UNIT-4
Classification

Syllabus:

Classification: Problem definition, General Approach to solving a classification problem, Evaluation of Classifiers, Classification techniques, Decision trees: Decision Tree Construction, Methods for expressing attribute test conditions, Measures for Selecting the Best split, Algorithm for Decision tree Induction, Naïve-Bayes Classifier, Bayesian Belief Networks; K-nearest neighbor classification-Algorithm and characteristics.

Classification

- Classification is the task of assigning objects to one of several predefined Categories.
- Classification is the task of learning a **target function** f that maps eachattribute set **x** to one of the predefined class labels y.
- The target function is also known informally as a **classification model** or
  - **classifier**

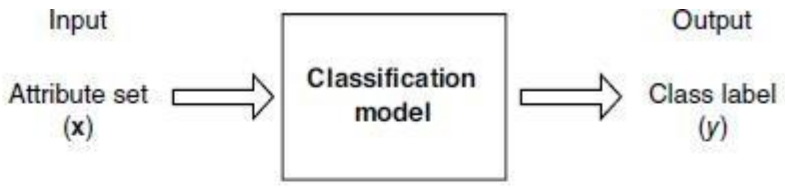


Figure 4.2. Classification as the task of mapping an input attribute set x into its class label y.

- A classification model is useful for the following purposes.
  - **Descriptive Modeling** A classification model can serve as an explanatory tool to distinguish between objects of different classes.
  - **Predictive Modeling** A classification model can also be used to predict the class label of unknown records
- **Examples of classification problems:**
  - A bank loans officer needs analysis of her data to learn which loanapplicants are “safe” and which are “risky” for the bank.
  - A marketing manager needs data analysis to help guess whether acustomer with a given profile will buy a new computer.
  - A medical researcher wants to analyze breast cancer data to predictwhich one of three specific treatments a patient should receive.
- Classification models predict categorical class labels. Regression models predict a continuous-valued function, or ordered value, as opposed to a class label.

- Classification techniques are most suited for predicting or describing datasets with binary or nominal categories.

- **Applications of Classification**:
    - detecting spam email messages based upon the message headerand content
    - categorizing cells as malignant or benign based upon the results ofMRI scans
    - classifying galaxies based upon their shapes
- Classification and numeric prediction (Regression) are the two majortypes of prediction problems. The term prediction refers to both numeric prediction and class labelprediction.

Classification vs Regression

Classification	**Regression**
1. Classification predicts categorical(discrete, unordered) labels. i.e., It is categorical prediction	Regression predicts numerical (continuous, ordered) values. i.e., It is numerical prediction.
2.Clssifficatio model is called nclassifier	Regression model is called predictor
3.It is evaluated by accuracy	It is evaluated by root mean square error
4.The classification algorithms involvedecision tree, logistic regression, Bayesian classification etc.	Regression tree (e.g. Random forest)and linear regression are the examples of regression algorithms
5. Example1: predicting ”will it be cold or hot tomorrow”	Example1: predicting “What is temperature going to be tomorrow”
6. Example2: Predicting whether a customer with a given profile will buy a new computer	Example2: Predicting how much a customer with a given profile will spend during a sale.

General Approach to solving a classification problem

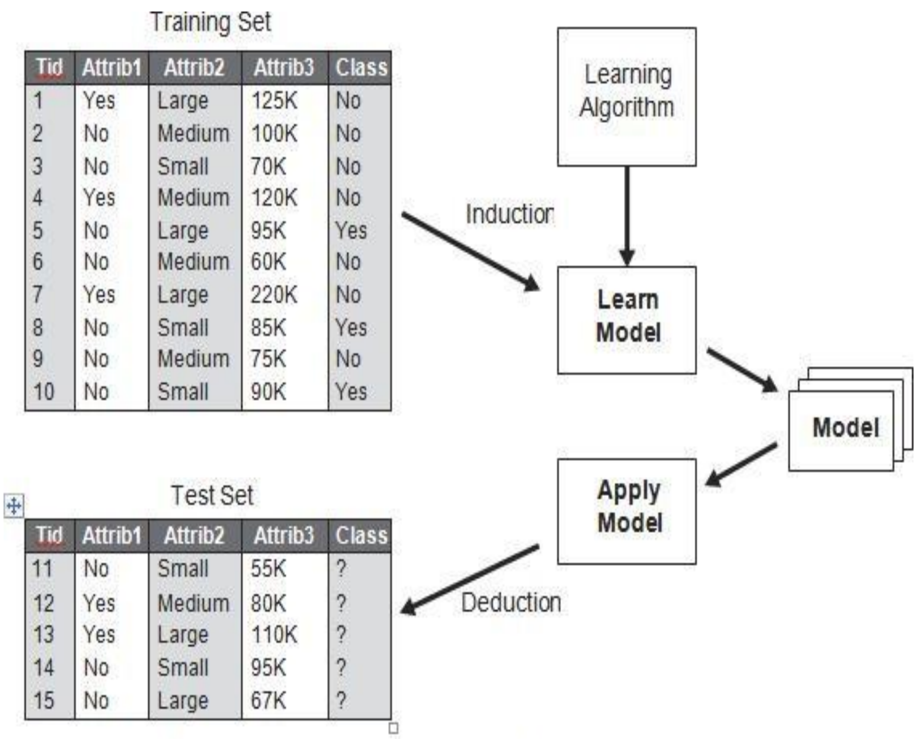
(How does classification work?)

Data classification is a two-step process.

1. **Learning Step (Training Phase):** In this step, a classifier is built describing a predetermined set of data classes or concepts. Here a classification algorithm builds the classifier by analyzing or “learning from” a **training set** made up of database tuples and their associated class labels.

2. **Classification Step (Testing Phase)**:**Test data** are used to estimate

the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new



Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Test Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Figure 4.3. General approach for building a classification model.

datatuples.

- A classification technique (or classifier) is a systematic approach to building classification models from an input data set.
- Each technique employs a **learning algorithm** to identify a model that best fits the relationship between the attribute set and class label of the input data. Therefore, a key objective of the learning algorithm is to build models with good generalization capability.
- A **training set** consisting of records whose class labels are known must be provided. The training set is used to build a classification model, which is subsequently applied to the **test set**, which consists of records with unknown class labels.

Terminology used in classification

Training data set vs Test data set:

The “training” data set is the general term for the samples used to create the model, while the “test” or “validation” data set is used to qualify performance.

Training set consisting of records whose labels are known where as test set consists of records with unknown class labels.

In the context of classification, data tuples or data records can be referred to as samples, examples, instances, data points, or objects.

Accuracy vs Error Rate

Accuracy(Recognition rate) of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier**.**

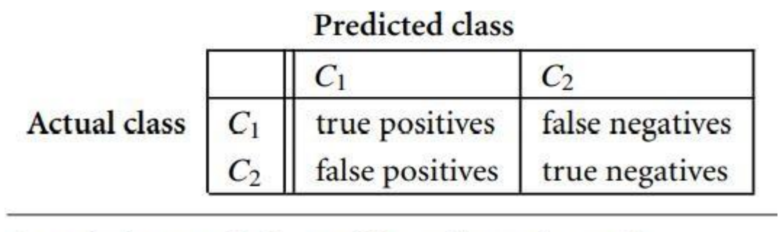
$$\text{Accuracy} = \frac{NumberOfCorrectprediction}{TotalNumberOf\ \mathrm{Pr}\ ediction}$$

Error Rate(Misclassification Rate):of a classifier on a given test set is the percentage of test set tuples that are incorrectly classified by the classifier.

$$\text{Error Rate} = \frac{\text{Number of wrong predictions}}{\text{Total Number of Predictions}}$$

Confusion Matrix:

- A confusion matrix is a technique for summarizing the performance of a classification algorithm.
- Counts of test records that are correctly and incorrectly predicted by the model are tabulated in a table called Confusion matrix.
- The confusion matrix is a useful tool for analyzing how well a classifier can recognize tuples of different classes.
- Given m classes, a confusion matrix is a table of at least size m by m
- Each row in a confusion matrix represents an actual class, while each column represents a predicted class.
- The table may have additional rows or columns to provide totals or recognition rates per class.



		Predicted class	
		C_1	C_2
Actual class	C_1	true positives	false negatives
	C_2	false positives	true negatives

A confusion matrix for positive and negative tuples.

- **Positive tuples**: tuples of the main class of interest, class label = yes
- **Negative tuples**: tuples where Class label= no
- **True Positives (TP or t-pos):** Positive tuples that were correctly labeled by the classifier. i.e., the no. of predictions where the classifier correctly predicts the positive class as positive.
- **True Negatives (TN or t-neg)**: Negative tuples that were correctly labeled by the classifier. i.e., correctly predict the negative class as negative.

- **False Positives (FP or f-pos)**: Negative tuples that were incorrectly labeled by the classifier. i.e., incorrectly predicts the negative class as positive.\
- **False Negatives (FN or f-neg):** Positive tuples that were incorrectly labeled by the classifier i.e., incorrectly predicts the positive class as negative.

Accuracy = Number of correct prediction/Total number of prediction
=(TP + TN)/(TP + FN + FP +TN)

Error Rate = Number of wrong prediction/Total number of prediction

= (FN + FP)/(TP + FN+ FP+TN)

Note:

- Error Rate = 1- Accuracy
- Most classification algorithms seek models that attain the highest accuracy or equivalently, the lowest error rate when applied to the test set
-

Supervised learning vs Unsupervised learning:

- In a **supervised learning** model, the algorithm learns on a labeled dataset, providing an answer key that the algorithm can use to evaluate itsaccuracy on training **data**.
- An unsupervised model, in contrast, provides unlabeled data that the algorithm tries to make sense of by extracting features and patterns on its own.

Ovefitting vs underfitting

- **Overfitting**: Good performance on the training data, poor generalization to other data.
- **Underfitting**: Poor performance on the training data and poor generalization to other data.

<u>**Evaluation of Classifiers**</u>

Evaluation Criteria: 1.Accuracy $\longrightarrow$ Alternatives to accuracy: 1.Error rate
2.Speed
3.Robustness
4.Scalability
5.Interpretability

Alternatives to accuracy:
1.Error rate
2.Sensitivity
3.Speecificity
4.Precision
5.F1-measure

Evaluation Techniques:
1. Holdout
2. Random Subsampling
3. Cross validation
4. Bootstrap

Evaluation Criteria:

Classification methods can be evaluated according to the following criteria:

- **Accuracy:** The accuracy of a classifier refers to the ability of a given classifier to correctly predict the class label of new or previously unseen data.
- **Speed:** This refers to the computational costs involved in generating and using the given classifier or predictor.
- **Robustness:** This is the ability of the classifier or predictor to make correct predictions given noisy data or data with missing values.
- **Scalability:** This refers to the ability to construct the classifier or predictor efficiently given large amounts of data.
- **Interpretability:** This refers to the level of understanding andinsight that is provided by the classifier. Interpretability is subjective and therefore more difficult to assess.

Alternatives to the Accuracy Measure:

- For classifiers, sensitivity, specificity, and precision are useful alternatives to the accuracy measure, particularly when the main class of interest is in the minority.
- **Sensitivity** is also referred to as the true positive (recognition) rate (that is, the proportion of positive tuples that are correctly identified)

  Sensitivity = TP/(TP+FN) = t-pos/pos

- **Specificity** is the true negative rate that is, the proportion of negative tuples that are correctly identified.

  Specificity = TN/(TN+FP) = t-neg/neg

- **Precision** is the percentage of tuples labeled as positive that actually are positive tuples.

  Precision= TP/(TP+FP) = t-pos/(t-pos+f-pos)

- **Accuracy** is a function of sensitivity and specificity:

$$accuracy = sensitivity\frac{pos}{(pos+neg)} + specificity\frac{neg}{(pos+neg)}.$$

- **F1 measure** represents a harmonic mean between recall(r) and precision (p).

$$F_1 = \frac{2rp}{r+p} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

Accuracy = (TP+TN)/(TP+TN+FP+FN) = (t-pos + t-neg)/(pos +neg)
Error Rate = (FP+FN)/(TP+TN+FP+FN) = (f-pos +f-neg)/(pos +neg)
Sensitivity = TP/(TP+FN) = t-pos/posor (recall)
Specificity = TN/(TN+FP) = t-neg/neg
Precision = TP/(TP+FP) = t-pos/(t-pos+f-pos)

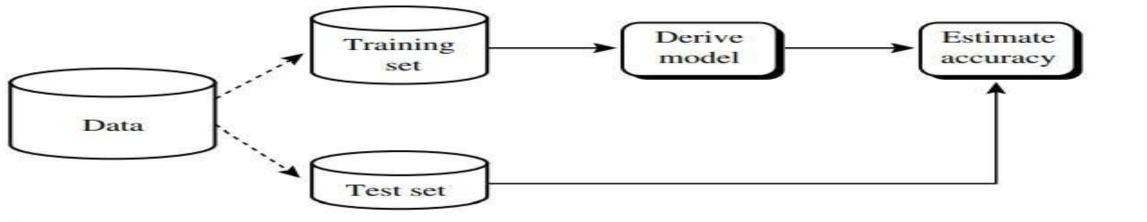
Evaluating Techniques:

Common techniques for estimating (assessing) of classifier accuracy are

- Holdout
- Random Subsampling
- Cross validation
- Bootstrap

Holdout

- In holdout method, the given data are randomly partitioned into two independent sets, a training set and a test set.
- Typically, two-thirds of the data are allocated to the training set, and the remaining one-third is allocated to the test set.
- The training set is used to derive the model, whose accuracy is estimated with the test set.
- The estimate is pessimistic because only a portion of the initial data is used to derive the model.



Estimating accuracy with the holdout method.

Random Subsampling

- Random sub sampling is a variation of the holdout method in which the holdout method is repeated k times.
- The overall accuracy estimate is taken as the average of the accuracies obtained from each of iteration.
- Let acc_i be the model accuracy during the *ith* iteration. The overall accuracy is given by

$$acc_{\text{sub}} = \sum_{i=1}^{k} acc_i/k.$$

- Disadvantages:
  - It does not utilize as much data as possible for training.
  - It also has no control over the number of times each record is usedfor testing and training.

Cross-Validation

- In k-fold cross-validation, the initial data are randomly partitioned into k mutually exclusive subsets or “folds,” D_1, D_2,...,D_k, each of approximately equal size.
- Training and testing is performed k times.
- In iteration i, partition Di 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_k collectively serve as the training set in order to obtain a first model, which is tested on D1;
- The second iteration is trained on subsets D1, D3,..., D_k and tested on D2;and so on.
- Unlike the holdout and random sub sampling methods above, here, each sample is used the same number of times for training and once for testing.
- The accuracy is estimated as the overall number of correct classifications from the k iterations, divided by the total number of tuples in the initial data.
- **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.
- In **stratified cross-validation**, the folds are stratified so that the class distribution of the tuples in each fold is approximately the same as that in the initial data.
- In general, stratified 10-fold cross-validation is recommended for estimating accuracy.

Bootstrap

- Unlike the accuracy estimation methods mentioned above, the bootstrap method samples the given training tuples uniformly with replacement.
- That is, each time a tuple is selected, it is equally likely to be selected again and re added to the training set.
- There are several bootstrap methods. A commonly used one is the .632 bootstrap, which works as follows.
- Suppose we are given a data set of d tuples. The data set is sampled d times, with replacement, resulting in a bootstrap sample or training set of d samples.
- It is very likely that some of the original data tuples will occur more than once in this sample. The data tuples that did not make it into the training set end up forming the test set.

- Suppose we were to try this out several times. As it turns out, on average, 63.2% of the original data tuples will end up in the bootstrap, and the remaining 36.8% will form the test set (hence, the name, .632 bootstrap.)
    - i.e; 36.8% of tuples will not be selected for training and thereby end up in the test set, and the remaining 63.2% will form the training set.
- This approximation follows from the fact that the probability a record is chosen by a bootstrap sample is $1- (1\text{-}1/d)^{d}$ .When N is sufficiently large, the probability asymptotically approaches $1- e^{-1}$ = 0.632.

Problem 1:
Using the confusion matrix below, answer the following questions

	FALSE	**TRUE**
0	94	23
1	24	100

a. Number of true positives?
b. Number of false negatives?
c. Number of true negatives?
d. Number of false positives?
e.

Solution:
TP=100
FN=24
TN=94
FP=23

Problem 2:
A binary classifier was evaluated using a set of 1,000 test examples in which 50% of all examples are negative. It was found that the classifier has 60% sensitivity and 70 % accuracy. Write the confusion matrix.

Solution:
Total examples =1000
Pos=50% = 500
Neg=50% = 500
sensitivity = 60%
t-pos/pos =0.6
t-pos/500 =0.6, **t-pos=0.6*500=300**
accuracy =70%
(t-pos +t-neg)/(pos+neg)=0.7

t-pos + f-neg =pos
300 + f-neg=500
f-neg=500-300=200

(300 +t-neg)/1000 =0.7
300+t-neg = 700
t-neg=700-300=400

t-neg + f-pos=neg
400 + f-pos= 500
f-pos=500-400=100

Confusion Matrix:

	+	**_**
+	300	200
_	100	400

Problem 3:

Suppose we design a model to identify iphones from a video that also contain android phones. If the program identifies 5 iphones in a scene containing 7 iphones and some android phones. if 3 of the identification are correct but 2 are actually android phones.What is the precision & sensitivity of the model?

Solution:

Pos=7
TP=3
FP=2
Precision = TP/(TP +
FP)=3/5=0.6Sensitivity=TP/Pos
= 3/7=0.43

<u>Classification Techniques:</u>

- Classification by Decision Tree Induction
- Bayesian Classification
- K-Nearest Neighbor Classification
- Rule-Based Classification
- Neural Networks
- Support Vector Machines
- Classification by back propagation
- Associative classification
- Logistic Regression
- Random Forest

<u>Classification by Decision Tree Induction</u>

- Decision tree induction is the learning of decision trees from class-labeledtraining tuples.
- A decision tree is a flowchart-like tree structure, where each

internalnode (non-leaf node) denotes a test on an attribute, each **branch** represents an outcome of the test, and each **leaf node (or terminal node)** holds a class label. The topmost node in a tree is the **root node**.

- A decision tree is a hierarchical structure consisting of nodes and directededges.
- In decision tree, **internal nodes** are denoted by **rectangles**, and **leaf nodes** are denoted by **ovals**.
- Some decision tree algorithms produce only *binary* trees (where each internal node branches to exactly two other nodes), whereas others can produce non binary trees.
- The decision tree has three types of nodes:
    - A **root node** that has no incoming edges and zero or more outgoingedges.
    - **Internal nodes**, each of which has exactly one incoming edge andtwo or more outgoing edges.
    - **Leaf** or **terminal** nodes, each of which has exactly one incoming edge and no outgoing edges.

Example1:

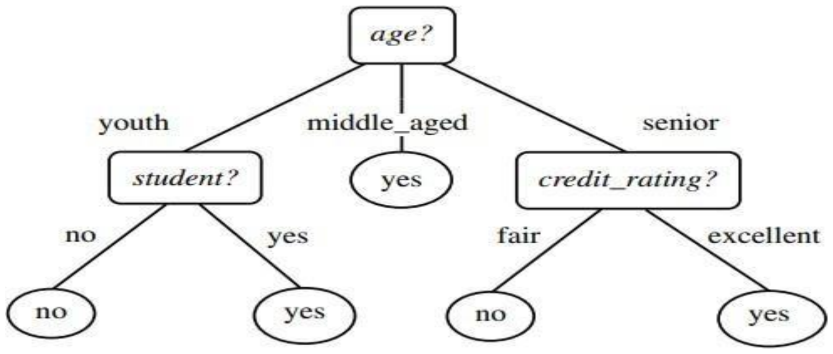


Figure: A decision tree for the concept *buys computer*

Example2:

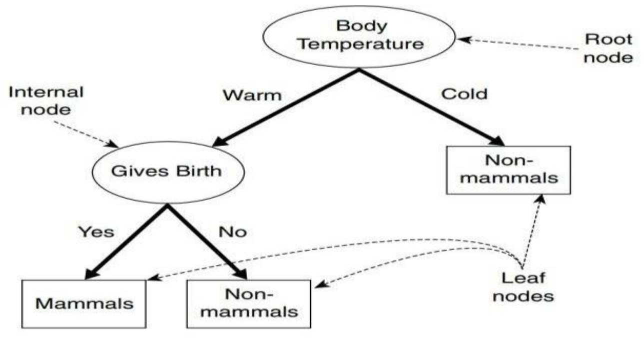
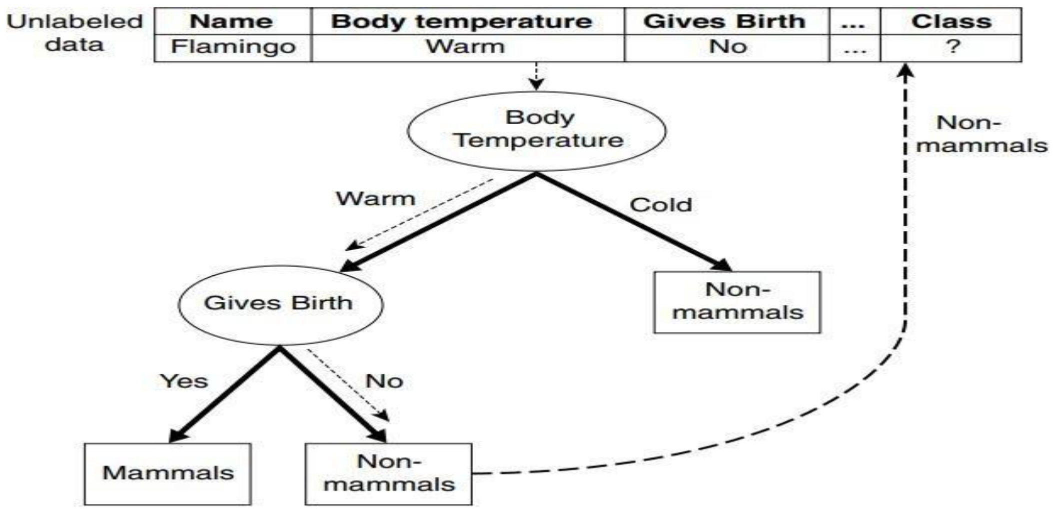


Figure 4.4. A decision tree for the mammal classification problem.

How are decision trees used for classification?

Given a tuple, X, for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traced from the root to a leaf node, which holds the class prediction for that tuple.



Why are decision tree classifiers so popular?

- The construction of decision tree classifiers does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery.
- The learning and classification steps of decision tree induction are simple and fast.
- In general, decision tree classifiers have good accuracy.
- Decision tree induction algorithms have been used for classification in many application areas, such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology.
- Decision trees are the basis of several commercial rule induction systems.

The most notable types of decision tree algorithms

1. **ID3 (Iterative Dichotomiser 3) :**This algorithm uses Information Gain to decide which attribute is to be used classify the current subset of the data. For each level of the tree, information gain is calculated for the remaining data recursively.
2. **C4.5:** This algorithm is the successor of the ID3 algorithm. This algorithm uses either Information gain or Gain ratio to decide upon the classifying attribute. It is a direct improvement from the ID3 algorithm as it can handle both continuous and missing attribute values.

3. **CART** (**Classification and Regression Tree):** It is a dynamic learning algorithm which can produce a regression tree as well as a classification tree depending upon the dependent variable.

Assumptions and Notes for Basic Algorithm

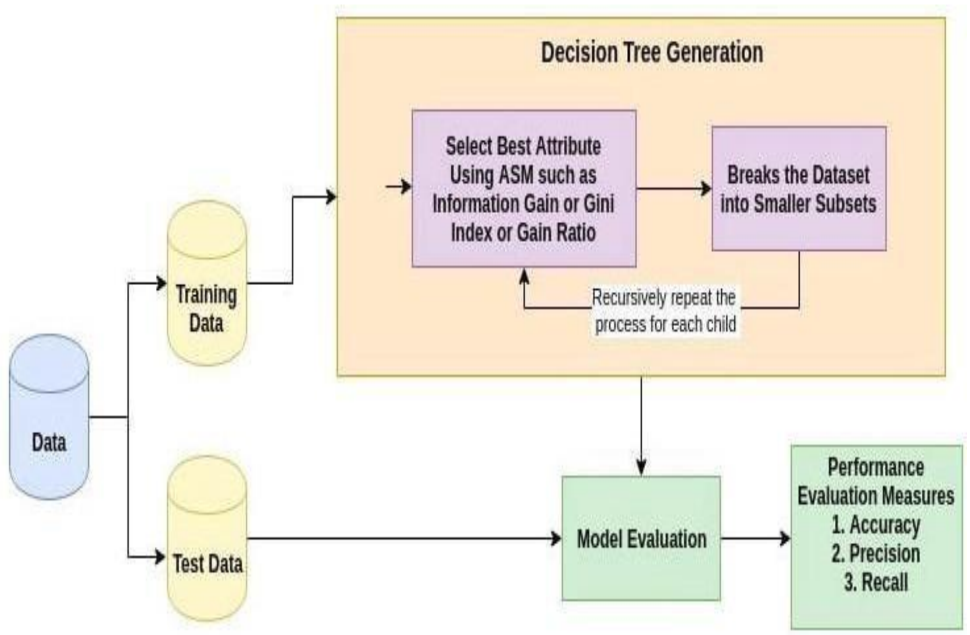
- Attributes are categorical. If continuous-valued, they are discretized in advanced
- Examples are partitioned recursively based on selected attributes
- Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- At start, all the training examples are at the root

Basic Algorithm

- The basic algorithm for decision tree construction is a greedy algorithm that constructs decision trees in a top-down recursive divide-and-conquer manner.
- ***Algorithm:***
  1. Create root node of decision tree with all training tuples.
  2. Select best attribute (split attribute) by Attribute Selection Measure topartition these tuples
  3. For each value of selected attribute a branch is created and the corresponding subset of tuples that have the attribute values specifiedby the branch is moved to the newly created child node.
  4. At any child node, if all tuples are of the same class, then leaf node iscreated and labeled with that class
  5. Algorithm applied recursively to each child node until
     i) all tuples at node are of one class or
     ii) no remaining attribute for further partitions or
     iii) no tuples for a given branch.

The recursive partitioning STOPS only when any one of the following conditions is true

- All the samples (records) in the partition are of the same class, thenthe node becomes the leaf labeled with that class
- There is no remaining attributes on which the data may be further partitioned, i.e. we have only class attribute left in this case. we apply MAJORITY VOTING to classify the node. MAJORITYVOTING involves converting the node into a leaf and labeling it with the most common class in the training data set
- There are no records (samples) left – a LEAF is created with majority vote for training data set.


Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Data
Training Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Attribute Selection Measures:

- A heuristic for selecting the splitting criterion that Decision Tree "best" separates a given data partition, D, of class labeled training instances into individual classes.
- Attribute selection measures are also known as splitting rules because they determine how the instances at a given node are to be split.
- The attribute selection measure provides a ranking for each attribute describing the given training instances.
- The attribute having the best score for the measure is chosen as the splitting attribute for the given instances.
- Three popular attribute selection measures are:
    - **Information Gain**
    - **Gain ratio**
    - **Gini index**

Information Gain

- ID3 uses information gain as its attribute selection measure.
- Let node N represents or hold the tuples of partition D. The attribute with the highest information gain is chosen as the splitting attribute for node
- N. This attribute minimizes the information 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 number of tests needed to classify a given tuple and guarantees that a simple (but not necessarily thesimplest) tree is found.
- The expected information needed to classify a tuple in D is given by:

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

- where pi is the probability that an arbitrary tuple in D belongs to class Ci and is estimated by |Ci,D|/|D|.
- Info(D) is also known as the **entropy** of D.
- Suppose we were to partition the tuples in D on some attribute A having v distinct values, {a1, a2,..., av}, 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,
- {D1, D2,..., Dv}, where Dj contains those tuples in D that have outcome aj of A.
- InfoA (D) is the expected information required to classify a tuple from D based on the partitioning by A.

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

- Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on A). That is,
- $Gain(A) = Info(D) - Info_A(D).$

Example:

Following Table presents a training set, D, of class-labeled tuples.

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

- The class label attribute, buys computer, has two distinct values (namely, {yes, no}); therefore, there are two distinct classes (that is, m = 2). Let class C1 correspond to yes and class C2 correspond to no.
- There are nine tuples of class yes and five tuples of class no. A (root) node N is created for the tuples in D. To find the splitting criterion for these tuples, we must compute the information gain of each attribute. Theexpected information needed to classify a tuple in D:

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

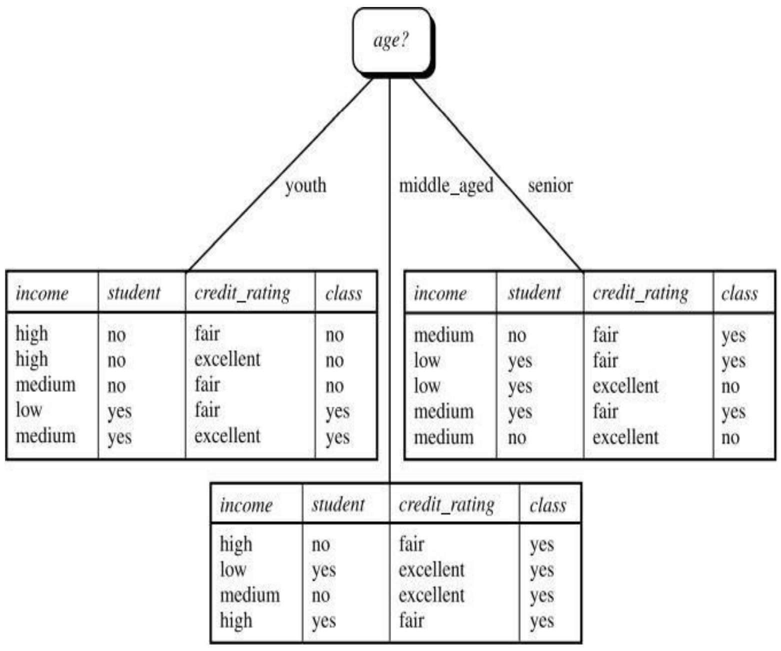
- Next, we need to compute the expected information requirement for each attribute. Let's start with the attribute age. We need to look at the distribution of yes and no tuples for each category of age.
- For the age category youth, there are two yes tuples and three no tuples. For the category middle aged, there are four yes tuples and zero no tuples.For the category senior, there are three yes tuples and two no tuples.
- The expected information needed to classify a tuple in D if the tuples are partitioned according to age is

$$\begin{aligned} Info_{age}(D) = {} & \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) \\ & + \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right) \\ & + \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) \\ = {} & 0.694 \text{ bits.} \end{aligned}$$

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

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

  - Similarly, we can compute Gain(income) = 0.029 bits, Gain(student)= 0.151 bits, and Gain(credit rating) = 0.048 bits. Because age has the highest information gain among the attributes, it is selected as the splitting attribute. Node N is labeled with age, and branches are grown for each of the attribute's values. The tuples are then partitioned accordingly.

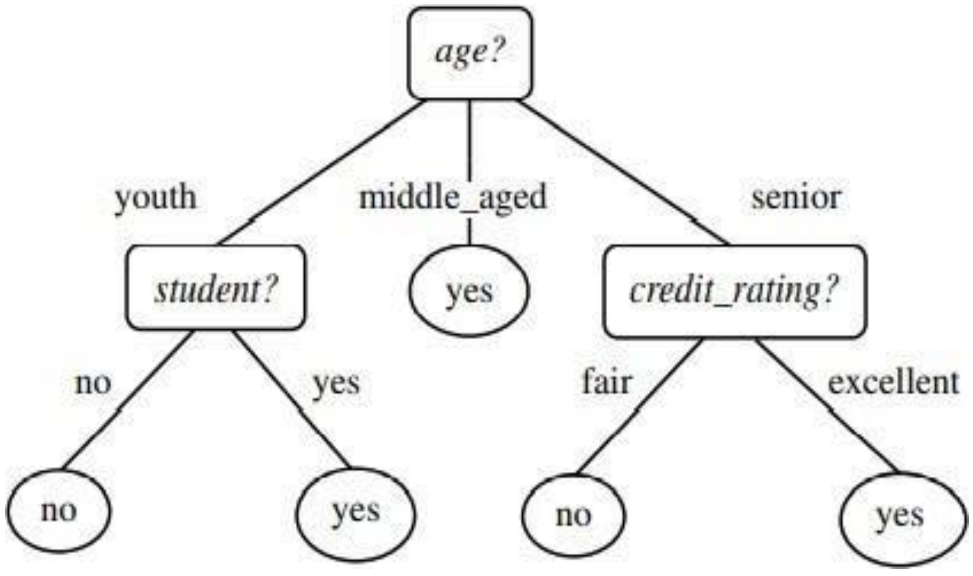


income	student	credit_rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

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

The final decision tree returned by the algorithm is:



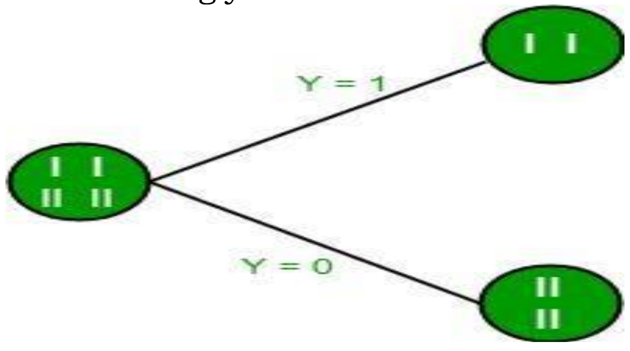
Example:
Draw a Decision Tree for the following data using Information gain.

X	Y	Z	Class
1	1	1	C1
1	1	0	C1
0	0	1	C2
1	0	0	C2

There are two tuples of class C1 and two tuples of class C2.

- The expected information needed to classify a tuple in above data set D is:
  Info(D) = Entropy(D)= -2/4 log(2/4)-2/4 log(2/4) = 1

- The expected information needed to classify a tuple in D if the tuples arepartitioned according to X is :
  InfoX(D)=1/4(-0/1 log(0/1)-1/1log(1/1))+3/4(-2/3log(2/3)-1/3log(1/3)=0.68865
  Gain(X) = Info(D)-infoX(D)=1-0.6887=0.3114
- The expected information needed to classify a tuple in D if the tuples arepartitioned according to Y is:
  InfoY(D)=2/4(-0/1log(0/2)-1/1log(0/2)+2/4(-2/2log(2/2)-0/2log(0/2) =0
  Gain(Y) = Info(D)-info$_Y$(D)=1
- The expected information needed to classify a tuple in D if the tuples arepartitioned according to Z is :
  InfoZ(D)=2/4(-1/2log(1/2)-1/2log(1/2)) +2/4(-1/2log(1/2)-1/2log(1/2) =1
  Gain(Z) = Info(D)-info$_Z$(D)=0
- Because Y has the highest information gain among the attributes, it is selected as the splitting attribute. Node N is labeled with Y, and branches are grown for each of the attribute's values. The tuples are then partitioned accordingly.



Gain ratio:

- The information gain measure is biased toward tests with many outcomes. That is, it prefers to select attributes having a large number of values.
- 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

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

- This value represents the potential information generated by splitting the training data set, D, into v partitions, corresponding to the v outcomes ofa test on attribute A.
- The gain ratio is defined as:

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

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

Example: Computation of gain ratio for the attribute income.

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

- A test on income splits the data of above Table into three partitions, namely low, medium, and high, containing four, six, and four tuples, respectively.

- To compute the gain ratio of income, we first find

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

- From previous Example we have Gain(income) = 0.029.
- Therefore, GainRatio(income) = 0.029/0.926 = 0.031.

Gini index

- The Gini index is used in CART.
- The Gini index is measure of the impurity(inequility) of D , a data partition or set of training tuples
- It is find using the formula:

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

  - where pi is the probability that a tuple in D belongs to class Ci and isestimated by |Ci,D|/|D|. The sum is computed over m classes.

Induction of a decision tree using gini index

1. Compute the gini index of D (entire training data set) using formula

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

2. Compute the gini index for each attribute. For each attribute, each of the possible binary splits is considered.
   - If attribute A has v possible values, then there are possible ways to form two partitions of the data, D, based on a binary split on A.
   - Compute a weighted sum of the impurity of each resulting partition. For example, if a binary split on A partitions D into D1 and D2, the gini index of D given that partitioning is

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

   - The subset that gives the minimum gini index for that attribute is selected as its splitting subset and this minimum gini index is gini index of that attribute.
3. Compute reduction in impurity that would be incurred by a binary attribute A using the formula

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

4. The attribute that maximizes the reduction in impurity (or, equivalently, has the minimum Gini index) is selected as the splitting attribute.

Example:

Let D be the training data of previous 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

First compute Gini index of D:

- To find the splitting criterion for the tuples in D, we need to compute the gini index for each attribute. Let's start with the attribute income and consider each of the possible splitting subsets. Consider the subset {low, medium}. This would result in 10 tuples in partition D1 satisfying the condition "income ∈ {low, medium}." The remaining four tuples of D would be assigned to partition D2. The Gini index value computed based on this partitioning is:

$$\begin{aligned}
&Gini_{income \in \{low, medium\}}(D) \\
&= \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2) \\
&= \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right) \\
&= 0.450 \\
&= Gini_{income \in \{high\}}(D).
\end{aligned}$$

- Similarly, the Gini index values for splits on the remaining subsets are: 0.315 (for the subsets {low, high} and {medium}) and 0.300 (for the subsets {medium, high} and {low}). Therefore, the best binary split for attribute income is on {medium, high} (or {low}) because it minimizes the gini index.
- Evaluating the attribute, we obtain {youth, senior} (or {middle aged}) as the best split for age with a Gini index of 0.375; the attributes {student} and {credit rating} are both binary, with Gini index values of 0.367 and 0.429, respectively. The attribute income and splitting subset {medium, high} therefore give the minimum Gini index overall, with a reduction in impurity of 0.459 − 0.300 = 0.159.
- Hence, the Gini index has selected income instead of age at the root node, unlike the tree created by information gain

Exercise:

Consider the following data set for a binary class problem:

A	B	Class Label
T	F	+
T	T	+
T	T	+
T	F	−
T	T	+
F	F	−
F	F	−
F	F	−
T	T	−
T	F	−

i) Calculate the information gain when splitting on A and B. Whichattribute would the decision tree induction algorithm choose?

ii) Calculate the gain ratio when splitting on A and B. Which attributewould the decision tree induction algorithm choose?

iii) Calculate the gain in the Gini index when splitting on A and B. Whichattribute would the decision tree induction algorithm choose?

Solution:

i)

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

Info(D)= $-4/10\log_2(4/10) - 6/10\log_2(6/10) = 0.9710$

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

$\text{Info}_A(D)=$ $7/10(-4/7\log_2(4/7) - 3/7\log_2(3/7) +$
$3/10(-0/3\log_2(0/3) - 3/3\log_2(3/3)= 0.68964 +0 = 0.68964$

$\text{Info}_B(D)=$ $4/10(-3/4\log_2(3/4) - 1/4\log_2(1/4) +$
$6/10(-1/6\log_2(1/6) - 5/6\log_2(5/6) =0.71452+ 0.39=1.1045$

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

Gain(A) = info(D) – info_A(D) = 0.9710-0.6896 = 0.2813
Gain(B) = info(D) – info_B(D) = 0.9710-0.71452=0.2565

Because A has the highest information gain, it is selected as the splitting attribute.

Attribute A will be chose to split the node

ii)

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

$SplitInfo_A(D) = -7/10\log_2(7/10) - 3/10\log_2(3/10)$
$= 0.3602+0.5210=0.8812$
$SplitInfo_B(D) = -4/10\log_2(4/10) - 3/10\log_2(6/10)$
$= 0.5287+0.4421=0.9708$

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

$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)} = (0.2813)/(0.8812)=0.3192$

GainRatio(B) =*Gain(B)/ SplitInfo(B) =(0.2565)/(0.9708)=0.2642*

Because A has the highest Gain Ratio, it is selected as the splittingattribute.
Attribute A will be chose to split the node

iii)

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

= 1-(4/10)²-(610)² =0.48

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

$Gini_A(D)$ =7/10(1-(4/7)^2-(3/7)^2) +3/10(1-(0/3)^2-(3/3)^2)
=7/10(0.4898)+3/10(0) = 0.3429
$Gini_B(D)$ =4/10(1-(1/4)^2-(3/4)^2) +6/10(1-(1/6)^2-(1/5)^2)
=4/10(0.3750) + 6/10(0.2778) = 0.3166

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

ΔGini(A) = Gini(D)-$Gini_A$(D) = 0.48-0.3429 = 0.1371
ΔGini(B) = Gini(D)-$Gini_B$(D) = 0.48-0.3167 = 0.1633
Because B has the highest reduction in impurity (i.e. minimum Gini index) , it is selected as the splitting attribute.
Attribute B will be chose to split the node

<u>Methods for Expressing Attribute Test Condition</u>

Decision tree induction algorithms must provide a method for expressing an attribute test condition and its corresponding outcomes for different attribute types.

Binary Attributes

The test condition for a binary attribute generates two potential outcomes,as shown in following Figure

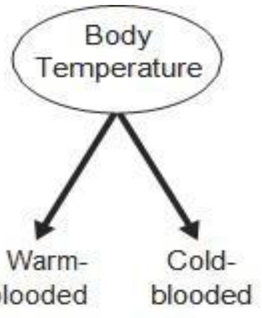


Figure 4.8. Test condition for binary attributes.

Nominal Attributes

Nominal attribute test condition can be expressed in two ways, as shownin following figure:

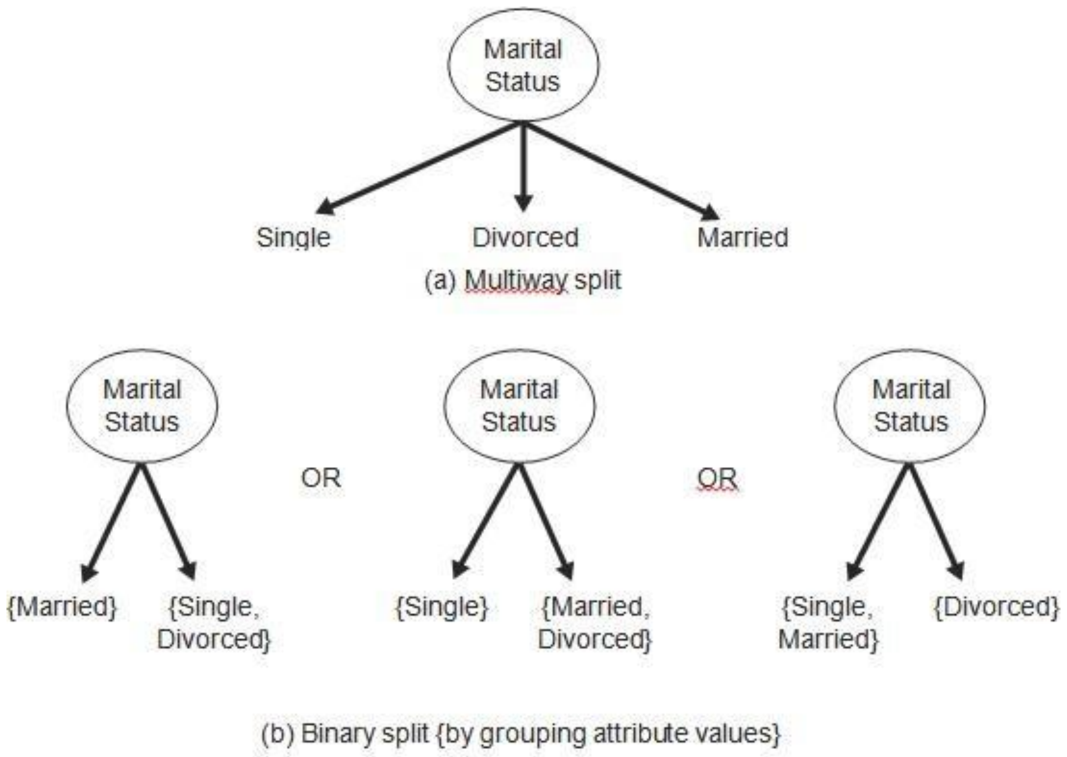


Figure 4.9. Test conditions for nominal attributes.

For a multi-way split, the number of outcomes depends on the number of distinct values for the corresponding attribute. For example, if an attribute such as marital status has three distinct values—single, married, or divorced—its test condition will produce a three-way split.

On the other hand, some decision tree algorithms, such as CART, produce only binary splits by considering all $2^{k-1} - 1$ ways of creating a binary partition of k attribute values.

Ordinal Attributes

- Ordinal attributes can also produce binary or multi-way splits.
- Ordinal attribute values can be grouped as long as the grouping does not violate the order property of the attribute values.
- Following figure illustrates various ways of splitting training records based on the Shirt Size attribute.

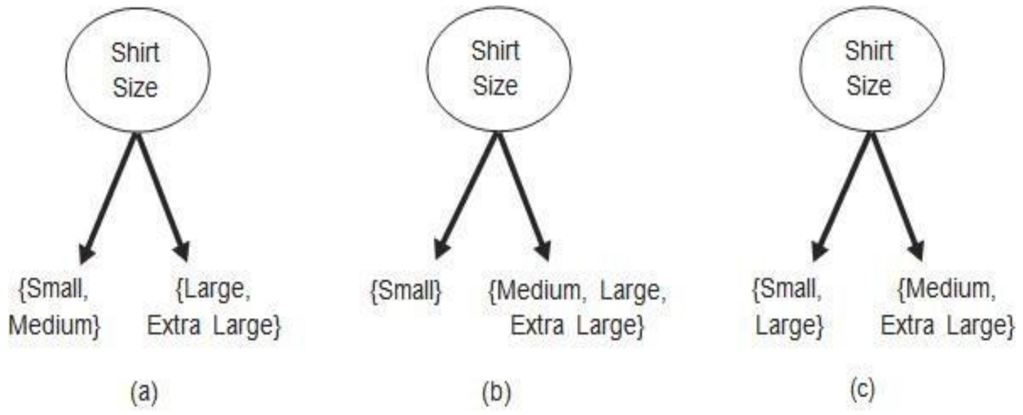


Figure 4.10. Different ways of grouping ordinal attribute values.

The groupings shown in Figures 4.10(a) and (b) preserve the order among the attribute values, where as the grouping shown in Figure 4.10(c) violates this property because it combines the attribute values Small and Large into the same partition while Medium and Extra Large are combined into another partition

Continuous Attributes

- For continuous attributes, the test condition can be expressed as a comparison test ($A<v$) or ($A \geq v$) with binary outcomes, or arrange query with outcomes of the form $vi \leq A < vi+1$,for $i=1,\ldots,k$
- For the binary case, the decision tree algorithm must consider all possible split positions v, and it selects the one that produces the best partition.
- For the multi-way split, the algorithm must consider all possible ranges ofcontinuous values

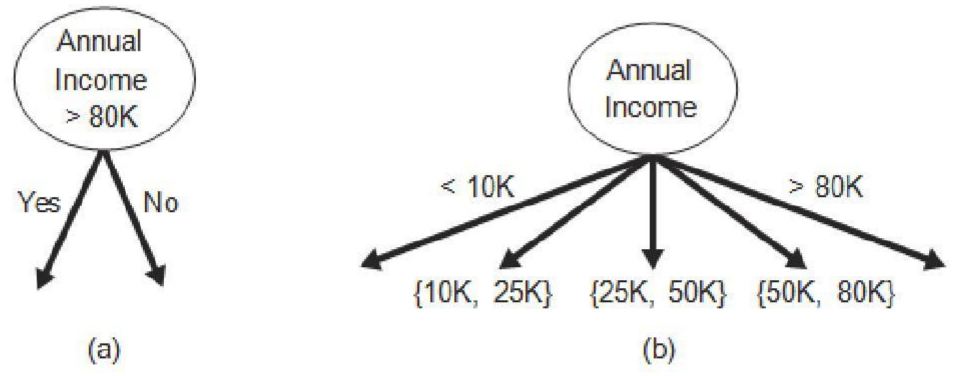


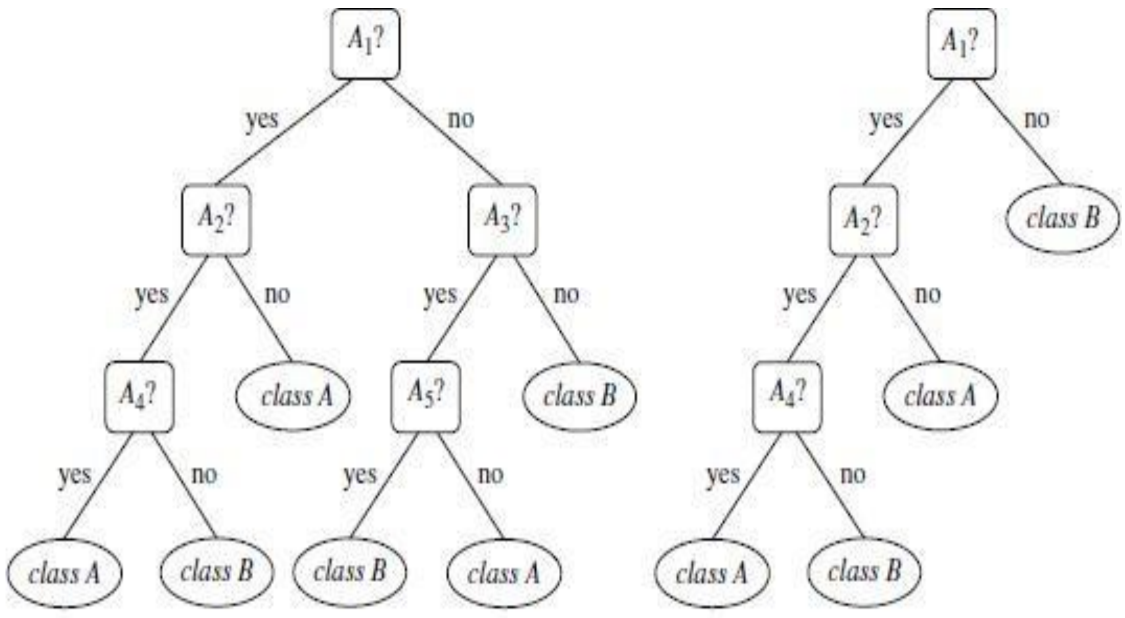
Figure 4.11. Test condition for continuous attributes.

Tree Pruning

- **Tree pruning** algorithms attempt to improve accuracy by removing tree branches reflecting noise in the data.
- Pruned trees tend to be smaller and less complex and, thus, easier to

comprehend. They are usually faster and better at correctly classifying independent test data (i.e., of previously unseen tuples) than unpruned trees.

- There are two common approaches to tree pruning: *prepruning* and
- *postpruning*.
- In the **prepruning** approach, a tree is "pruned" by halting its constructionearly (e.g., 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 or the probability distribution of those tuples. When constructing a tree, measures such as statistical significance, information gain, Gini index, and so on, can be used to assess the goodness of a split. If partitioning thetuples at a node would result in a split that falls below a pre specified threshold, then further partitioning of the given subset is halted.
- The second and more common approach is **postpruning**, which removes subtrees from a "fully grown" tree. A subtree at a given node is pruned byremoving its branches and replacing it with a leaf. The leaf is labeled withthe most frequent class among the subtree being replaced.



An unpruned decision tree and a pruned version of it.

Bayesian Classifier

- Bayesian Classifier is an approach for modeling probabilistic relationshipsbetween the attribute set and the class variable.
- Bayesian Classifier is a statistical classifier.
- It is based on Bayes theorem given by Thomas Bayes.
- There are two implementations of Bayesian classification:
    - **Naïve –Bayes Classifier(NBC):**
        - Class Conditional Independence: Effect of an attribute value on a given class is independent of the values of other attributes
        - -Simplifies Computations

    Bayesian Belief Networks (BBN)
    - -Graphical models
    - -Represent dependencies among subsets of attribute

- **Bayes Theorem:**

  Let X and Y be a pair of random variables. Their joint probability, P(X = x, Y = y), refers to the probability that variable X will take on the value x and variable Y will take on the value y.

- A conditional probability is the probability that a random variable will take on a particular value given that the outcome for another random variable is known.
- For example, the conditional probability P(Y = y|X = x) refers to the probability that the variable Y will take on the value y, given that the variable X is observed to have the value x.
- The joint and conditional probabilities for X and Y are related in the following way:
    - **P(X, Y) = P(Y |X) × P(X) = P (X|Y) × P(Y)**
  - Rearranging the last two expressions in above Equation leads to thefollowing formula, known as the Bayes theorem:

- Where P(X/Y): Y-conditional probability or Posterior
    - probability of X conditioned on Y P(Y/X): X-conditional probability or Posterior
    - probability of Y conditioned on XP(X): Prior probability of X
  - P(Y): Prior probability of Y

Example:

- Suppose we might be interested in finding out a patient's probability of having liver disease if they are an alcoholic. "Being an alcoholic" is the test (kind of like a litmus test) for liver disease.
- A could mean the event "Patient has liver disease." Past data tells you that 10% of patients entering your clinic have liver disease. P(A) = 0.10.
- B could mean the litmus test that "Patient is an alcoholic." Five percent ofthe clinic's patients are alcoholics. P(B) = 0.05.
- We might also know that among those patients diagnosed with liver disease, 7% are alcoholics. This is your B|A: the probability that a patient is alcoholic, given that they have liver disease, is 7%.
- Bayes' theorem tells :

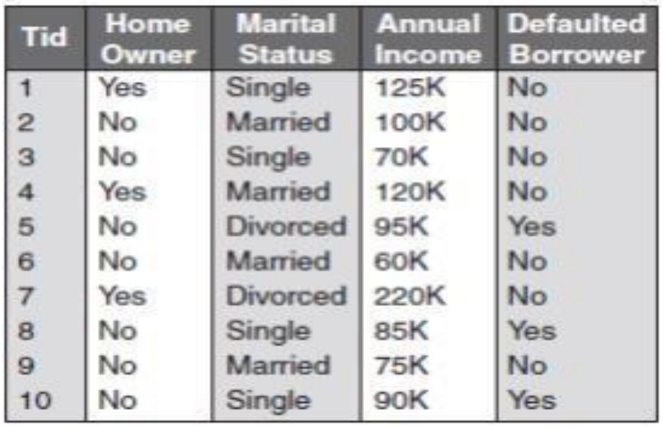
$$\mathbf{P(A|B)} = \frac{P(B / A) * P(A)}{P(B)} = \frac{0.07 * 0.1}{0.05} = \mathbf{0.14}$$

In other words, if the patient is an alcoholic, their chances of having liver disease are 0.14 (14%). This is a large increase from the 10% suggested by past data. But it's still unlikely that any particular patient has liver disease.

Using the Bayes Theorem for Classification

- Let X denotes the attribute set and Y denote the class variable.
- During the training phase, we need to learn the posterior probabilities P(Y|X) for every combination of X and Y based on information gathered from the training data.
- Test record X′ can be classified by finding the class Y ′ that maximizes the posterior probability, P(Y ′|X′).

Example: Following table shows a training set with the following attributes: Home Owner, Marital Status, and Annual Income. Loan borrowers who defaulted on their payments are classified as Yes, while those who repaid their loans are classified as No



Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Suppose we are given a test record with the following attribute set:
  - **X = (Home Owner = No, Marital Status = Married, Annual Income = $120K).**
  - To classify the record, we need to compute the posterior probabilities P(Yes|X) and P(No|X) based on information available

in the training data. If P(Yes|X) > P(No|X), then the record is classified as Yes; otherwise, it is classified as No

- The Bayes theorem is useful because it allows us to express the posterior probability in terms of the prior probability P(Y), the class-conditional probability P(X|Y), and the evidence, P(X):
  - The prior probability P(Y) can be easily estimated from the training set by computing the fraction of training records that belong to each class.

Naive Bayes Classifier

(Simple Bayesian classifier)

- A naive Bayes classifier estimates the class-conditional probability by assuming that the attributes are conditionally independent, given the classlabel y.
- The conditional independence assumption can be formally stated as follows:

$$P(Y|\mathbf{X}) = \frac{P(Y)\prod_{i=1}^{d} P(X_i|Y)}{P(\mathbf{X})}.$$

  - where each attribute set X = {X1,X2, . . . ,Xd} consists of d attributes.

Conditional Independence

- Let X, Y, and Z denote three sets of random variables. The variables in X are said to be conditionally independent of Y, given Z, if the following condition holds: P(X|Y,Z) = P(X|Z).
- An example of conditional independence is the relationship between a person's arm length and his or her reading skills. One might observe that people with longer arms tend to have higher levels of reading skills. This relationship can be explained by the presence of a confounding factor, which is age. A young child tends to have short arms and lacks the reading skills of an adult. If the age of a person is fixed, then the observedrelationship between arm length and reading skills disappears. Thus, we can conclude that arm length and reading skills are conditionally independent when the age variable is fixed

How a Naive Bayes Classifier Works

(***Algorithm for NBC***)

To classify a test record X , Let X = {X1,X2, . . . , Xd} denotes the attribute set and Y denotes the class variable in given data set

1. Compute the prior probabilities ,P(Y) of each of the class i.e.; P(Yes)and P(No) for binary problem.
2. Compute class-conditional probabilities ,P(X/Y) of each of the classbased on information available in the training data set using:

   P(X/Y) = $\prod_{i=1}^{d} P(X_i|Y)$. = P(X1/Y) × P(X2/Y) × … × P(Xd/Y)

5. Compute the posterior probabilities P(Y/X) for each of the class i.e.P(Yes/X) and P(No/X) for binary problems using:

$$P(Y|\mathbf{X}) = \frac{P(\mathbf{X}|Y) \times P(Y)}{P(\mathbf{X})}.$$

3. If P(Yes/X) > P(No/X) then the given record is classified as Yes; otherwise it is classified as No

Example1: We want to classify a record X={Red ,SUV, Domestic} for given following data set

Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

Step 1:
P(Stolen=Yes) = 5/10= 0.5 and
P(Stolen=No) = 5/10 = 0.5

Step 2:
P(X/Yes) = P(Red/Yes) * P(Domestic/Yes) * P(SUV/Yes)
= 3/5 *2/5 *1/5 = 0.048
P(X/No) = P(Red/No) * P(Domestic/No) * P(SUV/No)
= 2/5 *3/5 *3/5 = 0.144

Step 3:

$$\mathbf{P(Yes/X)} = \frac{P(X / Yes) \times P(Yes)}{P(X)} = \frac{0.024 \times 0.5}{P(X)} = \frac{0.012}{P(X)}$$

$$\mathbf{P(No/X)} = \frac{P(X / No) \times P(No)}{P(X)} = \frac{0.072 \times 0.5}{P(X)} = \frac{0.036}{P(X)}$$

Step 4:
Since P(No/X) > P(Yes/X) the given record is classified as **No**

Example 2: We want to classify a record following record in given data set

X= $\langle Outlk = sun, Temp = cool, Humid = high, Wind = strong \rangle$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Solution:

Step1:

P(Yes) = 9/ 14 and P(No) = 5 /14

Step 2:

P(Outlook=Sunny |Yes)=2/9

P(Temperature=Cool |Yes)=3/9

P(Humidity=High |Yes)=3/9

P(Wind = Strong |Yes) = 3 /9

P(Outlook=Sunny |No)=3/5

P(Temperature=Cool |No)=1/5

P(Humidity=High |No)=4/5

P(Wind = Strong |No) = 3/5

P(X |Yes) = P(Outlook=Sunny |Yes) ×P(Temperature=Cool |Yes) × P(Humidity=High |Yes)× P(Wind = Strong |Yes)
= 2/9 × 3/9 × 3/9 ×3/9 = 0.0082

P(X |No) = P(Outlook=Sunny |No) ×P(Temperature=Cool |No) × P(Humidity=High |No)× P(Wind = Strong |No)
= 3/5 × 1/5× 4/5 × 3/5 = 0.0576

Step 3:

$$\mathbf{P\ (Yes\ |X)} = \frac{P(X / Yes) \times P(Yes)}{P(X)} = \frac{0.0082 \times (9/14)}{P(X)} = \frac{0.0053}{P(X)}$$

$$\mathbf{P\ (No\ |X)} = \frac{P(X / No) \times P(No)}{P(X)} = \frac{0.0576 \times (5/14)}{P(X)} = \frac{0.0206}{P(X)}$$

Step 4:

Since P (No/X) > P (Yes/X) the given record is classified as **No**

Example 3: We wish to predict the class label of a tuple using naive Bayesian classification, given the following training data.

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

→ The tuple we wish to classify is

$$X = (age = youth,\ income = medium,\ student = yes,\ credit\_rating = fair)$$

Step 1:

$$P(buys\_computer = yes) = 9/14 = 0.643$$
$$P(buys\_computer = no) = 5/14 = 0.357$$

Step 2:

$$P(age = youth \mid buys\_computer = yes) = 2/9 = 0.222$$
$$P(age = youth \mid buys\_computer = no) = 3/5 = 0.600$$
$$P(income = medium \mid buys\_computer = yes) = 4/9 = 0.444$$
$$P(income = medium \mid buys\_computer = no) = 2/5 = 0.400$$
$$P(student = yes \mid buys\_computer = yes) = 6/9 = 0.667$$
$$P(student = yes \mid buys\_computer = no) = 1/5 = 0.200$$
$$P(credit\_rating = fair \mid buys\_computer = yes) = 6/9 = 0.667$$
$$P(credit\_rating = fair \mid buys\_computer = no) = 2/5 = 0.400$$

$$\begin{aligned} P(X \mid buys\_computer = yes) = {} & P(age = youth \mid buys\_computer = yes) \\ & \times P(income = medium \mid buys\_computer = yes) \\ & \times P(student = yes \mid buys\_computer = yes) \\ & \times P(credit\_rating = fair \mid buys\_computer = yes) \\ = {} & 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044. \end{aligned}$$

Similarly,

$$P(X \mid buys\_computer = no) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.$$

Step 3:

$$P(\text{buys-computer} = \text{yes}|X) = \frac{P(X \mid buys-computer = yes) \times P(buys-computer = yes)}{P(X)}$$

$$= \frac{0.044 \times .0.643}{P(X)} = \frac{0.028}{P(X)}$$

$$P(\text{buys-computer} = \text{no}|X) = \frac{P(X \mid buys-computer = no) \times P(buys-computer = no)}{P(X)}$$

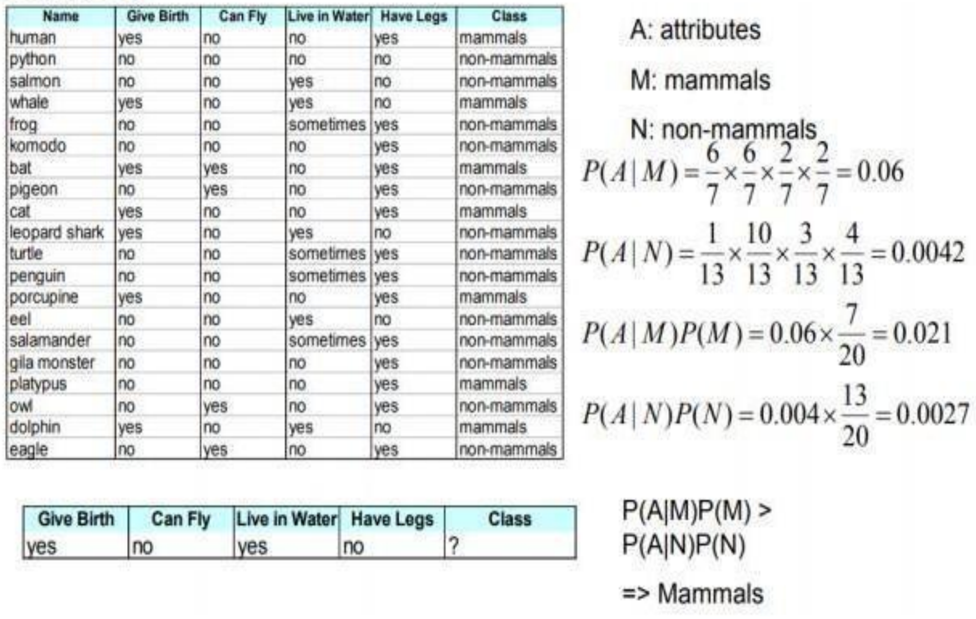
$$= \frac{0.019 \times .0.357}{P(X)} = \frac{0.007}{P(X)}$$

Step 4:

Since

P(buys-computer =no|X) > P(buys-computer =yes|X) the given record is classified as **NO**

Example 4:



Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A \mid M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A \mid N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A \mid M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A \mid N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

P(A|M)P(M) > P(A|N)P(N)

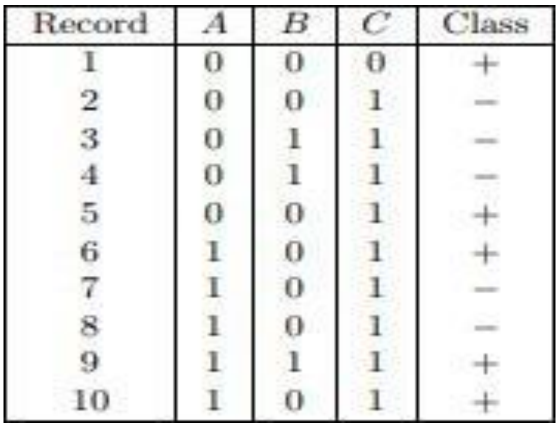
=> Mammals

Characteristics of NBC:

- Robust to isolated noise points.
- Handle missing values by ignoring the instance during probabilityestimate calculations.
- Robust to irrelevant attributes.
- Independence assumption may not hold for some attributes

Exercise:

Consider the data set shown below



Record	A	B	C	Class
1	0	0	0	+
2	0	0	1	−
3	0	1	1	−
4	0	1	1	−
5	0	0	1	+
6	1	0	1	+
7	1	0	1	−
8	1	0	1	−
9	1	1	1	+
10	1	0	1	+

Predict the class label for a test sample
(A = 0, B = 1, C = 0) using the Naıve Bayes approach.
Ans: Let X= (A = 0, B = 1, C = 0)
Step 1:
P(+)=5/10=0.5
P(-)=5/10=0.5

Step 2:
P(A=0|+) = 2/5=0.4
P(B=1|+) = 1/5=0.2
P(C=0|+) = 1/5=0.2
P(X|+) = 0.4 * 0.2 *0.2=0.016

P(A=0|-) = 3/5=0.6
P(B=1|-) = 2/5=0.4
P(C=0|-) = 0/5=0
P(X|-) = 0.6 * 0.4 *0=0

Step 3:
P(+|X)=P(X|+)*P(+)/P(X)
=0.016*0.5/P(X)
=0.08/P(X)

P(-|X)=P(X|-)*P(-)/P(X)
=0 * 0.5/P(X)=0

Step 4:Since P(+|X)>P(-|X)
The class label should be '+'

Bayesian Belief Networks (BBN)

- This approach is a more flexible approach for modeling the class conditional probabilities P(X|Y).
- Instead of requiring all the attributes to be conditionally independent given the class, this approach allows us to specify which pair of attributesare conditionally independent.

Model Representation:

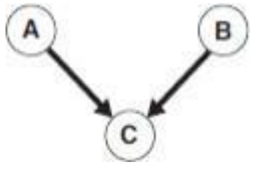
A Bayesian belief network (BBN), or simply, Bayesian network, providesa graphical representation of the probabilistic relationships among a set ofrandom variables.

There are two key elements of a Bayesian network:

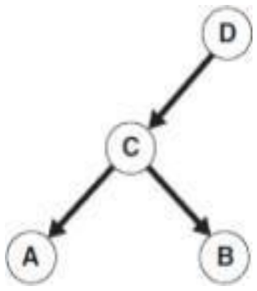
1. A directed acyclic graph (dag) encoding the dependence relationshipsamong a set of variables.
2. A probability table associating each node to its immediate parentnodes.

Example:

- Consider three random variables, A, B, and C, in which A and B are independent variables and each has a direct influence on a third variable, C.
- The relationships among the variables can be summarized into the directed acyclic graph shown in following figure.



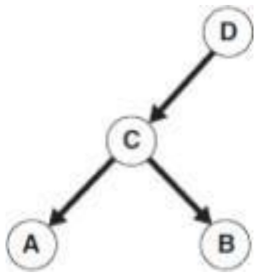
- Each node in the graph represents a variable, and each arc asserts thedependence relationship between the pair of variables.
- If there is a directed arc from X to Y, then X is the parent of Y and Y is the child of X.
- If there is a directed path in the network from X to Z, then X is an ancestor of Z, while Z is a descendant of X.
- For example, in the diagram shown in following figure, A is a



descendant of D and D is an ancestor of B. Both B and D are also non-descendants of A.

An important property of the Bayesian network can be stated as follows: Property: (Conditional Independence):A node in a Bayesian network isconditionally independent of its non-descendants, if its parents are known.

In the diagram shown in below figure, A is conditionally independent of both B and D given C because the nodes for B and D are non-descendantsof node A.



Each node in BBN is also associated with a probability table.

1. If a node X does not have any parents, then the table contains onlythe prior probability P(X).
2. If a node X has only one parent, Y, then the table contains theConditional probability P(X|Y).
3. If a node X has multiple parents, {Y1, Y2, . . . , Yk}, then the tablecontains the conditional probability P(X|Y1, Y2, . . . , Yk).

Example: A Bayesian network for modeling patients with heart disease or heartburn problems:

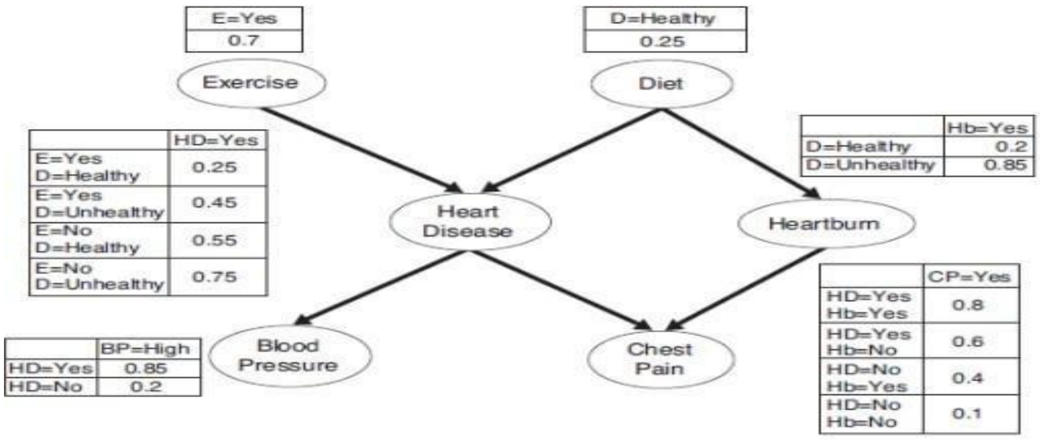


Figure 5.13. A Bayesian belief network for detecting heart disease and heartburn in patients.

The omitted probabilities can be recovered by noting that

P(X = x') = 1 − P(X = x) and P(X = x '|Y) = 1 − P(X = x|Y),

where x' denotes the opposite outcome of x.

For example, the conditional probability

$$
\begin{aligned}
&P(\text{Heart Disease} = \text{No}|\text{Exercise} = \text{No}, \text{Diet} = \text{Healthy})\\
=\ & 1 - P(\text{Heart Disease} = \text{Yes}|\text{Exercise} = \text{No}, \text{Diet} = \text{Healthy})\\
=\ & 1 - 0.55 = 0.45.
\end{aligned}
$$

Model Building:

- Model building in Bayesian networks involves two steps:
- Creating the structure of the network, and
- Estimating the probability values in the tables associated witheach node.
- The network topology can be obtained by encoding the subjectiveknowledge of domain experts.
- Following algorithm presents a systematic procedure for inducing thetopology of a Bayesian network.

Algorithm 5.3 Algorithm for generating the topology of a Bayesian network.

```
1: Let $T = (X_1, X_2, \ldots, X_d)$ denote a total order of the variables.
2: for $j = 1$ to $d$ do
3:     Let $X_{T(j)}$ denote the $j^{th}$ highest order variable in $T$.
4:     Let $\pi(X_{T(j)}) = \{X_{T(1)}, X_{T(2)}, \ldots, X_{T(j-1)}\}$ denote the set of variables preceding $X_{T(j)}$.
5:     Remove the variables from $\pi(X_{T(j)})$ that do not affect $X_j$ (using prior knowledge).
6:     Create an arc between $X_{T(j)}$ and the remaining variables in $\pi(X_{T(j)})$.
7: end for
```

Example:

Example 5.4. Consider the variables shown in Figure 5.13. After performing Step 1, let us assume that the variables are ordered in the following way: (E, D, HD, Hb, CP, BP). From Steps 2 to 7, starting with variable D, we obtain the following conditional probabilities:

- $P(D|E)$ is simplified to $P(D)$.
- $P(HD|E, D)$ cannot be simplified.
- $P(Hb|HD, E, D)$ is simplified to $P(Hb|D)$.
- $P(CP|Hb, HD, E, D)$ is simplified to $P(CP|Hb, HD)$.
- $P(BP|CP, Hb, HD, E, D)$ is simplified to $P(BP|HD)$.

Based on these conditional probabilities, we can create arcs between the nodes (E, HD), (D, HD), (D, Hb), (HD, CP), (Hb, CP), and (HD, BP). These arcs result in the network structure shown in Figure 5.13. ■

Characteristics of BBN

Following are some of the general characteristics of the BBN method:

1. BBN provides an approach for capturing the prior knowledge of a particular domain using a graphical model. The network can also be used to encode causal dependencies among variables.

2. Constructing the network can be time consuming and requires a large amount of effort. However, once the structure of the network has been determined, adding a new variable is quite straightforward.

3. Bayesian networks are well suited to dealing with incomplete data. Instances with missing attributes can be handled by summing or integrating the probabilities over all possible values of the attribute.

4. Because the data is combined probabilistically with prior knowledge, the method is quite robust to model overfitting.

The *k*-Nearest-Neighbor(k-NN) Classification

- In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors
- This method is labor intensive when given large training sets.
- Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuple with training tuples that are similar to it.
- The training tuples are described by *n* attributes. Each tuple represents a point in an *n*-dimensional space. In this way, all the training tuples are stored in an *n*-dimensional pattern space.
- When given an unknown tuple, a ***k*-nearest-neighbor classifier** searches the pattern space for the *k* training tuples that are closest to the unknown tuple. These *k* training tuples are the *k* nearest neighbors" of the unknowntuple.
- **Closeness** is defined in terms of a distance metric, such as Euclidean distance.
- Typically, we normalize the values of each attribute before finding closeness. Min-max normalization, for example, can be used to transform a value *v* of a numeric attribute *A* to *v* 0 in the range [0, 1].
- For *k*-nearest-neighbor classification, the unknown tuple is assigned the most common class among its *k*-nearest neighbors. When *k* = 1, the unknown tuple is assigned the class of the training tuple that is closest to it in pattern space.
- Nearest-neighbor classifiers can also be used for numeric prediction, that is, to return a real-valued prediction for a given unknown tuple. In this case, the classifier returns the average value of the real-valued labels associated with the *k*-nearest neighbors of the unknown tuple.
- Nearest-neighbor classifiers can be **extremely slow** when classifying test tuples.
- They can suffer from poor accuracy when given noisy or irrelevant attributes.
- A good value for k, the number of neighbours can be determined experimentally. Starting with *k* = 1, we use a test set to estimate the error rate of the classifier. This process can be repeated each time by incrementing *k* to allow for one more neighbor. The *k* value that gives the minimum error rate may be selected.

The following **two properties** would define KNN well:

- **Lazy learning algorithm** − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.

- **Non-parametric learning algorithm** − KNN is also a non-parametric learning algorithm because it doesn’t assume anything about the underlying data.

Working of KNN Algorithm

- KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).
- K-nearest neighbors (KNN) algorithm uses ‘feature similarity’ to predict the values of new data points which further means that the new data point will be assigned a value based on how closely it matches the points in the training set.

The KNN Algorithm

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data

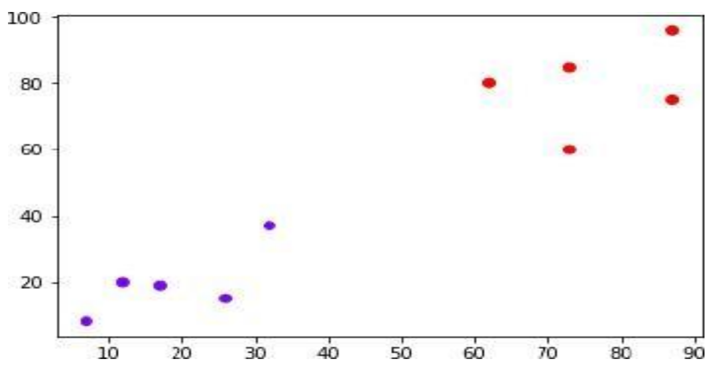
   3.1 Calculate the distance between the query example and the currentexample from the data.

   3.2 Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest tolargest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

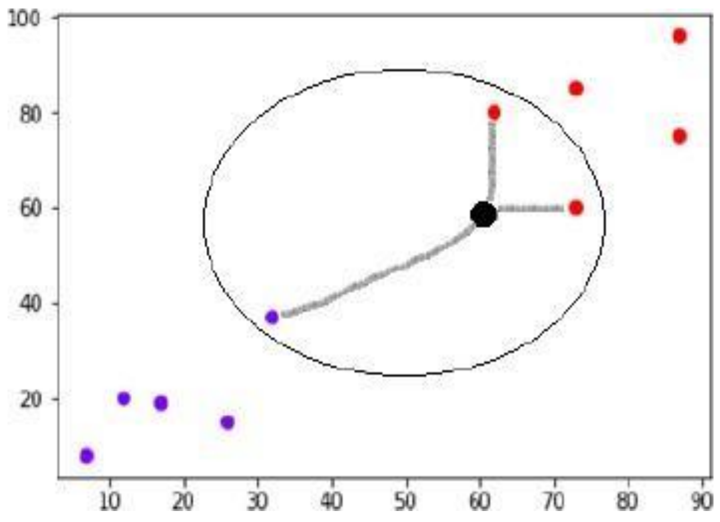
Example

The following is an example to understand the concept of K and workingof KNN algorithm:

Suppose we have a dataset which can be plotted as follows:



Now, we need to classify new data point with black dot (at point 60,60) into blue or red class. We are assuming K = 3 i.e. it would find three nearest data points. It is shown in the next diagram –



We can see in the above diagram the three nearest neighbors of the data point with black dot. Among those three, two of them lies in Red class hence the black dot will also be assigned in red class

Difference between eager learners and lazy learners:

- Decision tree and rule-based classifiers are examples of **eager learners** because they are designed to learn a model that maps the input attributes to the class label as soon as the training data becomes available. An opposite strategy would be to delay the process of modeling the training data until it is needed to classify the test examples. Techniques that employ this strategy are known as **lazy learners**
- Lazy learners such as nearest-neighbor classifiers do not require model building. However, classifying a test example can be quite expensive because we need to compute the proximity values individually between the test and training examples. In contrast, eager learners often spend the bulk of their computing resources for model building. Once a model has been built, classifying a test example is extremely fast.

Characteristics of Nearest-Neighbor Classifiers

Nearest-neighbor classification is part of a more general technique known as **instance-based learning**, which uses specific training instances to make predictions without having to maintain an abstraction (or model) derived from data. Instance-based learning algorithms require a proximity measure to determine the similarity or distance between instances and a classification function that returns the predicted class of a test instance based on its proximity to other instances.

Lazy learners such as nearest-neighbor classifiers do not require model building. However, classifying a test example can be quite expensive because we need to compute the proximity values individually between the test and training examples. In contrast, eager learners often spend the bulk of their computing resources for model building. Once a model has been built, classifying a test example is extremely fast

Nearest-neighbor classifiers can produce wrong predictions unless the appropriate proximity measure and data preprocessing steps are taken.

Pros and Cons of KNN

Pros

- It is very simple algorithm to understand and interpret.
- It is very useful for nonlinear data because there is no assumptionabout data in this algorithm.
- It is a versatile algorithm as we can use it for classification as well asregression.
- It has relatively high accuracy but there are much better supervisedlearning models than KNN.

Cons

- It is computationally a bit expensive algorithm because it stores all thetraining data.
- High memory storage required as compared to other supervisedlearning algorithms.
- Prediction is slow in case of big N.
- It is very sensitive to the scale of data as well as irrelevant features.