) = \frac{pos-neg}{pos+neg'}$$

→ coverage is the percentage of tuples that are covered by the rule.

→ Accuracy - what percentage of them the rule can correctly classify.

→ n covers be the number of tuples covered by R

n correct the number of tuples correctly classified by R.

Introducing Ensemble methods

→ An Ensemble combines a series of K learned models $M_1, M_2 \cdots M_K$ with the aim of creating an improved composite classification model. M^*

→ A given dataset D_1 is used to create K training sets $D_1, D_2 \cdots D_K$ is used to generate classifiers M_i

→ Given a new data tuple to classify, the base classifiers each vote by returning a class prediction

→ The Ensemble returns a class prediction based on the votes of the base classifiers.

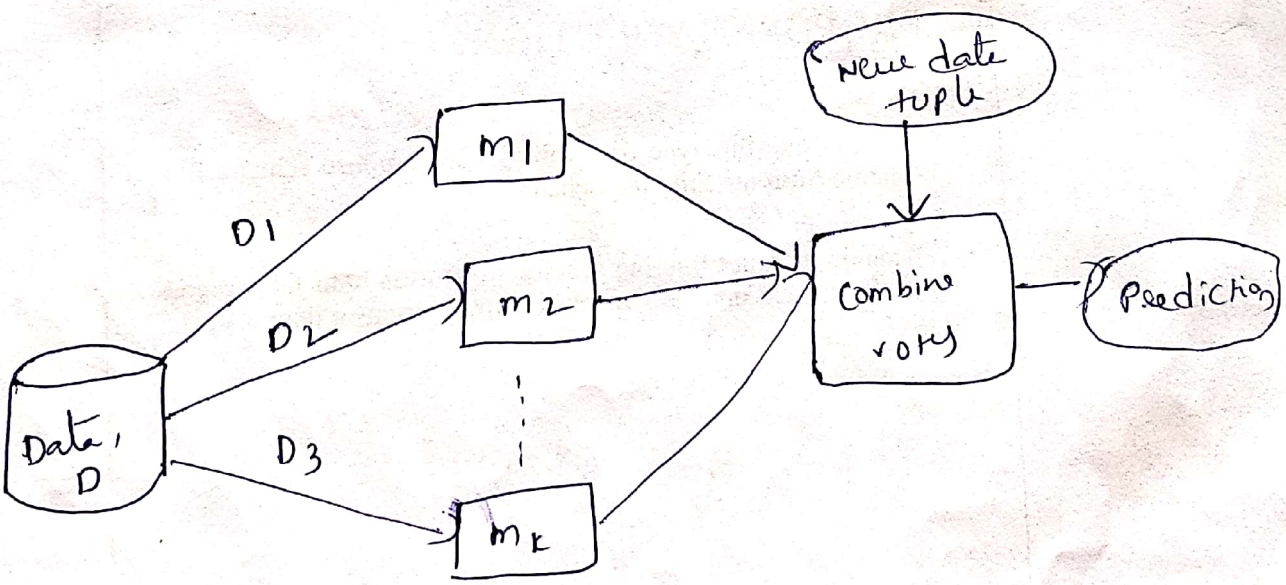


fig: Increasing classifiers accuracy.

Ensemble methods are

1. Bagging
2. Boosting & AdaBoosting
3. Random forests

Bagging: → Bagging works as a method of increasing accuracy. Suppose that you are patient and would like to have diagnosis made based on your symptoms.

→ Instead of asking one doctor, you may choose to ask several. Final diagnosis is made based on majority vote, where each doctor gets an equal vote.

→ Now replace doctor by a classifier. Combine a series of models to generate composite classification model.

→ Bagging is also called as bootstrap aggregation

Algorithm: The bagging algorithm - create an ensemble of classification models for a learning scheme where each model gives an equally weighted prediction

Input:
- → D, set of d Training tuples
- → K, number of models in the ensemble
- → a classification learning scheme (Decision Tree algo, Naive Bayesian

Output : A Composite model M*.

Method

(1) for i = 1 to k do // create k models

(2) create bootstrap sample D_i, by sampling D with replacement;

(3) use D_i and the learning scheme to derive a model M_i;

(4) end for.

Boosting and AdaBoost

→ Suppose that as a patient, you have certain symptoms. Instead of consulting one doctor, you choose to consult several.

→ Suppose you assign weights to the value (or) worth of each doctor's diagnosis, based on the accuracies of previous diagnoses they have made.

→ The final diagnosis is then a combination of the weighted diagnoses.

→ In Boosting, weights are also assigned to each training tuple. A series of k classifiers is learned iteratively.

Bagging

Bootstrap :- (or) Bootstrap aggregating - bagging

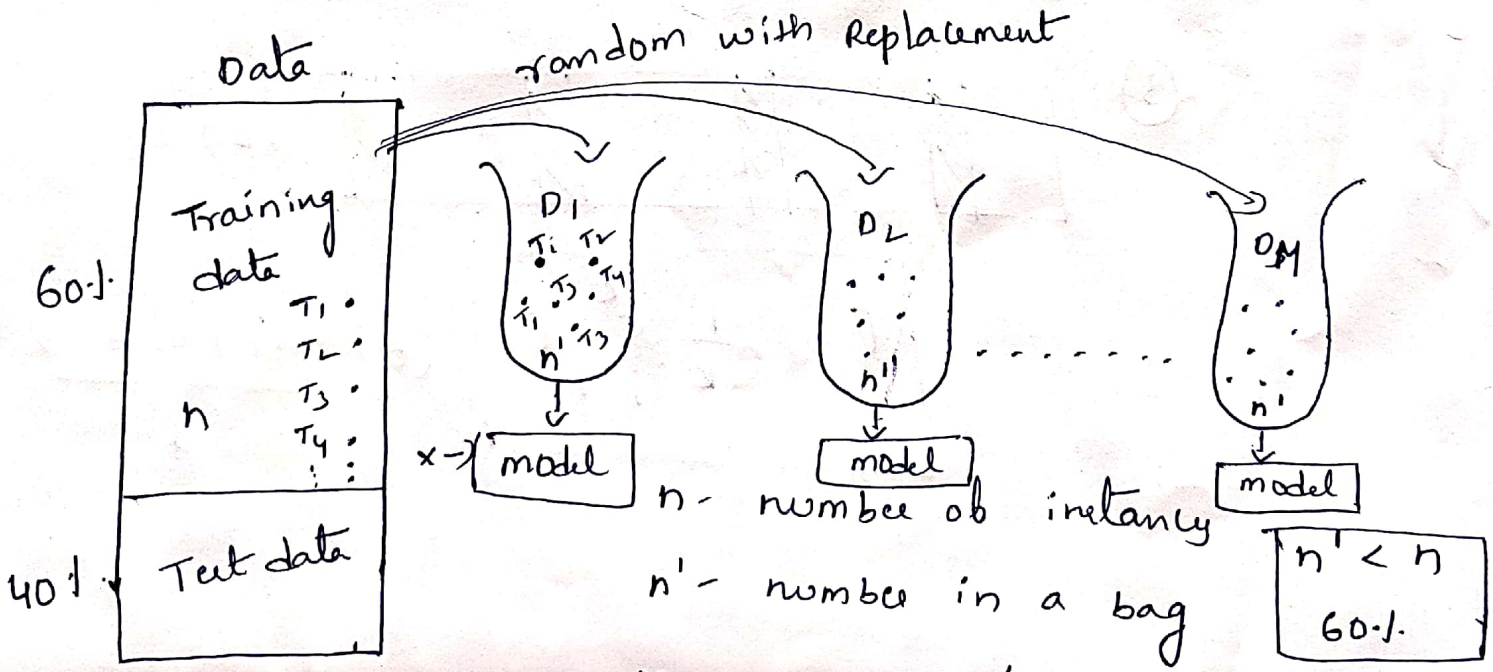
A Bootstrap sample is a random sample conducted with replacement

Steps :- ① Randomly select an observation from the original data

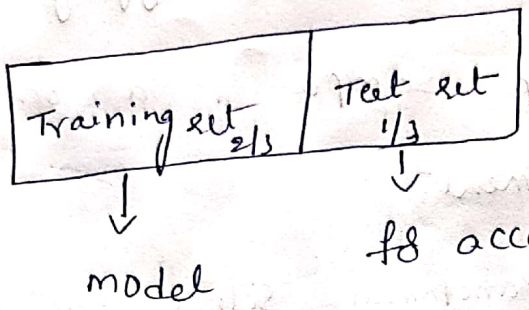
② "write it down"

③ "put it back" (i.e any observation can be selected more than once)

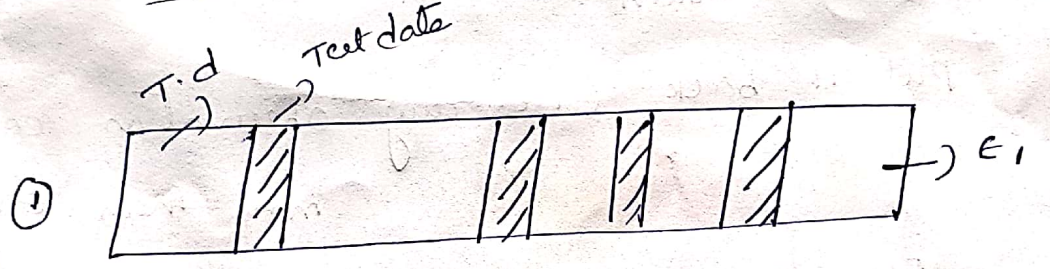
Repeat steps 1-3 N times, N is number of observations in the original sample.

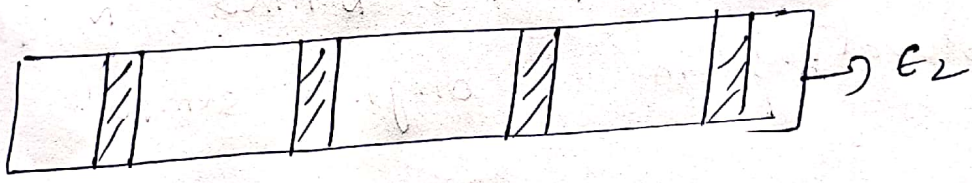


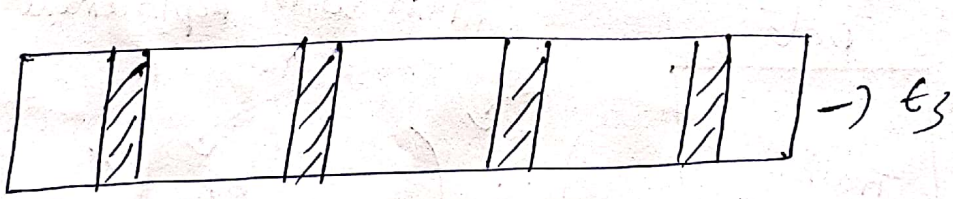
Hold out method :-



Random subsampling (or) Repeated Hold out method







$$\epsilon = \frac{1}{k}\sum_{i=1}^{k} \epsilon_i \quad (\text{or}) \quad \frac{\epsilon_1 + \epsilon_2 + \epsilon_3}{3}$$

Ada Boost.

→ Given a set of d class-labeled tuples $(X_1, Y_1) \cdots (X_d, Y_d)$.

→ 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 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.
* If error rate is calculated using D_i as a test set.
* If a tuple is misclassified, its weight is increased, otherwise it is decreased.

→ $\text{Error}(m_i) = \sum_{j}^{d} w_j \times \text{Err}(X_j)$

→ weight of classifier m_i's vote is

$$\log \frac{1 - \text{Error}(m_i)}{\text{Error}(m_i)}$$

Random Forests

Another Ensemble method called random forests Each of the classifiers in the Ensemble is a decision tree classifiers. so that the collection of classifiers is a "forest".

Two methods to construct Random Forest

Forest - RI :- (Random input selection):

Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximize size.

Forest - RC (random linear combinations):-

Create new attributes (features) that are a linear combination of existing attributes.