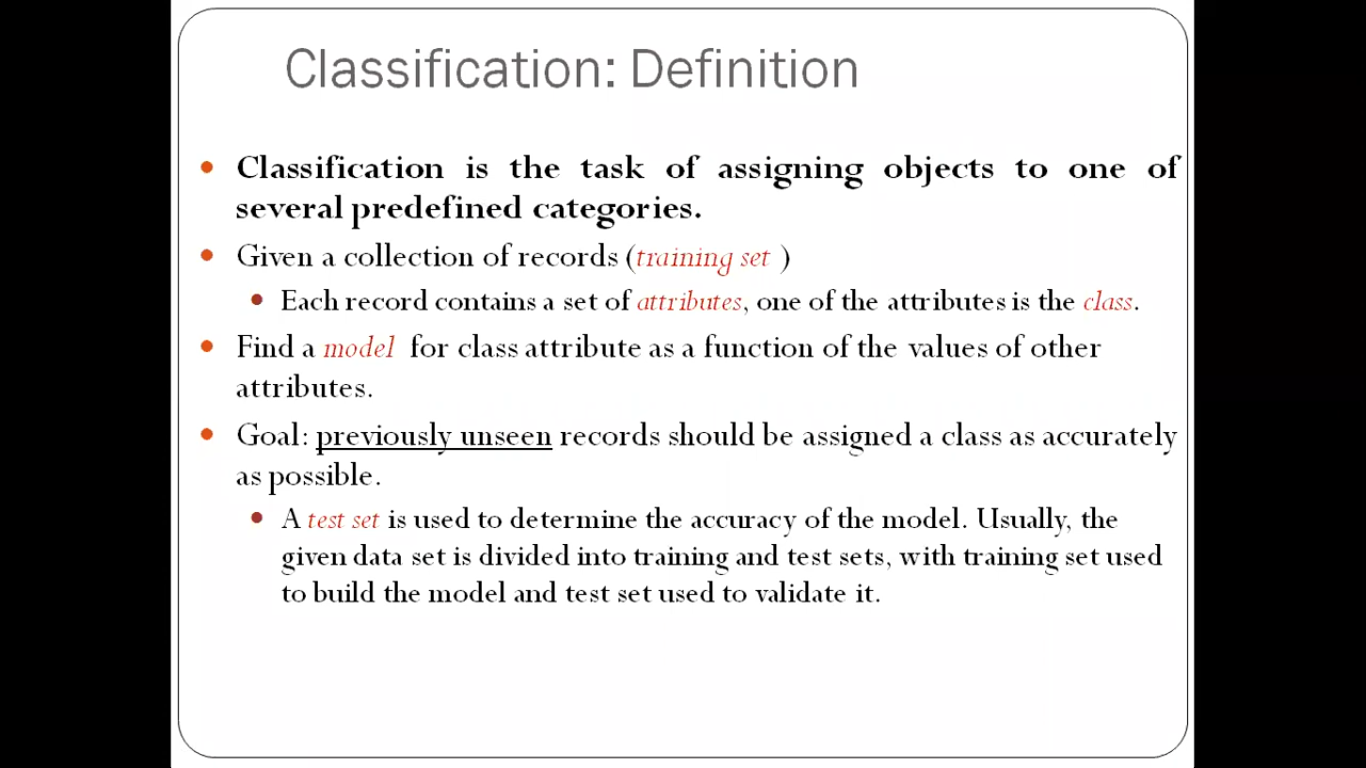
**UNIT-III**

**CLASSIFICATION**





The input data for a classification task is a collection of records. Each record is ,also known as an instance or example, is characterized by tuple(x ,y) ,where x is the attribute set and y is a special attribute designated as the class label.  
 **Goal**: previously unseen records should be assigned a class as accurately as possible.

Mining pattern that can classify future (new) data into known classes

**Why classification:**

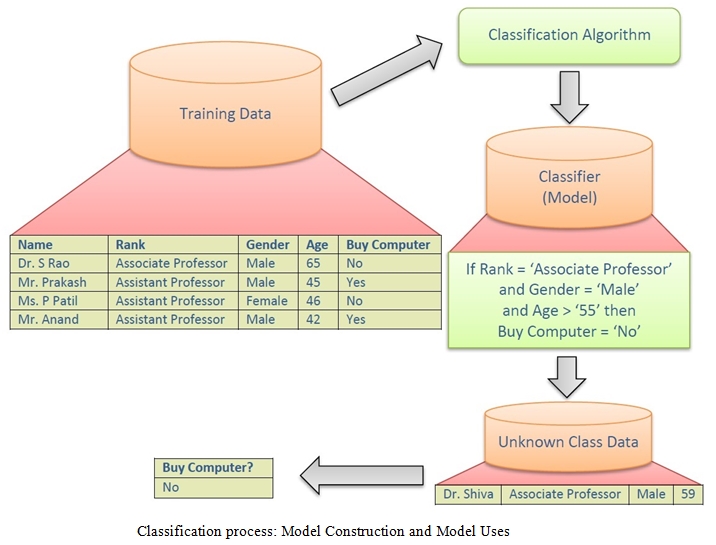
* Descriptive modeling: Explanatory tool to distinguish between objects of different classes (e.g., understand why people cheat on their taxes)
* Predictive modeling: Predict a class of a previously unseen record

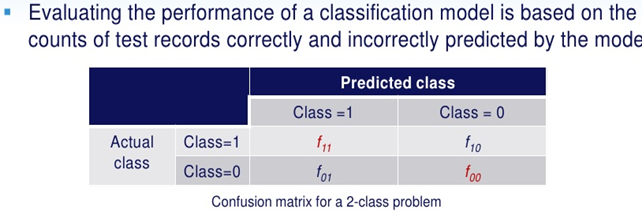
**Examples of Classification Tasks:**

1. Predicting tumor cells as benign or malignant
2. Classifying credit card transactions as legitimate or fraudulent
3. Categorizing news stories as finance,   
   weather, entertainment, sports, etc
4. Identifying spam email, spam web pages, adult content
5. Understanding if a web query has commercial intent or not

**General Approach to Solving a Classification Problem:**

* Training set consists of records with known class labels
* Training set is used to build a classification model
* A labeled test set of previously unseen data records is used to evaluate the quality of the model.
* The classification model is applied to new records with unknown class labels



**Evaluation of classification models:**

****

****

****

**Decision Tree Induction**: It is used for to make a decision. Decision tree induction is the learning of decision trees from class-labeled training tuples.

* A decision tree is a flowchart-like tree structure, where each internal node (nonleaf 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.
* Uses a tree structure to model the training set .

**Every tree has 3 types nodes:**

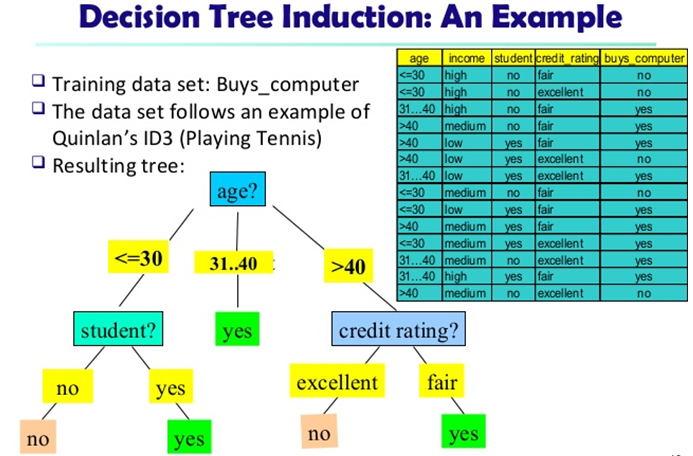
**1.ROOT NODE:** A Root node the tha no incoming edges and zero or more out goimg edges.

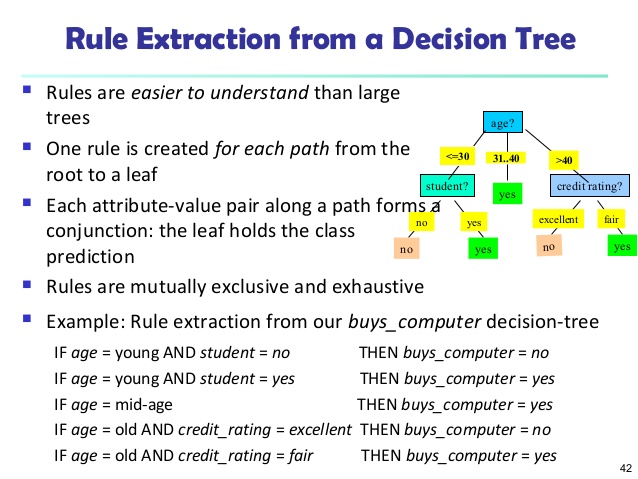
2.**INTERNAL NODES** :each of which has exactly one incoming edge and two or more out going edges.

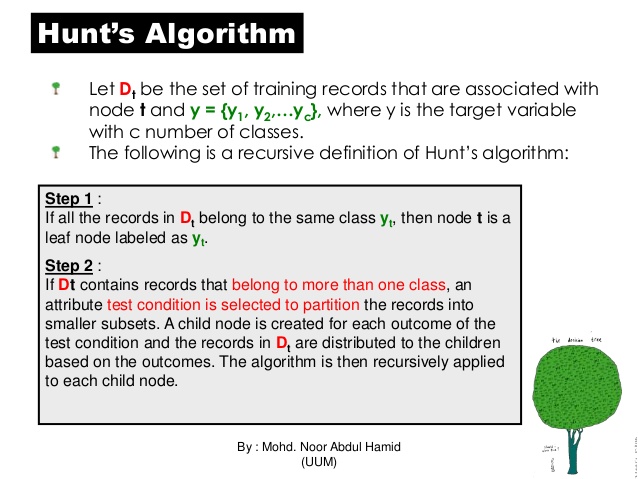
3. **LEAF(OR) TERMINAL NODES**: each of which has exactly one incoming edges and no outgoing edges.

**POPULARITY OF DECISION TREE:** The construction of decision tree classifier do not require ny domain knowledge. Decision tree handle high dimensional data.

Decision tree induction algorithm can be used in many application areas such as medicine,financial analisis and astronomy. A Decision tree algorithm known as ID3(iterative dichotomiser)this works as concept learning systems. later c4.5 is presented.







**Example of ID3**

Suppose we want ID3 to decide whether the weather is amenable to playing baseball. Over the course of 2 weeks, data is collected to help ID3 build a decision tree (see table 1).

The target classification is "should we play baseball?" which can be yes or no.

The weather attributes are outlook, temperature, humidity, and wind speed. They can have the following values:

outlook = { sunny, overcast, rain }

temperature = {hot, mild, cool }

humidity = { high, normal }

wind = {weak, strong }

Examples of set S are:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Day | Outlook | Temperature | Humidity | Wind | Play ball |
| 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 |

**Table 1**

We need to find which attribute will be the root node in our decision tree. The gain is calculated for all four attributes:

Gain(S, Outlook) = 0.246

Gain(S, Temperature) = 0.029

Gain(S, Humidity) = 0.151

Gain(S, Wind) = 0.048 (calculated in example 2)

Outlook attribute has the highest gain, therefore it is used as the decision attribute in the root node.Since Outlook has three possible values, the root node has three branches (sunny, overcast, rain). The next question is "what attribute should be tested at the Sunny branch node?" Since we=92ve used Outlook at the root, we only decide on the remaining three attributes: Humidity, Temperature, or Wind.

Ssunny = {D1, D2, D8, D9, D11} = 5 examples from table 1 with outlook = sunny

Gain(Ssunny, Humidity) = 0.970

Gain(Ssunny, Temperature) = 0.570

Gain(Ssunny, Wind) = 0.019

Humidity has the highest gain; therefore, it is used as the decision node. This process goes on until all data is classified perfectly or we run out of attributes.



The final decision = tree

The decision tree can also be expressed in rule format:

IF outlook = sunny AND humidity = high THEN playball = no

IF outlook = rain AND humidity = high THEN playball = no

IF outlook = rain AND wind = strong THEN playball = yes

IF outlook = overcast THEN playball = yes

IF outlook = rain AND wind = weak THEN playball = yes

**🡪How to Specify Test Condition?**

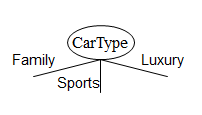
Depends on attribute types

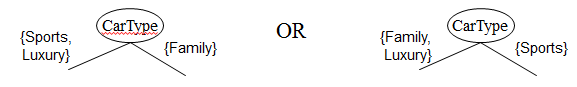
* Nominal
* Ordinal
* Continuous

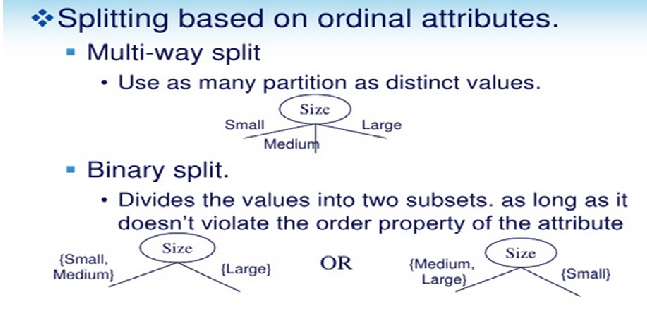
Depends on number of ways to split

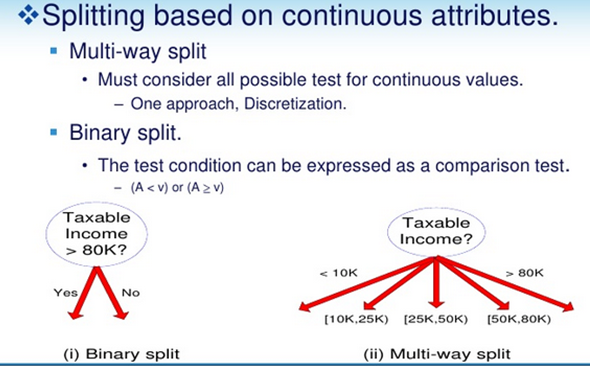
* 2-way split
* Multi-way split

**Splitting Based on Nominal Attributes**

* **Multi-way split**: Use as many partitions as distinct values. 
* **Binary split** : Divides values into two subsets.   
   Need to find optimal partitioning.



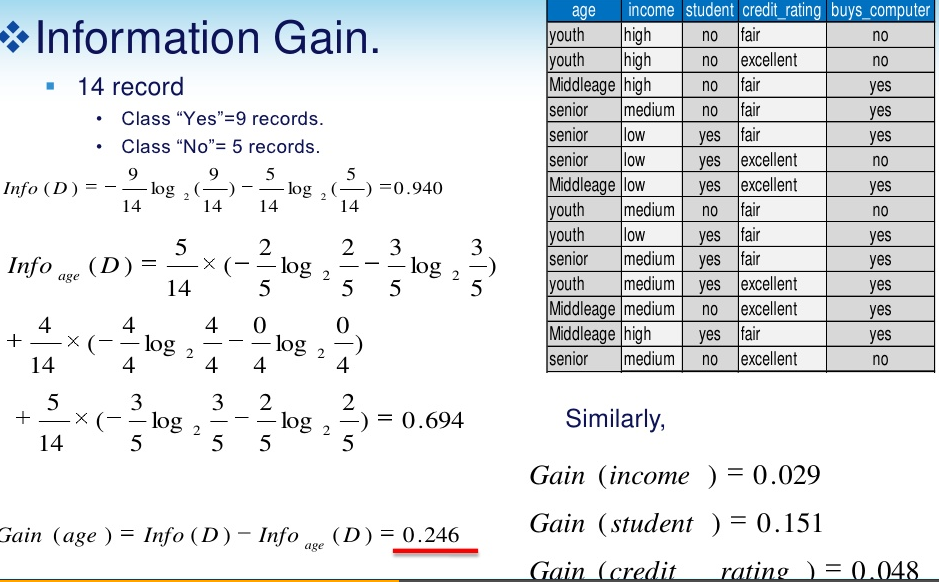
****

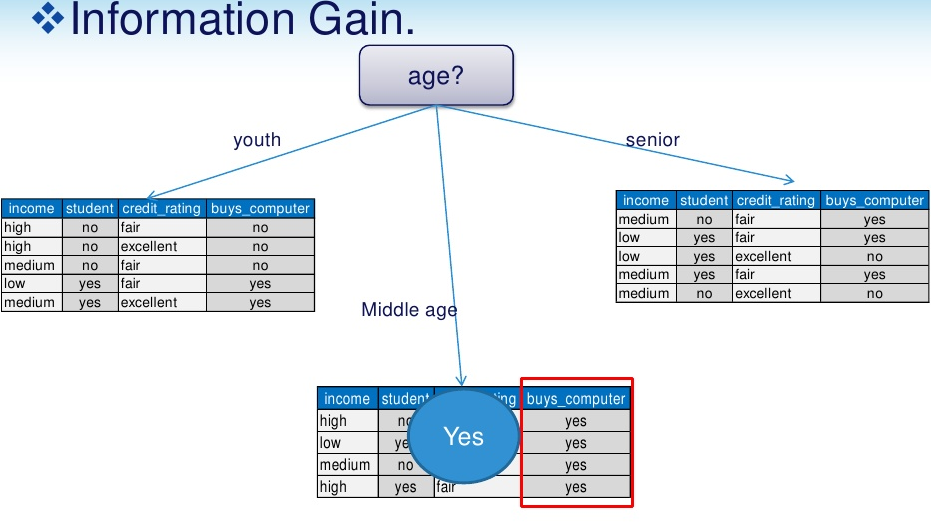


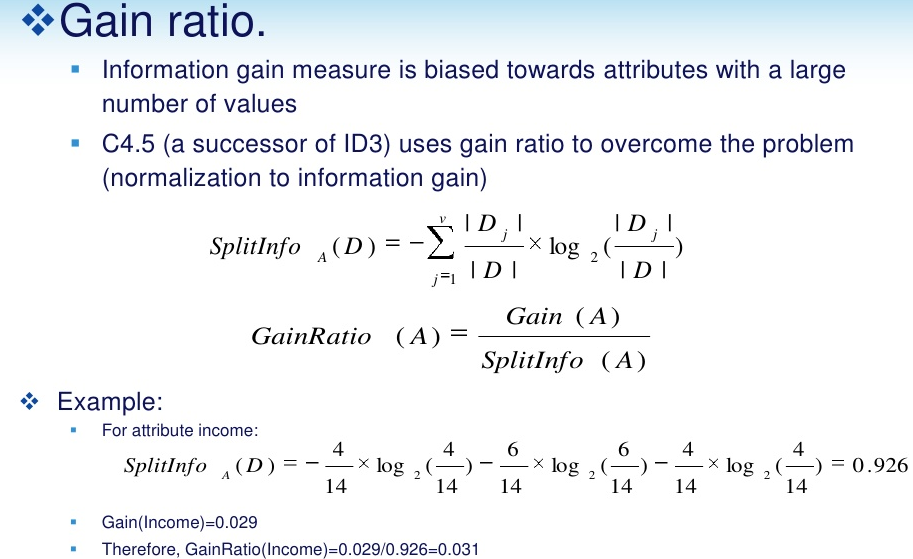


**Attribute Selection Measure:**









**Gini index:** Gini index is used in classification methods.Gini index measures the impurity of D. as

Gini(D)=1-∑(pi)2

Induction of decision tree using gini index: let d be the training set ,there are nine tuples belongs to class buys comp=yes and 5 tuples belongs to class buys comp=no

So gini index for D gini(D)=1-(9/14)2-(5/14)2 =0.45

**Decision tree induction:**

**Advantages:**

* inexpencive to constructive.
* Easy to construct.
* Easy to interrupt for small sized-trees.
* Extremely fast at classifying unknown records.

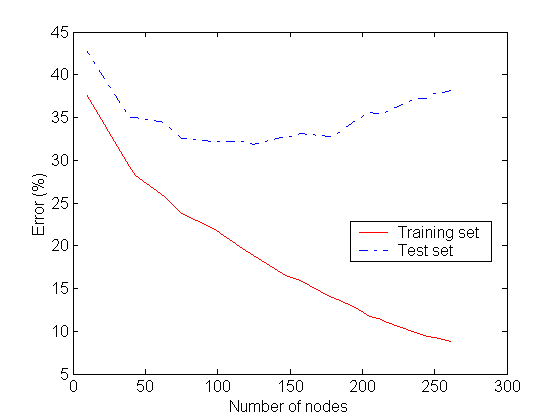
**Disadvantages: Decision tree could be suboptimal (i.e.,overfitting)**

**OVERFITTING THE DATA:** **If a decision tree is fully grown, it may lose some generalization capability. This is a phenomenon known as *overfitting*.**

**Errors committed by classification models are generally divided into two types:**

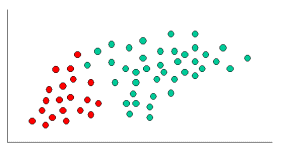
**1.Training Errors :The number of misclassification errors committed on training records; also called resubstitution error.**

**2.Generalization Errors :The expected error of the model on previously unseen records.**

* **A good classification model must have low TE as well as low GE**
* **A model that fits the training data too well can have high GE than a model with high TE**
* **This problem is known as model overfitting**
* **Underfitting and Overfitting.** 

|  |
| --- |
| **UNDERFITTING: when model is too simple, both training and test errors are large. TE & GE are large when the size of the tree is very small.**   * It occurs because the model is yet to learn the true structure of the data and as a result it performs poorly on both training and test sets   **OVER VIEW OF OVERFITTING**   * When a decision tree is built, many of the branches may reflect anomalies in the training data due to noise or outliers.   We may grow the tree just deeply enough to perfectly classify the training data set.  This problem is known as overfitting the data.  Overfitting & underfitting are two pathologies that are related to model complexity.   * **Problems of Overfitting :**Overfitting can lead to many difficulties:   Overfitted models are incorrect.  Require more space and more computational resources  Require collection of unnecessary features  They are more difficult to comprehend  **Overfitting can be due to:**   1. **Presence of Noise** 2. **Lack of representative samples**   **1.Presence of Noise:** The mammals classification problem.two of the ten training records  are mislabeled:bats and whales are classified as non-mammals instead  Of Mammals. |
| **How to Address Overfitting:**  **1.Pre-Pruning (Early Stopping Rule)**   * Stop the algorithm before it becomes a fully-grown tree   Typical stopping conditions for a node:   * Stop if all instances belong to the same class * Stop if all the attribute values are the same   **2.Post-pruning**   * Post-pruning : Post-pruning approach- removes branches of a fully grown tree. * Subtree replacement replaces a subtree with a single leaf node   **EVALUATING THE PERFORMANCE OF CLASSIFIER:** |
|  |
|  |
|  |
|  |

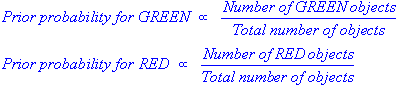
**Problem:** The Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.



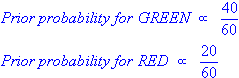
To demonstrate the concept of Naïve Bayes Classification, consider the example displayed in the illustration above. As indicated, the objects can be classified as either GREEN or RED. Our task is to classify new cases as they arrive, i.e., decide to which class label they belong, based on the currently exiting objects.

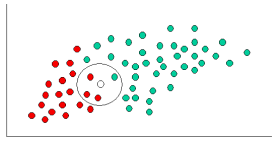
Since there are twice as many GREEN objects as RED, it is reasonable to believe that a new case (which hasn't been observed yet) is twice as likely to have membership GREEN rather than RED. In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience, in this case the percentage of GREEN and RED objects, and often used to predict outcomes before they actually happen.

Thus, we can write:

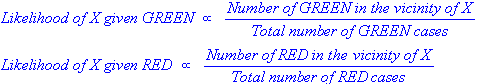


Since there is a total of 60 objects, 40 of which are GREEN and 20 RED, our prior probabilities for class membership are:





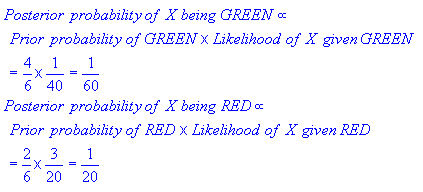
Having formulated our prior probability, we are now ready to classify a new object (WHITE circle). Since the objects are well clustered, it is reasonable to assume that the more GREEN (or RED) objects in the vicinity of X, the more likely that the new cases belong to that particular color. To measure this likelihood, we draw a circle around X which encompasses a number (to be chosen a priori) of points irrespective of their class labels. Then we calculate the number of points in the circle belonging to each class label. From this we calculate the likelihood:



From the illustration above, it is clear that Likelihood of X given GREEN is smaller than Likelihood of X given RED, since the circle encompasses 1 GREEN object and 3 RED ones. Thus:

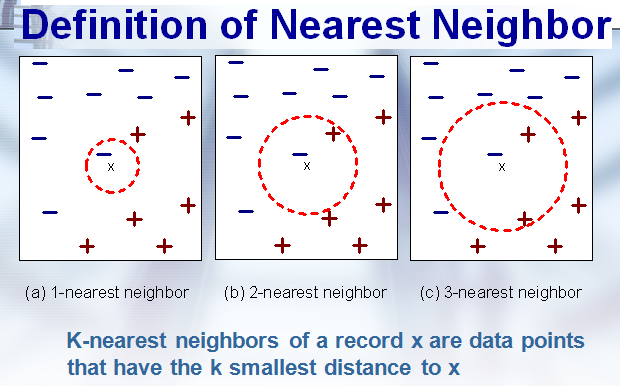
http://www.statsoft.com/textbook/NBEquation.gifhttp://www.statsoft.com/textbook/NaiveBayesIntro6.gif

Although the prior probabilities indicate that X may belong to GREEN (given that there are twice as many GREEN compared to RED) the likelihood indicates otherwise; that the class membership of X is RED (given that there are more RED objects in the vicinity of X than GREEN).



Finally, we classify X as RED since its class membership achieves the largest posterior probability.

**KNN CLASSIFIER:**

****

**Algorithm: KNN**

* The k-nearest neighbour method was first described in early 1950’s.
* k-nearest neighbor classifier are based on learning by by analogy(similarity).
* i.e by comparing .a given test tuple with training tuples that are similar to it.
* Training tuples are described by n –attributes.
* Each tuple represents point in n-dimensional space.
* All of the training Tuples are are stored in an n-dimensional space(pattern space).
* When an unknown tuple a k-nearest –neighbour classifier searches the pattern space for the k-train tuples that are closet to the unknown tuple.
* K-training tuples are the k-nearest neighbours of the unknown tuple.
* Close-ness is defined in terms of distance metric such as euclidean distance .
* Euclidean distance between two points or tuples say

X=(X1,X2,X3,X4…………XN)

Y=(Y1,Y2,Y3,Y4……………YN)

**DISTANCE(X,Y)= √(X1 – Y1)² + (X2 – Y2)²+…………(xn-yn)**

**COMPUTATION OF DISTANCE FOR ATTRIBUTE THAT NUMERIC BUT CATEGORICAL SUCH AS COLOR.**

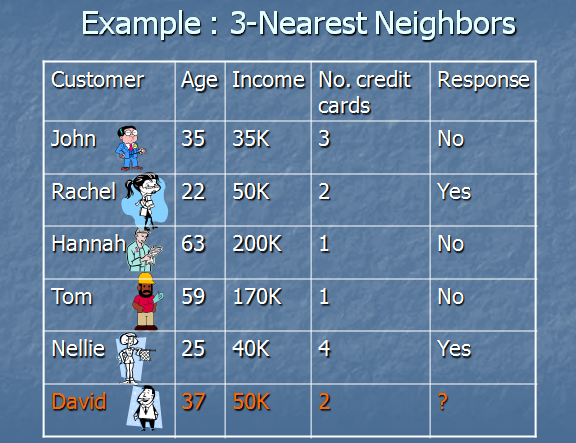
* For categorical attributes a simple method is to compare the corresponding value of that attributr in tuple x, with in tuple y.
* If 2 are identical (x1 and x2 both have same color blue) then difference is ‘0’.
* If 2 are different (tuple x1 is blue and tuple x2 is red) then difference is 1.

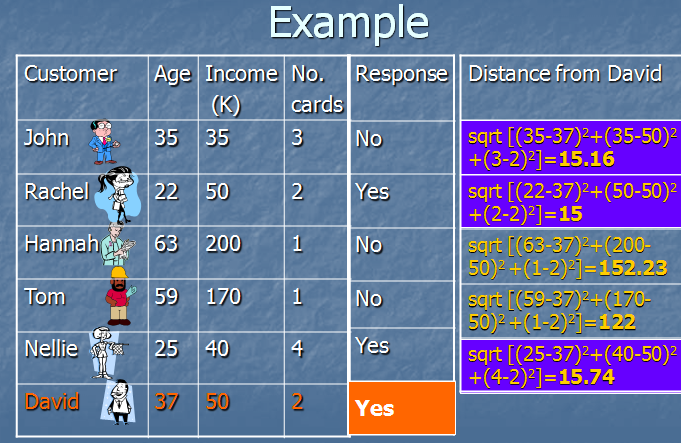
**ABOUT MISSING VALUES :**

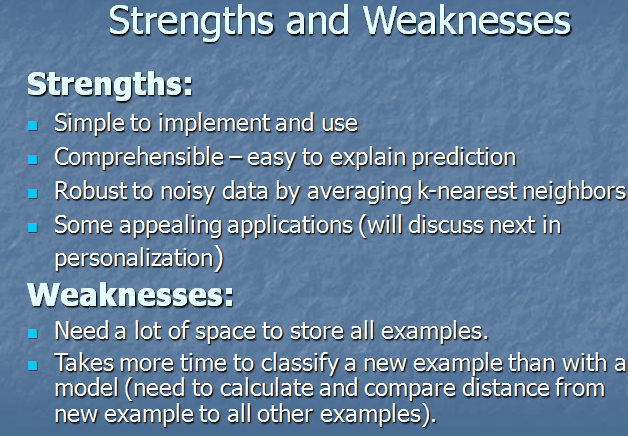
* If the value of a given attribute a is missing in tuple x and/ or in tuple y possible difference is maximum.
* For CATEGORICAL attributes the difference value is 1.
* If either one or both tuple s x and y of the corresponding values are missing.
* If a is numerical and missing from both tuples x and y then difference is also taken as 1

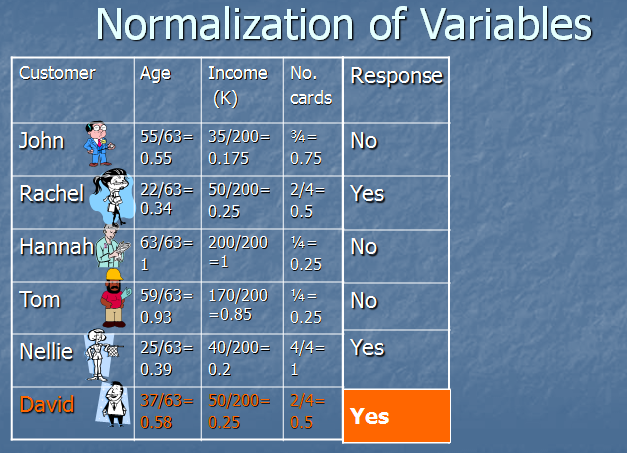
**DETERMINING A GOOD VALUE FOR K ,THE NO OF NEIGHBOURS:**

Start with k=1 test to Estimate error rate of the classifier. The process can be repeated each time by incrementing k to allow for one more neighbour. The k value that gives the minimum error rate may be selected.

****

****

****

****