DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

proposal for

Classification of Signal Data Using Decision Trees

submitted to fulfill the semester project requirement for

EE 4773/6773: Digital Signal Processing

August 31, 1997

submitted to:

Dr. Joseph Picone

Department of Electrical and Computer Engineering
413 Simrall, Hardy Rd.
Mississippi State University
Box 9571
MS State, MS 39762

submitted by:

Audrey Le, Julie Ngan, Janna Shaffer

Decision Tree Group
Department of Electrical and Computer Engineering
Mississippi State University
Box 9571
Mississippi State, Mississippi 39762
Tel: 601-325-8335
Fax: 601-325-3149
email: {le, ngan, shaffer}@isip.msstate.edu

I. ABSTRACT

In this project, we propose the use of a decision tree as a method of classifying different signal data. Two specific types of signal data that we will attempt to classify are the scenic beauty values of forestry images and the pronunciations of proper nouns. Decision trees will be constructed for each set of the data using three decision tree algorithms: Bayesian, C4, and CART. Evaluations will be conducted on the two data sets using each of these three algorithms. A Graphical User Interface (GUI) will be implemented to demonstrate the performance of these algorithms.

II. INTRODUCTION

Background

Decision trees are used in many disciplines and in various application domains for data exploration and data classification. In astronomy, decision trees are used in star-galaxy classification. In medicine, decision trees are used for detecting thyroid disorders. In object recognition, tree-based classification has been used for recognizing three dimensional objects. In physics, decision trees have been used for the detection of physical particles. The versatility and usefulness of decision trees in various disciplines motivates us to investigate the functionality of decision trees for the data in our domain—forestry images and proper noun pronunciations.

Terminology

Before the discussion of decision tree construction, we introduce some basic definitions and terminology used in the discussion so that it will facilitate reader's understanding.

A decision tree is built from a training set, which consists of objects. Each object is completely described by a set of attributes and a class label. Attributes can have ordered or unordered values. An example of an ordered value is an integer or a real value, while a Boolean value is an example of an unordered value.

A decision tree contains a root node, zero or more internal nodes, and one or more leaf nodes. All internal nodes have one or more child nodes. All non-terminal nodes contain splits. Each leaf node has a class label associated with it. The number of classes is finite.

A univariate decision tree is one in which the test at each internal node uses a single attribute. A multivariate decision tree is one in which the test at each internal node uses several attributes.

An object is misclassified by a tree if the class label output by the tree does not match the object's class label. The proportion of objects correctly classified by the tree is called accuracy and the proportion of objects incorrectly classified by the tree is called error [1].

Basics of Tree Construction

Tree construction consists of three steps: 1) the selection of the splits, 2) the decisions when to stop or when to continue splitting, and 3) the assignment of each terminal node to a class [1]. The selection of the splits is a set of one or more questions that is used as a splitting criterion. The selection of splits is either formulated from the data or is predetermined. Using this splitting

criterion, the data is split into subsets, and the splitting repeats until it meets some stopping criterion, at which each non-terminal node becomes a terminal node and is assigned to a class.

Splitting Algorithms

The idea of finding splits of nodes gives rise to purer descendant nodes. Splitting rules are defined by specifying a goodness of split function $\phi(s,t)$, where s is the split and t is the object in the current node, defined for every node t, and $s \in S$, where S is a set of binary splits. At every t, the split adopted is the split s^* which maximizes $\phi(s,t)$ [2]. There are many algorithms that are used to split a set of objects. We will limit our discussion to 1) twoing criterion, 2) Bayesian rule, 3) information gain, and 4) gain ratio.

The **twoing criterion** requires that selection at every node that conglomeration of classes be divided into two superclasses so that the problem can be considered as a two-class problem, where the greatest decrease in node impurity is realized. For example, there are $C = \{1, \ldots, J\}$ classes. At each node, the class is separated into two superclasses, $C_1 = \{j_1, \ldots, j_n\}$, $C_2 = C - C_1$. For any given split s of the node, $\Delta i(s, t)$, is computed as though it were a two-class problem. $s^*(C1^*)$ is found to maximize $\Delta i(s, t)$. $C1^*$ is found which maximizes $\Delta i(s^*(C_1), t)$. The split used on the node is $s^*(C1^*)$.

At each node, it sorts the classes into those two groups, which in some sense are most dissimilar, and outputs to the user the optimal grouping $C1^*$ and $C2^*$ as well as the best split s^*. This criterion attempts to group together large numbers of classes that are similar in some characteristics near the top of the tree and attempts to isolate single classes near the bottom of the tree.

Bayesian rule partitions the space of the samples into disjoint subsets with each leaf corresponding to one such subset and associating a conditional probability with each leaf. Let T be the tree structure that defines the partition and C be the number of classes, the probabilistic rule associated with each leaf can be modeled as a conditional probability distribution. If suppose example x falls to leaf l in the tree structure T then the tree gives a vector of class probabilities $\Phi_{j,l}$ for $j = 1, \ldots, C$, corresponding to a tree structure T and matrix of class proportions Φ_T. For a tree T, Φ_T represents a conditional probability distribution for class c given example x in the form of $\Phi_{j,l} = Pr((c = d_j)|(x, T, \Phi_T))$ where example x falls to leaf l in the tree structure T. For a given training sample $\vec{x}$, $\vec{c}$ consisting of N examples x_i with known classification given by class values c_i, the distribution of a single classified example x, c can be specified by a probability distribution on the example together with a conditional probability distribution on the class c given the example x, which corresponds to the class probability tree. Given a sample consisting of N examples with known classification, the posterior distribution of class probability

trees is given by Bayes rule:

$$Pr((T, \Phi_T)|(\vec{x}, \vec{c})) \alpha Pr(T, \Phi_T|\vec{x}) \Pi\Pi \varphi^{n_{j,l}}{}_{j,l}$$

where $n_{j,l}$ is the number of examples of class d_j falling in the l-th leaf of the tree structure T.

Information gain is another method used in hierarchical partitioning of the feature space. There is a simple method proposed by Sethi and Sarvarayudu [6] for the hierarchical partitioning of the feature space. The method is non-parametric and based on the concept of average mutual information. More specifically, let the average mutual information obtained about a set of classes C_k from the observation of an event X_k, at a node k in a tree T be defined as $I_k(C_k;X_k) = \sum_{C_k}\sum_{X_k} p(C_{ki};X_{kj}) \log\left[\frac{p(C_{ki}/X_{kj})}{p(C_{ki})}\right]$. Event X_k represents the measurement value of a feature selected at node *k* and has two possible outcomes; measurement values greater or smaller than a threshold associated with that feature at that node.

Then, the average mutual information between the entire set of classes, C, and the partitioning tree, T, can be expressed as $I(C;T) = \sum_{k=l}^{L} p_k \cdot I_k(C_k;X_k)$ where p_k is the probability of the class set C_k and L is the number of internal nodes in the tree T.

The probability of misclassification, p_e, of a decision tree classification T and the average mutual information $I(C;T)$ are also related as

$$I(C;T) \leq -\sum_{j=1}^{m} [p(C_j) \cdot \log p(C_j)] + p_e \cdot \log p_e + (1 - p_e) \cdot \log(1 - p_e) + p_e \cdot \log(m - 1)$$

with equality corresponding to the minimum required average mutual information for a prespecified probability of error. Then a goal for design of the tree could be to maximize the average mutual information gain at each node k. The algorithm terminates, when the tree average mutual information, $I(C;T)$, exceeds the required minimum tree average mutual information specified by the desired probability of error. An alternative stopping criterion proposed by Talmon is to test the statistical significance of the mutual information gain that results from further splitting a node.

Even though information gain provides quite good results, it has a deficiency of a strong bias in favor of tests with many outcomes. In other words, when one of the attributes contains unique information for all of the data, partitioning any set of training cases on the values of this attribute will lead to a large number of subsets, each containing just one case. Since all of these one-case subsets necessarily contain cases of a single class, the information gain will be maximal, but yet

quite useless [3].

Therefore, **gain ratio** is used instead of information gain as the splitting rule in many algorithms. Gain ratio is basically a normalization of the information gain where the apparent gain attributable to tests with many outcomes is adjusted. In order to generate a normalization, we define the potential information generated by splitting a decision tree T into n subsets as split information, with the class set C_i, can be expressed as, $Split(C;T) = -\sum_{i=1}^{n} [p(C_i) \cdot \log p(C_i)]$. Then the split information conveys the information relevant to classification that arises from the division.

As a result, the gain ratio, $GainRatio(C;T)$, expresses the proportion of information generated by the split that would be useful in the classification, and it is defined as, $GainRatio(C;T) = \frac{Gain(C;T)}{Split(C;T)}$. Therefore, when this ratio is maximized, the information gain must be large compared to the average gain over all test examined.

Pruning and Smoothing Algorithms

The recursive partitioning method of constructing decision tree subdivide the set of training cases until each subset in the partition contains cases of a single class. The resulting tree is often very complex and "overfits the data" by inferring more structure than is justified by the training classes. Therefore, the idea of tree pruning is introduced. Pruning of the decision tree is done by replacing a whole subtree by a leaf node. The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf. In the case of noisy data, zero probability can be found in leaf nodes. To obtain a better classification tree, sometimes smoothing is used instead of pruning. Smoothing of the decision tree is done by averaging multiple trees. In this project, we have limited our discussion of pruning and smoothing algorithms to the following:

Cost complexity pruning is also known as error complexity pruning. The complexity of a subtree S can be defined as the number of terminal nodes in that subtree. The complexity is denoted by $|\tilde{S}|$. Thus, the cost-complexity of a subtree is defined as $C_\alpha(S) = A(S) + \alpha|\tilde{S}|$, where A is the apparent error rate and α is a positive real number called the complexity parameter. The smallest minimizing subtree $S(\alpha)$ for α can be defined for the following conditions: $C_\alpha(S(\alpha)) = \min_{S \le S_{max}} C_\alpha(S)$ and if $C_\alpha(S) = C_\alpha(S(\alpha))$, then $S(\alpha) \le S$.

Thus, for every α there exists a smallest minimizing subtree. Even though α is continuous, there will be a finite number of subtrees because there is a finite number of branches off of the main tree. Therefore, a decreasing sequence of subtrees can be defined with $\alpha_1 = 0$ and

$\alpha_k \le \alpha < \alpha_{k+1}$ for which $S(\alpha) = S(\alpha_k) = S_k$ for $\alpha_k \le \alpha < \alpha_{k+1}$.

Now that a set of subtrees has been defined, the problem becomes finding the best subtree. This is performed on the basis of the apparent error rate for each subtree. If $A(S_{ko}) = \min_k \mathrm{A}(S_k)$, then S_{ko} is the optimum subtree. The subsequent choice of the subtree is made after cross-validation of the apparent error rates [8].

Bayesian Smoothing

The standard approach for classifying an example using a class probability tree is to send the example down to a leaf and then return the class probability at the leaf [10]. In the smoothing approach, the class probability vectors encountered at interior nodes along the way are averaged [9]. Given a particular tree structure T' grown as described by Bayes splitting rule, and given a pruned tree structure $pruned(T')$ obtained by pruning the tree structure T' in all possible ways, if the space given by the pruned tree structure $pruned(T')$ is restricted and the posterior on the tree structure is a muliplicative function over nodes in the tree, then the sum can be recursively calculated using the distributive law. The sum is computable in a number of step linear in the size of the tree. The sum takes the form of an average calculated along the branch traversed by the new example.

$$E_{T, \Phi|(\vec{x}, \vec{c})}(Pr(c = d_j)|(x, T, \Phi_T)) = \sum_{n \in traversed(x, T')} Pr(\text{n is leaf}|(\vec{x}, \vec{c}, \text{pruning of T}'))\frac{n_{j,n} + \alpha_j}{n_{0,n} + \alpha_0}$$

where $traversed(x, T')$ is the set of nodes on the path traversed by the example x as it falls to a leaf, and $Pr(\text{n is leaf}|(\vec{x}, \vec{c}, \text{pruning of T}'))$ is the posterior probability that the node n in the tree T' will be pruned back to a leaf given that the "true" tree is a pruned subtree of T'. It is given by

$$Pr(\text{n is leaf}|(\vec{x}, \vec{c}, \text{pruning of T}')) = (CPr(leaf(n), \vec{x}, \vec{c}))/(SPr(T', \vec{x}, \vec{c})) \cdot \Pi CPr(node(O))) \cdot SPr(B, \vec{x}, \vec{c}))$$

where ancestors $ancestors(T', n)$ is the set of ancestors of the node n in the tree T', $child(O, x)$ is the set of children trees of the node O,

$$SPr(T, \vec{x}, \vec{c}) = \sum_{S \in pruned(T)} Pr(S(T, \vec{x}, \vec{c})) \text{ [10]}$$

Pessimistic Pruning

Pessimistic pruning is developed by Quinlan [3], based on the idea of statistical correction. The resubstitution error, which is the error rate on pruning a subtree using the observation on the training set from which the tree was built, is estimated and adjusted to reflect this estimate's bias. Therefore, based on the estimated and adjusted resubstitution error, the tree is pruned. This is done by examining each nonleaf subtree, starting from the bottom of the tree. If replacement of this subtree with a leaf, or with its most frequently used branch, would lead to a lower predicted error rate, then the tree is pruned accordingly. Since the error rate for the whole tree decreases as

the error rate of any of its subtrees is reduced, this process will lead to a tree whose predicted error rate is minimal with respect to the allowable forms of pruning.

More specifically, the pessimistic estimate is described as follows. Consider a leaf covering N training cases, with E of them classified incorrectly. The resubstitution error rate for this leaf is then E/N. If we define this result as the probability of error over the entire population of cases covered by this leaf, for a given confidence level CF, the upper limit on this probability can be found from the confidence limits for the binomial distribution, denoted by $U_{CF}(E, N)$. On the argument that the tree has been constructed to minimize the observed error rate, this upper limit is then equated as the predicted error rate at a leaf.

To simplify the accounting, error estimates for leaves and subtrees are computed assuming that they were used to classify a set of unseen cases of the same size as the training set. So, a leaf covering N training cases with a predicted error rate of $U_{CF}(E, N)$ would give rise to a predicted $N \times U_{CF}(E, N)$ errors. Therefore, for a subtree t with k leaves, and each of the leaves cover n training cases with none of them classified incorrectly, the predicted error for k_i would be $n \times U_{CF}(0, n)$. The predicted error E_t for the subtree would be $\sum_{i=1}^{n} k_i$. Then if the subtree is replaced by a single leaf, it would cover the same number of training cases, n, but with error e, so the corresponding predicted errors E_t^* would be $n \times U_{CF}(e, n)$. If $E_t^* \geq E_t$, then the existing subtree has a higher number of predicted errors, and it is pruned to a leaf.

III. ALGORITHM DESCRIPTIONS

CART

CART refers to the software created in 1984 by Brieman, et al. It is an acronym for classification and regression trees. CART constructs a binary decision tree by recursively partitioning the training data. Its intent is to grow a large tree to cover all of the training cases, and then prune down the tree to balance the error rate with size of the tree [8]. CART uses the twoing criterion for splitting and cost-complexity cross-validation for pruning.

Bayesian Classifier

Bayesian classifier is based on the assumption that all of the relevant probability values are known. The apriori probabilities are assumed to be known. The random variable X can be determined to what class it belongs to based on a decision rule of probabilities. In our implementation of the Bayes' classifier, we use Bayes splitting rule to build multiple trees and use smoothing to average the trees.

C4

C4 is a decision tree algorithm which as its origins in Hunt's [7] Concept Learning Systems by

way of ID3. It is introduced by Quinlan [3] for inducing classification models from data. A typical C4 algorithm generates a decision tree using the gain ratio as splitting rule and the pessimistic pruning as the pruning rule.

IV. DATA DESCRIPTION

USFS Scenic Beauty Estimation Database

The database contains 638 images obtained from the USFS for algorithm research and development. The database contains thirteen features extracted from the images. These features are the distributions of green in different hues ranging from 0 to 255 divided into ten subgroups, the percentage of long lines in each image, and the entropy of color distribution (red, green, and blue) in each image. These features describe the scenic beauty values of the images. There are three classes of scenic beauty values: high scenic beauty estimate, medium scenic beauty estimate, and low scenic beauty estimate. These categories are determined by the subjective scenic beauty estimates (SBEs) that were obtained from human subjects judging the scenic value of the images. The SBEs of the images were averaged and the standard deviation was computed. Low scenic beauty falls below one standard deviation from the mean, medium scenic beauty falls between one standard deviation from the mean, and high scenic beauty falls one standard deviation above the mean.

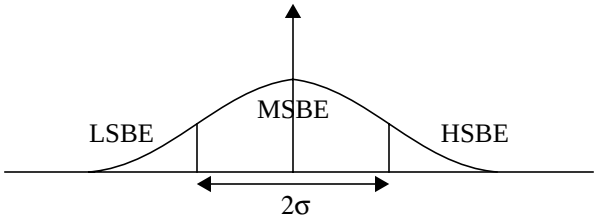

Figure 1 Categories of Scenic Beauty Estimate.

Proper Noun Pronunciation Database

A comprehensive public domain pronunciation dictionary of people’s last names (surnames) consists of 18,494 surnames from a diversity of ethnic origins and 25,648 corresponding pronunciations. The data, collected from a variety of sources, represents a reasonable mix of commonly found surnames, surnames with infrequent occurrence, and surnames that are known to present problems for letter-to-sound conversion due to complex morphology or difficult stress assignments [11].

The phonetic transcription was performed by hand using the Worldbet standards. Each surname was transcribed to obtain all the correct pronunciations possible. Transcription of name pronunciations was a difficult task as the surnames derive from dozens of source languages having different stress patterns. A number of foreign names have both ethnic as well as anglicized pronunciations and individual pronunciations are often peculiar in defying any kind of typical text-to-speech rules

V. PROJECT SUMMARY

The purpose of this project is to implement decision tree algorithms to classify signal data. We are using two types of data in testing our decision tree algorithms: USFS scenic beauty database and proper noun pronunciation database.

The main tasks involves investigating and experimenting with different decision tree techniques as described in the previous section, and extending the decision tree algorithms to accommodate other data. This can be broken down as below:

- Implement the three different decision tree algorithms as described previously. The code for these algorithms will be written in C++ and compiled using the gnu compiler. The different implemented decision tree algorithms will be trained on a set of data and then tested to find the best decision tree algorithm which produces the least misclassification rate.

- Implement a Graphical User Interface (GUI) to output the results of the different decision tree according to the user's input.

- Generalize our software to accommodate other data.

VI. EVALUATION

Decision trees are dependent on the domain of data on which they train. Therefore, a comprehensive evaluation plan is needed. The evaluation of the decision tree will be done in several steps. Two types of evaluation will be utilized: closed loop and open loop. Evaluation of closed loop involves testing the tree using the same data set that was used in training. Evaluation of open loop involves training the tree on one set of data and testing it on a different set of data. Each of these evaluation types will be used to determine the rate of misclassification of the decision tree.

Data	Method	Set Size (Train + Test)	Error
Image	CART		
	C4		
	Bayes		
Pronunciation	CART		
	C4		
	Bayes		

Table 1 Performance Statistics.

VII. PROJECT DEMONSTRATION

The demo for this project will bring all pieces of the project together into a graphical user interface (GUI). This GUI will be developed in tcl/tk. It will be an extension of an existing GUI built for a previous project on proper noun pronunciation. Thus, the demo will only exhibit the use of decision trees on the proper noun pronunciation database and not the USFS scenic beauty estimation database. The demo will allow the user to choose which decision tree algorithm he or she would like to use when finding the n-best pronunciations of the proper noun input to the system. The n-best pronunciations will appear in the lower left hand box of the demo, while the phoneme network of the pronunciation will appear in the lower right hand box. The demo also provides the audio speech of the pronunciation of the proper noun. A layout of the GUI is shown below with approximate locations of each of the features.

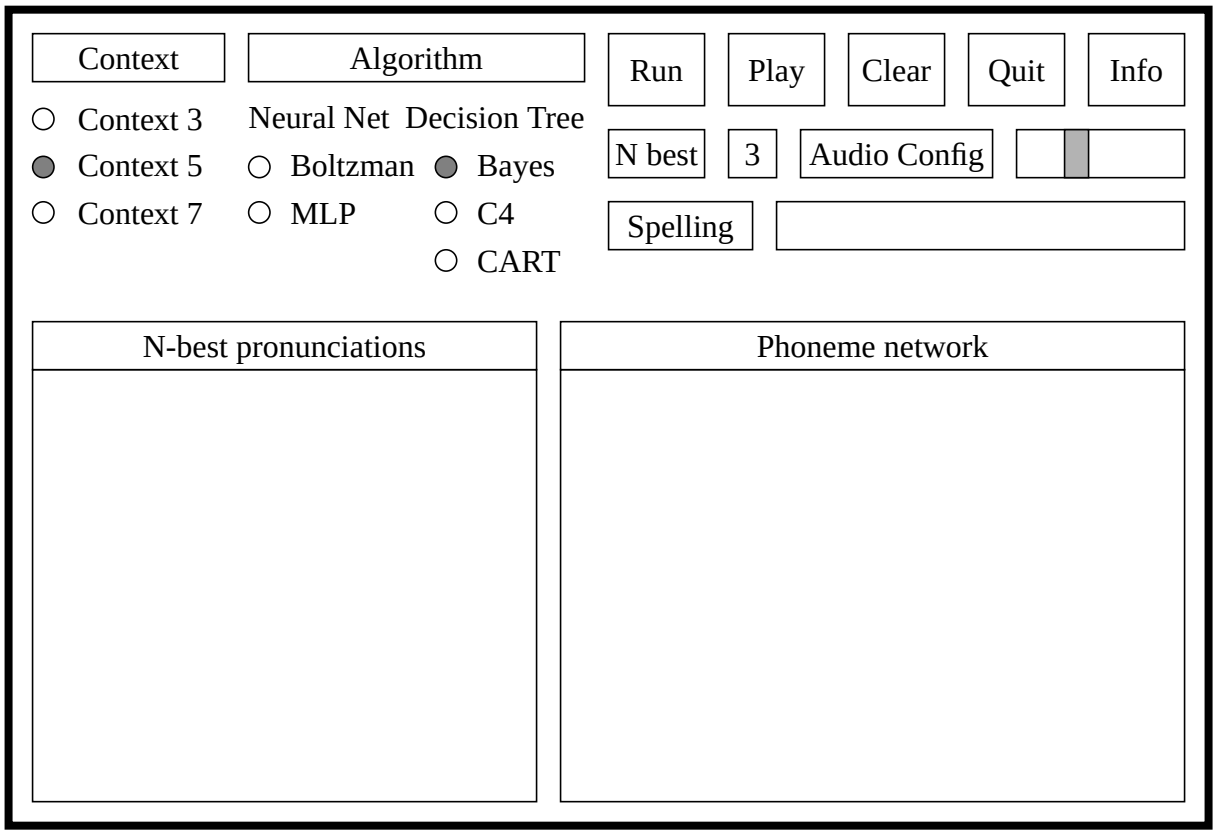

Figure 2 Graphical User Interface for Proper Noun Pronunciation Demo.

VIII. SCHEDULE

A schedule of the major tasks in this project is shown Figure 3.

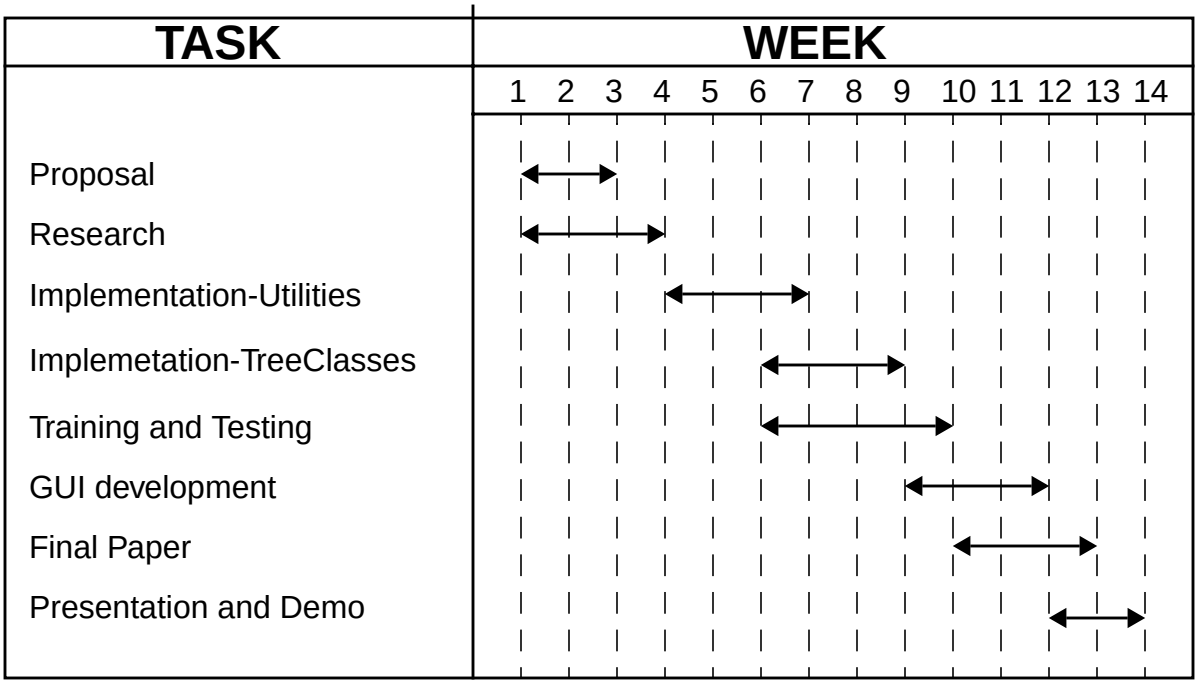

Figure 3 Schedule of Completion of Key Tasks.

Table 2 shows the contribution made by each member of our group.

	Audrey	Janna	Julie
Research	X	X	X
Proposal	X	X	X
Utility Classes	X	X	X
Algorithms	X-Bayes	X-CART	X-C4
Training/Testing	X	X	X
Demo	X		X
Presentation		X	
Paper	X	X	X

Table 2 Division of Labor

IX. REFERENCES

[1] S. K. V. Murthy. *On Growing Better Decision Trees from Data*, Ph.D. thesis, Johns Hopkins University, Baltimore, Maryland, 1996.

[2] L. Brieman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth International Group, 1984.

[3] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, 1993.

[4] P. C. Taylor and B. W. Silverman. “Block Diagrams and Splitting Criteria for Classification Trees”. *Statistics and Computing*, vol. 3, pp 147-161, Dec. 1993.

[5] S. R. Safavin and D. Landgrebe. “A Survey of Decision Tree Classifier Methodology”, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 3, pp. 660-674, May/June 1991.

[6] I. K. Sethi and M. Otten, “Hierarchical Classifier Design Using Mutual Information,” IEEE Trans Patt. Anal. Mach. Intell., vol. PAMI-4, pp.441-445, 1982.

[7] E. B. Hunt, J. Marin, and P. J. Stone. *Experiments in Induction.* Academic Press, 1966.

[8] G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition.* John Wiley & Sons, Inc., 1992.

[9] L. Bahl, P. Brown, P. de Souza, and R. Mercer, “A Tree-Based Language Model for Natural Language Speech Recognition”, *IEEE Transactions Acoustic Speech and Speech Processing*, vol. 37, pp. 1001-1008.

[10] W. Buntine, “Learning Classification Trees”, *Statistics and Computing*, vol. 2, pp 63-73, 1992.

[11] E.C. Smith, *American Surnames*, Genealogical Publishing Co. Inc., Baltimore, MD, 1986.