**UNIT-IV**

**Classification:** Statistical-Based algorithms, Distance-Based algorithms, Decision Tree-Based algorithms, Neural Network-Based algorithms, Rule-Based algorithms, Combining Techniques.

**Objectives:**

To learn techniques to classify datasets and to provide categorical labels

To understand how models are used to predict certain future behaviors.

**UNIT-IV**

**Classification**

Data Classification is a two step process. In the first step, a model is built and which describes a set of data classes or concepts. If the class label of each training sample is known in advance then it is called supervised learning otherwise called unsupervised learning. In the second step predictive accuracy of the model is estimated.

The accuracy of a model on a given test set is the percentage of test set samples are correctly classified by the model. i.e., out of 100 test samples if the model able to classify 98 samples correctly then the accuracy of the model is defined as 98%.

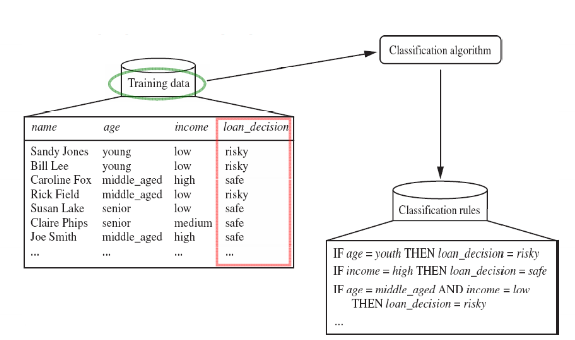


Fig 4.1: Learning in data classification

**4.1 Issues Regarding Classification and Prediction**

Preparing the Data for Classification and Prediction:

The following preprocessing steps may be applied to the data in order to improve the accuracy, efficiency, and scalability of the classification or prediction process.

**Data Cleaning**: This refers to the preprocessing of data in order to remove or reduce noise (by applying smoothing techniques) and the treatment of missing values(e.g., by replacing a missing value with the most commonly occurring value for that attribute, or with the most probable value based on statistics.). Although most classification algorithms have some mechanisms for handling noisy or missing data, this step can help reduce confusion during learning.

**Relevance Analysis:** Many of the attributes in the data may be irrelevant to the classification or prediction task. For example, data recording the day of the week on which a bank loan application was filed is unlikely to be relevant to the success of the application. Furthermore, other attributes may be redundant. Hence, relevance analysis may be performed on the data with the aim of removing any irrelevant or redundant attributes from the learning process. In machine learning, this step is known as feature selection. Including such attributes may otherwise slow down, and possibly mislead, the learning step.

**Data Transformation:** The data can be generalized to higher level concepts. Concept hierarchies may be used for this purpose. This is particularly useful for continuous valued attributes. For example, numeric values for the attribute income may be generalized to discrete ranges such as low, medium, and high. Similarly, nominal valued attributes like street, can be generalized to higher level concepts, like city. Since generalization compresses the original training data, fewer input / output operations may be involved during learning.

The data may also be normalized, particularly when neural networks or methods involving distance measurements are used in the learning step. Normalization involves scaling all values for a given attribute so that they fall within a small specified range, such as – 1.0 to 1.0, or 0.0 to 1.0. In methods that use distance measurements, for example, this would prevent attributes with initially large ranges (like, say, income) from outweighing attributes with initially smaller ranges (such as binary attributes).

**4.2 Comparison of Classification Methods**

Classification and prediction methods can be compared and evaluated according to the following criteria:

**Predictive Accuracy:** This refers to the ability of the model to correctly predict the class label of new or previously unseen data.

**Speed:** This refers to the computation costs involved in generating and using the model.

**Robustness**: This is the ability of the model to make correct predictions given noisy data or data with missing values.

**Scalability**: This refers to the ability to construct the model efficiently given large amount of data.

**Interpretability**: This refers to the level of understanding and insight that is provided by the model.

**STATI STICAL-BASED ALGORITHMS**

**4.2.1 Regression**

Regression problems deal with estimation of an output value based on input values. When used for classification, the input values are values from the database D and the output values represent the classes. Regression can be used to solve classification problems, but it can also be used for other applications such as forecasting. Although not explicitly described in this text, regression can be performed using many different types of techniques, including NNs. In actuality, regression takes a set of data and fits the data to a formula.

linear regression using the formula

Y = CO + C lX l + · · · + CnXn

By determining the regression coefficients co, C[ , . . • , Cn the relationship between the output parameter, y, and the input parameters, x 1 , . . . , Xn can be estimated. All high school algebra students are familiar with determining the formula for a straight line,

y = mx + b, given two points in the xy plane. They are determining the regression coefficients m and b. Here the two points represent the training data.

Admittedly, Example 3.5 is an extremely simple problem. However, it illustrates

how we all use the basic classification or prediction techniques frequently. Figure 4.5 illustrates the more general use of linear regression with one input value. Here there is a sample of data that we wish to model (shown by the scatter dots) using a linear model. The line generated by the linear regression technique is shown in the figure.

Notice, however, that the actual data points do not fit the linear model exactly. Thus, this model is an estimate of what the actual input-output relationship is. We can use the generated linear model to predict an output value given an input value, but unlike that for Example 3.5, the prediction is an estimate rather than the actual output value. If we attempt to fit data that are not linear to a linear model, the results will be a poor model

of the data, as illustrated by Figure 4.5.



**Bayes Theorem:**

Bayes theorem is a statistical classifier which performs probabilistic prediction, i.e., predicts class membership probabilities. A simple Bayesian classifier, naïve Bayesian classifier, has comparable performance with decision tree and selected neural network classifiers. Each training example can incrementally increase/decrease the probability that a hypothesis is correct, prior knowledge can be combined with observed data.

**Algorithm:**

Let **X** be a data sample (“evidence”): class label is unknown

Let H be a hypothesis that X belongs to class C

Classification is to determine P(H|**X**), the probability that the hypothesis holds given the observed data sample **X**

P(H) (prior probability), the initial probability

E.g., **X** will buy computer, regardless of age, income,

P(**X**): probability that sample data is observed

P(**X**|H) (posteriori probability), the probability of observing the sample **X**, given that the hypothesis holds.

E.g.,Given that **X** will buy computer, the prob. that X is 31..40, medium income

Given training data **X**, posteriori probability of a hypothesis H, P(H|**X**), follows the Bayes theorem



Posteriori = likelihood x prior/evidence

Each data sample is represented by an n-dimensional feature vector X=(x1,x2…..xn) measurements over n attributes respectively.

Predicts **X** belongs to Ci iff the probability P(Ci|**X**) is the highest among all the P(Ck|X) for all the k classes

P(Ci/X)>P(Cj/X) for 1<=j<=m j!=i

Therefore Posteriori hypothesis P(Ci/X) is maximized.



P(X) is constant for all classes,  need to be maximized. If the prior probabilities are not known in advance, then it is assumed that all the classes are equally likely.

P(C1)=P(C2)=P(C3)=…P(Cm)

To reduce computational complexity, class conditional independence is used. It is presumed that the values of the attributes are conditionally independent.

)

If attribute is categorical, then , Sik is the No. of training samples of class Ci, having the value xk for Ak, and Si is the number of training samples belonging to Ci.

Test sample X is assigned to class Ci iff P(X/Ci)P(Ci) > P(X/Cj)P(Cj)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **age** | **income** | **student** | **credit\_rating** | **buys\_computer** |
| <=30 | high | no | fair | no |
| <=30 | high | no | excellent | no |
| 31…40 | high | no | fair | yes |
| >40 | medium | no | fair | yes |
| >40 | low | yes | fair | yes |
| >40 | low | yes | excellent | no |
| 31…40 | low | yes | excellent | yes |
| <=30 | medium | no | fair | no |
| <=30 | low | yes | fair | yes |
| >40 | medium | yes | fair | yes |
| <=30 | medium | yes | excellent | yes |
| 31…40 | medium | no | excellent | yes |
| 31…40 | high | yes | fair | yes |
| >40 | medium | no | excellent | no |

**Table 7.2:** Customer information of All Electronics Shoppe

Class: C1:buys\_computer = ‘yes’ C2:buys\_computer = ‘no’

Predict the class label for given Test sample: X = (age <=30, Income = medium, Student = yes, Credit rating = Fair) using Bayes Theorem.

P(Ci): P(buys\_computer = “yes”) = 9/14 = 0.643

P(buys\_computer = “no”) = 5/14 = 0.357

Compute P(X|Ci) for each class

P(age = “<=30” | buys\_computer = “yes”) = 2/9 = 0.222

P(age = “<= 30” | buys\_computer = “no”) = 3/5 = 0.6

P(income = “medium” | buys\_computer = “yes”) = 4/9 = 0.444

P(income = “medium” | buys\_computer = “no”) = 2/5 = 0.4

P(student = “yes” | buys\_computer = “yes) = 6/9 = 0.667

P(student = “yes” | buys\_computer = “no”) = 1/5 = 0.2

P(credit\_rating = “fair” | buys\_computer = “yes”) = 6/9 = 0.667

P(credit\_rating = “fair” | buys\_computer = “no”) = 2/5 = 0.4

For the given test sample X = (age <= 30 , income = medium, student = yes, credit\_rating = fair)

**P(X|Ci) :**

P(X|buys\_computer = “yes”) = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(X|buys\_computer = “no”) = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

**P(X|Ci)\*P(Ci) :**

P(X|buys\_computer = “yes”) \* P(buys\_computer = “yes”) = 0.028

P(X|buys\_computer = “no”) \* P(buys\_computer = “no”) = 0.007

Therefore, the given test sample X belongs to class (“buys\_computer = yes”)

**Assignment:**

Predict the class label for given Test sample X using Bayes Theorem: X = (age <=31...40, Income = High, Student = yes, Credit rating = Excellent). Use Table 7.2 data for prediction.

**Distance based Algorithms**

**K nearest neighbor Classifier**

For an unknown given sample, k-nearest neighbor classifier searches the pattern space for the k training samples that are closest to the unknown sample. The closeness among the samples are defined in terms of Euclidean distance.

*d(x,y)=*

When an unknown sample is assigned to most common class among its k nearest neighbors. When k=1, unknown sample is assigned to training sample that is closest in sample space.

**4.4 Classification by Decision Tree based Algortihms**

Decision tree is a flow chart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and leaf nodes represent classes or class distributions. Top most node in a tree is the root node. In order to classify an unknown sample, the attribute values of the sample are tested against the decision tree. A path is traced from the root to a leaf node that holds the class prediction for that sample. Decision trees can easily be converted into classification rules. Many of the branches of Decision trees may reflect noise or outliers while in the training. Tree pruning attempts to identify and remove such branches, with the goal of improving classification accuracy on unseen data.

**Basic algorithm for induction of a Decision Tree**

**Algorithm**: Generate \_ decision \_ tree. Generate a decision tree from the given training data.

**Input**: The training samples, samples, represented by discrete – valued attributes; the set of candidate attributes, attribute – list.

**Output**: A decision tree

**Method**:

Create a node N;

If samples are all of the same class, C then

Return N as a leaf node labeled with the class C;

If attribute – list is empty then

Return N as a leaf node labeled with the most common class in samples ;

// majority voting

Select test attribute, the attribute among attribute – list with the highest information gain;

Label node N with test – attribute;

For each known value ai of test – attribute // partition the sample

Grow a branch from node N for the condition test – attribute = ai;

Let si be the set of samples in samples for which test – attribute = ai; // a partition

if si is empty then

attach a leaf labeled with the most common class in samples;

else attach the node returned by Generate\_decision\_tree

**Calculation of Information Gain of an Attribute:**



Select the attribute with the highest information gain

Let *pi* be the probability of an arbitrary tuple in D belongs to class Ci, estimated by |C*i*, D|/|D|

Expected information (entropy) needed to classify a tuple in D:

Information needed (after using A to split D into v partitions) to classify D:



Information gained by branching on attribute A

**Entropy:** Entropy is a measure of the uncertainty associated with a random variable and it is an uncertainty and or randomness increases for a result set so does the entropy values range from 0 to 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **age** | **income** | **student** | **credit\_rating** | **buys\_computer** |
| <=30 | high | no | fair | no |
| <=30 | high | no | excellent | no |
| 31…40 | high | no | fair | yes |
| >40 | medium | no | fair | yes |
| >40 | low | yes | fair | yes |
| >40 | low | yes | excellent | no |
| 31…40 | low | yes | excellent | yes |
| <=30 | medium | no | fair | no |
| <=30 | low | yes | fair | yes |
| >40 | medium | yes | fair | yes |
| <=30 | medium | yes | excellent | yes |
| 31…40 | medium | no | excellent | yes |
| 31…40 | high | yes | fair | yes |
| >40 | medium | no | excellent | no |

Table 4.1: Training data tuples from All Electronics Customer Database

Class P: buys computer = “yes”

Class N: buys computer= “no”

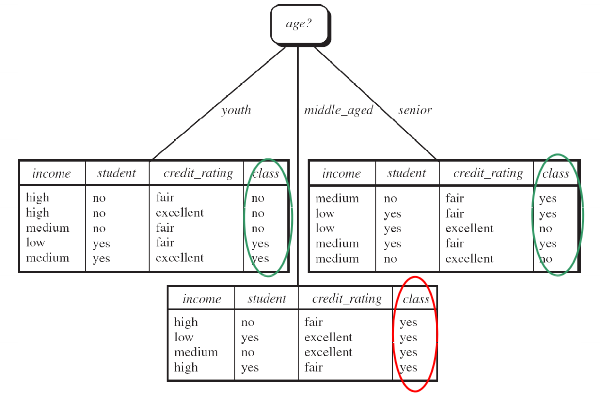
Info (D)= -9/14 log(9/14)2-5/14 log (5/14)2=0.94

Entropy(age)={5/14(-2/5 log(2/5)2-3/5 log(3/5)2) }

+ {4/14(-4/14 log(4/14)2-0/14 log(0/14)2)}

+ {5/14(-3/5 log(3/5)2-2/5 log(2/5)2)} = 0.694

Gain(age)=Info(D)-E(age)=0.246

****

**Fig 4.2:** The attribute age has highest information gain and therefore  becomes splitting attribute at the root node of the decision tree.

Branches are grown for each outcome of age.

**Assignment:** Construct a Decision Tree for the following example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Weather** | **Temperature** | **Humidity** | **Wind** | **Golf Play** |
| fine | Hot | high | none | no |
| fine | Hot | high | few | no |
| cloudy | Hot | high | none | yes |
| rain | Warm | high | none | yes |
| rain | Cold | medium | none | yes |
| rain | Cold | medium | few | no |
| cloudy | Cold | medium | few | yes |
| fine | Warm | high | none | no |
| fine | Cold | medium | none | yes |
| rain | Warm | medium | none | yes |
| fine | Warm | medium | few | yes |
| cloudy | Warm | high | few | yes |
| cloudy | Hot | medium | none | yes |
| rain | Warm | high | few | no |

**Tree Pruning**

Many of the branches in a decision tree will reflect anomalies in the training data which occur due to noise or outliers. Tree pruning methods use statistical measures to identify least reliable branches and those can be removed.

Pruning is defined commonly in two approaches called Pre pruning & Post pruning. In Pre Pruning, a tree is pruned by halting its construction early. Upon halting each node will become a leaf. The leaf may hold the most frequent class among the subset samples.

In Post Pruning, branches are removed from a fully grown tree. A tree node is pruned by removing its branches. The lowest un pruned node becomes a leaf and is labelled by the most frequent class among its former branches.

**4.5 NEURAL NETWORK-BASED ALGORITHMS**

With neural networks (NNs), just as with decision trees, a model representing how to classify any given database tuple is constructed. The activation functions typically are sigmoidal. When a tuple must be classified, certain attribute values from that tuple are input into the directed graph at the corresponding source nodes. There often is one sink node for each class. The output value that is generated indicates the probability that the corresponding input tuple belongs to that class. The tuple will then be assigned to the class with the highest probability of membership. The learning process modifies the labeling of the arcs to better classify tuples. Given a starting structure and value for all the labels in the graph, as each tuple in the training set is sent through the network, the projected classification made by the graph can be compared with the actual classification.

Based on the accuracy of the prediction, various labelings in the graph can change. This Solving a classification problem using NNs involves several steps:

1. Determine the number of output nodes as well as what attributes should be used as input. The number of hidden layers (between the source and the sink nodes) also must be decided. This step is performed by a domain expert.

2. Determine weights (labels) and functions to be used for the graph.

3. For each tuple in the training set, propagate it through the network and evaluate the output prediction to the actual result. If the prediction is accurate, adjust labels to ensure that this prediction has a higher output weight the next time. If the prediction is not correct, adjust the weights to provide a lower output value for this class.

4. For each tuple ti E D, propagate t; through the network and make the appropriate classification.

1 There are many issues to be examined:

• Attributes (number of source nodes): This is the same issue as determining which attributes to use as splitting attributes.

• Number of hidden layers: In the simplest case, there is only one hidden layer.

• Number of hidden nodes: Choosing the best number of hidden nodes per hidden layer is one of the most difficult problems when using NNs. There have been many empirical and theoretical studies attempting to answer this question. The answer depends on the structure of the NN, types of activation functions, training algorithm, and problem being solved. If too few hidden nodes are used, the target function may not be learned (underfitting). If too many nodes are used, overfitting may occur. Rules of thumb are often given that are based on the size of the training set.

• Training data: As with DTs, with too much training data the NN may suffer from overfitting, while too little and it may not be able to classify accurately enough.

• Number of sinks: Although it is usually assumed that the number of output nodes is the same as the number of classes, this is not always the case. For example, with two classes there could only be one output node, with the resulting value being the probability of being in the associated class. Subtracting this value from one would give the probability of being in the second class.

• Interconnections: In the simplest case, each node is connected to all nodes in the next level.

• Weights: The weight assigned to an arc indicates the relative weight between those two nodes. Initial weights are usually assumed to be small positive numbers and are assigned randomly.

• Activation functions: Many different types of activation functions can be used.

Learning technique: The technique for adjusting the weights is called the learning technique. Although many approaches can be used, the most common approach is some form of backpropagation, which is discussed in a subsequent subsection.

• Stop: The learning may stop when all the training tuples have propagated through the network or may be based on time or error rate.

There are many advantages to the use of NNs for classification:

• NNs are more robust than DTs because of the weights.

• The NN improves its performance by learning. This may continue even after the training set has been applied.

• The use of NNs can be parallelized for better perfom1ance.

• There is a low error rate and thus a high degree of accuracy once the appropriate training has been performed.

• NNs are more robust than DTs in noisy environments. Conversely, NNs have many disadvantages:

• NNs are difficult to understand. Nontechnical users may have difficulty understanding how NNs work. While it is easy to explain decision trees, NNs are much more difficult to understand.

• Generating rules from NNs is not straightforward.

• Input attribute values must be numeric.

• Testing

• Verification

• As with DTs, overfitting may result.

• The learning phase may fail to converge.

• NNs may be quite expensive to use.

**4.6 RULE-BASED ALGORITHMS**

One straightforward way to perform classification is to generate if-then rules that cover all cases. For example, we could have the following rules to determine classification of grades:

If 90 ::S grade, then class=A

If 80 ::S grade and grade < 90, then class=B

If 70 ::S grade and grade < 80, then class=C

i 60 ::S grade and grade < 70, then class D

If grade < 60, then class F

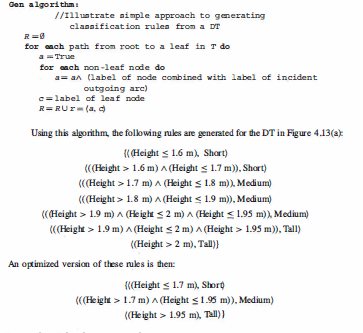
A classification rule, r = (a , c ) , consists of the if or antecedent, a, part and the then or consequent portion, c. The antecedent contains a predicate that can be evaluated as true or false against each tuple in the database (and obviously in the training data). These rules relate directly to the corresponding DT that could be created. A DT can always be used to generate rules, but they are not equivalent. There are differences between rules and trees :

• The tree has an implied order in which the splitting is performed. Rules have no order.

• A tree is created based on looking at all classes. When generating rules, only one class must be examined at a time. There are algorithms that generate rules from trees as well as algorithms that generate rules without first creating DTs.

**4.6. 1 Generating Rules from a DT**

The process to generate a rule from a DT is straightforward and is outlined in Algorithm This algorithm will generate a rule for each leaf node in the decision tree. All rules with the same consequent could be combined together by ORing the antecedents of the simpler rules.

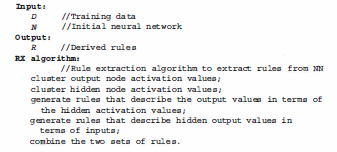


**4.6.2 Generating Rules from a Neural Net**

To increase the understanding of an NN, classification rules may be derived from it. While the source NN may still be used for classification, the derived rules can be used to verify or interpret the network. The problem is that the rules do not explicitly exist. They are buried in the structure of graph itself. In addition, if learning is still occurring, the rules themselves are dynaffiic. The rules generated tend both to be more concise and to have a lower error rate than rules used with DTs. The basic idea of the RX algorithm is to cluster output values with the associated hidden nodes and input. A major problem with rule extraction is the potential size that these rules should be. For example, if you have a node with n inputs each having 5 values, there are 5n different input combinations to this one node alone. These patterns would all have to be accounted for when constructing rules. To overcome this problem and that of having continuous ranges of output values from nodes, the output values for both the hidden and output layers are first discretized. This is accomplished by clustering the values and dividing continuous values into disjoint ranges. The rule extraction algorithm, RX, shown in Algorithm 4.9

is derived from [LSL95]

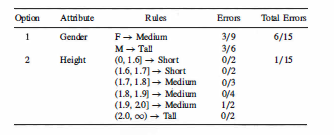
Algorithm.



Generating Rules iNithout a DT or N N

These techniques are sometimes called covering algorithms because they attempt to generate rules exactly cover a specific class [WFOO]. Tree algorithms work in a topdown divide and conquer approach, but this need not be the case for covering algorithms. They generate the best rule possible by optimizing the desired classification probability.

Usually the "best" attribute-value pair is chosen, as opposed to the best attribute with the tree-based algorithms. Suppose that we wished to generate a rule to classify persons as tall. The basic format for the rule is then If ? then class = tall



Another approach to generating rules without first having a DT is called **PRISM.**

PRISM generates rules for each class by looking at the training data and adding rules that completely describe all tuples in that class. Its accuracy is 100 percent. Example 4. 12 illustrates the use of PRISM. Algorithm 4. 1 1, which is adapted from [WFOO], shows the process. Note that the algorithm refers to attribute-value pairs. Note that the values will include an operator so that in Example 4.12 the first attribute-value pair chosen is with attribute height and value 72.0. As with earlier classification techniques, this must be

modified to handle continuous attributes. In the example, we have again used the ranges of height values used in earlier examples.

EXAMPLE 4.1.2

Using the data in Table 4.1 and the Outputl classification, the following shows the basic probability of putting a tuple in the tall class based on the given attribute-value pair:

Gender = F 0/9

Gender = M 3/6

Height <= 1.6 0/2

1 .6 < Height <= 1.7 0/2

1.7 < Height <= 1.8 0/3

1.8 < Height <= 1.9 1/2

1.9 < Height <= 2.0 0/4

2.0 < Height 2/2

Based on this analysis, we would generate the rule

If 2.0 < height, then class = tall

Since all tuples that satisfy this predicate are tall, we do not add any additional predicates to this rule. We now need to generate additional rules for the tall class. We thus look at the remaining 13 tuples in the training set and recalculate the accuracy of the corresponding predicates:

Gender = F 0/9

Gender = M 1/4

Height <= 1.6 0/2

1.6 <Height <= 1.7 0/2

1.7 < Height <= 1.8 0/3

1.8 <Height <= 1.9 0/4

1.9 < Height <= 2.0 1/2

Based on the analysis, we see that the last height range is the most accurate and thus generate the rule:

If 2.0 < height, then class = tall

However, only one of the tuples that satisfies this is actually tall, so we need to add another predicate to it. We then look only at the other predicates affecting

. these two tuples. We now see a problem in that both of these are males. The problem 1s actually caused by our "arbitrary" range divisions. We now divide the range into two subranges:

1.9 < Height <= 1.95 0/

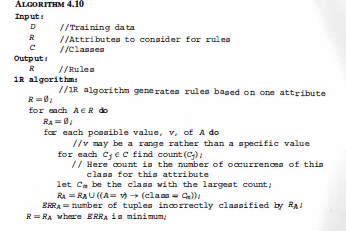
1.95 < Height <= 2.0 1/1

We thus add this second predicate to the rule to obtain

If 2.0 < height and 1.95 < height <= 2.0, then class = tall or

If 1.95 < height, then class = tall

This problem does not exist if we look at tuples individually using the attribute:-value pairs. However, in that case we would not generate the needed ranges for class1fymg the actual data. At this point, we have classified all tall tuples. The algonthm would then proceed by classifying the short and medium classes. This is left as an exercise.



**4.7 COMBINING TECHNIQUES**

Given a classification problem, no one classification technique always yields the best results. Therefore, there have been some proposals that look at combining techniques.

one technique for combining classifiers called boosting. A basic techniques can be used to accomplish this:

• A synthesis of approaches takes multiple techniques and blends them into a new approach. An example of this would be using a prediction technique, such as linear regression, to predict a future value for an attribute th at is then used as input to a classification NN . In this way the NN is used to predict a future classification value.

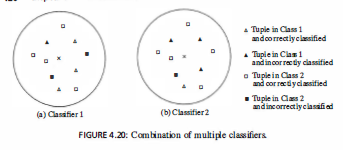
• Multiple independent approaches can be applied to a classification problem, each yielding its own class prediction. The results of these individual techniques can then be combined in some manner. This approach has been referred to as combination of multiple classifiers ( CMC) .

One approach to combine independent classifiers assumes that there are n independent classifiers and that each generates the posterior probability Pk ( C J 1 ti ) for each

class. The values are combined with a weighted linear combination

n

L Wk Pk (Cj I ti )

****

**SUMMARY**

Classification and prediction are two forms of data analysis which can be used to extract models describing important data classes or to predict future data trends. While classification predicts categorical labels (classes), prediction models continuous-valued functions.

\_ Preprocessing of the data in preparation for classification and prediction can involve data cleaning to reduce noise or handle missing values, relevance analysis to remove irrelevant or redundant attributes, and data transformation, such as generalizing the data to higher level concepts, or normalizing the data.

\_ Predictive accuracy, computational speed, robustness, scalability, and interpretability are effective criteria for the evaluation of classification and prediction methods.

\_ ID3 and C4.5 are greedy algorithms for the induction of decision trees. Each algorithm uses an information theoretic measure to select the attribute tested for each non-leaf node in the tree. Pruning algorithms attempt to improve accuracy by removing tree branches reflecting noise in the data. Early decision tree algorithms typically assume that the data are memory resident

- a limitation to data mining on large databases. Since then, several scalable algorithms have been proposed to address this issue, such as SLIQ, SPRINT, and RainForest. Decision trees can easily be converted to classification IF-THEN rules.

\_ Naive Bayesian classification and Bayesian belief networks are based on Bayes theorem of posterior probability. Unlike naive Bayesian classification (which assumes class conditional independence), Bayesian belief networks allow class conditional independencies to be defined between subsets of variables.

\_ Backpropagation is a neural network algorithm for classification which employs a method of gradient descent. It searches for a set of weights which can model the data so as to minimize the mean squared distance between the network's class prediction and the actual class label of data samples. Rules may be extracted from trained neural networks in order to help improve the interpretability of the learned network.

\_ Association mining techniques, which search for frequently occurring patterns in large databases, can be applied to and used for classification.

\_ Nearest neighbor classifiers and cased-based reasoning classifiers are instance-based methods of classification in that they store all of the training samples in pattern space. Hence, both require efficient indexing techniques.

In genetic algorithms, populations of rules \evolve" via operations of crossover and mutation until all rules within a population satisfy a specified threshold. Rough set theory can be used to approximately define classes that are not distinguishable based on the available attributes. Fuzzy set approaches replace \brittle" threshold cutoffs for continuous-valued attributes with degree of membership functions.

\_ Linear, nonlinear, and generalized linear models of regression can be used for prediction. Many nonlinear problems can be converted to linear problems by performing transformations on the predictor variables.

\_ Data warehousing techniques, such as attribute-oriented induction and the use of multidimensional data cubes, can be integrated with classification methods in order to allow fast multilevel mining. Classification tasks may be specified using a data mining query language, promoting interactive data mining.

\_ Stratified k-fold cross validation is a recommended method for estimating classifier accuracy. Bagging and boosting methods can be used to increase overall classification accuracy by learning and combining a series of individual classifiers.