


CMR INSTITUTE OF TECHNOLOGY		USN <input type="text"/>							
Internal Assesment Test - II									
Sub:	Data warehousing & Data mining						Code:	10IS74	
Date:	03 / 11 / 2016	Duration:	90 mins	Max Marks:	50	Sem:	7	Branch:	ISE
Answer Any FIVE FULL Questions									
							Marks	OBE	
								CO	RBT
1(a)	What is classification analysis? Explain the general Approach for Classification.						[10]	CO5	L3
2(a)	Write in detail about Bayesian classifier with an example? Explain the Continuous data attributes using an example.						[10]	CO3	L1
3(a)	Explain how to improve the accuracy of classification methods?						[10]	CO5	L3
4(a)	Discuss in detail about the web mining with an example?						[10]	CO6	L1
5(a)	List out the approaches involved in Text mining process and discuss in detail.						[10]	CO6	L1
6(a)	Explain how to estimate the accuracy of classification methods?						[10]	CO5	L3
7(a)	Discuss in detail about the Decision Tree with an example? List out the metrics for classification.						[10]	CO5	L3

Course Outcomes		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1:	Discuss the role of data warehousing and Data ware house Architecture.	1	-	2	-	-	-	-	-	-	1	-	-
CO2:	Describe the importance of Data cube design with OLAP operations.	1	2	1	-	-	-	-	-	-	1	-	-
CO3:	Identify the scope and necessity of Data Mining and their data sets, data attributes.	1	1	-	-	-	-	-	-	-	1	-	-
CO4:	Apply data mining technique of association analysis and cluster technique to design models for solving real world problems	2	3	1	-	-	-	-	-	-	1	-	-
CO5:	Apply classification techniques to solve real world problems.	2	3	-	-	-	-	-	-	-	-	-	-
CO6:	Describe the significance of Web mining, Text mining and temporal aspects of Data mining.	1	-	-	-	-	-	-	-	-	-	-	-

Cognitive level	KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PO1 - *Engineering knowledge*; PO2 - *Problem analysis*; PO3 - *Design/development of solutions*; PO4 - *Conduct investigations of complex problems*; PO5 - *Modern tool usage*; PO6 - *The Engineer and society*; PO7- *Environment and sustainability*; PO8 - *Ethics*; PO9 - *Individual and team work*; PO10 - *Communication*; PO11 - *Project management and finance*; PO12 - *Life-long learning*

Internal Assessment test - II

Scheme and solutions

Data warehousing & Data mining

1. What is classification analysis? Explain how to improve the accuracy of classification methods? [10]

Eliminate Low Quality Features (Words)

Low quality features in your training data-set is more likely to contribute negatively to your classification results (particularly since they might not be classified correctly), eliminating these low quality features can often lead to better classification results. In my experience this generally leads to a very healthy increase in over-all accuracy.

It is sometimes difficult to select a cut-off point for the most important features, generally it is recommended to design a research bench-work application that recursively tries different cut-off points and selects the one with the best accuracy (against a test data-set), for example for a Topic classifier I found that considering only the top **15,000** most frequent words (in my training set) leads to the best average performance against my multiple test data-sets.

A nice bi-product of eliminating low quality features is that your algorithm can be trained a lot faster, since the probability space is much smaller, which opens up the possibility of tweaking the algorithm more readily.

There is an excellent article by Jacob on Eliminating Low Information Features in NLTK, which can also be generalized to many classification algorithms.

Recursively Grow your Stopword List

I usually have at least 5 different stopwords list per classification project, each of which grows as the algorithm is re-optimised and tweaked throughout the life-time of the project, in order for the classifier to meet the target accuracy figure, some of the stopwords lists include:

- **Frequently used English (or any language) words:** this will include about 500 words that doesn't really contribute to context, such as: "*and, the, for, if, I*" etc.
- **Countries**
- **Cities**
- **Names**
- **Adjectives**
- **Temporal words:** this will include about 100 words such as: "*Tuesday, tomorrow, January*", etc.

And many others, obviously not all classification project will use all stopwords list, you will need to match the right stopwords with the right classification problem. You could grow your stopwords list by iteratively analyzing the top features in your algorithm for words that shouldn't be in there... and of course use logic!

Look Beyond Unigram into Bigrams and Trigrams

In text classification, Unigrams are single words, Bigrams are two related words (appear frequently next to each other in text), and Trigram is just the next extension of that concept.

I found that often considering Bigrams in a classification algorithm tends to really boost performance, since the increased long-tail specificity of the word means that the classifier can easily determine which class has a higher probability, leading to better classifications. In my experience Trigrams do not have offer the same boost as Bigrams, but they are worth considering and could be essential for certain types of classifiers. You could also go beyond Trigrams if you felt that the classification problem requires it.

The important thing to remember here is to apply the same logic for eliminating low quality bigrams and trigrams as you would with unigrams.

Diversify your Training Corpus

This point cannot be stressed enough, particularly if you wish to create a production-ready text classifier that behaves as expected in the lab as it would in the real world.

Diversity in the training corpus helps dilute word features that are specific to one particular corpus, allowing the classification algorithm to only select features that have a root contribution towards the text classification problem at hand. Obviously you need to select the corpus intelligently and do not just add more data for the sake of adding more data, it is all about the context of the classification problem.

For example if you were building a sentiment classification algorithm that will be used to classify social media sentiment, you need to make sure that your training set includes a variety of sources and not just training data from Twitter, ignoring communication from other social hubs like Facebook (a space in which you intend to run your algorithm). This will lead to features specific to Twitter data to appear in your classification probability space, leading to poor results when applied to other input sources.

Tweak Precision and Recall

I have written an article that discusses precision and recall in the context of the Confusion Matrix. The idea here is to tweak the system so when it fails, it does so in a manner that is more tolerable. This can be done by shifting *False Positive (or Precision)* under-performance to *False Negative (or recall)* under-performance, and viseversa according to you what is best for your system.

Eliminate Low Quality Predictions (Learn to Say “I Dont Know”)

Sometimes the algorithm might not be sure about which class the input text belongs to, this could be because

- The text does not contain features that the algorithm has been trained on. For example words that do not exist enough in the training set.

- The input text contains words from a different number of classes, resulting in evenly distributing the probability across those classes. For example a sentiment classifier trying to classify the input “I am happy and angry”

In these scenarios the classifier usually returns the item with the highest probability even though it is a very low quality guess.

If your model can tolerate a reduction in the coverage of what it can classify, you could greatly improve the accuracy of what is being classified by returning “Class Unknown” when the classifier is too uncertain (highest probability is lower than threshold), this can be done by analyzing probability filter threshold against accuracy and coverage.

Canonicalize Words through Lemma Reduction

The same word can have different formats depending on its grammatical usage (verb, adjective, noun, etc.), the idea of canonicalization is to reduce words to their lowest format (lemma), assuming that the grammatical placement of words is an irrelevant feature to your classifier. For example the words:

- Running
- Runner
- Runs
- Ran
- Runners

All can be reduced down to the word “**Run**” as far as the classifier is concerned.

This approach in reducing the word space can sometimes be extremely powerful **when used in the right context**, since it does not only reduce the probability space of the algorithm generated by the training set (giving the same word 1 score is better and more accurate than 10 different scores), but also helps in reducing the chances of encountering new words that the algorithm has not been trained on (when the algorithm is deployed), since all text will be reduced to its lowest canonical form, leading to improved *practical accuracy* of the algorithm.

This could also extend to **normalizing exaggerations in speech**, which is a very common problem when classifying social data, this will include reducing words like “*haaaaaappppppyyyy*” to “*happy*“, or even better, reducing all exaggerated lengths of the word “*happy*” to a canonical format different from the non-exaggerated form, for example reducing both “*haaaaaappppppyyyy*” and “*haaaaaaaaaaaaaappppppyyy*” to “*haapppy*“, this will differentiate it from the non-exaggerated form when scoring it for classification, but still reduces the over-all probability space by normalizing the word. A good example of where this might be applicable is when classifying *conversational intensity*.

2. Write in detail about Bayesian classifier with an example? [10]

A Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem (from Bayesian statistics) with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model".

In simple terms, a naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.

Depending on the precise nature of the probability model, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without believing in Bayesian probability or using any Bayesian methods.

In spite of their naive design and apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, analysis of the Bayesian classification problem has shown that there are some theoretical reasons for the apparently unreasonable efficacy of naive Bayes classifiers.^[1] Still, a comprehensive comparison with other classification methods in 2006 showed that Bayes classification is outperformed by more current approaches, such as boosted trees or random forests.^[2]

An advantage of the naive Bayes classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \dots, x_n)$ representing some n features (dependent variables), it assigns to this instance probabilities

$$p(C_k | x_1, \dots, x_n)$$

for each of K possible outcomes or classes.^[7]

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k | \mathbf{x}) = \frac{p(C_k) p(\mathbf{x} | C_k)}{p(\mathbf{x})}.$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C_k, x_1, \dots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(C_k) p(x_1, \dots, x_n | C_k) \\ &= p(C_k) p(x_1 | C_k) p(x_2, \dots, x_n | C_k, x_1) \\ &= p(C_k) p(x_1 | C_k) p(x_2 | C_k, x_1) p(x_3, \dots, x_n | C_k, x_1, x_2) \\ &= p(C_k) p(x_1 | C_k) p(x_2 | C_k, x_1) \dots p(x_n | C_k, x_1, x_2, x_3, \dots, x_{n-1}) \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$, given the category C . This means that

$$p(x_i | C_k, x_j) = p(x_i | C_k),$$

$$p(x_i | C_k, x_j, x_k) = p(x_i | C_k),$$

$$p(x_i | C_k, x_j, x_k, x_l) = p(x_i | C_k),$$

and so on, for $i \neq j, k, l$. Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k | x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots \\ &\propto p(C_k) \prod_{i=1}^n p(x_i | C_k). \end{aligned}$$

This means that under the above independence assumptions, the conditional distribution over the class variable C is:

$$p(C_k|x_1, \dots, x_n) = \frac{1}{Z}p(C_k) \prod_{i=1}^n p(x_i|C_k)$$

where the evidence $Z = p(\mathbf{x})$ is a scaling factor dependent only on x_1, \dots, x_n , that is, a constant if the values of the feature variables are known.

Constructing a classifier from the probability model

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the maximum a posteriori or MAP decision rule. The corresponding classifier, a Bayes classifier, is the function that assigns a class label $\hat{y} = C_k$ for some k as follows:

$$\hat{y} = \operatorname{argmax}_{k \in \{1, \dots, K\}} p(C_k) \prod_{i=1}^n p(x_i|C_k).$$

3. Improving Classification Accuracy Using Automatically Extracted Training Data [10]

Classification is a core task in knowledge discovery and data mining, and there has been substantial research effort in developing sophisticated classification models. In a parallel thread, recent work from the NLP community suggests that for tasks such as natural language disambiguation even a simple algorithm can outperform a sophisticated one, if it is provided with large quantities of high quality training data. In those applications, training data occurs naturally in text corpora, and high quality training data sets running into billions of words have been reportedly used.

We explore how we can apply the lessons from the NLP community to KDD tasks. Specifically, we investigate how to identify data sources that can yield training data at low cost and study whether the quantity of the automatically extracted training data can compensate for its lower quality. We carry out this investigation for the specific task of inferring whether a search query has commercial intent. We mine toolbar and click logs to extract queries from sites that are predominantly commercial (e.g., Amazon) and non-commercial (e.g., Wikipedia). We compare the accuracy obtained using such training data against manually labeled training data. Our results show that we can have large accuracy gains using automatically extracted training data at much lower cost.

4. Discuss in detail about the web mining? [10]

web Content mining :

The content based approach exploits semantic connections between documents and parts thereof, and semantic connections between queries and documents. Most content based document retrieval systems use an inverted index algorithm.

A signature file is a technique that creates a quick and dirty filter, for example a Bloom filter, that will keep all the documents that match to the query and hopefully a few ones that do not. The way this is done is by creating for each file a signature, typically a hash coded version. One method is superimposed coding. A post-processing step is done to discard the false alarms. Since in most cases this structure is inferior to inverted files in terms of speed, size and functionality, it is not used widely. However, with proper parameters it can beat the inverted files in certain environments.

5. List out the approaches involved in Text mining process? [10]

Text mining, also referred to as text data mining, roughly equivalent to text analytics, refers to the process of deriving high-quality information from text. High-quality information is typically derived through the devising of patterns and trends through means such as statistical pattern learning. Text mining usually involves the process of structuring the input text (usually parsing, along with the addition of some derived linguistic features and the removal of others, and subsequent insertion into a database), deriving patterns within the structured data, and finally evaluation and interpretation of the output. 'High quality' in text mining usually refers to some combination of relevance, novelty, and interestingness. Typical text mining tasks include text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity relation modeling (i.e., learning relations between named entities).

Text analysis involves information retrieval, lexical analysis to study word frequency distributions, pattern recognition, tagging/annotation, information extraction, data mining techniques including link and association analysis, visualization, and predictive analytics. The overarching goal is, essentially, to turn text into data for analysis, via application of natural language processing (NLP) and analytical methods.

A typical application is to scan a set of documents written in a natural language and either model the document set for predictive classification purposes or populate a database or search index with the information extracted.

6. Explain how to estimate the accuracy of classification methods? [10]

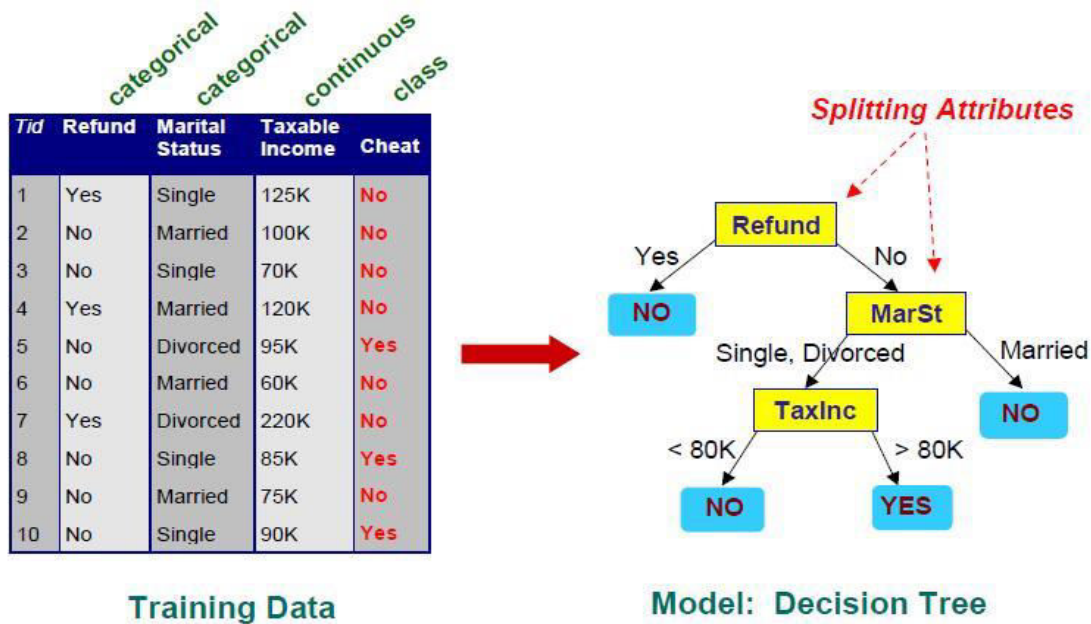
When we apply cluster analysis to a dataset, we let the values of the variables that were measured tell us if there is any structure to the observations in the data set, by choosing a suitable metric and seeing if groups of observations that are all close together can be found. If we have an auxiliary variable (like the country of origin from the cars example), it may be interesting to see if the natural clustering of the data corresponds to this variable, but it's important to remember that the idea of clustering is just to see if any groups form naturally, not to see if we can actually figure out which group an observation belongs to based on the values of the variables that we have.

When the true goal of our data analysis is to be able to predict which of several non-overlapping groups an observation belongs to, the techniques we use are known as classification techniques. We'll take a look at three classification techniques: kth nearest neighbor classification, linear discriminant analysis, and recursive partitioning.

7. What is Decision Tree? How is it built? How does it work? [10]

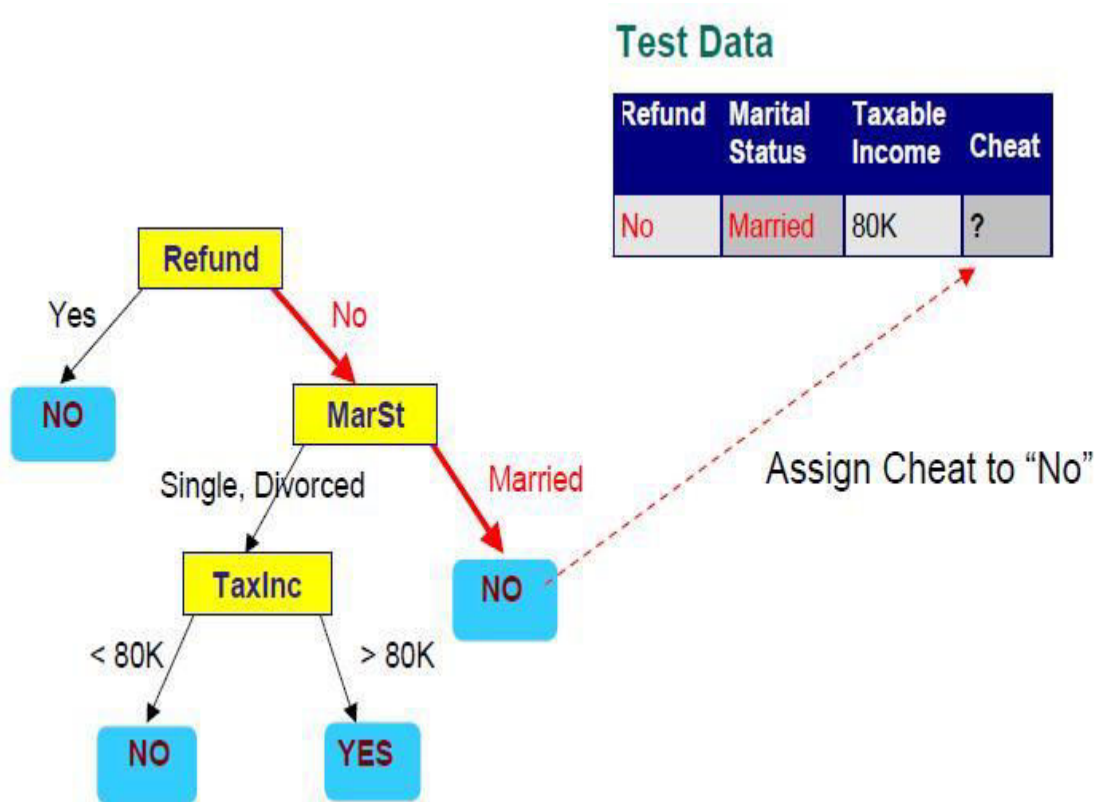
Decision Tree Based Method

The decision tree classifiers organized a series of test questions and conditions in a tree structure. The following figure [1] shows an example decision tree for predicting whether the person cheats. In the decision tree, the root and internal nodes contain attribute test conditions to separate records that have different characteristics. All the terminal nodes are assigned a class label Yes or No.



Once the decision tree has been constructed, classifying a test record is straightforward. Starting from the root node, we apply the test condition to the record and follow the appropriate branch based on the outcome of the test. It then leads us either to another internal node, for which a new test condition is applied, or to a leaf node. When we reach the leaf node, the class label

associated with the leaf node is then assigned to the record, As shown in the following figure [1], it traces the path in the decision tree to predict the class label of the test record, and the path terminates at a leaf node labeled NO.



Build A Decision Tree

Build a optimal decision tree is key problem in decision tree classifier. In general, many decision trees can be constructed from a given set of attributes. While some of the trees are more accurate than others, finding the optimal tree is computationally infeasible because of the exponential size of the search space.

However, various efficient algorithms have been developed to construct a reasonably accurate, albeit suboptimal, decision tree in a reasonable amount of time. 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. For example, Hunt's algorithm, ID3, C4.5, CART, SPRINT are greedy decision tree induction algorithms.

Hunt's Algorithm

Hunt's algorithm grows a decision tree in a recursive fashion by partitioning the training records into successively purer subsets. Let D_t be the set of training records that reach a node t . The general recursive procedure is defined as below: [1]

1. If D_t contains records that belong to the same class y_t , then t is a leaf node labeled as y_t
2. If D_t is an empty set, then t is a leaf node labeled by the default class, y_d
3. If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets.

It recursively applies the procedure to each subset until all the records in the subset belong to the same class. The Hunt's algorithm assumes that each combination of attribute sets has a unique class label during the procedure. If all the records associated with D_t have identical attribute values except for the class label, then it is not possible to split these records any further. In this case, the node is declared a leaf node with the same class label as the majority class of training records associated with this node.

Determine the best attribute test Condition

The decision tree inducing algorithm must provide a method for specifying the test condition for different attribute types as well as an objective measure for evaluating the goodness of each test condition.

First, the specification of an attribute test condition and its corresponding outcomes depends on the attribute types. We can do two-way split or multi-way split, discretize or group attribute values as needed. The binary attributes lead to two-way split test condition. For nominal attributes which have many values, the test condition can be expressed into multiway split on each distinct value, or two-way split by grouping the attribute values into two subsets. Similarly, the ordinal attributes can also produce binary or multiway splits as long as the grouping does not violate the order property of the attribute values. For continuous attributes, the test condition can be expressed as a comparison test with two outcomes, or a range query. Or we can discretize the continuous value into nominal attribute and then perform two-way or multi-way split.

Since there are many choices to specify the test conditions from the given training set, we need use a measurement to determine the best way to split the records. The goal of best test conditions is whether it leads a homogeneous class distribution in the nodes, which is the purity of the child nodes before and after splitting. The larger the degree of purity, the better the class distribution.

To determine how well a test condition performs, we need to compare the degree of impurity of the parent before splitting with degree of the impurity of the child nodes after splitting. The larger their difference, the better the test condition. The measurement of node impurity/purity are:

- Gini Index
- Entropy

- Misclassification Error
Stop the Split Procedure

A stop condition is also needed to terminate the tree-growing process. A possible strategy is to continue expanding a node until either all the records belong to the same class or all the records have identical attribute values. Although they are sufficient conditions to stop decision tree induction algorithm, some algorithm also applies other criteria to terminate the tree-growing procedure earlier.

Algorithm for Decision Tree Induction

The decision tree induction algorithm works by recursively selecting the best attribute to split the data and expanding the leaf nodes of the tree until the stopping criterion is met. The choice of best split test condition is determined by comparing the impurity of child nodes and also depends on which impurity measurement is used. After building the decision tree, a tree-pruning step can be performed to reduce the size of decision tree. Decision trees that are too large are susceptible to a phenomenon known as overfitting. Pruning helps by trimming the branches of the initial tree in a way that improves the generalization capability of the decision tree.

Here is an example recursive function [2] that builds the tree by choosing the best dividing criteria for the given data set. It is called with list of rows and then loops through every column (except the last one, which has the result in it), finds every possible value for that column, and divides the dataset into two new subsets. It calculates the weighted average entropy for every pair of new subsets by multiplying each set's entropy by the fraction of the items that ended up in each set, and remembers which pair has the lowest entropy. If the best pair of subsets doesn't have a lower weighted-average entropy than the current set, that branch ends and the counts of the possible outcomes are stored. Otherwise, buildtree is called on each set and they are added to the tree. The results of the calls on each subset are attached to the True and False branches of the nodes, eventually constructing an entire tree.