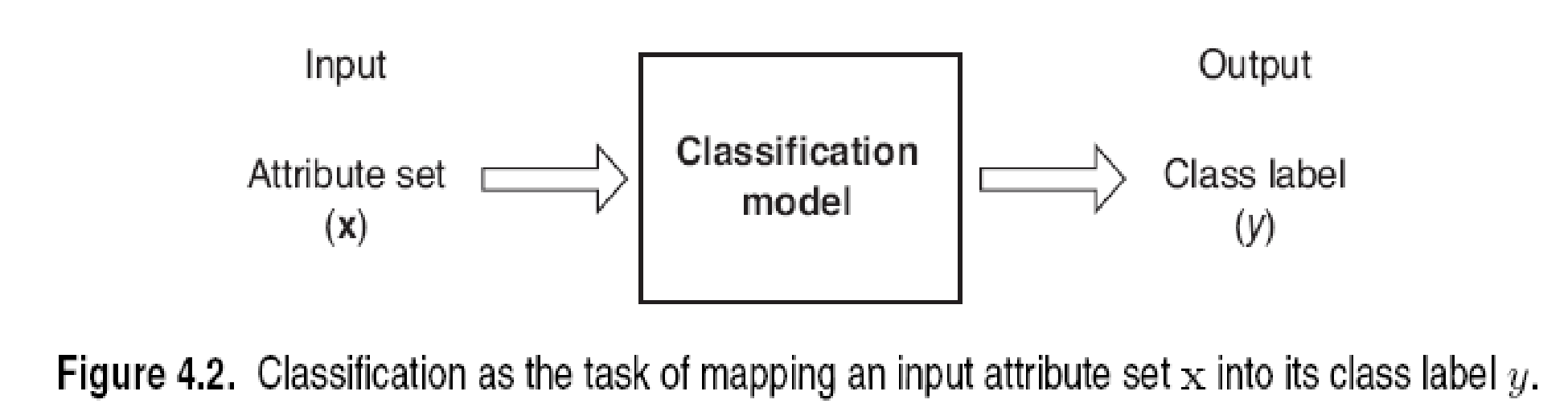
**UNIT 3: CLASSIFICATION**

|  |
| --- |
| Basic Concepts |
| General approach to solving a classification problem |
| Decision Tree induction:  --Working of decision tree  --Building a decision tree  -- Methods for expressing attribute test conditions  -- Measures for selecting the best split  -- Algorithm for decision tree induction |
| Model over fitting:  -- Due to presence of noise  -- Due to lack of representation samples |
| Evaluating the performance of classifier:  --Holdout method  --Random sub sampling  --Cross-validation  --Bootstrap |

**🡪Classification: Basic concepts:**

**🡪Definition:**

Classification is the task of learning a target function ‘**f**’ that maps each attribute set ‘**x**’ to one of the predefined class label ‘**y**’.



In this attribute set ‘**x’** can be any number of attributes and the attributes can be binary, categorical and continuous. The class label ‘**y**’ must be a discrete attribute; i.e., either binary or categorical (nominal or ordinal).

**🡪Classification models:**

**--Descriptive modeling** is a classification model used for summarizing the data.

**--Predictive modeling** is a classification model used to predict the class label of unknown records.

**🡪Applications:**

1. Detecting spam email messages based upon the message header and content.
2. Classifying galaxies based upon their shapes.
3. Classifying the Students based on their Grades.
4. Classifying the Patients according to their Medical records.
5. Classification can be used in credit approval.

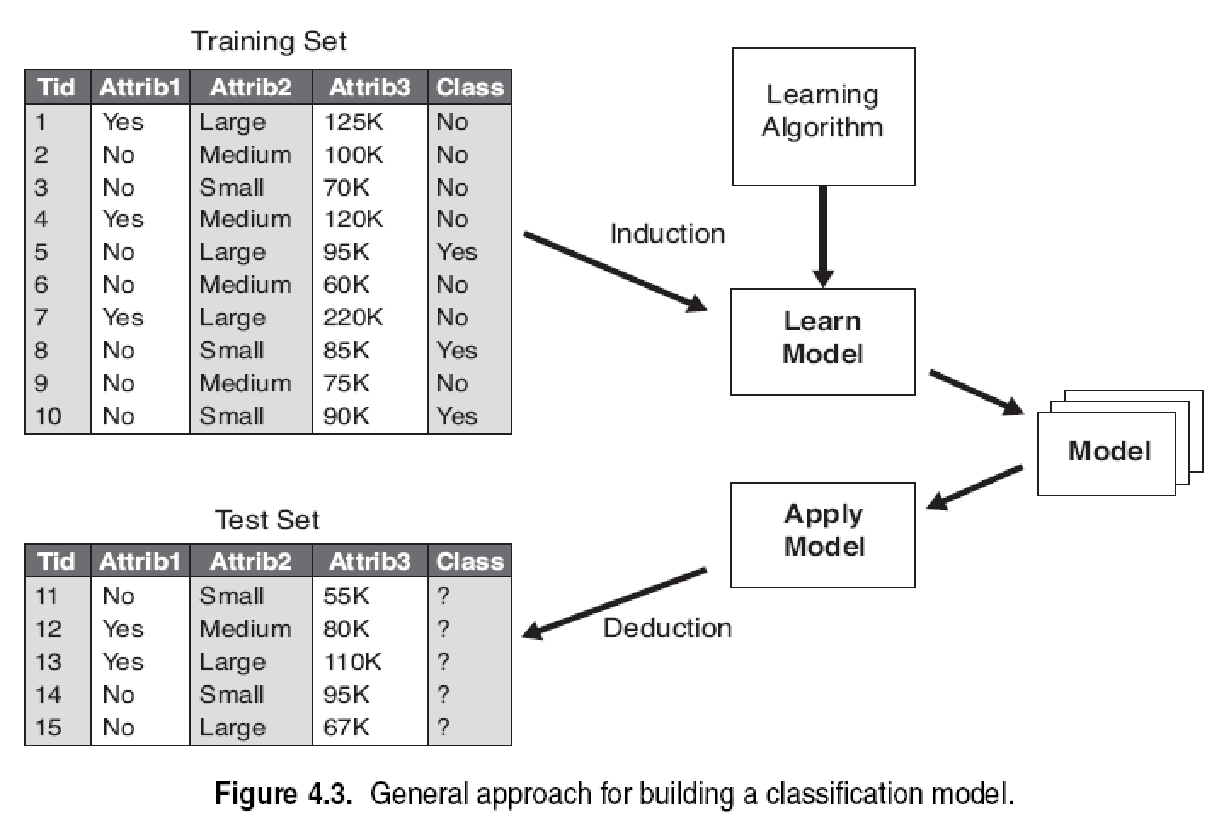
**🡪General approach to solve a classification problem:**

--A classification technique is a systematic approach to build classification models based on a data set.

**--**Examples are decision tree classifiers, rule-based classifiers, neural networks, support vector machines and naïve Bayes classifier.

--Each technique employs a learning algorithm to identify a model that best fits the relationship between the attribute set and the class label of the input data.

--A **training set** consists of records whose class labels are known must be provided. The training test is used to build a classification model, which is applied to the test set. The **test set** consists of records whose class label is unknown



--Evaluation of the performance of a classification model is based on the counts of test records correctly and incorrectly predicted by the model.

--These counts are tabulated in a table known as **confusion matrix.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Predicted Class | |
|  | | Class 1 | Class 0 |
| Actual Class | Class=1 | f11 | f10 |
| Class=0 | f01 | f00 |

**--**Each entry **fij**  in the table denotes the number of records from the class ‘**i**’ predicted to be of class ‘**j**’.

--For example, f01 refers to the number of records from class 0 incorrectly predicted as class 1.

--Based on the entries in the confusion matrix, the total number of correct predictions made by the model is (f11+f00) and the total number of incorrect predictions is (f01+f10).

--Although a confusion matrix provides the information needed to determine how well a classification model performs, summarizing this information with a single number would make it more convenient to compare the performance of different models.

--This can be done using a **performance metric.**

**--Accuracy** can be expresses as:

Accuracy= Number of correct predictions/ Total number of predictions

. Accuracy= (f11+f00)/(f11+f10+f00+f01)

--Equivalently, **Error rate** can be expresses as:

**Error rate**=Number of wrong predictions/ Total number of predictions

. **Error rate** = (f10+f01)/(f11+f10+f00+f01)

**🡪Decision Tree Induction:** Decision tree induction is a technique used for identifying unknown class labels in classification. The topics are:

--Working of decision tree

--Building a decision tree

--Methods for expressing attribute test conditions

--Measures for selecting the best split

--Algorithm for decision tree induction

**🡪Working of a decision tree:**

The tree has three types of nodes.

1. A **root node** has no incoming edges and zero or more outgoing edges.
2. **Internal nodes**, each of which has exactly one incoming edge and two or more outgoing edges.
3. **Leaf** or **terminal nodes**, each of which has exactly one incoming edge and no outgoing edges.

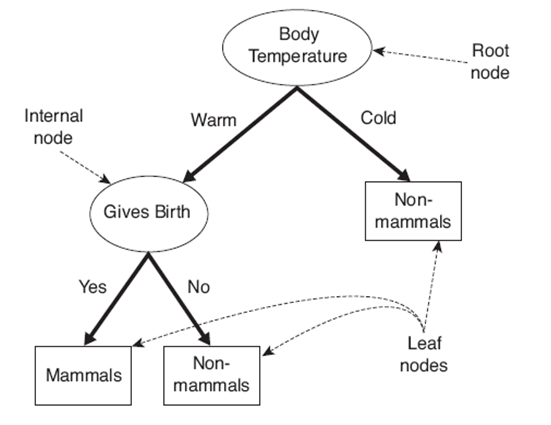


Fig: A decision tree for mammal classification problem

In this example, we are classifying whether vertebrate is a mammal or non-mammal. From this decision tree, we can identify a new vertebrate as mammal or non-mammal. If the vertebrate is cold-blooded, then it is a non-mammal. If the vertebrate is warm-blooded, then check the next node gives berth. If it gives berth, then it is a mammal, else, non-mammal.

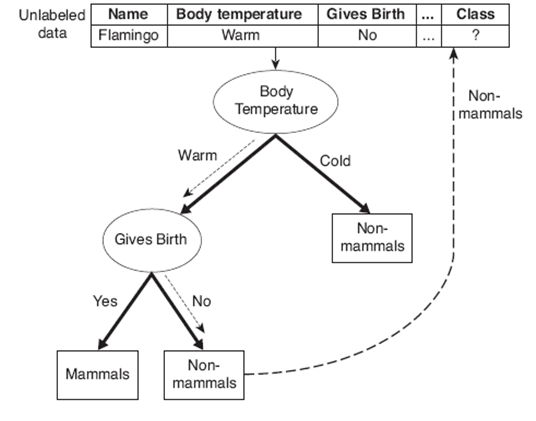


Fig: Classifying an unlabelled vertebrate

**🡪Building of a decision tree:**

--There are various algorithms devised for constructing a decision tree. They are:

1. Hunt’s algorithm
2. ID3 (Iterative Dichotomiser 3)
3. C4.5 (Classification 4.5)
4. CART (Classification Algorithm and Regression Tree)

--These algorithms usually employ a greedy strategy that grows a decision tree by making a series of locally optimum decisions about which attribute to use for partitioning the data. One such algorithm is **Hunts algorithm**.

**Hunt’s algorithm**

--In Hunt’s algorithm, a decision tree is grown in a recursive fashion by partitioning the training records into subsets.

--Let Dt be a set of training records that are associated with node t and y={y1,y2,…,yc} be the class labels.

--The recursive procedure for hunt’s algorithm is as follows:

**STEP 1**

If all the records in Dt belong to same class yt, then t is a leaf node labeled as yt.

**STEP 2**

If Dt contains records that belong to more than one class, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome and the records in Dt are distributed based on the outcomes. The algorithm is then recursively applied for each node.

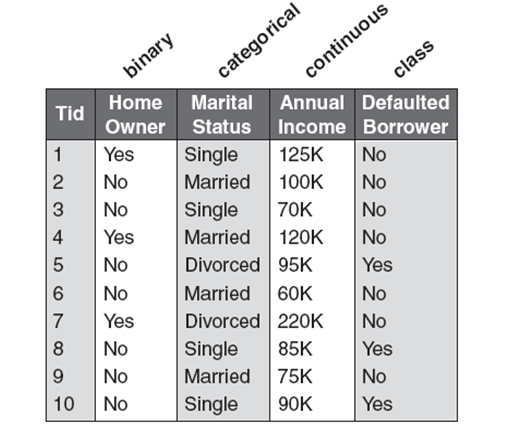
****

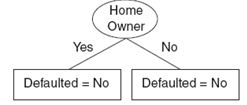
Fig: Training set for predicting borrowers who will default on loan payments

--In the above data set, the class labels for all the 10 records are not same, so step 1 cannot be satisfied. We need to construct the decision tree using step 2.

--The class label has maximum number of records with “no”. So, label the node as follows:

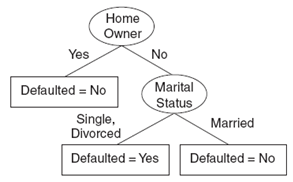


--Select one of the attribute as root node, say, home owner since home owner with entry “yes” need not require any further splitting. There are 3 records with home owner =yes and records with home owner=no.



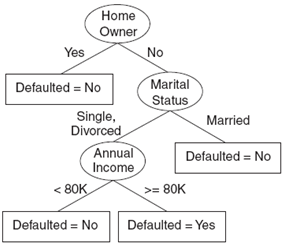
--The records with home owner=yes are classified and we now need to classify other 7 records i.e., home owner=no. The attribute test condition can be applied either on marital status or annual income.

--Let us select marital status, where we apply binary split. Here marital status=married need not require further splitting.



--The records with marital status=married are classified and we now need to classify other 4 records i.e., home owner=no and marital status=single, divorced.

--The left out attribute is annual income. Here we select the range since it is a continuous attribute.



--Now the other 4 records are also classified.

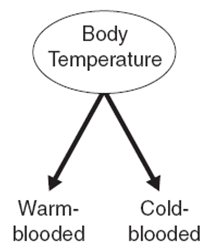
Additional conditions are needed to handle some special cases:

1. It is possible for some of the child nodes created in step 2 to be empty; i.e., there are no records associated with these nodes. In such cases assign the same class label as the majority class of training records associated with its parent node; i.e., in our example majority class is no, so assign ‘no’ for the new record.
2. If all the records in Dt have identical attribute values but the class label is different in such cases, assign the majority class label.

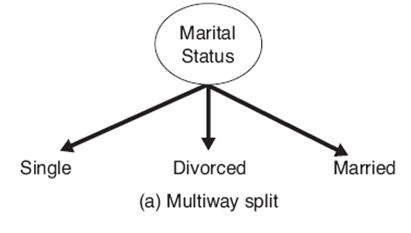
**🡪Methods for expressing attribute test conditions:**

The following are the methods for expressing attribute test conditions. They are:

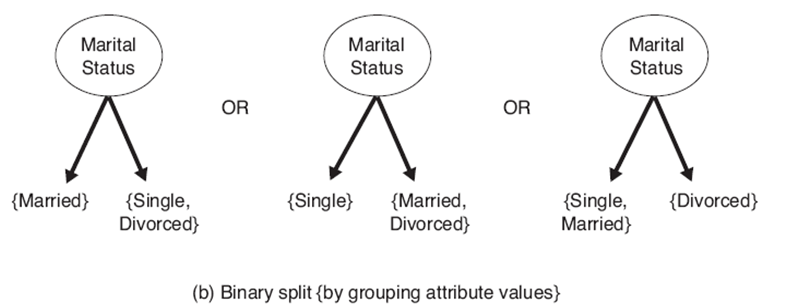
1. **Binary attribute:** The test condition for binary attribute generate two outcomes as shown below:

****

1. **Nominal attributes:** since a nominal attribute can have many values, its test condition can be expressed in two ways as shown below:

****

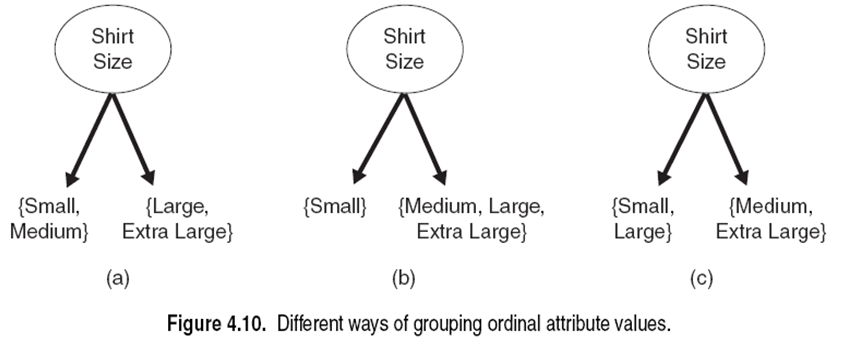
For a multi way split, the number of outcomes depends on the number of distinct values for the corresponding attribute.



Some algorithms, such as CART supports only binary splits. In such case we can partition the k-attribute values into 2k-1-1 ways.

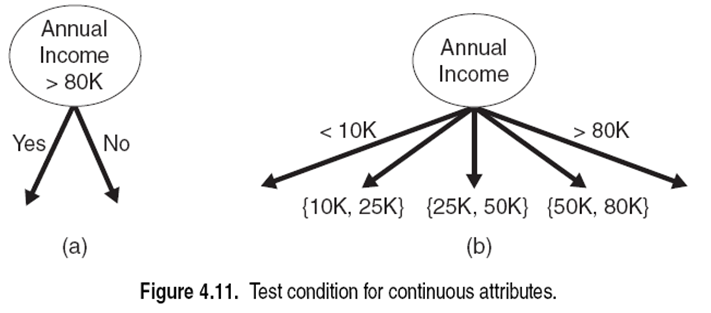
For example, marital status is a 3-attribute value, we can split it in 22-1-1; i.e., 3 ways.

1. **Ordinal attribute:** It 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.

****

In the above example, condition a and condition b satisfies order but condition c violates the order property.

1. **Continuous attributes:** The test condition can be expressed as a comparison test (A<v) or (A>=v) with binary outcomes, or a range query with outcomes of the form vi<=A<vi+1 for i=1,2,…,k.

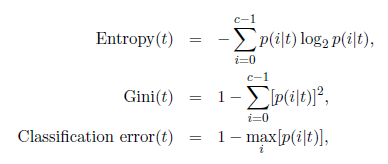
****

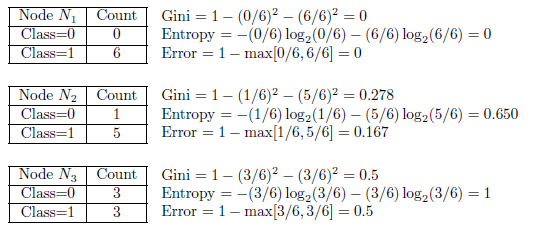
**🡪Measures for selecting the best split:**

There are many measures that can be used to determine the best way to split the records.

Let P(i|t) denote the fraction of records belonging to class i at a node t. the measures for selecting the best split are often based on the degree of impurity of the child nodes. The smaller the degree of impurity, the more skewed the class distribution. For example, a node with class distribution (0,1) has zero impurity, whereas a node with uniform class distribution (0.5,0.5) has the highest impurity.

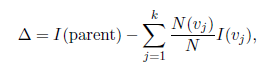
**Examples** of impurity measures include:





The 3 measures attain maximum values when the class distribution is uniform and minimum when all the records belong to same class.

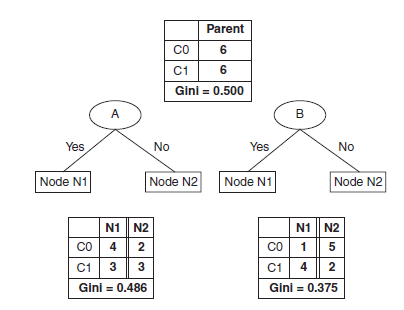
Compare the degree of impurity of the parent node with the degree of impurity of the child node. The larger their difference, the better the test condition. The gain, ∆, is a criterion that can be used to determine the goodness of a split.



Where I(.) is the impurity measure of a given node, N is the total number of records at the parent node, k is the attribute values and N(vj) is the number of records associated with node vj. when entropy is used as impurity measure the difference in entropy is known as information gain, ∆info.

**Splitting of binary attributes**

Suppose there are two ways to split the data into smaller subsets, say, A and B. before splitting the GINI index is 0.5 since there are equal number of records from both the classes.



For attribute A,

For node N1, the GINI index is 1-[(4/7)2+(3/7)2]=0.4898

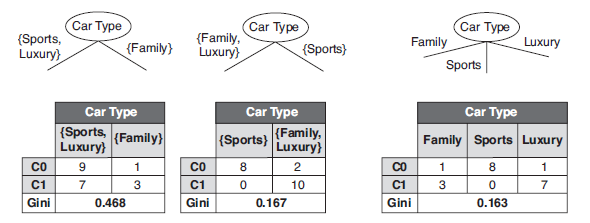
For node N2, the GINI index is 1-[(2/5)2+(3/5)2]=0.48

The average weighted GINI index is (7/12)(0.4898)+(5/12)(0.48)=0.486

For attribute B, the average weighted GINI index is 0.375, since the subsets for attribute B have smaller GINI index than A, attribute B is preferable.

**Splitting of nominal attributes**

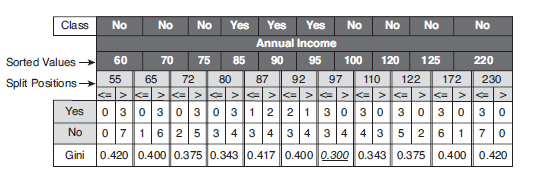
A nominal attribute can produce either binary or multi way split.



The computation of GINI index is same as for binary attributes. The smaller the average GINI index is the best split. In our example, multi way split has the lowest GINI index, so it is the best split.

**Splitting of continuous attributes**

In order to split a continuous attribute, we select a range.

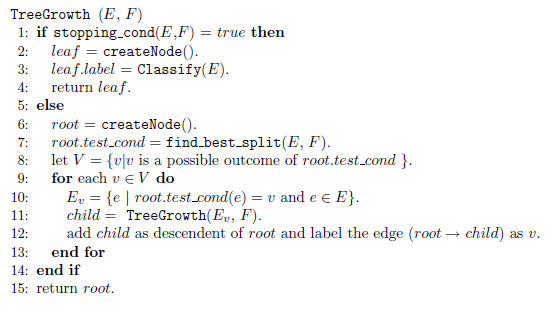


In our example, the **sorted values** represents the ascending order of distinct values in continuous attribute.

**Split positions** represent mean between two adjacent sorted values.

Calculate the GINI index for every split position and the smaller GINI index split position can be chosen as the range for continuous attribute

**🡪Algorithm for decision tree induction:**

****

1. The create node() function extends the decision tree by creating a new node. A node in the decision tree has either a test condition, denoted as node.test\_cond, or a class label, denoted as node.label.
2. The find.best\_split () function determines which attribute should be selected as the test condition for splitting the training records.
3. The classify() function determines the class label to be assigned to a leaf node.
4. The stopping\_cond() function is used to terminate the tree-growing process by testing whether all the records are classified or not.

**🡪Model Overfitting:**

--The errors committed by a classification model are generally divided into two types:

1. Training errors
2. Generalization errors

**--Training errors** is the number of misclassification errors committed on training records. For example, a record in test data is already existed in training data, but the class label is wrongly predicted. This type of errors is known as training errors.

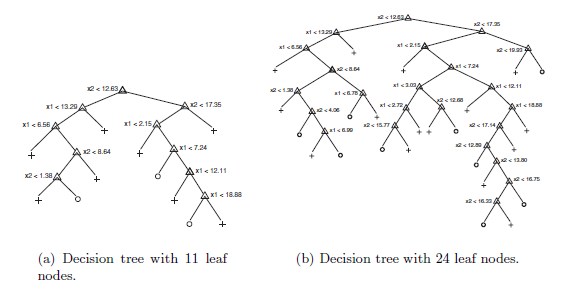
**--Generalization errors** is the expected error of the model. For example, the class label for the record in the test data is known but it is wrongly predicted. This type of errors is known as Generalization errors.

**--A good model should have low training errors as well as low testing errors.**

--The training and test error rates are large when the size of the tree is very small. This situation is known as **model underfitting**.

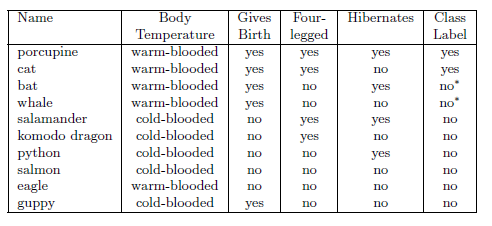
--When the tree becomes large, the test error rate increases and training error rate decreases. This situation is known as **model overfitting**.

--In the below two trees, the tree with less nodes has high training errors and less test errors.

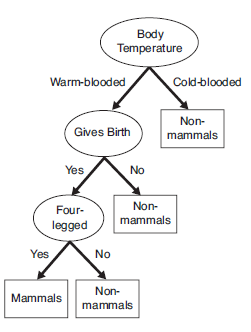


**🡪Overfitting due to presence of noise:**

--Consider the training and test sets for the mammal classification problem. Two of the ten records are mislabeled. Bats and whales are classified as non mammals instead of mammals.

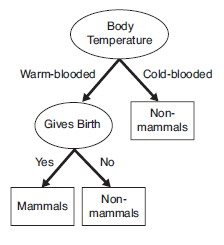


--The decision tree for the above data set is



--The class label for {name=’human’, body-temperature=’warm-blooded’, gives berth=’yes’, four-legged=’no’, hibernates=’no’} is non-mammals from above decision tree. But humans are mammals. The prediction is wrong due to presence of noise in data.

--So, change the class labels bat and whale. The decision tree is redrawn as follows:



--After removing the noise, the predictions are right.

**🡪Overfitting due to lack of representative samples:**

--If the numbers of records in training data set are less, then there are more test errors.

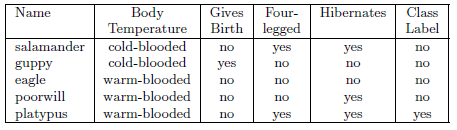


Fig: training data

--The decision tree for the above training data is as follows:

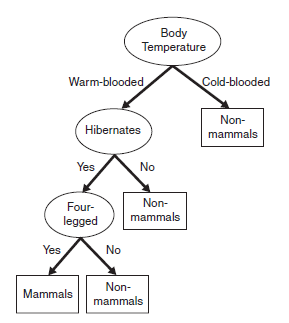


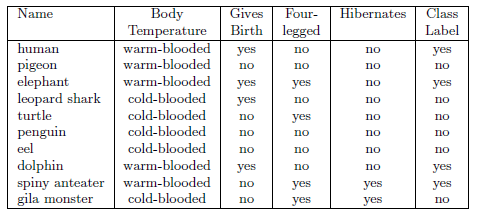
Fig: decision tree

Fig: test set

--From the above decision tree, humans, elephants and dolphins are misclassified since tree is constructed with less number of records.

**🡪Evaluating the performance of a classifier:**

**--**A classification algorithm should be judged before using it for real time data. The accuracy and error rate is judged by finding the class labels of test sets whose class labels are already known in advance.

--The following methods are used for evaluating the performance of a classifier:

--Holdout method

--Random sub sampling

--Cross-validation

--Bootstrap

**🡪Holdout Method**:

In this method the original data sets is divided into two parts, 50% or 2/3rd of original data is considered as training sets and another 50% or 1/3rd of original data as test sets respectively. Now, the classification model is trained on training tests and then applied on test sets. The performance of the classification algorithm is based on number of correct predictions made on the test set.

**Limitations**

1. Less number of samples for training( since the original samples are spitted)
2. The model is highly dependent on the composition of the training and test sets

**🡪Random Sampling:**

Multiple repetition of holdout method is known as random sampling. Here the original data is divided randomly into training sets and test sets and the accuracy is calculated as in holdout method. This random sampling is then repeated k times and the accuracy is calculated for each time. The overall accuracy is:



--Here acci is the model accuracyduring *i* th iteration

**Limitations**

1. Less number of samples for training( since the original samples are spitted)
2. A record may be used more than once in training and test tests.

**🡪Cross-Validation:**

--There are three variations of cross-validation approach

1. **Two fold cross validation**

In this approach data is partitioned into two parts. The first part is considered as training set and the second part as test set. Now they are swapped and the first part is considered as test set and second one as training set. The total error is the sum of both the errors.

1. **K-fold cross validation**

In this approach the data is partitioned into k subsets. One of the partitions is considered as test set and remaining sets are considered as training set. This process is repeated k times and the total error is the sum of all the k runs.

1. **Leave-one-out approach**

In this approach one record is considered as test set and rest of the samples are considered as training set. This process is repeated k times (k= number of records) and the total error is the sum of all the k runs. But this process is computationally very expensive.

**🡪Bootstrap**

In this approach a record may be sampled more than once. Means a record when sampled is again kept back in the original data. So it is likely that the record may be sampled again and again. Consider original data of size N. The probability of a record to be chosen as bootstrap sample is 1-(1-1/N)N .When the Size of N is very large then the probability is 1-e-1. The sampling is repeated B times to generate b bootstrap samples.