NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

CART PROGRAM: THE IMPLEMENTATION OF THE CART PROGRAM AND ITS APPLICATION TO ESTIMATING ATTRITION RATES

by

Amin Elseramegy, Hamdy

December 1985

Thesis Advisor: Robert R. Read

UNCLASSIFIED
CURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
. DECLASSIFICATION / DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
. NAME OF PERFORMING ORGANIZATION: aval Postgraduate School 6b. OFFICE SYMBOL (If applicable): 55	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
ADDRESS (City, State, and ZIP Code): onterey, California 93943-5100	7b. ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5100
NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

TITLE (Include Security Classification)
ART PROGRAM: THE IMPLEMENTATION OF THE CART PROGRAM AND ITS APPLICATION O ESTIMATING ATTRITION RATES

PERSONAL AUTHOR(S)
min Elseramegy, Hamdy

13. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
aster's Thesis	FROM _____ TO _____	1985 December	174

SUPPLEMENTARY NOTATION

COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
CART Classification Regression
Binary Tree Resubstitution
Implementation Application

ABSTRACT (Continue on reverse if necessary and identify by block number)

The main purpose of this paper is the implementation of the lassification and regression trees (CART) programs on the NPS Computer ystem. An additional goal is the forecasting of officer attrition rates f the U.S. Marine Corps. The use of this program requires a deep under- tanding of the algorithmic structure of CART and the user must experiment ith it in order to develop a reasonable approach to the management of the omputer's memory. The complete set of commands required to run the rograms, and the complete results, are included.

DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
. NAME OF RESPONSIBLE INDIVIDUAL Robert R. Read	22b. TELEPHONE (Include Area Code): (408) 646-2382 22c. OFFICE SYMBOL: 55 Re

FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Approved for public release; distribution is unlimited

CART Program
The Implementation of the CART Program and Its Application to Estimating Attrition Rates

by

Amin Elseramegy, Hamdy
Colonel, Egyptian Air Force
B.S.C., Military Technical College, 1972

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1985

ABSTRACT

The main purpose of this paper is the implementation of the classification and regression trees (CART) programs on the NPS Computer System. An additional goal is the forecasting of officer attrition rates for the highly cross-classified manpower structure of the U.S. Marine Corps. The use of this program requires a deep understanding of the algorithmic structure of CART and the user must experiment with it in order to develop a reasonable approach to the management of the computer's memory. The complete set of commands required to run the programs, and the complete results, are included.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. GENERAL

The main purpose of this paper is the implementation of the CART (Classification and Regression Trees) programs on the NPS computer system (IBM 3030) by describing the procedure for CART Version 1.1, which is being distributed as of December 20, 1984, and the application of these programs to real data; the Marine Corps officer manpower data.

A summary of necessary graph theoretic terminology is presented in this introduction, but first we describe the contents of each chapter.

Chapter II is concerned with the goals of the CART programs, and the kinds of problems they solve. In particular, the classification problem is introduced; the concepts of learning samples and classifiers are discussed.

Chapter III is concerned with the idea of the CART technique. The sequence of solution steps followed by the CART program is explained. It is instructive to explain first the initial algorithm that was followed and then to present the modified algorithm which is being used now. A numerical example is worked out in detail in this chapter so as to better explain the sequence of solution steps.

Chapter IV introduces the Marine Corps officer manpower attrition rate data. The forecasting of these rates is the

main application of this paper. After describing the data structure and source, the steps followed for data preparation and extraction of samples are presented.

Chapter V, the main chapter of this paper, is concerned with the implementation of the CART programs on the NPS computer system. It contains the commands and files required to run these programs. Typical examples are presented in detail. The content of output forms, notation, and special terms are explained.

Chapter VI is concerned with the results obtained from our application to Marine Corps manpower data.

Chapter VII contains our evaluation to the CART programs based on more then 50 runs of these programs.

Finally, the Appendix includes all the complete outputs of the CART programs when the Marine Corps manpower data files were used as input files.

B. BACKGROUND

The CART programs use the binary tree as a main data structure. The following lines define what we mean by a binary tree. These definitions are standard. For further information, see any good text on graph theory.

A "graph" G=[V,E] consists of a vertex set V and an edge set E. Either G is undirected, in which case every edge is an unordered pair of distinct vertices, or G is directed,

in which case every edge is an ordered pair of distince vertices.

A "path" in a graph from vertex V_1 to vertex V_k is a list of vertices $[V_1, V_2, \ldots, V_k]$ such that (V_i, V_{i+1}) is an edge for $i \in \{1, \ldots, k-1\}$. The path contains vertex V_i for $i \in \{1, \ldots, k\}$ and edge (v_i, v_{i+1}) for $i \in \{1, \ldots, k-1\}$ and avoids all other vertices and edges. Vertices V_1 and V_k are the ends of the path. The path is "simple" if all its vertices are distinct. If the graph is directed, the path is a "cycle" if $k>1$ and $V_1 = V_k$, and a "simple cycle" if in addition $V_1, V_2, \ldots, V_{k-1}$ are distinct. If the graph is undirected, the path is a cycle if $k>1$, $V_1 = V_k$ and no edge is repeated. It is a simple cycle if in addition $V_1, V_2, \ldots, V_{k-1}$ are distinct. A graph without cycles is a cyclic. If there is a path from a vertex v to a vertex w then w is reachable from v.

An undirected graph G is "connected" if every vertex is reachable from every other vertex and "disconnected" otherwise.

A "free tree" is an undirected graph that is connected and acyclic.

A "rooted tree" is a free tree with a distinguished vertex r, called the root. If v and w are vertices such that v is on the path from r to w, v is an ancestor of w and w is a descendant of v. If in addition $v \neq w$, v is a proper ancestor of w and w is a proper descendant of v. If v is a proper ancestor of w, and v and w are adjacent, v is the parent of w

and w is a child of v. Every vertex v except the root has a unique parent, and zero or more children.

A "full binary tree" is a rooted tree in which each vertex v has either two children, its left child left(v) and its right child right(v), or no children. In subsequent chapters, we refer to the full binary trees by simply "binary trees."

The CART system consists of three main programs:

(1) CART-BUILD program,

(2) CART-CASE program, and

(3) CART-TEST program.

The purpose of each of these programs and how to use them are the main issues of this paper.

II. CART: PROBLEMS AND GOALS

Depending on the application, the basic purpose of a classification study can be either to produce an accurate classifier or to uncover the predictive structure of the problem. If we are aiming at the latter, then we are trying to get an understanding of what variables or interactions of variables drive the phenomenon - that is, to give simple characterization of the conditions (in terms of the measurement variables) that determine when an object is in one class rather than another. These two goals are not exclusive. Most often, they will include both accurate prediction and understanding. An important criterion for a good classification procedure is that it not only produce accurate classifiers (within the limits of the data) but that it also provide insight and understanding into the predictive structure of the data.

A. PROBLEM DEFINITION

In general classification problems, the goal is the same: Given a set of measurements on a case or object, find a systematic way of predicting what class it is in. In any problem, a classifier or a classification rule is a systematic way of predicting what class a case is in. [Ref. 1]

To give a more precise formulation, arrange the set of measurements on a case in some preassigned order, e.g., take the measurements to be $x_1, x_2, \ldots$, where, say, x_1 is the temperature of a patient and x_2 is the blood pressure, etc. Define the measurements $(x_1, x_2, \ldots)$ made on a case as the measurement vector x corresponding to the case. Take the measurement space to be defined as containing all possible measurement vectors.

Suppose that the cases or objects fall into J classes. Number the classes $1,2,\ldots J$ and let C be the set of classes, that is, $C = \{1,\ldots,J\}$. The CART provides a systematic way of predicting class membership in C to every measurement vector x in $\mathcal{X}$. That is, given any $x \varepsilon \mathcal{X}$, the rule assigns x to one of the classes $\{1,\ldots,J\}$.

Definition: A classifier or classification rule is a function d(x) defined on $\mathcal{X}$ so that for every x, d(x) is equal to one of the numbers $2,1,\ldots,J$. Another way of looking at a classifier is to define A_j as the subset of $\mathcal{X}$ on which $d(x)=J$; that is, $A_j = \{x; d(x)=J\}$.

The sets $A_1,\ldots,A_J$ are disjoint and $\mathcal{X} = U_j A_j$; thus the $\{A_j\}$ form a partition of $\mathcal{X}$. This gives the following equivalent definition:

A classifier is a partition of $\mathcal{X}$ into J disjoint subsets $A_1, \ldots, A_J$, $\mathcal{X} = U_j A_j$ such that for every $x \varepsilon A_j$ the predicted class is J.

Classifiers are not constructed whimsically. They are based on past experience. In systematic classifier construction, past experience is summarized by a "learning sample." This consists of the measurement data on N cases observed in the past together with their actual classification.

(1) Definition. A learning sample consists of data $(x_1, J_1), \ldots (x_n, J_n) \ldots$ on N cases where $x_n \in \mathcal{X}$ and $J_n \in \{1, \ldots, J\}$, $n=1, \ldots, N$. Two general types of variables can appear in the measurement vector:

(2) Definition: A variable is called ordered or numerical if its measured values are real numbers. A variable is categorical if it takes values in a finite set not having any natural ordering.

For example, a categorical variable might be a color and could take values in the set {red,blue,...}, while age can be treated as numerical variable.

To summarize up to this point, we conclude that in order to build a classifier or classification rule, we need historical data, i.e., a "learning sample." Then we use this classifier to predict the classes of a new set of data with unknown classes.

Tree structured classifiers, or more correctly, binary tree structured classifiers, are constructed by repeated splits of subsets of $\mathcal{X}$ into descendent subsets, beginning with $\mathcal{X}$ itself, the split of a subset should be selected so that the data in each of the descendant subsets are "purer"

(fewer variables) than the data in the parent subset. This process of splitting continues until we reach subsets with acceptable level of impurity. These subsets are called terminal subsets. The terminal subsets form a partition of $\mathcal{X}$. Each terminal subset is designated by a class label. There may be one, two, or more terminal subsets with the same class label. The partition corresponding to the classifier is obtained by uniting together all terminal subsets corresponding to the same class.

Thus, the entire construction of a tree revolves around three elements:

(1) The selection of the splits.

(2) The decision of when to declare a node terminal or to continue splitting.

(3) The assignment of each terminal node to a class.

B. CART GOALS AND CHARACTERISTICS

(1) Accurate prediction.

(2) Uncover predictive relationships in the problem. What variables or interactions of variables drive the phenomenon.

In addition to these goals announced by designers, the following characteristics are possessed by CART:

(1) Can be applied to any data structure by a proper formulation of questions.

(2) With standard data structure it handles both orderable and categorical variables in a natural way.

(3) With standard data structure it is invariant under all monotone transformations of the orderable variables (only uses the order information).

(4) The prediction rule has a simple form that is efficiently stored and executed (Binary tree representation).

(5) Handles nonhomogenious (nonlinear) relationships between the variables (makes powerful use of conditional information), especially in large data sets.

(6) Automatic stepwise variable selection and complexity reduction.

(7) In classification, it gives, with no additional effort, an estimate for misclassification risk associated with each classified object.

(8) The output provides considerable data analytic information, easily understood and interpreted (Binary tree representation).

(9) It has proved a powerful tool in a wide variety of applications in many fields.

III. SEQUENCE OF SOLUTION STEPS

A. INTRODUCTION

This chapter is concerned with the explanation of the steps followed by the CART-BUILD program to solve the classification problem. As we said before, two types of binary trees are built by CART, classification trees and regression trees. Indeed the procedure for building these two types of trees are completely different. Thus for each of these types, the sequence of steps is introduced, followed by a numerical example.

Recall that tree structured classifiers, or, more correctly, binary tree structured classifiers, are constructed by repeated splits of subsets of X into two descendant subsets, beginning with X itself. This process, for a hypothetical six-class tree, is pictured in Figure III-1. In Figure III-1 X_2 and X_3 are disjoint, with $X = X_2 \cup X_3$. Similarly, X_4 and X_5 are disjoint with $X_4 \cup X_5 = X_2$, and $X_6 \cup X_7 = X_3$. Those subsets which are not split, in this case X_6, X_8, X_{10}, X_{11}, X_{12}, X_{14}, X_{15}, X_{16}, and X_{17}, are called terminal subsets. This is indicated by a rectangular box; nonterminal subsets are indicated by circles.

The terminal subsets form a partition of X. Each terminal subset is designated by a class label. There may be two or more terminal subsets with the same class label. The

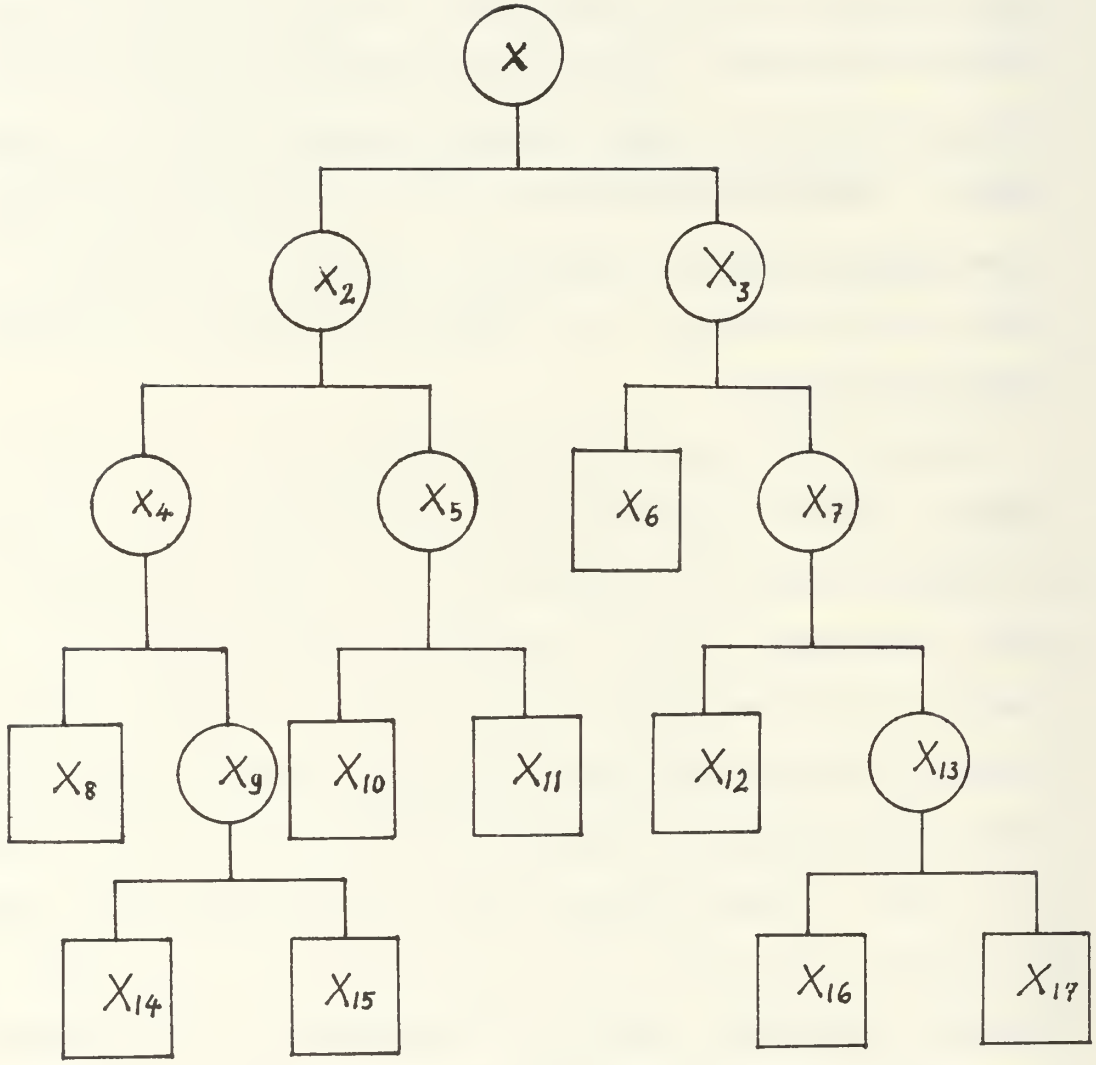


Figure III-1. A Hypothetical Six-Class Tree

partition corresponding to the classifier is obtained by putting together all terminal subsets corresponding to the same class, thus,

$A_1 = X_{15}$ $\qquad$ $A_2 = X_{11} \cup X_{14}$

$A_3 = X_{10} \cup X_{16}$ $\qquad$ $A_4 = X_6 \cup X_{17}$

$A_5 = X_8$ $\qquad$ $A_6 = X_{12}$

The splits are formed by conditions on the coordinates of x = (x1,x2,.....). For example, split 1 of X into X_2 and X_3 could be of the form:

$$X_2 = \{ x \; ; \; x_4 \leq 7 \} \quad , \quad X_3 = \{ x \; ; \; x_4 > 7 \}$$

The tree classifier predicts a class for the measurement vector x in this way: from the definition of the first split, it is determined whether x goes into X_2 or X_3 . For example, suppose x goes into X_2, then from the definition of split 2, it is determined whether x goes into X_4 or X_5. When x finally moves into a terminal subset, its predicted class is given by the class label attached to that terminal subset.

At this point we change terminology to that of tree theory. From now on,

A node t = a subset of X

and

The root node t_1 = X

Terminal subsets become terminal nodes, and nonterminal subsets are nonterminal nodes.

The following terms and notations will be utilized within this chapter:

N : Learning sample size.

N_j : Number of cases in class j in the learning sample.

$\{\pi(j)\}$: The set of prior probabilities estimated either from the data as N_j/N , or supplied by the user.

$N(t)$: Number of cases in node t.

$N_j(t)$: Number of class j cases in node t.

$P(j,t) = \pi(j)N_j(t)/N_j$: Is the resubstitution estimate for the probability that a case will both be in class j and fall into node t.

$P(t) = \Sigma_j P(j,t)$: Is the resubstitution estimate for the probability that any case falls into node t.

$P(j/t) = P(j,t)/P(t)$: Is the resubstitution estimate of the probability that a case is in class j given that it falls into node t.

$i(t)$: The impurity measure of node t.

$\tilde{T}$: The set of terminal nodes.

B. CLASSIFICATION TREES

Four elements were needed in the initial tree growing procedure:

(1) A set W of binary questions of the form $\{Is\ x \in A?\}$, $A \subset X$.

(2) A goodness of split criterion $\Phi(s,t)$ that can be evaluated for any split s of any node t.

(3) A stop splitting rule.

(4) A rule for assigning every terminal node to a class.

This procedure is called "the initial procedure" because it has since been modified. The new one will be presented in B.2.d. Sections B.2.a. through B.2.c. present the initial procedure in detail, followed by the justification of the modification.

1. The Standard Set of Questions

If the data have standard structure (fixed dimensionality), the class W of questions can be standardized. Assume that the measurement vectors have the form $x = (x_1, \ldots, x_m)$, where M is the fixed dimensionality and the variables $x_1, \ldots, x_m$ can be a mixture of numerical and categorical types. The standardized set of questions W is defined as follows:

(1) Each split depends on the value of only a single variable.

(2) For any numerical variable x_m, W includes all questions of the form {IS $x_m \leq D$?} for all D ranging over $(-\infty, \infty)$.

(3) If x_m is categorical, taking values, say, in $\{b_1, b_2, \ldots, b_L\}$, then W includes all questions of the form {IS $x_m \in \tilde{S}$?}, as $\tilde{S}$ ranges over all subsets of $\{b_1, b_2, \ldots, b_L\}$.

There can only be a finite number of distinct splits of the data. For example, if x_1 is numerical, then the data points in the learning sample contain at most N distinct values $x_{1,1}, x_{1,2}, \ldots, x_{1,N}$ of x_1. There are at most N different splits generated by the set of questions {IS $x_1 \leq C$?}

These are given by {IS $x_l \leq C_n$?}, n=1,...,N'≤N, where the C_n are taken halfway between consecutive distinct data values of x_l.

For a categorical variable x_m, since $\{x_m \in S\}$ and $\{x_m \notin S\}$ generate the same split with t_L and t_R reversed, if x_m takes on L distinct values, then 2^{L-1} splits are defined on the values of x_m.

At each node the tree algorithm searches through the variables one by one, beginning with x_l and continuing up to x_m. For each variable it finds the best split. Then it compares the M best single variable splits and selects the best of the best.

2. The Splitting and Stop-Splitting Rules

The goodness of split criterion was originally derived from an impurity function. [Ref. 1]

(1) Definition 1. An impurity function is a function defined on the set of all J-tuples of numbers $P_1,...,P_J$) satisfying $P_J \geq 0$, j=1,...,J, $\Sigma_j P_j = 1$ with the properties.

(a) Φ is a maximum only at the point $(\frac{1}{j},\frac{1}{j},...,\frac{1}{j})$,

(b) Φ achieves its minimum only at the points $(1,0,...,0)$, $(0,1,0,...,0)$,...,$(0,0,...,0,1)$,

(c) Φ is a symmetric function of $P_1,...,P_j$.

(2) Definition 2. Given an impurity function Φ, define the impurity measure i(t) of any node t as

$$i(t) = \Phi(p(1/t), ..., p(J/t)) .$$

If a split s of a node t sends a proportion P_R of the data cases in t to t_R and the proportion P_L to t_L, define the decrease in impurity to be

$$\Delta i(s,t) = i(t) - P_R i(t_R) - P_L i(t_L) \quad .$$

Then take the goodness of split $\Phi(s,t)$ to be $\Delta i(s,t)$. It is easy to see that selecting the splits that maximize $\Delta i(s,t)$ at each node t is equivalent to selecting those splits that minimize the overall tree impurity I(T).

Various splitting rules could be used. Two are used by CART. One rule uses the GINI index of diversity as a measure of node impurity, i.e.,

$$i(t) = \sum_{i \neq j} P(i/t)P(j/t) = 1 - \sum_j P^2(j/t) \quad .$$

The other is the TOWING rule: At a node t, with s splitting t into t_L and t_R, choose the split s that maximizes

$$\frac{P_L P_R}{4}\left[\sum_j \left| P(j/t_L) - P(j/t_R) \right|\right] \quad .$$

This later rule is not related to a node impurity measure.

By way of contrast, the initial stop-splitting rule was simple (and unsatisfactory). One simply set a threshold B>0, and declared a node t terminal (stop-splitting) if

$$\max_{s \in S} \Delta i(s,t)\, P(t) < B \quad .$$

3. The Class Assignment Rule

A class assignment rule assigns a class $j \in \{1,\ldots,J\}$ to every terminal node $t \in \tilde{T}$. The class assigned to node $t \in \tilde{T}$ is denoted by $j(t)$.

(1) Definition: True Misclassification Rate. Given a set of classes C, and a classifier, that is, given a function d(x) on $\mathcal{X}$ taking values in C, let R*(d) be the "true misclassification rate"; taking $(x,y), x \in \mathcal{X}$, $y \in C$, to be a new sample (independent of the learning sample) from the probability distribution P(A,j); i.e.,

$$P(x \in A, y = j) = P(A, j) \quad ,$$

then,

$$R^*(d) = P(d(x) \neq y) \quad .$$

(2) Definition: Resubstitution Estimate R(d). Define the indicator function Q(.) to be 1 if the statement inside the parentheses is true, otherwise zero, then

$$R(d) = \frac{1}{N} \sum_n Q(d(x_n) \neq j_n)$$

is the resubstitution estimate of the "true misclassification rate."

For any prior probabilities and class assignment rule $j(t)$, $\sum_{j \neq j(t)} P(j \mid t)$ is the resubstitution estimate of the probability of misclassification given that a case falls into node t.

We take as our class assignment rule the rule that minimizes this estimate, i.e., the class assignment rule is given by:

$$\text{If } p(j|t) = \max_i p(i|t) \text{ , then } j^*(t) = j \quad .$$

If the maximum is achieved for two or more different classes, assign j*(t) arbitrarily as any one of the maximizing classes.

Using this rule we get:

Definition: The resubstitution estimate r(t) of the probability of misclassification, given that a case falls into node t, is

$$r(t) = 1 - \max p(j|t) \quad .$$

Then the resubstitution estimate for the overall misclassification rate R*(T) of the tree classifier T is

$$R(T) = \sum_{t \in \tilde{T}} r(t)\, p(t) \quad .$$

Now, suppose that the threshold stopping rule is used, with the threshold set so low that every terminal node has only one data point. Then the P(j/t) are all zero or 1, and the resubstitution estimate R(t) for the misclassification rate is zero. In general, R(t) decreases as the number of terminal nodes increases. The more you split, the better you think you are doing.

Unfortunately, when the previous procedure was applied, and the trees generated were tested using test samples, poor results were obtained. The real misclassification rates were high in spite of the corresponding resubstitution estimates were unrealistically low.

It was proved that the resubstitution estimate R(t) was not an accurate estimate of the real misclassification rate, besides, depending on the thresholding, the splitting was either stopped too soon at some terminal nodes or continued too far in other parts of the tree.

A satisfactory resolution came only after a fundamental shift in focus. Instead of attempting to stop the splitting at the right set of terminal nodes, one should continue the splitting until all terminal nodes are very small. The result is a large tree, but then one can prune (recombine) this large tree upward, obtaining a decreasing sequence of subtrees. Then by using a more accurate estimate for the misclassification rate, one can pick out that subtree having the lowest estimated misclassification rate. This process is a central element in the methodology and is covered in the following section.

4. Right Sized Trees and Honest Estimates

The poor accuracy of R(t), and the problems associated with the use of a threshold level as a stopping rule led to the conclusion that looking for the right stopping rule was the wrong way of looking at the problem. [Ref. 1] A

more satisfactory procedure was found consisting of two key elements:

(1) Prune instead of stop. Grow a tree that is much too large and prune it upward in the "right way" until we finally cut back to the root node.

(2) Use more accurate estimates of $R^*(t)$ to select the right sized tree from among the pruned subtrees.

a. The Pruning Process

(1) Definition 1: A node t' lower down on the tree is called a descendant of a higher node t if there is a connected path down the tree leading from t to t' . Then, also, it is called an ancestor of t'.

(2) Definition 2. A branch T_t of T with root node $t \varepsilon T$ consists of the node t and all descendants of t in T.

(3) Definition 3. Pruning a branch T_t from a tree T consists of deleting from T all descendants of t, that is, cutting off all of T_t except its root node. The tree pruned this way will be denoted by $T-T_t$.

The first step is to grow a very large tree, T_{max}, by letting the splitting procedure continue until all terminal nodes are either small or pure, or contain only identical measurement vectors. Here, pure means that the node cases are all in one class. The best way of growing this initial tree would be to continue splitting until each terminal node contained exactly one sample case. The compromise method adopted for growing a sufficiently large initial tree T_{max} , specifies a number N_{min} and continues

splitting until each terminal node either is pure, or satisfies $N(t) \leq N_{min}$, or contains only identical measurement vectors.

The second step uses the minimal cost-complexity pruning procedure to get a sequence of subtrees T_1, T_2, T_3, ... with progressively fewer terminal nodes.

Before introducing the minimal cost complexity pruning procedure, let us examine the idea behind it.

Definition: For any subtree $T \subset T_{max}$, define its complexity as $|\tilde{T}|$, the number of terminal nodes in T. Let $\alpha \geq 0$ be a real number called the complexity parameter and define the cost-complexity measure $R_\alpha(T)$ as

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}| \quad .$$

Thus, $R_\alpha(T)$ is a linear combination of the cost of the tree and its complexity. If we think of α as the complexity cost per terminal node, $R_\alpha(T)$ is formed by adding to the misclassification cost of the tree a cost penalty for complexity.

Now, for each value of α, find that subtree $T(\alpha) \subset T_{max}$ which minimizes $R_\alpha(T)$, i.e.,

$$R_\alpha(T(\alpha)) = \min_{T \subset T_{max}} R_\alpha(T) \quad .$$

If α is small, the penalty for having a large number of terminal nodes is small and $T(\alpha)$ will be large. For instance, if T_{max} is so large that each terminal node

contains only one case, then every case is classified correctly; $R(T_{max}) = 0$, so that T_{max} minimizes $R_\alpha(T)$. As the penalty α per terminal node increases, the minimizing subtree $T(\alpha)$ will have fewer terminal nodes. Finally, for α sufficiently large, the minimizing subtree $T(\alpha)$ will consist of the root node only, and the tree T_{max} will have been completely pruned.

Although α runs through a continuum of values, there are at most a finite number of subtrees of T_{max}. Thus, the pruning process produces a finite sequence of subtrees $T_1, T_2, T_3, \ldots$ with progressively fewer terminal nodes.

Now, the steps of the minimal cost complexity procedure are as follows:

(1) Get T_1 from T_{max}: Let t_L, t_R be any two terminal nodes in T_{max} resulting from a split of the immediate ancestor node t. It is easy to prove that $R(t) \geq R(t_L) + R(t_R)$. If $R(t) = R(t_L) + R(t_R)$, then prune off t_L and t_R. Continue this process until no more pruning is possible. The resulting tree is t_1.

(2) Get T_2 from T_1: For each node $t \in T_1$, calculate the value of the function $g_1(t)$ defined by:

$$g_1(t) = \begin{cases} \dfrac{R(t) - R(T_t)}{|\tilde{T}| - 1} & , \quad t \notin \tilde{T}_1 \\ \\ +\infty & , \quad t \in \tilde{T}_1 \end{cases} .$$

Then define the weakest link $\tilde{t}_1$ in T_1 as the node such that:

$$g_1(\tilde{t}_1) = \min_{t \in T_1} g_1(t)$$

and put

$$\alpha 2 = g_1(\tilde{t}_1) \quad .$$

Define a new tree $T_2 \subseteq T_1$ by pruning away the branch $T_{\tilde{t}_1}$, that is,

$$T_2 = T_1 - T_{\tilde{t}_1} \quad .$$

(3) Using T_2 instead of T_1, repeat the same procedure in the last step to find T_3, α_3, then T_4, α_4, ... and so on.

b. The Best Pruned Subtree

The method of pruning discussed results in a decreasing sequence of subtrees $T_1, T_2, \ldots, \{t_1\}$. The problem is now reduced to selecting one of these as the optimum-sized tree.

If the resubstitution estimate $R(T_k)$ is used as a criterion, the largest tree T_1 would be selected. But if one had an "honest" estimate $R(\hat{T}_k)$ of the misclassification cost, then the best subtree T_{ko} could be defined as the subtree that minimizes $\hat{R}(T_k)$; i.e.,

$$\hat{R}(T_{ko}) = \min_k \hat{R}(T_k) \quad .$$

Two methods of estimation are used by CART: Use of an independent test sample, and cross-validation. The

independent test sample approach is computationally more efficient and is preferred when the learning sample contains a large number of cases. Cross-validation is computationally more expensive, but makes more effective use of all cases and gives useful information regarding the stability of the tree structure.

(1) Test Sample Estimates. The idea used is very simple; select a fixed number $N^{(2)}$ of cases at random from the learning sample L to form a test sample L_2. The remainder L_1 form the new learning sample.

The tree T_{max} is grown using only L_1 and pruned upward to give the sequence $T_1, T_2, \ldots, \{t_1\}$. That is, the $\{T_k\}$ sequence of trees is constructed without ever seeing any of the cases in L_2.

Now take the cases in L_2 and drop them through T_1. Each tree T_k assigns a predicted classification to each case in L_2. Since the true class of each case in L_2 is known, the misclassificatioan cost of T_k operating on L_2 can be computed. This produces the estimate $R^{ts}(T_k)$.

In more detail, denote by $N_j^{(2)}$ the number of class j cases in L_2. For T any one of the trees $T_1, T_2, \ldots,$ take $N_{ij}^{(2)}$ to be the number of class j cases in L_2 whose predicted classification by T is class i.

If the prior probabilities are estimated from the data, use L_2 to estimate them as $\pi(j) = N_j^{(2)} / N^{(2)}$.

In this case:

$$R^{ts}(T) = \frac{1}{N^{(2)}} \sum_{i,j} C(i/j)\, N_{ij}^{(2)} \quad ,$$

where C(i/j) is the cost of misclassifying a class j object as a class i object.

The last expression has a simple interpretation. Compute the misclassification cost for every case in L_2 dropped through T and then take the average.

The last sample estimates can be used to select the right sized tree T_{ko} by the rule:

$$R^{ts}(T_{ko}) = \min_k R^{ts}(T_k)$$

(2) Cross-Validation Estimates. In V-fold cross-validation, the original learning sample L is divided by random selection into V subsets, L_v, v=1,..., V, each containing the same number of cases (as nearly as possible). The vth learning subsample is

$$L^{(v)} = L - L_v \quad , \quad v = 1, \ldots, V \quad ,$$

where the minus sign is used to mean "remove", so that $L^{(v)}$ contains the fraction (V-1)/V of the total data cases.

In V-fold cross validation, V auxiliary trees are grown together with the main tree grown on L. The vth auxiliary is grown using the learning sample $L^{(v)}$. Start by

growing the overly large trees $T_{max}^{(v)}$, v=1,..., V, and T_{max} as well.

Recall that a sequence of (α_k) values was calculated corresponding to the sequence of (T_k) subtrees. Define:

$$\alpha'_k = \sqrt{\alpha_k \, \alpha_{k+1}} \quad ,$$

e.g.,

$$\alpha'_1 = \sqrt{\alpha_1 \, \alpha_2} \quad .$$

With $\alpha = \alpha'_1$, prune $L^{(v)}$, v=1,..., V, to get the sequence of optimally pruned subtrees $(T^{(v)}(\alpha))$.

Drop L_v down the corresponding optimally pruned subtree. For every value of v, i, j, define:

$N_{ij}^{(v)}$ = the number of class j cases in L_v classified as i by the tree $T^{(v)}(\alpha)$, and set

$$N_{ij} = \sum_v N_{ij}^{(v)} \quad ,$$

so that N_{ij} is the total number of class j test cases classified as i. Each case in L appears in one and only one test sample L_v, therefore, the total number of class j cases in all test samples is N_j, the number of class j cases in L.

If the prior probabilities are estimated from the data, then

$$R^{CV}(T_k) = \frac{1}{N} \sum_{i,j} C(i/j) \, N_{ij} \quad .$$

Now the rule for selecting the right sized tree is: Select the tree T_{ko} such that

$$R^{cv}(T_{ko}) = \min_{k} R^{cv}(T_k) \quad .$$

Then use $R^{cv}(T_{ko})$ as an estimate of the misclassification cost.

5. Numerical Example

Consider the problem of classifying the prices of the cars in a warehouse of used cars. The owner partitioned the cars into four classes which, to him, were important combinations of the variables: car model, fuel consumption rate, and mileage in thousands of miles. Given data from 15 cars, we want to build a classification tree using these data as a learning sample. TABLE III-1 contains data codes, and TABLE III-2 contains the coded data.

SOLUTION:

N = 15; learning sample size.
J = 4 ; number of classes.

Variable Information:

Variable Name	Type	Min Value	Max Value
MODEL	categorical (3)	---	---
FUEL	categorical (3)	---	---
MILEAGE	numerical	20	90

j (classes)	N_j	π_j
1	4	4/15 = 0.267
2	2	2/15 = 0.133
3	5	5/15 = 0.333
4	4	4/15 = 0.267

TABLE III-1. DATA CODES

MODEL	MODEL CODE	FUEL (CONS.RATE)	CODE
A	1	LOW	1
B	2	MEDIUM	2
C	3	HIGH	3

TABLE III-2. LEARNING SAMPLE

CASE	MODEL	FUEL	MILEAGE	CLASS
1	1	1	30	1
2	2	1	40	1
3	3	3	70	4
4	1	2	70	3
5	2	2	60	4
6	3	1	80	3
7	2	2	30	3
8	1	3	80	4
9	2	1	90	4
10	3	2	60	3
11	3	3	50	3
12	1	2	50	2
13	2	3	30	2
14	3	1	20	1
15	1	1	40	1

a. Step 1: Growing the Maximum Tree

At each split, assume that the cases answering "YES" to a question split to the right node (t_R) while the rest of the cases split to the left node (t_L).

For the first node t_1:

$$P(j/1) = \pi_j$$

$$i(t) = 1 - \sum_j P^2(j/t) \quad ; \quad GINI$$

$$= 0.729$$

Using

$$\Delta i(s,t) = i(t) - P_R i(t_R) - P_L i(t_L) \quad ,$$

TABLE III-3 is produced, where $\Delta i(s,t)$ is the improvement in the tree impurity.

TABLE III-3. CANDIDATE SPLITS AT NODE 1

QUESTION #	SET OF QUESTIONS	$N(t_L)$	$N(t_R)$	$i(t_L)$	$i(t_R)$	IMPROV.
1	Is Model = 1 ?	10	5	0.7	0.72	0.023
2	Is Model = 2 ?	10	5	0.7	0.72	0.023
3	Is Model = 3 ?	10	5	0.74	0.56	0.049
4	Is Fuel = 1 ?	9	6	0.642	0.5	0.144
5	Is Fuel = 2 ?	10	5	0.7	0.56	0.076
6	Is Fuel = 3 ?	11	4	0.694	0.625	0.053
*7	Is Mileage $\leq$ 55? (i.e. median range)	7	8	0.490	0.625	0.167

From TABLE III-3 it is clear that the Question 7 provides the most improvement relative to node 1 (marked with *), and the first split will be according to Mileage ≤ 55.

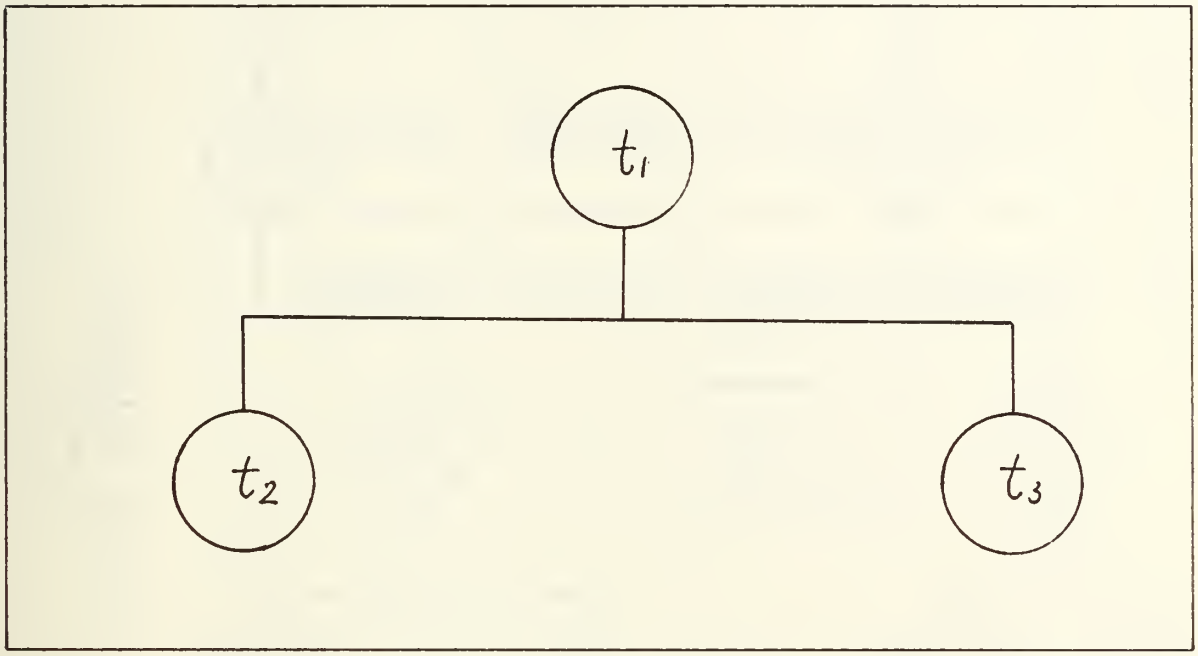


Figure III-2. The Result of Node 1 Splitting

Now, cases of learning sample fall into node 2 are: cases 3, 4, 5, 6, 8, 9, and 10, cases fall into node 3 are cases 1, 2, 7, 11, 12, 13, 14, and 15.

Using the same procedure, the operation of selecting the best splitting could be done at every nonterminal node. Figure III-3 shows the generated maximum tree (T_{max}), while TABLE III-4 contains the calculations required at each node.

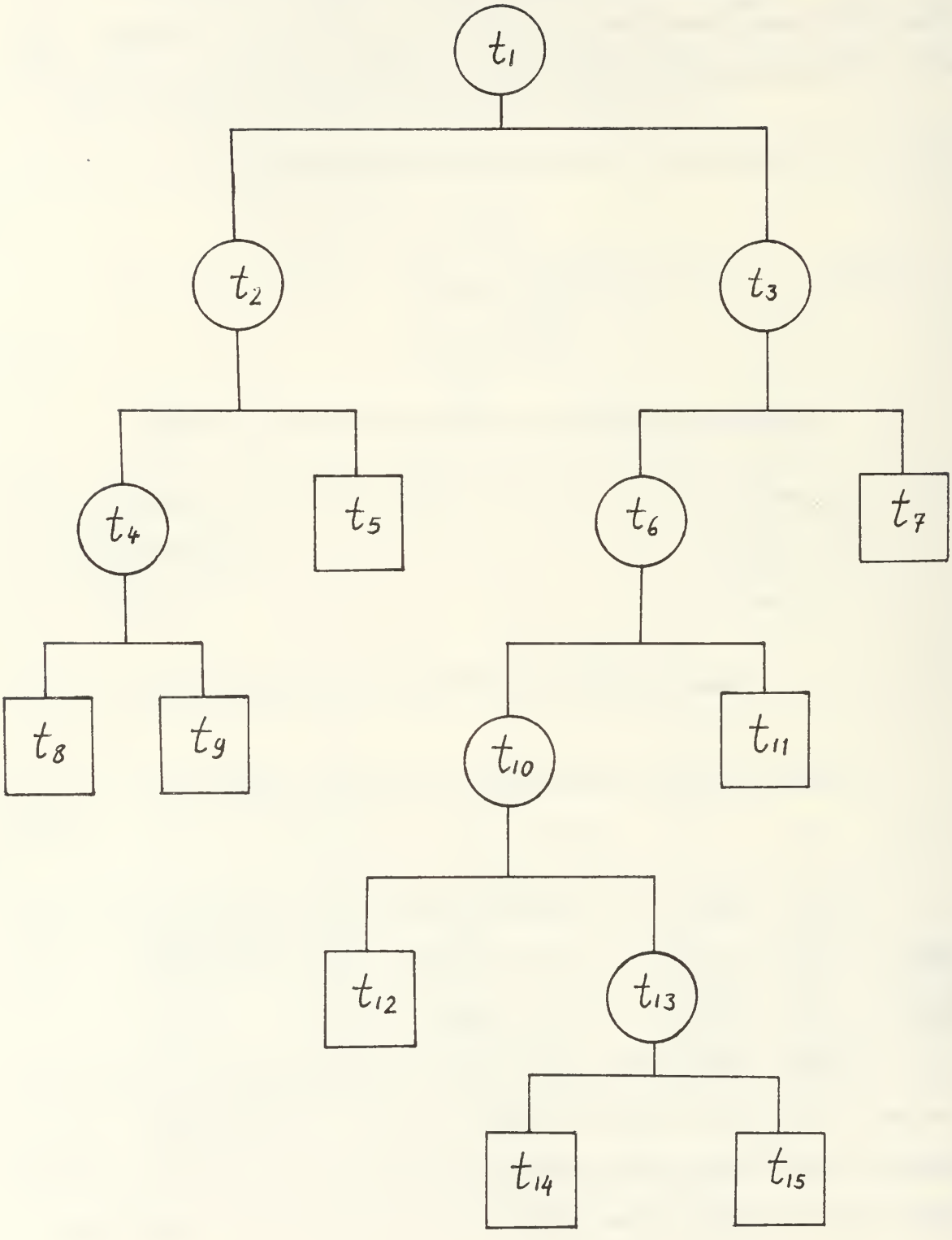


Figure III-3. Maximum Tree

TABLE III-4. SPLITTING SELECTION

NODE	SPLIT	SET OF QUESTIONS	$N(t_L)$	$N(t_R)$	$i(t_L)$	$i(t_R)$	$\Delta i(s,t)$	
t_2	1	Is model = 1 ?	5	2	0.48	0.5	0.0043	
	2	Is model = 2 ?	5	2	0.48	0.0	0.147	*
	3	Is model = 3 ?	4	3	0.375	0.444	0.085	
	4	Is fuel = 1 ?	5	2	0.48	0.5	0.0043	
	5	Is fuel = 2 ?	4	3	0.375	0.444	0.085	
	6	Is fuel = 3 ?	5	2	0.48	0.0	0.147	
	7	Is mileage $\leq$ 72.5 ?	3	4	0.444	0.5	0.014	
t_3	1	Is model = 1 ?	5	3	0.64	0.444	0.058	
	2	Is model = 2 ?	5	3	0.56	0.667	0.025	
	3	Is model = 3 ?	6	2	0.611	0.5	0.042	
	4	Is fuel = 1 ?	4	4	0.5	0.0	0.375	*
	5	Is fuel = 2 ?	6	2	0.5	0.5	0.125	
	6	Is fuel = 3 ?	6	2	0.5	0.5	0.125	
	7	Is mileage $\leq$ 37.5 ?	4	4	0.625	0.625	0.0	
t_4	1	Is model = 1 ?	3	2	0.444	0.5	0.013	
	2	Is model = 3 ?	2	3	0.5	0.444	0.013	
	3	Is fuel = 1 ?	4	1	0.5	0.0	0.08	
	4	Is fuel = 2 ?	3	2	0.444	0.0	0.213	
	5	Is fuel = 3 ?	3	2	0.0	0.0	0.480	*
	6	Is mileage $\leq$ 72.5 ?	2	3	0.5	0.444	0.013	
t_6	1	Is model = 1 ?	3	1	0.444	0.0	0.167	*
	2	Is model = 2 ?	2	2	0.5	0.5	0.0	
	3	Is model = 3 ?	3	1	0.444	0.0	0.167	
	4	Is fuel = 2 ?	2	2	0.5	0.5	0.0	
	5	Is fuel = 3 ?	2	2	0.5	0.5	0.0	
	6	Is mileage $\leq$ 37.5 ?	2	2	0.5	0.5	0.0	

b. Step 2: Pruning

(1) Get T_1 from T_{max}:

- Calculate $R(t) = p(t)r(t)$, $t \in T_{max}$
- If $R(t) = R(t_L) + R(t_R)$, $t_R, t_L \quad T_{max}$

then prune off t_L and t_R.

TABLE III-5 contains the whole required calculations. It is clear that the condition for pruning is not satisfied at any terminal nodes, so $T_1 = T_{max}$.

(2) Get T_2 from T_1

- For each node $t \in T_1$, calculate

$$g_1(t) = \begin{cases} \dfrac{R(t) - R(T_t)}{|\tilde{T}| - 1} & , \quad t \notin \tilde{\tilde{T}}_1 \\ +\infty & , \quad t \in \tilde{\tilde{T}}_1 \end{cases}$$

- Find the weakest link $\bar{t}_1$ as the node such that

$$g_1(\bar{t}_1) = \min_{t \in T_1} g_1(t)$$

TABLE III-6 contains the calculations of $g_1(t)$. From this table we can get:

$$\bar{t}_1 = t_{10} \quad , \quad \alpha 2 = g(t) = 0.0335$$

Figure III-4 shows the generated tree T_2.

TABLE III-5. REQUIRED CALCULATIONS TO GET T_1 FROM T_{max}

NODE	P(j,t)				P(t)	r(t)	R(t)
	j=1	2	3	4			
t_1	0.267	0.133	0.333	0.267	1	0.667	0.667
t_2	0	0	0.2	0.267	0.467	0.428	0.2
t_3	0.267	0.133	0.133	0	0.533	0.5	0.266
t_4	0	0	0.2	0.133	0.333	0.4	0.133
t_5	0	0	0	0.133	0.133	0	0
t_6	0	0.133	0.133	0	0.266	0.5	0.133
t_7	0.267	0	0	0	0.267	0	0
t_8	0	0	0.2	0	0.2	0	0
t_9	0	0	0	0.133	0.133	0	0
t_{10}	0	0.067	0.133	0	0.2	0.335	0.067
t_{11}	0	0.067	0	0	0.067	0	0
t_{12}	0	0	0.067	0	0.067	0	0
t_{13}	0	0.067	0.067	0	0.134	0.5	0.067
t_{14}	0	0.067	0	0	0.067	0	0
t_{15}	0	0	0.067	0	0.067	0	0

(3) Using the same procedure, we can get T_3 from T_2, and then T_4 from T_3,... and so on, until T_6 which is only the root node. TABLES III-6 through III-10 contains the whole calculations, and Figures III-4 through III-8 show the corresponding trees.

c. Step 3: Selection of the Right Sized Tree

Now, we have six trees, and it is required to choose the optimum-sized tree. Using one of the two methods, independent test sample and cross-validation, one can select the optimum tree. Both methods are explained in Section B.2.d.(2)

C. REGRESSION TREES

The tree-structured approach in regression is simpler than in classification. The same impurity criterion used to grow the tree is also used to prune the tree. Besides this, there are no prior probabilities to deal with.

In regression, a case consists of data (x,y) where x falls in a measurement space X and y is a real-valued number. The variable y is usually called the response or dependent variable. The values in x are referred to as the independent variables or carriers.

A tree-structured predictor is similar to a tree-structured classifier. The space X is partitioned by a sequence of binary splits into terminal nodes. In each terminal node t, the predicted response value y(t) is constant.

TABLE III-6. REQUIRED CALCULATIONS TO GET T_2 FROM T_1

$t \varepsilon T_1$	$R(t)$	$R(T_t)$	$\lvert\tilde{T}_t\rvert - 1$	$g_1(t)$
t_1	0.667	0	7	0.0953
t_2	0.2	0	2	0.1
t_3	0.266	0	4	0.0665
t_4	0.133	0	1	0.133
t_6	0.133	0	3	0.0443
t_{10}	0.067	0	2	0.0335*
t_{13}	0.067	0	1	0.067

* min $= \alpha_2$

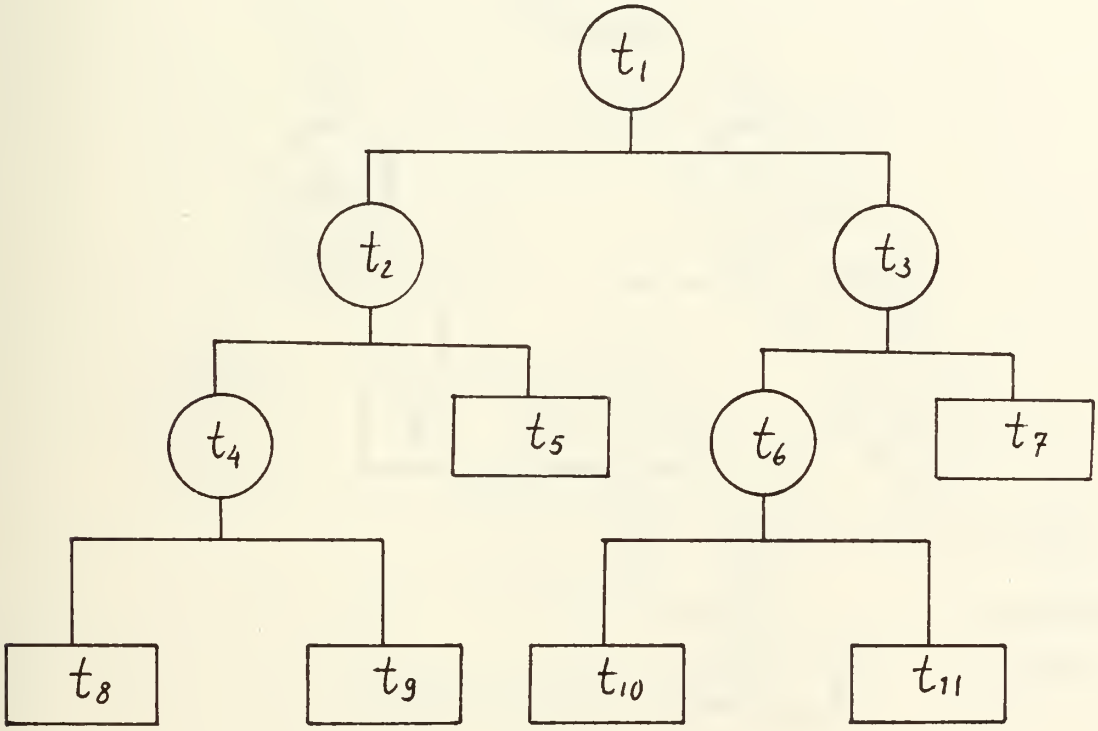


Figure III-4. Tree T_2

TABLE III-7. REQUIRED CALCULATIONS TO GET T_3 FROM T_2

$t \varepsilon T_2$	R(t)	$R(T_t)$	$\lvert\tilde{T}_t\rvert - 1$	$g_2(t)$
t_1	0.667	0.067	5	0.12
t_2	0.2	0	2	0.1
t_3	0.266	0.067	2	0.1
t_4	0.133	0	1	0.133
t_6	0.133	0.067	1	0.066*

* min = α 3

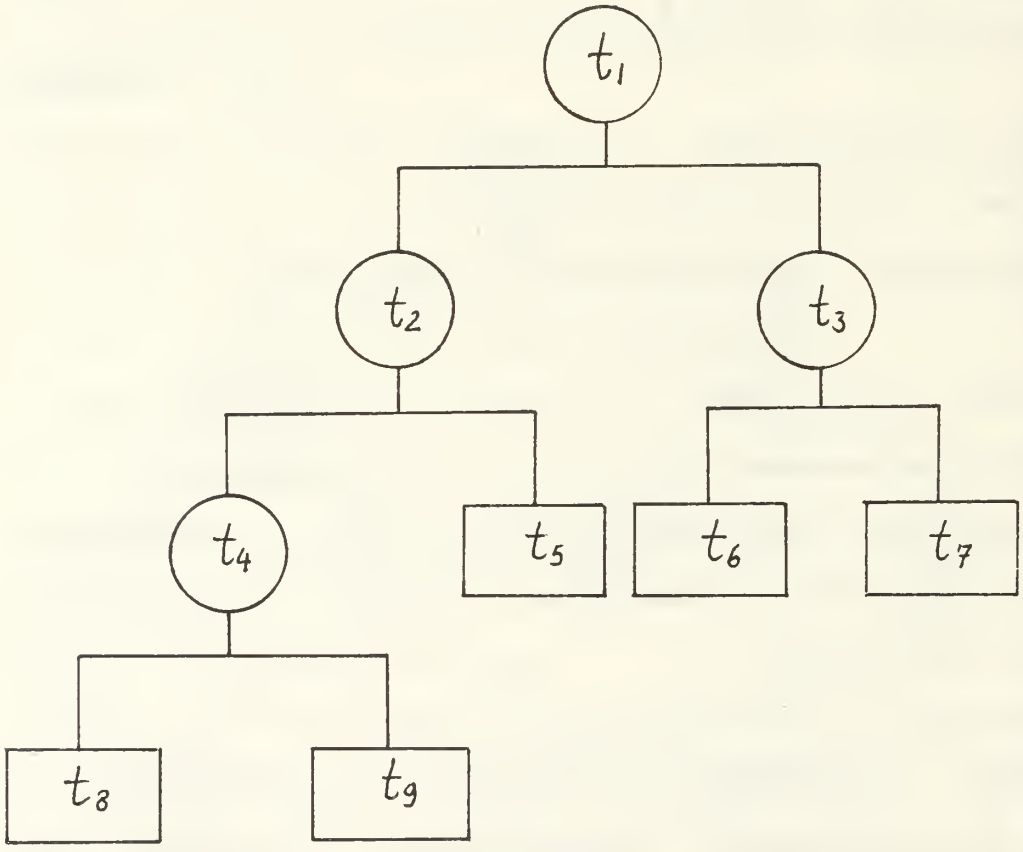


Figure III-5. Tree T_3

TABLE III-8. REQUIRED CALCULATIONS TO GET T_4 FROM T_3

$t \varepsilon T_3$	$R(t)$	$R(T_t)$	$\lvert \tilde{T}_t \rvert - 1$	$g_3(t)$
t_1	0.667	0.133	4	0.1335
t_2	0.2	0	2	0.1*
t_3	0.266	0.133	1	0.133
t_4	0.133	0	1	0.133

* min = $\alpha 4$

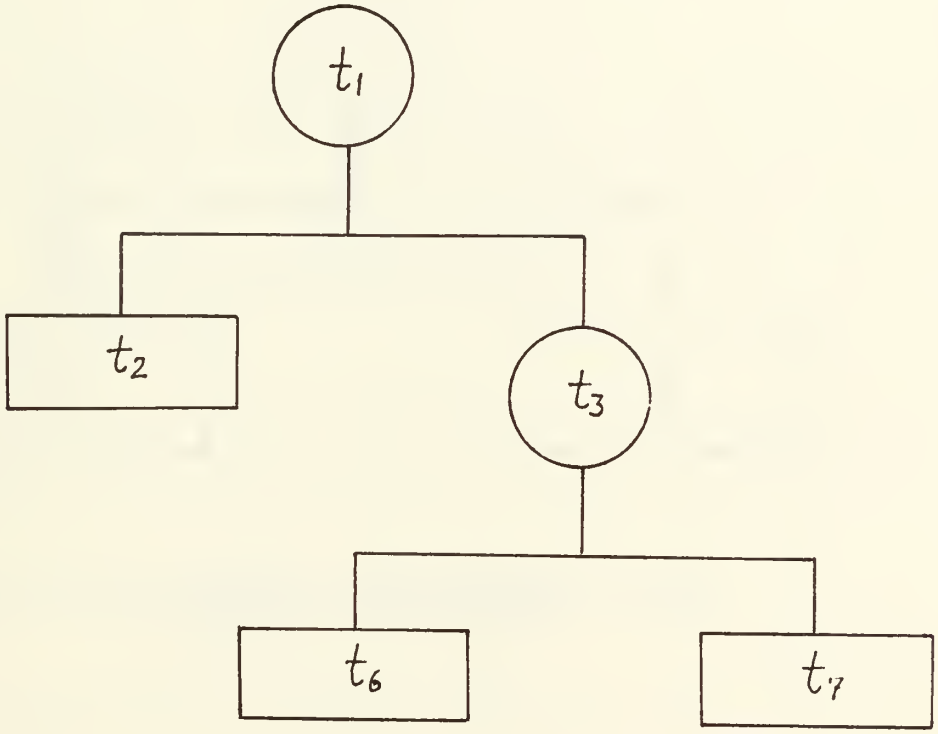


Figure III-6. Tree T_4

TABLE III-9. REQUIRED CALCULATIONS TO GET T_5 FROM T_4

$t \varepsilon T_4$	$R(t)$	$R(T_t)$	$\lvert\tilde{T}_t\rvert - 1$	$g_4(t)$
t_1	0.667	0.333	2	0.167
t_3	0.266	0.133	1	0.133*

* min = 5

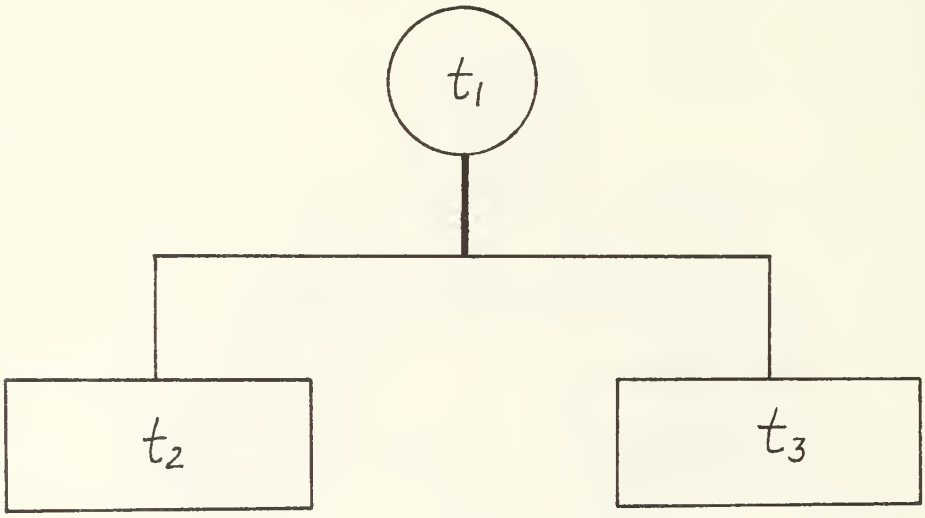


Figure III-7. Tree T_5

TABLE III-10. REQUIRED CALCULATION TO GET T_6 FROM T_5

$t \varepsilon T_5$	$R(t)$	$R(T_t)$	$\lvert\tilde{T}_t\rvert - 1$	$g_5(t)$
t_1	0.667	0.466	1	0.201

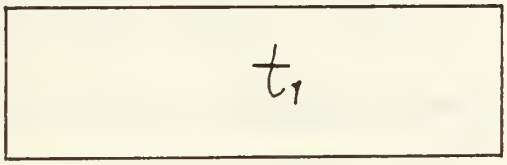


Figure III-8. Tree T_6 (One Terminal Node)

Starting with a learning sample, three elements are necessary to determine a tree predictor:

(1) A rule to select a split at every intermediate node.

(2) A rule for determining when a node is terminal.

(3) A rule for assigning a value y(t) to every terminal node t.

Before proceeding, let us define the following notation:

t	: given node
y(t)	: predicted value at t
y	: true value of object in t
y-y(t)	: error

$$\text{cost}(y-y(t)) = \begin{cases} (y-y(t))^2 & \text{, LS regression} \\ |y-y(t)| & \text{, LAD regression} \end{cases}$$

LS	: least square
LAD	: least absolute deviations.

a. Splitting, Stop-Splitting, and Assignment Rules

(1) Splitting Rule: Choose the split to miminize:

$$\sum_{N(t_L)} COST(y - y(t)) + \sum_{N(t_R)} COST(y - y(t))$$

(2) Stop-Splitting Rule: Node t is a terminal node if:

$$N(t) \leq N_{min}$$

(3) Assignment Rule: Choose y(t) to minimize:

$$\sum_{N(t)} \text{cost}\,(y-y(t))$$

LS: $y(t) = \underset{N(t)}{\text{Average}}\ y$

LAD: $y(t) = \underset{N(t)}{\text{Median}}\ y$

b. Pruning

The method used to select a tree is exactly the same as that used to select a classification tree. First, a large tree T_{max} is grown by successively splitting until for every $t \in \tilde{T}_{max}$, $N(t) \leq N_{min}$.

The minimal error-complexity pruning is done exactly as minimal cost-complexity pruning in classification. The result is a decreasing sequence of trees

$$T_1 , T_2 , \ldots , \{t_1\} .$$

To select the right sized tree from this sequence, test sample estimates, and cross-validation estimates are used.

IV. REAL DATA: MANPOWER DATA

A. GENERAL

No one can deny the importance of accurate loss forecasts in military manpower systems. The following lines are extracted from Tucker, D. D. [Ref. 2] in order to amplify this point.

> In military manpower systems, most personnel flows are initiated by the creation of vacancies within the system. Vacancies are largely the result of losses. Losses in the paygrade hierarchy trigger promotions from lower grades. Vacancies also generate a need for new accessions to replenish or expand the force. Because accessions and promotions share this direct relationship with losses, accession and promotion flows, as well as manpower budgeting, are dependent on accurate loss forecasts. An underestimate of losses can lead to too few accessions, too few promotions, erroneous budget projections, and ultimately, a readiness problem. In contrast, an overestimate can cause too many accessions, delays in promotions, and potential budget overruns.

A goal of this paper is to apply the new technique, CART, to estimate and forecast the loss rates in the Marine Corps officer manpower system.

Losses and loss rates are composed of flows of officers from particular cells. The cells are characterized by a cross-classification of military occupational speciality (MOS), length of service (LOS), and grade. When the flow goes from one cell to another within the Marine Corps, then it is referred to as a strength loss. Cell changes are not losses from the Marine Corps to the civilian labor market,

however, from an occupational field manager's point of view, changes to another occupatiional field are the same as a strength loss. Similarly, for the receiving occupational field, the changes are represented by a gain. Changes such as promotion to a higher pay grade or a lateral cell shift due to an officer aging from one length of service to the next, are not recognized as losses.

Losses from within the Marine Corps to the civilian labor market can be voluntary or involuntary. Voluntary losses occur when officers resign, retire, or are released by choice. Involuntary losses occur due to discharge, death, disability, release from active duty, and retirements.

B. DATA SOURCE

The data used in this project was obtained from a summary data file from the Commander, Navy Personnel Research and Development Centre (NPRDC). The summary data file was generated from two source files: the Headquarters Master File (HMF) and the Quarterly Statistical Transaction File (STATS). The Headquarters Master File was used to produce historical officer inventories as of the beginning of the fiscal year. Inventories were generated for each fiscal year from 1977 through fiscal year 1983. The inventories were identified by distinct characteristics. These characteristics were Military Occupational Specialty (MOS), grade, and Length of Service (LOS).

The Quarterly Statistical Transaction File was used to generate historical losses. Within this file, if the Type Transaction Code indicated a loss, then the effective date of action field would specify the year and month of the loss. Losses were classified into eight categories for each fiscal year 1977 through 1983. The losses were further classified into distinct elements by MOS, grade, and LOS.

C. DATA FORMAT

The summary data file classified the Marine Corps officer inventory into 40 military occupation specialties, 10 grade levels, 31 lengths of service, and 8 loss categories. The data format is defined by TABLE IV-1 and TABLES of Codes IV-2 and IV-3. [Ref. 3]

D. DATA PREPARATION AND EXTRACTION OF SAMPLES

Before proceeding in the data preparation, let us define some useful terms:

(1) Attrition: Any departure from the Marine Corps by an officer.

(2) Central Attrition Rate: The number of leavers during the period who were in this class when they left divided by the average number in this class during the period.

To calculate the central attribute rate for a particular cell, the following policy was followed:

(1) Let, t=1,...,T refer to the years 1977,1978,...

(2) Let, y(t) = number of losses in year t.

(3) INV(t) = inventory in the beginning of year t.

TABLE IV-1. DATA FORMAT

COLUMN	DATA
1-2	YEAR - (FISCAL) (1977...1983)
3-4	MOS GROUP (0...39)
5-6	GRADE
7-8	LENGTH OF SERVICE (0...30)
9-13	BEGINNING OF YEAR INVENTORY
14-53	LOSS COUNTS (TYPES 1...8)

TABLE IV-2. GRADE AND TYPE LOSS CODES

CODE	GRADE
0	WARRANT OFFICER (W-1)
1	CHIEF WARRANT OFFICER (CWD-2)
2	CHIEF WARRANT OFFICER (CWD-3)
3	CHIEF WARRANT OFFICER (CWD-4)
4	SECOND LIEUTENANT
5	CAPTAIN
7	MAJOR
8	LIEUTENANT COLONEL
9	COLONEL

CODE	TYPE LOSS
1	VOLUNTARY RESIGNATION
2	VOLUNTARY RETIREMENT
3	INVOLUNTARY - DEATH
4	INVOLUNTARY - DISCHARGE
5	INVOLUNTARY - DISABILITY
6	RELEASE FROM ACTIVE DUTY
7	DISABILITY RETIREMENT
8	INVOLUNTARY RETIREMENT

TABLE IV-3. MILITARY OCCUPATIONAL SPECIALTIES (MOS)

DATA CODE	ACTUAL MOS	MOS TITLE
00	UN	UNKNOWN
01	01	PERSONNEL AND ADMINISTRATION
02	02	INTELLIGENCE
03	03	INFANTRY
04	04	LOGISTICS
05	08	FIELD ARTILLERY
06	11	UTILITIES
07	13	ENGINEER, CONSTRUCTION, AND EQUIPMENT
08	14	DRAFTING, SURVEYING, AND MAPPING
09	15	PRINTING AND REPRODUCTION
10	18	TANK AND AMPHIBIAN TRACTOR
11	21	ORDNANCE
12	23	AMMUNITION AND EXPLOSIVE ORDNANCE DISPOSAL
13	25	OPERATIONAL COMMUNICATIONS
14	26	SIGNALS INTELLIGENCE/GROUND ELECTRONIC WARFARE
15	28	DATA/COMMUNICATIONS MAINTENANCE
16	30	SUPPLY ADMINISTRATION AND OPERATIONS
17	31	TRANSPORTATION
18	33	FOOD SERVICE
19	34	AUDITING, FINANCE, AND ACCOUNTING
20	35	MOTOR TRANSPORT
21	40	DATA SYSTEMS
22	41	MARINE CORPS EXCHANGE
23	43	PUBLIC AFFAIRS
24	44	LEGAL SERVICES
25	46	TRAINING AND AUDIOVISUAL SUPPORT
26	55	BAND
27	57	NUCLEAR, BIOLOGICAL, AND CHEMICAL
28	58	MILITARY POLICE AND CORRECTIONS
29	59	ELECTRONICS MAINTENANCE
30	60	60XX
31	61	AIRCRAFT MAINTENANCE
32	63	AVIONICS
33	65	AVIATION ORDNANCE
34	68	WEATHER SERVICE
35	70	AIRFIELD SERVICES
36	72	AIR CONTROL, AIR SUPPORT, AND ANTI-AIR WARFARE
37	73	AIR TRAFFIC CONTROL
38	75	PILOTS AND NAVAL FLIGHT OFFICERS
39	99	IDENTIFYING MOS AND REPORTING MOS

OVERLOOKED IN ORDER TO BE COMPATIBLE WITH THE PROJECT FROM NPRDC.

(4) Let, N(t) = maximum of y(t) and the average inventory in year t and t+1 and computing their average:

(5) Let R = sum of y(t) divided by the sum of N(t), (both sums over t) represents the actual central attrition rate for the particular cell.

In the rest of this paper, we will refer to the actual central attrition rate by simply actual attrition rate or actual loss rate.

1. Data Preparation

As we said before, two data files are required as input files to execute CART programs: a learning sample data file and a test (validation) data file. To create these files, the following steps were followed:

(1) The summary data file (16093 records) was divided, according to the fiscal year, into two files:

- LEARN file: contains data for fiscal years from 1977 to 1981.
- TEST file: contains data for fiscal years from 1981 to 1983.

(2) The last two files were used as input files to a program to calculate the actual attrition rate for each cell. Neglecting the cells with zero actual rate, two output files were generated, MLEARN (1401 records) and MTEST (885 records). The number of records neglected in the first output file is 2996 records. The FORMAT of these two files is as follows:

COLUMN	DATA
1-2	MOS group
3-4	Grade
5-6	Length of Service
7-13	Attrition Rate

2. Extraction of Samples

Each of the last two files, MLEARN and MTEST, was divided into 9 data subsets, each data subset contains one of the general MOS categorization (TABLE IV-4). At this point, we had 18 files (TABLE IV-5), and we were ready to execute CART programs.

Before going on to the next chapter, someone may ask why we divided each of the two files MLEARN and MTEST, into 9 files? The answer will be presented in Chapter VII in detail, but for the time being, the need to do this is one of the limitations of the CART program.

TABLE IV-4. GENERAL MOS CATEGORIZATION

NO.	GENERAL MOS CATEGORIES	DATA MOS CATEGORY
1	GROUND COMBAT	03, 05, 10
2	COMBAT SUPPORT	07, 13, 20
3	AVIATION	38
4	MANAGEMENT	01, 02, 16, 19, 23, 24, 28
5	TECHNICIANS	30, 36, 37
6	INFORMATION	04, 14, 21
7	ADMINISTRATION	09, 17, 18, 22, 25, 26, 39
8	ORDNANCE GROUP	06, 11, 12, 33, 35
9	SKILLED	08, 15, 27, 29, 31, 32, 34

TABLE IV-5. DATA FILES

GENERAL MOS CATEGORY	FILE NAME	PURPOSE	RECORDS	NO. OF MOS CATEGORIES
1	BUILD1	LEARNING SAMPLE	195	3
	TEST1	VALIDATION	149	3
2	BUILD2	LEARNING SAMPLE	191	3
	TEST2	VALIDATION	125	3
3	BUILD3	LEARNING SAMPLE	78	1
	TEST3	VALIDATION	67	1
4	BUILD4	LEARNING SAMPLE	403	7
	TEST4	VALIDATION	267	7
5	BUILD5	LEARNING SAMPLE	136	3
	TEST5	VALIDATION	79	3
6	BUILD6	LEARNING SAMPLE	116	3
	TEST6	VALIDATION	75	3
7	BUILD7	LEARNING SAMPLE	88	7
	TEST7	VALIDATION	49	7
8	BUILD8	LEARNING SAMPLE	85	5
	TEST8	VALIDATION	46	5
9	BUILD9	LEARNING SAMPLE	71	7
	TEST9	VALIDATION	27	7

More details about the neglected and considered records in the learning sample files are included in Appendix J.

V. CART PROGRAMS: IMPLEMENTATION AND DESCRIPTION OF RESULTS

A. GENERAL

CART can be executed only under the batch operating system on the IBM 3030 at NPS (MVS). To execute it, appropriate job control language commands (JCL) must be used together with the data to construct the code to be submitted to MVS. These (JCL) commands are general and used with all jobs submitted to MVS, so these commands will not be described here but will be included in the complete programs introduced.

To set up a batch run, we have to do two things:

(1) Set up the execution commands, which includes telling CART where certain input and output files are located.

(2) Construct two input files, the data specifications file and the options file (if new data is being run down a previously constructed tree, then at most an options file is necessary).

Here is a summary of the files that CART uses.

1. The Data File

The data must be in a text file, and have the form of a matrix whose rows are the cases and whose columns are the variables. Variables may be categorical or numerical and may have missing values. The data file must be formatted to these specifications by the user. In particular all categoricals must be coded to have serial integer values.

2. The Options and Data Specifications Files

These are input files constructed by the user in character form. The data specifications file describes the data and may be used in all CART runs on the same or similar data sets. The options file contains the values of the options and parameters chosen for the current run. Once set up, the options file may be modified (by editing) to get different analyses on the same data set. There are three different forms of the options file: one for classification, one for regression, and one for case-by-case output from a previously constructed tree.

3. Output Files

Each CART run produces an output file containing the results of the run in character form. If a run constructs a tree, then this tree can be saved in machine readable form together with the data specifications file. Other data to be run down this tree are assumed by CART to conform to the data specifications stored with the tree.

CART contains three major programs called BUILD, TEST, and CASE. These do different things:

(1) BUILD - constructs a new tree.

(2) TEST - runs data down a previously constructed tree with summary output.

(3) CASE - runs data down a previously constructed tree producing case-by-case output.

The files needed by these three programs are different. Also there are different logical unit numbers given to these files in the FORTRAN statements opening them.

4. The Syntax of the Data Specifications and Options Files

(1) Each data attribute or option is entered by typing a keyword of which only the first three letters are read. The letters may be typed in either upper or lower case.

(2) If a value needs to be assigned to the keyword, it has to be separated from the word by a space(s) or equal sign.

(3) Spaces and new lines are always delimiters. The same line can contain a number of keywords and their assigned values as long as they are separated by spaces. Commas may only occasionally be used as delimiters as noted below. All blank lines, tabs, and new lines are ignored except where they delimit keywords and values.

(4) Specifications or options can be given in any order. There are a few exceptions to this noted below.

(5) If a keyword for a specification or option does not appear in the file, then CART will assign the default value (if one exists). Thus, if a default value is acceptable to the user, there is no need to enter its keyword into the file.

(6) Occasionally, an * has to be typed in to signal the end of the values assigned to a keyword.

B. PROGRAMS REQUIREMENTS

In this section all files required by each program are described in detail, particularly the contents of input files.

1. The BUILD Program (CARTMVS1)

FILES NEEDED	UNIT NUMBER
INPUT FILES	
data	7
data specifications	1
option specifications	2
test set (option)	8
OUTPUT FILES	
printable output	9
machine readable tree	4

a. Data Specifications File

The following is a sample data specifications file for the Marine Corps officer manpower data described in the previous chapter, followed by the comments on the keywords, the values assigned to them, and certain other features.

```
VARIABLES = 4
NAMES  (1) MOS  (2) GRADE  (3) LOS  (4) RATE*
CATEGORIES  40(1) 10(2)*
MINIMUMS    0(1) 0(2)*
MISSING     27(1)31(1)32(1)*
DATA
FORMAT
(3F3.0, F7.3)
FILE
/*            (end of specifications file)
```

(1) VARIABLES. This is the first exception to the general rule that options or specifications can be entered in any order. The specification of the number of variables must be the first thing entered in this file.

(2) NAMES. The numbers in parentheses refer to the order in which variables appear in the data set. Variables not named will be referred to in the output by these numbers. The variables do not have to be named in order and they do not have to be separated by spaces. If the names are entered in the same order as the variables appear in the data, then the numbers are not needed, thus an acceptable format is:

```
NAMES   MOS   GRADE   LOS   RATE*.
```

In the data specifications file the asterisk must be used to indicate the end of the assigned values to NAMES, CATEGORIES, MINIMUMS, and MISSING.

(3) CATEGORIES. This indicates that the first variable (MOS) is a 40-valued categorical, and the second variable (GRADE) is a 10-valued categorical. By default, CART will assume that all variables not listed under the keyword CATEGORIES are numerical. The general format for entering the categories information is:

(a) B(M) : variable M has B categories

(b) B(M-N) : variables M through N have B categories

(c) B(M,N) : variables M and N both have B categories.

For instance:

```
CAT
    3(5,7)2(11-15)8(19)
or
    3(5,7)2(11-15)8(19)
or
    3(5,7)2(11,12,13,14,15)8(19)
```

are all read as -- variables 5 and 7 are 3-valued categoricals, variables 11, 12, 13, 14, and 15 are 2-valued categoricals, and variable 19 is an 8-valued categorical. Combinations are allowed so 3(17,5-9) is a valid expression meaning that variables 5, 6, 7, 8, 9, and 17 are 3-valued categoricals. If variables have been named, these names can be used in the specifications above instead of the variable numbers, so

that 40(MOS) is an acceptable assignment. Notice that commas can be used inside the parentheses as spacers. So can spaces, so that 2(11 12 13 14 15) is an acceptable specification.

(4) MINIMUMS. This indicates that the minimum values of the first and second variables are zeros. CART must know the minimum value of every variable declared as a categorical. The default value for a minimum is 1. If the minimum of a categorical variable is not 1 then its minimum must be entered following the keyword "minimums." The format for entering this information is:

(a) K(M) : variable M has the minimum value K.

(b) K(M-N) : variable M through N have minimum value K.

(c) K(M,N) : variables M,N have minimum value K.

Recall that K must always be an integer value. Thus,

the expression:

```
min
   0(5,3,8)2(11-13),-1(7)*
```

will be read as -- variables 5, 3, and 8 have minimum value 0, variables 11, 12, and 13 have minimum value 2, and variable 7 has minimum value -1. Combinations of the form 0(9,3-6) are valid and again, names can be used to designate variables. Note that commas are again acceptable as spacers inside of parentheses. This is the general rule for their use.

CART assumes categoricals have consecutive integer values. If a categorical is coded as 0 or 4 and it is declared as 2-valued categorical with minimum value 0 then CART will die. However, if it is declared as a 5-valued categorical with minimum value zero and if the variable is not the response variable, then CART will accept it as having values (0,1,2,3,4). The spurious additional categorical values will be harmless.

(5) MISSING. This indicates that the first variable (MOS) has the missing value codes 27,31,32. CART does not have a default missing value code. The format for entering missing value codes is:

(a) X(M) : variable M has missing value code X.

(b) X(M-N) : variables M through N have the missing value code X.

(c) X(M,N) : variables M and N have the missing value code X.

Again, combinations such as X(1,5,7-10) are valid as is the use of variable names. The missing value may be in any FORTRAN format. If scientific notation is used, it must be in a form that contains E or e and the sign of the exponent, i.e., 9.9E+10.

(6) DATA. These three lines give the FORTRAN format for one data case. Note that the entire format specification must be enclosed by parentheses. Also, only F_._ formats are allowed and no integer (I) declarations must appear. CART can read formatted or free format data. Free format is the default and if a format specification is not made, then CART will attempt to read the data file assuming that it can be read in free format form. The word "format" must appear following DATA but above the line containing the FORTRAN formatting.

(7) FILE. This is the second exception. The keyword file must be the last keyword in the data specifications file. In MVS the line following FILE only contains a * in the second column. The data file must be defined by a DD statement in the execution commands.

b. The Options File

There are two different forms of the options file for that program. An example will be given on each and comment made on them.

(1) Regression

The example below is the options file for a BUILD run to construct a regression tree using the Marine Corps officer data. Unlike data specifications, every option has a default value.

```
RESPONSE = RATE

REGRESSION

------------------- primary options ----------------

SPLIT = LAD

CROSS = 5

SERULE = 1

LINEAR SIZE = 50 PARAM = .2*

COPIOUS

COMPLEXITY = 50.0

NTREES = 20

------------------- secondary options ---------------

LIMITS

    ATOM = 5

    SAMPLE = 100

    SURROGATES 10 5

    COMPETITORS = 5

    LEARNSIZE = 1500

    TESTSIZE = 0

    NODEMAX = 500

    CATMAX = 100

    LINMAX = 50

    STACK = 50
    *
/*          (end of options file)
```

Comments on Keywords Used.

(1) REGRESSION. This indicates that we want to grow a regression tree. In this case the response variable must be numerical (not categorical).

(2) SPLIT. CART can do either least-squares regression or least-absolute deviation regression. The value "LS" for SPLIT selects the former, the value "LAD" selects the latter. The default is "LS."

(3) CROSS. CART has three methods for estimating the regression error.

(a) The first is cross-validation. If a keyword corresponding to an estimation method does not appear in the file, then CART, as the default, will use 10-fold cross-validation. If we want to use V-fold cross-validation with V not equal to 10, then we type in the keyword CROSS followed by a V. In our example we have selected 5-fold cross-validation.

(b) The second method is using test set estimation by typing in the keyword TEST. If we do not type in anything else, then CART will select .33 of the data cases at random to serve as a test sample. If we want a different proportion selected, say 0.25, then we follow the word TEST by "proportion = 0.25."

If we already have a test set contained in a separate data file that we want CART to use, then the keyword TEST followed by the word "file" must be used.

(c) The third method of estimation is resubstitution. To specify this method use the keyword EXPLORATORY.

(4) SERULE. This selects the number of standard errors to be used in the tree selection rule. The value assigned may be any non-negative real number. The default is 1.0 so that actually, this keyword did not need to appear in the above file.

(5) DELETE. This keyword does not appear in our example, but when it is used associated with some variable names, it indicates that we want to delete these variables from the regression. If this keyword is omitted, then CART will include all variables in the data specifications in the regression. The general format for assigning values is:

DELETE-(M,N,...)*

where M,N,..., are the names or numbers of the variables to be deleted. Notations such as:

DELETE-(K,L,M-N)*

are acceptable and will result in the deletion of variables K, L, and M through N, where again K, L, M, and N can be names or numbers.

(6) LINEAR. The appearance of this keyword tells CART to do linear combination splits on the nodes. If this word does not appear CART will find splits on only one variable at a time. If this keyword appears, then values should be assigned to the keywords SIZE and PARAM. In the above example, SIZE has been set to equal 50. This means that CART will use linear combination splits on all notes containing 50 or more cases, but only single variable splits if the node contains less than 50 cases. The default value is 32000. This default value is a signal for no linear combination splits. The value of PARAM governs the backwards deletion of variables from the linear split. The default value is .2. If this option is selected, then the value assigned to the keyword LINMAX, which appears after the keyword LIMITS, must be changed from its default value.

(7) COPIOUS. If cross-validation is selected as the estimation method, either by default or by typing CROSS, then there is another option available which can be called by the keyword COPIOUS. This option outputs all node information for all of the largest cross-validation trees. If COPIOUS is typed into the file, but cross-validation is not selected as the estimation method, then CART will ignore the word. This option may generate truly copious output and should be used judiciously.

(8) COMPLEXITY. This is the value of the initial complexity parameter: its default value is zero.

(9) NTREES. This is the maximum number of trees printed out in the tree sequence: its default is 100.

(10) LIMITS. This keyword must appear before the list of secondary option keywords. Recall that there is no need for them to be on separate lines. We could have typed "LIMITS ATOM=5 SAMPLE=100 ...etc."

(11) ATOM. The minimum size below which a node will not be split. The default is 5.

(12) SAMPLE: The node size above which subsampling will be used. The default is 20000.

(13) SURROGATES. This refers to two things. The first is the maximum number of surrogate splits used to fill in missing values. The second is how many surrogate splits will be printed out for each node. The default values are 10 and 5 respectively. The order in which these two numbers are entered does not matter. The larger of the two will automatically be set equal to the number of surrogate splits used for missing values and the smaller for the number of splits printed out.

(14) COMPETITORS. This is the number of competing splits printed out for each node split: the default is 5.

(15) LEARNSIZE. The maximum number of cases allowed in the learning sample: the default is 20000.

(16) TESTSIZE. The maximum number of cases to be put into the test sample. The default is 20000 if TEST is selected as a primary option. Otherwise the default is zero.

(17) NODEMAX. If the value of K is assigned to this word, then CART sets aside storage space for K nodes in the largest tree it grows. The default is 750.

(18) CATMAX. If this value is set at K, then CART sets storage space for K categorical primary and surrogate splits in the largest tree grown. The default is 1000.

(19) LINMAX. If this value is K then CART allots storage space for K linear combination splits in the largest tree grown. The default is 0. If we have selected LINEAR as a primary option, LINMAX must be set to a positive value.

(20) STACK. If this is set to K then CART allots storage space for trees of approximately depth K. The default is 750.

(21) * (Asterisk). The * after the STACK value indicates the end of LIMITS. Also the values assigned to DELETE and LINEAR must be terminated by *.

How to estimate the values that should be assigned to these options? The answer will be clear in Chapter VI.

(2) Classification

The options file for classification is very similar to that for regression. The major difference is that more primary option information must be given for classification. The secondary options and keywords are exactly the same as for regression except that one more secondary option is added. The sample options file below is for the same data as used before but with the assumption that variable RATE (response variable) is a 10-valued categorical.

```
    RESPONSE = RATE
    CLASSES = 10
    SPLIT = GINI
    PRIORS = EQUAL*
    COST = UNIT*
    SERULE = 1.0
        .
        .
        .
    LIMITS
        .
        .
        .
    STACK = 50
    WEIGHT = 0.0
    *
/*
```

The similarity with the REGRESSION options is apparent, with the same keywords having the same default values, comments will be given only on those keywords that differ.

(1) RESPONSE. This again designates the response variable, in classification, the variable that indicates class membership.

(2) CLASSES. This keyword has two functions. First, its appearance tells CART to grow a classification tree. Secondly, it sets the number of classes in the data, 10 in our example.

(3) SPLIT. The allowable values of SPLIT are:

(a) gini (default): the gini rule with priors altered to incorporate costs.

(b) symgini: the gini rule with symmetrization to incorporate costs.

(c) twoing: the twoing rule with priors altered to incorporate costs

(d) ordered: ordered twoing.

(e) class: class probability.

The response to this keyword sets the type of splitting rule to be used in constructing the tree. For unit misclassification costs, gini and symgini are the same.

(4) PRIORS. This sets the prior class probabilities. The first thing typed in after PRIORS must be one of the three words:

(a) data (default): sets the class J prior equal to the proportion of class J cases in the data set.

(b) equal: sets all priors equal to one divided by the number of classes.

(c) thus: user specified priors.

If either "data" or "equal" is typed, no more information has to be entered. Typing "thus" tells CART that

the user wants to specify the priors and must be followed by the entry of the values of the priors. The general format for entering the values of the priors is:

(a) P(J) : class J has prior value proportional to P.

(b) P(I,J) : classes I and J have prior values proportional to P.

(c) P(I-J) : classes I through J have prior values proportional to P.

Normally, the sum of the class priors has to be 1.0. However, the reason for using the word "proportional" above, is that CART will automatically normalize the values of the priors entered so that they sum to 1.0. It does this by dividing the value entered for each class by the sum of these values. The values of P must be non-negative numbers in any FORTRAN format. Scientific notation may be used, i.e., .15E-1. Just as in data specifications combinations of the above formats are allowable.

(5) COST. The values assigned to COST are the misclassification costs. That is, the costs C(I/J) of classifying a class J case as a class I case. COST must be followed by one of the two words:

(a) unit (default)

(b) thus.

If "unit" is typed no more information needs to be entered. The costs C(I/J) are set equal to one if I is not equal to J, and to zero for I equal to J. If "thus" is typed it signifies that the user wants to specify the misclassification costs and must be followed by the entry of these values. the format for entry is:

X(I/J) : C(I/J) = X
X(I,K/J) : C(I/J) = C(K/J) = X
X(I/K,J) : C(I/K) = C(I/J) = X
X(I-K/J) : C(L/J) = X for all L from I through K
X(I/J-K) : C(I/L) = X for all L from J through K

If any C(I/J) is not specified following "thus", then CART sets it equal to one, if I is not equal to J. CART sets all values of C(J/J) equal to zero even if the user enters a nonzero value.

The costs can be any non-negative numbers in any FORTRAN format or in scientific notation. Combinations of formats are acceptable.

(6) * (asterisk). Entries for both PRIORS and COST have to be ended by *.

(7) WEIGHT. This sets the center cutting exponent. The default is zero.

2. The CASE Program (CARTMVS3)

FILES NEEDED	UNIT NUMBER
INPUT FILES	
new data	7
option specifications	2
machine readable tree	4
OUTPUT FILES	
printable output	9

This program gives case-by-case output only. It is useful in those situations where we want to get predicted responses for data having unknown response, that is, to get a predicted classification or a predicted numerical response. It can also be used to give a case-by-case comparison of predicted with actual response for data with known responses.

It also gives a way to get case-by-case output for the original data set used to construct the tree. That is, construct and save the tree (using BUILD program), then using CASE program, run the original data down the saved tree.

a. The Option Specifications File

The case-by-case option file formats the output. The values given to the various options tell CART the order for printing the cases, which variables to print, and in what format. Here is a sample options file for the Marine Corps officers data used in the data specifications and regression option examples:

```
ORDER = node
CASEID = sequence
VARIABLE (MOS, GRADE, LOS, RATE)*
DECIMALS  3(RATE)*
```

Here are the keywords and their permissable values:

(1) ORDER. This sets the order in which the cases are printed. It must be followed by one of the following words:

   (a) input (default): order by appearance in the data file.

   (b) response: order by magnitude of the predicted response.

   (c) node: order by terminal node number.

(2) CASEID. This allows the user to attach an identification number to each case printed out. Following CASEID one can type either:

   (a) sequence (default): number sequentially starting from 1.

   (b) none: no id numbers will be printed.

(3) VARIABLES. This selects the variables to be printed for each case. Up to nine variables can be printed on a standard printer. This is in addition to the predicted response, terminal node number, and the id

number. The format to use in specifying variables is of the form

VARIABLES(M,N,...)*

where M,N,... are variable names or numbers. The notation (...,M-N,...) will print out all variables from name or number M through name or number N. The order of listing for a case is:

predicted response,
terminal node number, and
id number

followed by either:

M,N,... order in which the variables were listed

or

(default) response variable plus the first eight variables that were not deleted in the tree construction in the order of their appearance in the data file.

Note that if the response variable is missing in the data being run down the tree, then if the default is chosen the missing value code will be printed out for the response variable (but not for the predicted response).

(4) DECIMALS. To make the output more readable, the user can specify the number of decimal places (number of places to the right of the decimal point) to be printed for each variable. The format is:

D(M) : variable M will be printed to D decimal places.

D(M-N) : variables M through N will be printed to D decimal places.

D(M,N) : variables M and N will be printed to D decimal places.

The field for each variable printout is 7 characters wide. In choosing the number of decimal places, one should leave enough space for the decimal point and the integer part of the largest value of the variable. Notice that the variables MOS, GRADE were left out of the specification. This is because categorical variables are automatically printed out in their integer coding. The predicted response will be printed out with the same decimal format as specified for the

response variable. This latter specification can be made even if the user is not requiring that the response variable be printed out. The default for DECIMALS is set by a little algorithm in CART which tries on its own to give tidy output.

(5) * (asterisk). Note that the entries following both VARIABLES and DECIMALS have to be terminated by a *.

3. The TEST Program (CARTMVS2)

FILES NEEDED	UNIT NUMBER
INPUT FILES	
new data	8
machine readable tree	4
OUTPUT FILES	
printed output	9

There are no special files needed by this program.

C. MEMORY MANAGEMENT

CART has a maximum workspace that was set by the programmer who installed CART on the system (about 8000K bytes). The workspace is for the data and a number of arrays that are filled with tree information.

Some of these array sizes are set by the user values assigned to parameter options in options files. These will be outlined below and with some experimentation, can be specified to increase the data space available to CART.

CART occupies about 300-400K bytes, exclusive of the workspace. If the data set is large and contains many variables, then it occupies most of the workspace.

The following are the secondary options effecting the memory managements with our comments.

(1) SAMPLE. Whenever a node is to be split, some scratch space must be set aside for the computations. This space is proportional to the sample size used in splitting the node. In the default mode, this sample size is set equal to the number of cases in the data set. If the subsampling parameter is set lower than the total number of data cases, the scratch space required will be reduced.

(2) NODEMAX. Sets the maximum number of nodes allowed in the largest tree grown. Since a considerable amount of storage must be set aside to describe the characteristics of each node, setting this parameter unrealistically high will significantly reduce the workspace available for data.

(3) SURROGATES. This sets the number of surrogate splits used and stored at each node. The initial default sets a maximum of 10. Setting it lower will free storage space.

(4) CATMAX. More memory is needed to store specifics of splits on a categorical variable than splits on a numerical variable. This option sets up the additional memory space necessary.

(5) LINMAX. Storing linear combination splits also requires more memory that splits on individual numerical variables. This option sets up the additional storage memory for such splits. If the linear combination option is not used, it defaults to zero.

(6) STACK. Some computations in CART need space in proportion to the number of nodes between the root (top) and a terminal node. This option sets the maximum for this number.

Some of the above parameters are difficult to estimate in advance. To assist the user the printout of every run contains, at the bottom, the settings of these options together

with the actual values used in the CART run. Given this information on a number of runs, the user should acquire some instinct as to appropriate settings.

In classification problems with many classes, the tree information will take up a large amount of memory if cross-validation is used. This is because the misclassification matrices have to be stored for every tree in all cross-validation tree sequences.

There are two possible remedies. First, use a test set only. Second, do a preliminary set run to determine the appropriate range of the complexity parameter. Then for the option "COMPLEXITY" in the options file, set the complexity parameter to the lower limit of the range. This will keep the cross-validation trees smaller and less storage will be required.

D. UNDERSTANDING OF RESULTS

The printed outputs of CASE program and TEST program are self explanatory and don't need any additional comments. The following are comments on the contents of the printed output of the BUILD program (CARTMVS1).

The printed output consists of the following segments:

- a header,
- tree sequence,
- tree diagram,
- node information,
- summary information, and
- option settings.

1. Tree Sequence Output

This gives the sequence of pruned subtrees and estimates of their errors. The tree selected by the tree selection rule is indicated by an * following the tree number.

In regression, below the tree sequence is printed out the initial sample variance of the data and the initial average of the response variable (least squares) or the initial mean absolute deviation from the data median, and the initial median of the response variable (least absolute deviation regression).

The relative error estimates in least squares regression are the ratios of the estimated mean squared errors after regression to the initial sample variance. Therefore, the estimated mean squared error after regression can be computed by multiplying the relative error by the initial sample variance.

In least absolute deviation regression, the relative error is the ratio of the estimated mean absolute error after regression to the initial mean absolute deviation. Thus, multiplying the relative error by the initial mean absolute deviation gives the estimated mean absolute error. Note that the tree selected may not be consistent with the printed standard errors. The latter are computed using an estimate of the standard error of the relative error ratio. Tree selection is based on the standard error of the estimate of the numerator only.

The estimated misclassification costs are made relative by dividing them by the initial misclassification cost. The initial misclassification cost is printed out underneath the tree sequence so that the actual estimated misclassification costs can be computed by multiplication just as the regression case.

The initial misclassification cost is computed by predicting all cases to be in the class selected by the class node assignment rule. For instance, in the unit cost case, the class assigned is the class with the largest initial probability. The initial misclassification cost is the sum of the initial probabilities of the other classes. The initial class assignment is also printed out at the end of the tree sequence.

Unlike regression, the printed standard errors of the relativized costs are not estimates of the standard errors of ratios. They are simply the estimated standard errors of the estimated misclassification costs divided by the initial misclassification cost. Therefore, the standard error of numerators can be recovered by multiplication.

2. The Tree Diagram

This gives, in graphical form, a picture of the overall tree topology. It is useful in keeping track of which nodes are ancestors or descendents of one another.

3. Node Information

The node information is mainly self explanatory. The improvement in classification is the decrease in mean residual sum-of-squares or in mean absolute deviation. "C.T." stands for the complexity threshold of the node relative to the selected tree. Thus the node having the lowest "C.T." value would be the node having its descendents pruned in the next smallest tree.

Beneath the node summary is a list of those surrogate splits having the largest measures of association. The number of such splits printed is controlled in the secondary options menu. No matter how many surrogate splits are requested, only those splits with positive measures of association (to 2 decimals) will be printed. An "R" or "S" in front of the split value indicates that the association is with the reversed split (R) or with the split as printed (S).

The competing splits listed below the surrogate splits are those splits on other variables, having the largest improvements (regardless of the association). This information can be important in uncovering alternative tree structures.

4. Summary Information

The summary information consists first of the terminal node summaries, then an overall summary, followed by variable importance ranking. If a test set is used, the terminal node summary contains both test and learning set results

listed separately. In the summary, the misclassification matrix for classes is given separately for the resubstitution estimates and for the cross-validation or test set estimates.

VI. CART-MANPOWER DATA: APPLICATION AND RESULTS

A. INTRODUCTION

This chapter is concerned with two main issues. The first is the options selection in options files required by CART, and the second is the results obtained through the application of CART on the Marine Corps manpower data. Recall that CART contains three programs: BUILD, CASE, and TEST. Options files are needed only by the first two programs.

Options selection is not a simple operation. Often, it is a trial and error operation. It requires a deep understanding of the classification and regression trees technique, in addition to the features of the computer installation.

B. PREPARATION FOR EXECUTION

Before execution of CART programs, the data files, and options files must be prepared in the specific form required.

Execution of the BUILD program requires three files as input: data file, data specifications file, and option specifications file. Each of these files could be in a separate file whose name must be specified in a data definition statement (JCL) in any job needs this file, or all files required could be included in one file associated with the suitable data definition statements. The last case was chosen.

1. Final Step of Data Preparation

Recall from Chapter IV that nine files were prepared to be as input files for the CART-BUILD program. Each of these files contains one of the general MOS categorization groups, e.g., BUILD1 file contains the MOS codes 03, 05, and 10 which are the subgroups of the ground combat group. This means that MOS is a categorical variable that takes values from the set 03,05,10 . However these values, as required by CART programs, must be integers and the difference between any two adjacent values must equal 1. For this reason, all MOS codes within each file are temporarily changed to be serial integers, e.g., BUILD1 file contains now the MOS codes 51, 52, and 53.

2. Options Selected

The following lines are the contents of the options file selected to associate the execution of BUILD program using BUILD1 file as an input, followed by the justification of this selection.

```
RESPONSE = RATE
REGRESSTION
SPLIT = LAD
CROSS  5
LIMITS
    ATOM = 2
    SAMPLE = 100
    NODEMAX = 300
```

CATMAX = 100

STACK = 50

*

By the keyword REGRESSION, we selected regression trees rather than classification trees, because our response variable RATE, is a numerical variable.

The type of splitting is selected to be LAD; the least-absolute deviation regression rather than least-squares regression. The former is specially designed to be used by CART.

The number of cross-validation trees is selected to be 5. The greater the number of cross-validation trees, the greater the accuracy of misclassification estimate, the greater the time of execution, and vice-versa.

The minimum size below which a node will be split is chosen to be 2. The smaller the minimum size, the larger is the maximum tree grown T_{max}, and the larger is the memory required to hold the tree. Choosing greater values for the minimum size of a node can lead to generation of a maximum tree which is smaller than the optimal tree.

The node size above which subsampling will be used is selected to be 100. The greater the selected value, the greater the time of execution and the greater the memory required to do the calculations at each node. If a small value is assigned to the keyword SAMPLE, only a small number of cases will be considered at each node to select the best

split, which in turn can lead to a poor selection for the optimal tree.

The maximum number of nodes of the maximum tree is chosen to be 300. The greater the number selected, the greater the memory allocated to hold the proposed maximum tree, and this can lead to a program fault because of memory limitations. A compromise is required between this number and the minimum node size selected.

CATMAX is assigned a value of 100. This is a trial and error operation.

The maximum depth of the tree, in terms of the number of nodes between the root and terminal nodes, is selected to be 50. It is also a trial and error operation.

It may seem to someone that the options selection is a simple operation. It is a time and effort consuming operation, especially in the case of a large data set and categorical variables that take values from a set of many elements. This point will be discussed in detail in the next chapter.

C. STEPS OF EXECUTION

1. Disk Space Allocation

Before execution of the BUILD progoram, a space must be allocated on the MVS disk storage system to hold the generated tree. A typical program that can be used to allocate six cylinders is as follows:

```
//ALLOCAT JOB (3041,111), 'ALLOCATE JCL', CLASS=A
//*MAIN ORG = NPGVM1.3041P
//  EXEC PGM = IEFBR14
//DD1 DD UNIT = 3330V, MSVGP = PUB4A, DISP = (NEW,CATLG),
//       DISNAME = MSS.S3041.TREE
//
```

2. Execution of CART-BUILD Program

One file containing data, data specifications, and options is constructed instead of three separate files. This file is submitted to MVS by the command SUBMIT. The data contained in this file are those contained in one of the BUILD1,...,BUILD9 files. The following is a typical example of such job.

```
//HMAIN1  JOB(1111,0001),'MY NAME', CLASS = G
//*MAIN ORG=NPGVM1.3041P
//EXEC PGM=CARTMVS1,REGION=5120K
//STEPLIB  DD DISP=SHR,DSN=MSS.SYS3.CART.LOADMOD
           DD DISP=SHR,DSN=SYS1.PP.VFORTLIB
//FT04F001 DD DISP=(OLD,KEEP),DSN=MSS.S3041.TREE
//FT07F001 DD*
65.00.07.  0.0513
65.01.06.  0.6667
      .
      .
      .
  rest of data
      .
      .
67.07.28.  1.0000
/
*
```

```
FILE: MAIN5    BUILD    A1

//FT06F001 DD SYSOUT=A
//FT09F001 DD SYSOUT=A
//FT01F001 DD *
VARIABLES=4
NAMES (1)MOS (2)GRADE (3)LOS   (4)RATE*
CATEGORIES  3(1)10(2)*
MINIMUMS 65(1)0(2)*
MISSING 9(2)*
DATA
FORMAT
(3F2.0,F7.4 )
FILE
/*
//FT02F001 DD *
RESPONSE=RATE
REGRESSION
SPLIT=LS
CROSS FOLD = 10*
TREE
LIMITS
            ATOM=1
            SAMPLE=1000
            LEARNSIZE=200
            NODEMAX=200
            STACK=50
            *
/*
//
```

3. Execution of CART-CASE Program

The goal of this run is the validation of the results obtained from the previous run. The input for this program is one of the 9 files, TEST1,...,TEST9. Data files must be in the same format as those files used to build the corresponding trees.

Selection of options for this run is very simple since it specifies only the format of required output. An example for this run is as follows:

```
//HCASE1 JOB(1111,0001),'MY NAME', CLASS=C
//*MAIN ORG=NPGVM1.3041P
//  EXEC PGM=CARTMVS3, REGION=2048K
//STEPLIB  DD DISP=SHR,DSN=MSS.SYS3.CART.LOADMOD
//         DD DISP=SHR,DSN=SYS1.PP.VFORTLIB
//FT04F001 DD DISP=(OLD,KEEP),DSN=MSS.S3041.TREE
//FT07F001 DD*
65.00.14.
    .
    .
rest of data
67.08.25.
/*
//FT06F001 DD SYSOUT=A
//FT09F001 DD SYSOUT=A
//FT02F001 DD*
ORDER=INPUT
CASEID=SEQUENCE
VARIABLES(RATE,MOS,GRADE,LOS)*
DECIMALS 1(LOS)4(RATE)*
/*
//
```

4. Execution of CART-TEST Program

The goal of this program is to print a summary of results obtained from both of the previous runs. No options

selection, and no input files, except the binary tree, are required. A typical example is as follows:

```
//HTEST1 JOB(1111,1111),'MY NAME',CLASS=C
//*MAIN ORG=NPGVM1.3041P
//  EXEC PGM=CARTMVS2,REGION=2048K
//STEPLIB  DD DISP=SHR,DSN=MSS.SYS3.CART.LOADMON
//         DD DISP=SHR,DSN=SYS1.PP.VFORTL2B
//FT04F001 DD DISP=(OLD,KEEP),DSN=MSS.S3041.TREE
//FT08F001 DD*
66.01.13.  0.1111
    .
    .
    .
 rest of data
    .
    .
    .
67.03.02.  0.0500
/*
//FT06F001  DD SYSOUT=A
//FT09F001  DD SYSOUT=A
//
```

D. RESULTS

The complete printed results are found in the appendix. Here is a summary of results obtained from BUILD and CASE runs.

1. Results of CART-BUILD Run

This run generates two outputs, the first is a binary tree saved on a file in binary format, the second is a printed output. This run was executed nine times. TABLE VI-1 contains a summary for these runs.

It is clear from TABLE VI-1 that LOS is the most important variable affecting the loss rate in all the nine runs. A general conclusion could be reached concerning the relative importance of variables in all runs is as follows:

VARIABLE NAME	RELATIVE IMPORTANCE
LOS	
GRADE	
MOS	

2. Results of CART-CASE Run

The CART-CASE run was executed nine times, each time one of the TEST files was used as an input file. The printed output of this run contains the predicted loss rates in addition to the actual loss rates (if it is required) for all cases, case-by-case, contained in the input files.

TABLE VI-1. SUMMARY OF BUILD PROGRAM RESULTS

GENERAL MOS CATELOG	INPUT FILE NAME	NO. OF TERMINAL NODES	RELATIVE IMPORTANCE OF VARIABLES					
			(1)		(2)		(3)	
1	BUILD1	7	LOS	100	GRADE	78	MOS	16
2	BUILD2	6	LOS	100	GRADE	70	MOS	13
3	BUILD3	2	LOS	100	GRADE	82	MOS	0
4	BUILD4	13	LOS	100	GRADE	76	MOS	62
5	BUILD5	1	LOS	100	GRADE		MOS	
6	BUILD6	1	LOS	100	GRADE	53	MOS	0
7	BUILD7	3	MOS	100	GRADE	79	LOS	43
8	BUILD8	1	LOS	100	GRADE	68	MOS	23
9	BUILD9	1	LOS	100	GRADE	52	MOS	5

VII. CART PROGRAM: LIMITATIONS AND EVALUATION

A. INTRODUCTION

Two main points are discussed when evaluating any computer program: memory requirements and CPU time requirements. In our case, both requirements are greatly affected by the number and types of variables in addition to the data size.

It appears that no effort was made to optimize the memory and CPU time requirements in the development of CART. This view resulted from our attempt to execute the CART-BUILD program on data with the following specifications.

NUMBER OF CASES = 1401

NUMBER OF VARIABLES = 4

TYPES OF VARIABLES: 2 numerical variables
2 categorical variables

DEPENDENT VARIABLE: Numerical variable.

The first categorical variable takes values from the set {0,1,...,9}, the second takes values from the set {0,1,...,39}.

It is enough to say that 5000K bytes of memory and more than 10 hours of CPU time were not enough to execute the job on IBM 3030 computer system.

Two main reasons are behind this lack of efficiency.

(1) This technique depends on testing all possible combinations and alternatives at each step of the algorithm to choose the best of them, e.g., selection of the best split at each node, selection of the weakest link in pruning the maximum tree, selection of the best pruned tree from the set of all pruned trees, etc. Looking for the best conditions is a time consuming operation. Many techniques were developed to replace the previous technique, these techniques look for sufficient conditions rather than best conditions.

(2) The responsibility of memory management is given to the user who has to do it by trial and error. Designers don't appear to use dynamic allocation techniques to enable them to optimize the memory requirements during the execution time.

Few seconds are required to execute CART program on data with numerical variables only, but when categorical variables are taken into consideration, the situation is completely different.

B. CPU TIME REQUIREMENTS

Ten files were prepared to test the CPU time requirements of CART-BUILD program. The common specifications of these files are:

NUMBER OF RECORDS = 40

NUMBER OF VARIABLES = 4

TYPES OF VARIABLES = 2 numerical variables
2 categorical variables.

The first categorical variable takes values from the set {1,...,9}

INDEPENDENT VARIABLE: numerical variable.

The only difference among these files is the set of categories assigned to the second categorical variable. TABLE VII-1 contains the number of categories of the second categorical variable and the results obtained.

It is clear from TABLE VII-1 that CART-BUILD program enters an infinite loop if the number of categories for a categorical variable is greater than 15.

The effect of the learning sample size is neglegible if it is compared with that of the number of categories with respect to CPU time, but with respect to memory requirements, the both factors have the same great effect.

C. TEST OF EFFECTIVENESS

To check the effectiveness of this technique, the Figure of Merit is used as a measure of effectiveness:

where: $$F = \sum_i (r_i - P_i)^2 n_i / P_i \quad ,$$

i : number of cells,

P_i : predicted loss rate for cell i,

r_i : actual loss rate for cell i,

n_i : inventory in "actual" cells.

Figure of Merit was calculated for each one of the nine outputs of the CART-CASE program. The results are tabulated in TABLE VII-2.

TABLE VII-2 shows the lack of effectiveness of CART when the average inventory per cell is large, specifically, when TEST1, TEST3, and TEST7 files were used as input files.

TABLE VII-1. CPU TIME REQUIREMENTS

FILE NAME	NUMBER OF CATEGORIES	CPU TIME
F1	40	>60 MINUTES
F2	37	>60 MINUTES
F3	34	>60 MINUTES
F4	31	>60 MINUTES
F5	28	>60 MINUTES
F6	25	>60 MINUTES
F7	22	>60 MINUTES
F8	19	>60 MINUTES
F9	15	05 MIN., 06 SECONDS
F10	10	00 MIN., 21 SECONDS

TABLE VII-2. TEST OF EFFECTIVENESS

INPUT FILE	NUMBER OF CELLS	AVG. NO. OF INVENTORY PER CELL	F	$\chi^2_{0.95}$	
TEST1	149	26.757	217.394	178.5	*
TEST2	125	7.869	142.016	152.1	
TEST3	66	78.179	360.885	85.96	*
TEST4	267	6.351	207.535	305.0	
TEST5	79	5.006	73.494	100.7	
TEST6	75	2.704	31.528	96.2	
TEST7	49	24.884	111.247	66.3	*
TEST8	46	0.837	8.903	62.8	
TEST9	27	1.113	7.239	40.1	

D. EVALUATION

The program designers have assumed that the users are professionals who fully understand everything about the classification and regression tree technique. Failing this, the selection of options will be time and effort consuming, if not impossible.

The program depends upon a user-trial-and-error operation for memory management. There can be waste of computer time, and the goal could be nonachievable.

From the users point of view, the program is efficient when data variables are only numerical variables, but this efficiency will be greatly reduced if categorical variables are considered.

From the test of effectiveness, we can conclude that CART is effective only when the average inventory per cell is not large.

The program can be used with categorical variables provided that the number of categories for each doesn't exceed 15.

APPENDIX A

OUTPUTS OF THE CART PROGRAMS USING
1ST GENERAL CATEGORIZATION DATA AS INPUT

OUTPUT OF THE PROGRAM : CART-BUILD

PROGRAM PURPOSE : BUILDING A REGRESSION TREE

INPUT FILE :
NAME : BUILD1
CONTENTS : GENERAL MOS CAT. 1 DATA

TREE SEQUENCE

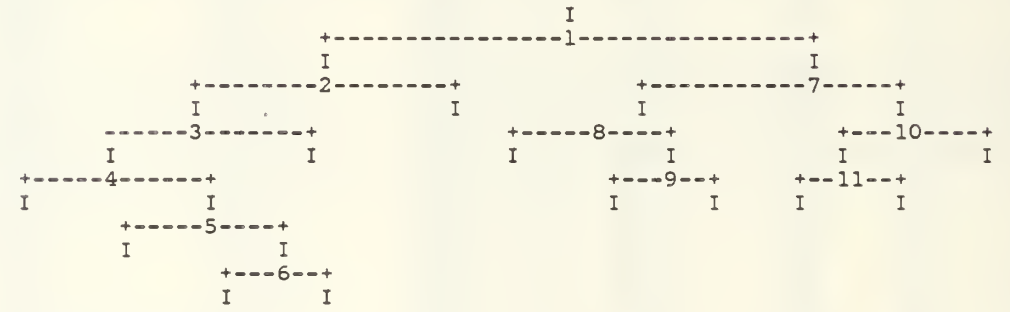
TREE	TERMINAL NODES	CROSS-VALIDATION RELATIVE ERROR	RESUBSTITUTION RELATIVE ERROR	COMPLEXITY PARAMETER
1	68	0.68 +/- 0.000	0.25	0.0
2	67	0.68 +/- 0.000	0.25	0.003
3	66	0.67 +/- 0.000	0.25	0.004
4	65	0.67 +/- 0.000	0.25	0.005
5	64	0.67 +/- 0.000	0.25	0.0051
6	63	0.67 +/- 0.000	0.25	0.011
7	62	0.67 +/- 0.000	0.25	0.012
8	61	0.67 +/- 0.000	0.25	0.014
9	60	0.67 +/- 0.000	0.25	0.016
10	59	0.67 +/- 0.000	0.25	0.020
11	58	0.67 +/- 0.000	0.25	0.022
12	57	0.67 +/- 0.000	0.26	0.023
13	56	0.67 +/- 0.000	0.26	0.023
14	55	0.67 +/- 0.000	0.26	0.032
15	54	0.67 +/- 0.000	0.26	0.035
16	53	0.67 +/- 0.000	0.26	0.041
17	52	0.67 +/- 0.000	0.26	0.041
18	50	0.66 +/- 0.000	0.26	0.049
19	49	0.66 +/- 0.000	0.27	0.052
20	48	0.66 +/- 0.000	0.27	0.054
21	45	0.66 +/- 0.000	0.27	0.059
22	44	0.66 +/- 0.000	0.27	0.060
23	43	0.66 +/- 0.000	0.28	0.061
24	42	0.66 +/- 0.000	0.28	0.064
25	41	0.66 +/- 0.000	0.28	0.067
26	40	0.66 +/- 0.000	0.28	0.068
27	39	0.67 +/- 0.000	0.28	0.076
28	38	0.64 +/- 0.000	0.29	0.088
29	36	0.63 +/- 0.000	0.29	0.110
30	35	0.63 +/- 0.000	0.30	0.114
31	34	0.63 +/- 0.000	0.30	0.125
32	32	0.63 +/- 0.000	0.31	0.126
33	31	0.63 +/- 0.000	0.31	0.135

FILE: APPENDIX A A1

34	30	0.63	+/-	0.000	0.32	0.161
35	29	0.61	+/-	0.000	0.32	0.175
36	25	0.62	+/-	0.000	0.34	0.182
37	20	0.61	+/-	0.000	0.37	0.194
38	19	0.61	+/-	0.000	0.38	0.250
39	18	0.61	+/-	0.000	0.39	0.281
40	15	0.64	+/-	0.000	0.42	0.289
41	14	0.64	+/-	0.000	0.43	0.333
42*	12	0.63	+/-	0.000	0.45	0.362
43	10	0.68	+/-	0.000	0.48	0.590
44	9	0.72	+/-	0.000	0.51	0.784
45	7	0.72	+/-	0.000	0.56	0.910
46	5	0.69	+/-	0.000	0.62	0.987
47	4	0.71	+/-	0.000	0.66	1.28
48	3	0.77	+/-	0.000	0.71	1.79
49	2	0.82	+/-	0.000	0.82	3.48
50	1	1.00	+/-	0.000	1.00	6.10

Initial median = 0.178
Initial mean absolute deviation = 0.171

CLASSIFICATION/REGRESSION TREE



Terminal Regions

1 2 3 4 5 6 7 8 9 0 11 12

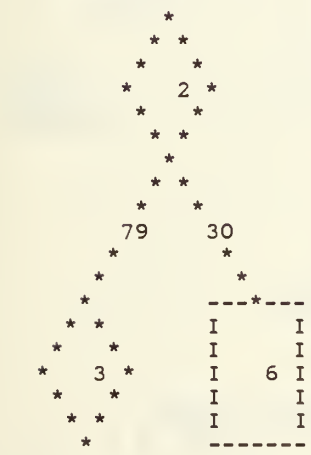
PARTITIONING TREE

Node 1 was split on variable LOS
A case goes left if variable LOS .le. 1.95E+01
Improvement = 3.1E-02 (C. T. = 6.1E+00)

Node	Cases	Median	Mean Abs Dev.
1	195	0.18	0.17
2	109	0.84E-01	0.94E-01
7	86	0.29	0.20

	Surrogate		Split	Assoc.	Improve.
1	GRADE	r	0, 1, 2, 3, 7, 8, 9	0.65	7.7E-03

	Competitor	Split	Improve.
1	GRADE	4, 5, 6, 7, 8	1.7E-02
2	MOS	51	5.0E-04

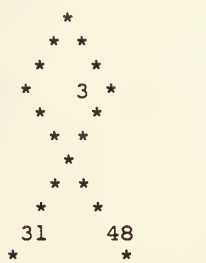


Node 2 was split on variable GRADE
A CASE GOES LEFT IF VARIABLE GRADE IS IN (5, 6)
IMPROVEMENT = 9.1E-03 (C. T. = 1.7E+00)

NODE	CASES	MEDIAN	MEAN ABS DEV.
2	109	0.84E-01	0.94E-01
3	79	0.11	0.93E-01
-6	30	0.19E-01	0.38E-01

	Surrogate		Split	Assoc.	Improve.
1	LOS	r	1.50E+00	0.13	1.1E-03

	Competitor	Split	Improve.
1	LOS	7.50E+00	2.8E-03
2	MOS	51, 52	6.8E-04

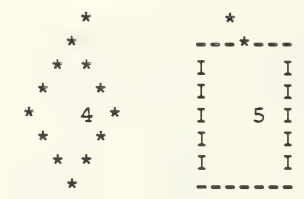


Node 3 was split on variable LOS
A CASE GOES LEFT IF VARIABLE LOS .LE. 7.50E+00
IMPROVEMENT = 6.5E-03 (C. T. = 1.2E+00)

NODE	CASES	MEDIAN	MEAN ABS DEV.
3	79	0.11	0.93E-01
4	31	0.22	0.12
-5	48	0.96E-01	0.50E-01

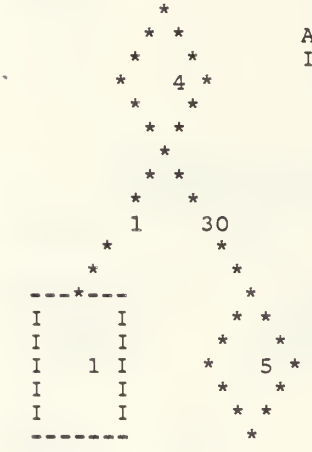
SURROGATE	SPLIT	ASSOC.	IMPROVE.

FILE: APPENDIX A A1



1 GRADE R 0, 1, 2, 3, 4, 6, 7, 8, 9 0.03 3.5E-03

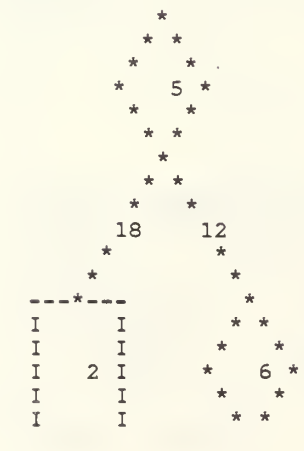
	Competitor	Split	Improve.
1	GRADE	5	3.5E-03
2	MOS	52	3.3E-05



Node 4 was split on variable LOS
A CASE GOES LEFT IF VARIABLE LOS .LE. 1.00E+00
IMPROVEMENT = 4.0E-03 (C. T. = 7.8E-01)

NODE	CASES	MEDIAN	MEAN ABS DEV.
4	31	0.22	0.12
-1	1	1.0	0.00E+00
5	30	0.22	0.97E-01

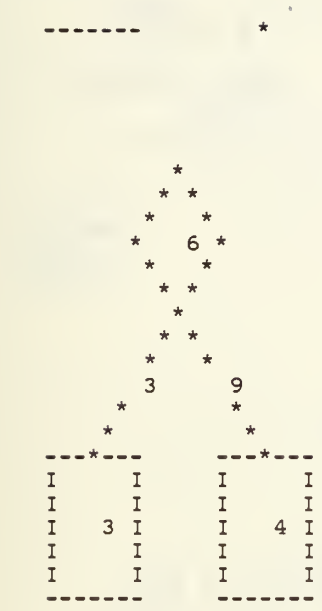
	Competitor	Split	Improve.
1	GRADE	5	1.2E-03
2	MOS	51	3.4E-04



Node 5 was split on variable GRADE
A case goes left if variable GRADE is in (5)
Improvement = 1.7E-03 (C. T. = 5.9E-01)

Node	Cases	Median	Mean Abs Dev.
5	30	0.22	0.97E-01
-2	18	0.24	0.51E-01
6	12	0.12	0.14

	Competitor	Split	Improve.
1	LOS	4.50E+00	8.8E-04
2	MOS	51	4.6E-04



Node 6 was split on variable LOS
A case goes left if variable LOS .le. 4.50E+00
Improvement = 4.3E-03 (C. T. = 8.5E-01)

Node	Cases	Median	Mean Abs Dev.
6	12	0.12	0.14
-3	3	0.67	0.16
-4	9	0.11	0.36E-01

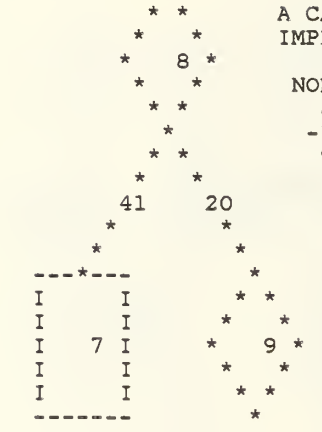
NODE 7 WAS SPLIT ON VARIABLE GRADE
A CASE GOES LEFT IF VARIABLE GRADE IS IN (7, 8)
IMPROVEMENT = 1.7E-02 (C. T. = 3.4E+00)

NODE	CASES	MEDIAN	MEAN ABS DEV.
7	86	0.29	0.20
8	61	0.24	0.12
10	25	0.67	.0.23

	Competitor	Split	Improve.
1	MOS	51	4.1E-03
2	LOS	2.55E+01	3.8E-03

NODE 8 WAS SPLIT ON VARIABLE LOS

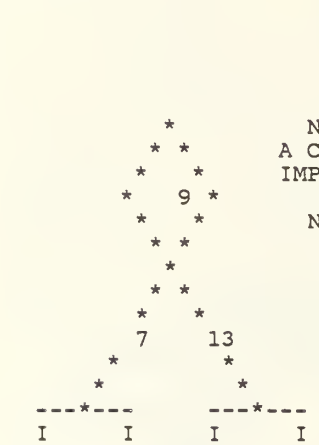
FILE: APPENDIX A A1



A CASE GOES LEFT IF VARIABLE LOS .LE. 2.65E+01
IMPROVEMENT = 3.5E-03 (C. T. = 9.8E-01)

NODE	CASES	MEDIAN	MEAN ABS DEV.
8	61	0.24	0.12
-7	41	0.21	0.88E-01
9	20	0.38	0.16

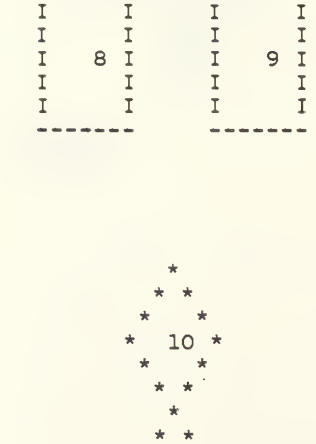
	Competitor	Split	Improve.
1	MOS	52	1.5E-03
2	GRADE	7	1.3E-04



NODE 9 WAS SPLIT ON VARIABLE MOS
A CASE GOES LEFT IF VARIABLE MOS IS IN (52)
IMPROVEMENT = 6.5E-03 (C. T. = 1.2E+00)

NODE	CASES	MEDIAN	MEAN ABS DEV.
9	20	0.38	0.16
-8	7	0.67	0.12
-9	13	0.29	0.91E-01

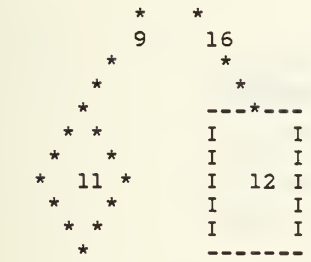
	COMPETITOR	SPLIT	IMPROVE.
1	LOS	2.95E+01	1.4E-03



Node 10 was split on variable MOS
A case goes left if variable MOS is in (51)
Improvement = 2.7E-03 (C. T. = 9.1E-01)

Node	Cases	Median	Mean Abs Dev.
10	25	0.67	0.23
11	9	0.47	0.24
-12	16	0.67	0.19

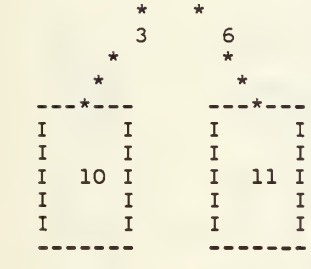
FILE: APPENDIX A A1



	Competitor	Split	Improve.
1	LOS	2.65E+01	2.2E-03
2	GRADE	6	8.5E-04

NODE 11 WAS SPLIT ON VARIABLE GRADE
A CASE GOES LEFT IF VARIABLE GRADE IS IN (2, 3)
IMPROVEMENT = 6.6E-03 (C. T. = 1.3E+00)

Node	Cases	Median	Mean Abs Dev.
11	9	0.47	0.24
-10	3	1.0	0.11
-11	6	0.29	0.91E-01



	Competitor	Split	Improve.
1	LOS	2.35E+01	1.1E-03

1 12 TERMINAL NODES

Node	Cases	Median	Mean Ad
1	1	1.00	0.00E+00
2	18	0.239	0.51E-01
3	3	0.667	0.16
4	9	0.105	0.36E-01
5	48	0.962E-01	0.50E-01
6	30	0.186E-01	0.38E-01
7	41	0.209	0.88E-01
8	7	0.667	0.12
9	13	0.289	0.91E-01
10	3	1.00	0.11
11	6	0.286	0.91E-01

FILE: APPENDIX A A1

12 16 0.667 0.19

	Relative Importance	Number Of Categories	Minimum Category
LOS	100.	numerical	
GRADE	70.	10	0
MOS	17.	3	51

Number of cases in the learning sample = 195

PRIMARY OPTION SETTINGS

construction rule	least absolute deviation
estimation method	10-fold cross-validation
tree selection	1.0 se rule
variables used	See variable importance list above. response is variable RATE
linear combinations	no

SECONDARY OPTION SETTINGS

1	Minimum node size	=	1
2	Minimum size below which node will not be split	=	5
3	Number of surrogate splits printed	=	2
4	Number of competing splits printed	=	2
5	Maximum number of trees for which errors are printed	=	100
6	INITIAL VALUE OF THE COMPLEXITY PARAMETER	=	0.000
7	Maximum number of cases to put into learning sample	=	20000
8	Maximum number of cases to put into test sample	=	20000
9	Maximum node size without sub-sampling the node	=	196
10	Maximum number of surrogates used	=	2
11	Maximum number of nodes allowed for in large tree	=	750
	(Actual maximum number of nodes	=	70)
12	Max. categorical primary + surrogate splits in a tree	=	1000
13	Max. linear combination splits in a tree	=	0
	(Actual number cat. + linear combination splits	=	35)
	(Actual number categorical competitor splits	=	51)
14	Maximum height of tree	=	750
	(Actual maximum height of tree	=	11)
	Maximum size of memory available	=	70000
	(Actual size of memory used in run	=	45070)

OUTPUT OF THE PROGRAM : CART-CASE

PROGRAM PURPOSE : VALIDATION OF RESULTS

INPUT FILE :
NAME : TEST1
CONTENTS : GENERAL MOS CAT. 1 DATA

CASE	NODE	PREDICTED	RATE	MOS	GRADE	LOS
1.	10	1.0000	1.0000	51.	3.	20.0
2.	6	0.0186	0.0071	51.	4.	1.0
3.	6	0.0186	0.0140	51.	4.	2.0
4.	6	0.0186	0.0103	51.	4.	3.0
5.	6	0.0186	0.0088	51.	4.	4.0
6.	6	0.0186	0.1132	51.	4.	8.0
7.	6	0.0186	0.0588	51.	4.	9.0
8.	2	0.2390	0.0059	51.	5.	2.0
9.	2	0.2390	0.0805	51.	5.	3.0
10.	2	0.2390	0.1573	51.	5.	4.0
11.	2	0.2390	0.1241	51.	5.	5.0
12.	2	0.2390	0.2237	51.	5.	6.0
13.	2	0.2390	0.1615	51.	5.	7.0
14.	5	0.0962	0.2278	51.	5.	8.0
15.	5	0.0962	0.2090	51.	5.	9.0
16.	5	0.0962	0.0580	51.	5.	10.0
17.	5	0.0962	0.1587	51.	5.	11.0
18.	5	0.0962	0.6667	51.	5.	18.0
19.	3	0.6667	0.8000	51.	6.	4.0
20.	4	0.1053	0.1020	51.	6.	5.0
21.	4	0.1053	0.0640	51.	6.	6.0
22.	4	0.1053	0.0641	51.	6.	7.0
23.	5	0.0962	0.0393	51.	6.	8.0
24.	5	0.0962	0.0358	51.	6.	9.0
25.	5	0.0962	0.0813	51.	6.	10.0
26.	5	0.0962	0.1061	51.	6.	11.0
27.	5	0.0962	0.1624	51.	6.	12.0
28.	5	0.0962	0.0989	51.	6.	13.0
29.	5	0.0962	0.0882	51.	6.	14.0
30.	5	0.0962	0.1709	51.	6.	15.0
31.	5	0.0962	0.1613	51.	6.	16.0
32.	5	0.0962	0.2727	51.	6.	17.0
33.	5	0.0962	0.1667	51.	6.	19.0
34.	11	0.2857	1.0000	51.	6.	20.0
35.	11	0.2857	1.0000	51.	6.	22.0
36.	6	0.0186	0.0351	51.	7.	10.0
37.	6	0.0186	0.0115	51.	7.	11.0
38.	6	0.0186	0.0153	51.	7.	12.0

39.	6	0.0186	0.0114	51.	7.	13.0
40.	6	0.0186	0.0050	51.	7.	14.0
41.	6	0.0186	0.0093	51.	7.	16.0
42.	6	0.0186	0.0171	51.	7.	18.0
43.	6	0.0186	0.0204	51.	7.	19.0
44.	7	0.2090	0.3784	51.	7.	20.0
45.	7	0.2090	0.3600	51.	7.	21.0
46.	7	0.2090	0.2927	51.	7.	22.0
47.	7	0.2090	0.0769	51.	7.	23.0
48.	7	0.2090	0.2105	51.	7.	24.0
49.	7	0.2090	0.3077	51.	7.	25.0
50.	7	0.2090	0.3077	51.	7.	26.0
51.	9	0.2889	0.3333	51.	7.	27.0
52.	9	0.2889	0.2500	51.	7.	28.0
53.	9	0.2889	0.3636	51.	7.	30.0
54.	7	0.2090	0.0956	51.	8.	20.0
55.	7	0.2090	0.0884	51.	8.	21.0
56.	7	0.2090	0.1566	51.	8.	22.0
57.	7	0.2090	0.1475	51.	8.	23.0
58.	7	0.2090	0.1942	51.	8.	24.0
59.	7	0.2090	0.2727	51.	8.	25.0
60.	7	0.2090	0.1333	51.	8.	26.0
61.	9	0.2889	0.1702	51.	8.	27.0
62.	9	0.2889	0.3200	51.	8.	28.0
63.	9	0.2889	0.2500	51.	8.	29.0
64.	9	0.2889	0.5806	51.	8.	30.0
65.	6	0.0186	0.0556	52.	4.	0.0
66.	6	0.0186	0.0195	52.	4.	1.0
67.	6	0.0186	0.0119	52.	4.	2.0
68.	6	0.0186	0.1250	52.	4.	9.0
69.	2	0.2390	0.1597	52.	5.	3.0
70.	2	0.2390	0.1508	52.	5.	4.0
71.	2	0.2390	0.1780	52.	5.	5.0
72.	2	0.2390	0.1656	52.	5.	6.0
73.	2	0.2390	0.2759	52.	5.	7.0
74.	5	0.0962	0.0488	52.	5.	8.0
75.	5	0.0962	0.2424	52.	5.	9.0
76.	5	0.0962	0.0667	52.	5.	10.0
77.	5	0.0962	0.0769	52.	5.	11.0
78.	5	0.0962	0.1111	52.	5.	12.0
79.	3	0.6667	0.3333	52.	6.	4.0
80.	4	0.1053	0.1053	52.	6.	5.0
81.	4	0.1053	0.0485	52.	6.	6.0
82.	4	0.1053	0.0494	52.	6.	7.0
83.	5	0.0962	0.0111	52.	6.	8.0
84.	5	0.0962	0.0112	52.	6.	9.0
85.	5	0.0962	0.0526	52.	6.	10.0
86.	5	0.0962	0.1728	52.	6.	11.0
87.	5	0.0962	0.0606	52.	6.	12.0
88.	5	0.0962	0.0615	52.	6.	13.0
89.	5	0.0962	0.0500	52.	6.	15.0
90.	5	0.0962	0.5000	52.	6.	18.0
91.	5	0.0962	0.5000	52.	6.	19.0
92.	12	0.6667	1.0000	52.	6.	20.0
93.	6	0.0186	0.0189	52.	7.	13.0

FILE: APPENDIX A A1

94.	6	0.0186	0.0139	52.	7.	14.0
95.	6	0.0186	0.0144	52.	7.	15.0
96.	6	0.0186	0.0313	52.	7.	16.0
97.	7	0.2090	0.4762	52.	7.	20.0
98.	7	0.2090	0.2353	52.	7.	21.0
99.	7	0.2090	0.3333	52.	7.	22.0
100.	7	0.2090	0.3333	52.	7.	23.0
101.	7	0.2090	1.0000	52.	7.	24.0
102.	8	0.6667	0.6667	52.	7.	29.0
103.	6	0.0186	0.0238	52.	8.	19.0
104.	7	0.2090	0.2069	52.	8.	20.0
105.	7	0.2090	0.0750	52.	8.	21.0
106.	7	0.2090	0.0784	52.	8.	22.0
107.	7	0.2090	0.0476	52.	8.	23.0
108.	7	0.2090	0.2069	52.	8.	24.0
109.	7	0.2090	0.7143	52.	8.	26.0
110.	8	0.6667	0.4444	52.	8.	27.0
111.	8	0.6667	0.6667	52.	8.	28.0
112.	8	0.6667	0.6667	52.	8.	29.0
113.	8	0.6667	1.0000	52.	8.	30.0
114.	12	0.6667	0.6667	53.	3.	23.0
115.	6	0.0186	0.0870	53.	4.	0.0
116.	2	0.2390	0.1538	53.	5.	3.0
117.	2	0.2390	0.0917	53.	5.	4.0
118.	2	0.2390	0.1636	53.	5.	5.0
119.	2	0.2390	0.0769	53.	5.	6.0
120.	2	0.2390	0.2083	53.	5.	7.0
121.	5	0.0962	0.3158	53.	5.	8.0
122.	5	0.0962	0.3333	53.	5.	12.0
123.	5	0.0962	1.0000	53.	5.	15.0
124.	4	0.1053	0.0400	53.	6.	5.0
125.	4	0.1053	0.0299	53.	6.	6.0
126.	5	0.0962	0.0920	53.	6.	8.0
127.	5	0.0962	0.0256	53.	6.	9.0
128.	5	0.0962	0.0580	53.	6.	10.0
129.	5	0.0962	0.0500	53.	6.	11.0
130.	5	0.0962	0.0645	53.	6.	12.0
131.	5	0.0962	0.1481	53.	6.	13.0
132.	5	0.0962	0.1176	53.	6.	14.0
133.	5	0.0962	0.4000	53.	6.	15.0
134.	5	0.0962	0.4000	53.	6.	16.0
135.	5	0.0962	1.0000	53.	6.	18.0
136.	12	0.6667	1.0000	53.	6.	20.0
137.	6	0.0186	0.0714	53.	7.	16.0
138.	7	0.2090	0.2000	53.	7.	20.0
139.	7	0.2090	0.4615	53.	7.	21.0
140.	7	0.2090	0.2500	53.	7.	22.0
141.	7	0.2090	0.6667	53.	7.	23.0
142.	7	0.2090	0.6667	53.	7.	24.0
143.	9	0.2889	0.6667	53.	7.	28.0
144.	6	0.0186	0.1111	53.	8.	19.0
145.	7	0.2090	0.0714	53.	8.	20.0
146.	7	0.2090	0.1667	53.	8.	22.0
147.	7	0.2090	0.1538	53.	8.	24.0
148.	7	0.2090	0.2000	53.	8.	25.0

FILE: APPENDIX A A1

149.	9	0.2889	0.5000	53.	8.	29.0

OUTPUT OF THE PROGRAM : CART-TEST

PROGRAM PURPOSE : PROVIDES A SUMMARY OF RESULTS OBTAINED FROM BUILD AND CASE PROGRAMS.

1 WELCOME TO CART (TM) Version 1.0 November 20, 1984

Summary statistics for a test sample with 149 cases.

Relative error based on tree = 0.120 +/- 0.965
Initial mean absolute deviation = 0.178
Initial median = 0.154

1 12 TERMINAL NODES

	LEARNING SAMPLE			TEST SAMPLE			
NODE	CASES	MEDIAN	MEAN AD	CASES	MEDIAN	MEAN AD	MEAN ERROR
1	1	1.00	0.00E+00	0			
2	18	0.239	0.51E-01	16	0.160	0.04	0.95
3	3	0.667	0.16	2	0.800	0.23	0.23
4	9	0.105	0.36E-01	8	0.640E-01	0.21E-01	0.04
5	48	0.096	0.50E-01	43	0.111	0.14	0.14
6	30	0.018	0.38E-01	26	0.189E-01	0.24E-01	0.024
7	41	0.209	0.88E-01	34	0.211	0.15	0.15
8	7	0.667	0.12	5	0.667	0.11	0.11
9	13	0.289	0.91E-01	9	0.333	0.12	0.14
10	3	1.00	0.11	1	1.00	0.00E+00	0.00
11	6	0.286	0.91E-01	2	1.00	0.00E+00	0.71
12	16	0.667	0.19	3	1.00	0.11	0.22

APPENDIX B

OUTPUTS OF THE CART PROGRAMS USING
2ND GENERAL CATEGORIZATION DATA AS INPUT

```
OUTPUT OF THE PROGRAM : CART-BUILD

PROGRAM PURPOSE : BUILDING A REGRESSION TREE

INPUT FILE :
             NAME     : BUILD2
             CONTENTS : GENERAL MOS CAT. 2 DATA
```

TREE SEQUENCE

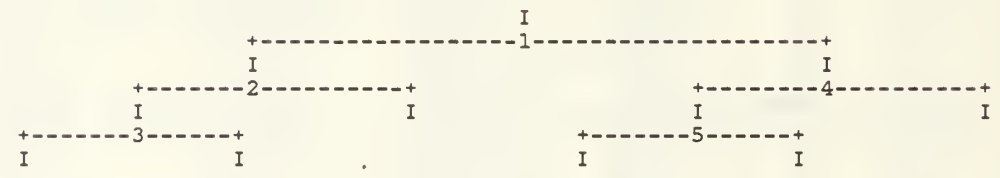
Tree	Terminal Nodes	Cross-Validated Relative Error	Resubstitution Relative Error	Complexity Parameter
1	72	0.86 +/- 0.000	0.34	0.0
2	71	0.86 +/- 0.000	0.34	0.002
3	70	0.86 +/- 0.000	0.34	0.004
4	69	0.86 +/- 0.000	0.34	0.008
5	68	0.86 +/- 0.000	0.34	0.009
6	67	0.86 +/- 0.000	0.34	0.009
7	66	0.86 +/- 0.000	0.34	0.017
8	65	0.86 +/- 0.000	0.34	0.018
9	64	0.86 +/- 0.000	0.34	0.023
10	63	0.86 +/- 0.000	0.34	0.023
11	62	0.86 +/- 0.000	0.34	0.025
12	61	0.86 +/- 0.000	0.34	0.025
13	60	0.86 +/- 0.000	0.35	0.030
14	58	0.86 +/- 0.000	0.35	0.031
15	57	0.86 +/- 0.000	0.35	0.033
16	56	0.86 +/- 0.000	0.35	0.034
17	53	0.86 +/- 0.000	0.35	0.042
18	52	0.86 +/- 0.000	0.35	0.043
19	51	0.86 +/- 0.000	0.36	0.043
20	50	0.86 +/- 0.000	0.36	0.048
21	47	0.86 +/- 0.000	0.36	0.056
22	45	0.85 +/- 0.000	0.37	0.065
23	44	0.86 +/- 0.000	0.37	0.070
24	43	0.86 +/- 0.000	0.37	0.079
25	42	0.86 +/- 0.000	0.37	0.080
26	39	0.86 +/- 0.000	0.38	0.081
27	36	0.86 +/- 0.000	0.39	0.088
28	35	0.85 +/- 0.000	0.39	0.114
29	34	0.86 +/- 0.000	0.40	0.133
30	33	0.87 +/- 0.000	0.40	0.141
31	32	0.87 +/- 0.000	0.40	0.167
32	30	0.82 +/- 0.000	0.42	0.179

FILE: APPENDIX B A1

33	28	0.82	+/-	0.000	0.43	0.210
34	27	0.83	+/-	0.000	0.43	0.214
35	22	0.83	+/-	0.000	0.47	0.242
36	19	0.82	+/-	0.000	0.49	0.244
37	17	0.82	+/-	0.000	0.51	0.250
38	16	0.84	+/-	0.000	0.52	0.303
39	12	0.81	+/-	0.000	0.55	0.312
40	11	0.78	+/-	0.000	0.56	0.362
41	8	0.74	+/-	0.000	0.60	0.410
42*	6	0.78	+/-	0.000	0.64	0.709
43	5	0.81	+/-	0.000	0.67	0.841
44	4	0.87	+/-	0.000	0.71	1.55
45	3	0.89	+/-	0.000	0.77	1.91
46	2	0.89	+/-	0.000	0.83	2.02
47	1	1.00	+/-	0.000	1.00	5.66

Initial median = 0.247
Initial mean absolute deviation = 0.176

CLASSIFICATION/REGRESSION TREE



Terminal Regions

1 2 3 4 5 6

PARTITIONING TREE

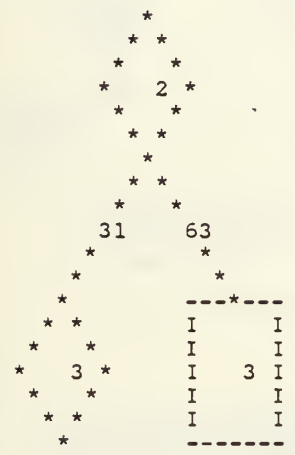
Node 1 was split on variable LOS
A case goes left if variable LOS .le. 1.95E+01
Improvement = 2.9E-02 (C. T. = 5.6E+00)

Node	Cases	Median	Mean Abs Dev.
1	191	0.25	0.18
2	94	0.13	0.11
4	97	0.36	0.18

	Surrogate	Split	Assoc.	Improve.
1	GRADE r	2, 3, 7, 8, 9	0.64	1.7E-02

2	MOS	r	54, 56	0.04 5.2E-04

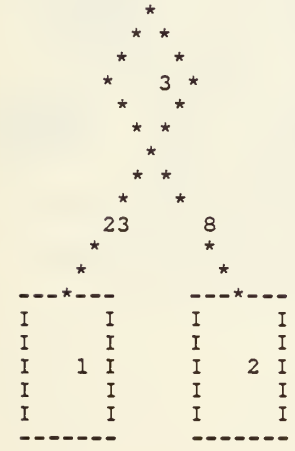
	Competitor	Split	Improve.
1	GRADE	0, 1, 4, 6	1.8E-02
2	MOS	54, 55	1.9E-03



Node 2 was split on variable GRADE
A case goes left if variable GRADE is in (5)
Improvement = 1.0E-02 (C. T. = 1.9E+00)

Node	Cases	Median	Mean Abs Dev.
2	94	0.13	0.11
3	31	0.24	0.11
-3	63	0.81E-01	0.80E-01

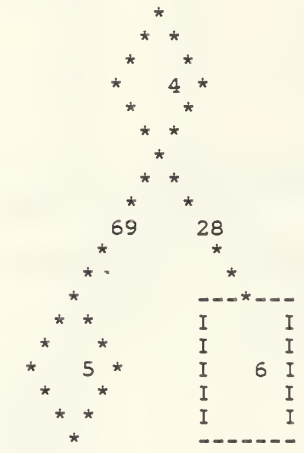
	Competitor	Split	Improve.
1	LOS	9.50E+00	5.0E-03
2	MOS	54, 55	1.6E-03



Node 3 was split on variable LOS
A case goes left if variable LOS .le. 9.50E+00
Improvement = 4.4E-03 (C. T. = 8.4E-01)

Node	Cases	Median	Mean Abs Dev.
3	31	0.24	0.11
-1	23	0.26	0.97E-01
-2	8	0.13	0.33E-01

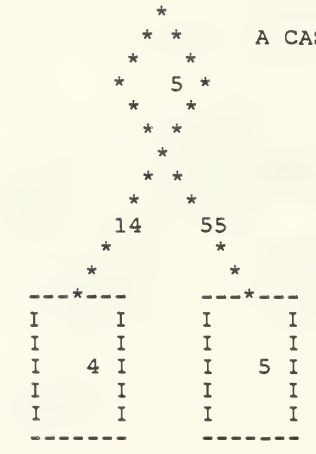
	Competitor	Split	Improve.
1	MOS	54	3.3E-04



Node 4 was split on variable LOS
A case goes left if variable LOS .le. 2.65E+01
Improvement = 1.0E-02 (C. T. = 2.0E+00)

Node	Cases	Median	Mean Abs Dev.
4	97	0.36	0.18
5	69	0.29	0.16
-6	28	0.50	0.17

	Competitor	Split	Improve.
1	GRADE	1, 2, 3	9.8E-03
2	MOS	54, 55	8.9E-04



Node 5 was split on variable GRADE
A CASE GOES LEFT IF VARIABLE GRADE IS IN (1, 2, 3)
Improvement = 8.1E-03 (C. T. = 1.5E+00)

Node	Cases	Median	Mean Abs Dev.
5	69	0.29	0.16
-4	14	0.50	0.13
-5	55	0.26	0.14

	Competitor	Split	Improve.
1	MOS	54, 55	6.7E-04
2	LOS	2.15E+01	5.3E-04

1 6 TERMINAL NODES

FILE: APPENDIX B A1

Node	Cases	Median	Mean Ad
1	23	0.263	0.97E-01
2	8	0.125	0.33E-01
3	63	0.806E-01	0.80E-01
4	14	0.500	0.13
5	55	0.261	0.14
6	28	0.500	0.17

	Relative Importance	Number Of Categories	Minimum Category
LOS	100.	numerical	
GRADE	70.	10	0
MOS	13.	3	54

Number of cases in the learning sample = 191

PRIMARY OPTION SETTINGS

construction rule	least absolute deviation
estimation method	5-fold cross-validation
tree selection	1.0 se rule
variables used	See variable importance list above. response is variable RATE
linear combinations	no

SECONDARY OPTION SETTINGS

1	Minimum node size	=	1
2	Minimum size below which node will not be split	=	5
3	Number of surrogate splits printed	=	2
4	Number of competing splits printed	=	2
5	Maximum number of trees for which errors are printed	=	100
6	Initial value of the complexity parameter	=	0.0000E+00
7	Maximum number of cases to put into learning sample	=	20000
8	Maximum number of cases to put into test sample	=	20000
9	Maximum node size without sub-sampling the node	=	192
10	Maximum number of surrogates used	=	2
11	Maximum number of nodes allowed for in large tree	=	750
	(Actual maximum number of nodes	=	73)
12	Max. categorical primary + surrogate splits in a tree	=	1000
13	Max. linear combination splits in a tree	=	0
	(Actual number cat. + linear combination splits	=	44)
	(Actual number categorical competitor splits	=	64)
14	Maximum height of tree	=	750
	(Actual maximum height of tree	=	12)
	Maximum size of memory available	=	70000
	(Actual size of memory used in run	=	45067)

FILE: APPENDIX B A1

OUTPUT OF THE PROGRAM : CART-CASE

PROGRAM PURPOSE : VALIDATION OF RESULTS

INPUT FILE :
NAME : TEST2
CONTENTS : GENERAL MOS CAT. 2 DATA

CASE	NODE	PREDICTED	RATE	MOS	GRADE	LOS
1.	3	0.0806	0.2500	54.	0.	11.0
2.	5	0.2609	1.0000	54.	0.	23.0
3.	3	0.0806	0.4000	54.	1.	13.0
4.	3	0.0806	0.1538	54.	1.	14.0
5.	4	0.5000	1.0000	54.	2.	22.0
6.	4	0.5000	1.0000	54.	2.	25.0
7.	6	0.5000	0.6667	54.	3.	28.0
8.	6	0.5000	0.5000	54.	3.	30.0
9.	3	0.0806	0.0952	54.	4.	0.0
10.	1	0.2632	0.0317	54.	5.	2.0
11.	1	0.2632	0.1356	54.	5.	3.0
12.	1	0.2632	0.2282	54.	5.	4.0
13.	1	0.2632	0.2162	54.	5.	5.0
14.	1	0.2632	0.2250	54.	5.	6.0
15.	1	0.2632	0.3684	54.	5.	7.0
16.	1	0.2632	0.7500	54.	5.	8.0
17.	1	0.2632	0.5714	54.	5.	9.0
18.	2	0.1250	0.2857	54.	5.	10.0
19.	5	0.2609	1.0000	54.	5.	20.0
20.	3	0.0806	0.6667	54.	6.	4.0
21.	3	0.0806	0.2154	54.	6.	5.0
22.	3	0.0806	0.0455	54.	6.	6.0
23.	3	0.0806	0.0244	54.	6.	7.0
24.	3	0.0806	0.0563	54.	6.	8.0
25.	3	0.0806	0.0253	54.	6.	9.0
26.	3	0.0806	0.0294	54.	6.	10.0
27.	3	0.0806	0.2439	54.	6.	11.0
28.	3	0.0806	0.3448	54.	6.	12.0
29.	3	0.0806	0.0870	54.	6.	13.0
30.	3	0.0806	0.0800	54.	6.	14.0
31.	3	0.0806	0.1379	54.	6.	15.0
32.	3	0.0806	0.0833	54.	6.	16.0
33.	5	0.2609	0.3750	54.	6.	20.0
34.	5	0.2609	0.4444	54.	6.	21.0
35.	5	0.2609	0.2857	54.	6.	22.0
36.	5	0.2609	0.2222	54.	6.	26.0
37.	6	0.5000	0.5000	54.	6.	28.0
38.	3	0.0806	0.0488	54.	7.	12.0
39.	3	0.0806	0.0364	54.	7.	15.0
40.	5	0.2609	0.2857	54.	7.	20.0

FILE: APPENDIX B A1

41.	5	0.2609	0.7143	54.	7.	21.0
42.	5	0.2609	0.6667	54.	7.	23.0
43.	5	0.2609	0.6667	54.	7.	26.0
44.	5	0.2609	0.0500	54.	8.	20.0
45.	5	0.2609	0.0625	54.	8.	21.0
46.	5	0.2609	0.2857	54.	8.	22.0
47.	5	0.2609	0.3636	54.	8.	23.0
48.	5	0.2609	0.1818	54.	8.	24.0
49.	5	0.2609	0.2222	54.	8.	25.0
50.	5	0.2609	0.4000	54.	8.	26.0
51.	6	0.5000	1.0000	55.	3.	30.0
52.	3	0.0806	0.1538	55.	4.	0.0
53.	3	0.0806	0.1053	55.	4.	3.0
54.	1	0.2632	0.1745	55.	5.	3.0
55.	1	0.2632	0.2543	55.	5.	4.0
56.	1	0.2632	0.2105	55.	5.	5.0
57.	1	0.2632	0.1758	55.	5.	6.0
58.	1	0.2632	0.3265	55.	5.	7.0
59.	1	0.2632	0.2400	55.	5.	9.0
60.	2	0.1250	0.1667	55.	5.	10.0
61.	2	0.1250	0.1765	55.	5.	11.0
62.	2	0.1250	0.1538	55.	5.	13.0
63.	2	0.1250	0.6667	55.	5.	16.0
64.	3	0.0806	0.1818	55.	6.	5.0
65.	3	0.0806	0.0460	55.	6.	6.0
66.	3	0.0806	0.0455	55.	6.	7.0
67.	3	0.0806	0.1687	55.	6.	8.0
68.	3	0.0806	0.0217	55.	6.	9.0
69.	3	0.0806	0.0541	55.	6.	11.0
70.	3	0.0806	0.1290	55.	6.	12.0
71.	3	0.0806	0.0597	55.	6.	13.0
72.	3	0.0806	0.0755	55.	6.	14.0
73.	3	0.0806	0.1081	55.	6.	15.0
74.	3	0.0806	0.6667	55.	6.	17.0
75.	3	0.0806	0.8000	55.	6.	19.0
76.	5	0.2609	1.0000	55.	6.	20.0
77.	3	0.0806	0.0417	55.	7.	15.0
78.	5	0.2609	0.6667	55.	7.	20.0
79.	5	0.2609	0.4000	55.	7.	21.0
80.	5	0.2609	0.6667	55.	7.	22.0
81.	5	0.2609	0.6000	55.	7.	23.0
82.	5	0.2609	0.2500	55.	7.	24.0
83.	5	0.2609	0.8571	55.	7.	25.0
84.	6	0.5000	0.5000	55.	7.	27.0
85.	6	0.5000	0.2857	55.	7.	28.0
86.	6	0.5000	1.0000	55.	7.	30.0
87.	5	0.2609	0.0588	55.	8.	20.0
88.	5	0.2609	0.4000	55.	8.	22.0
89.	5	0.2609	0.3158	55.	8.	23.0
90.	5	0.2609	0.3636	55.	8.	24.0
91.	5	0.2609	0.4444	55.	8.	25.0
92.	5	0.2609	0.1818	55.	8.	26.0
93.	6	0.5000	0.8000	55.	8.	27.0
94.	6	0.5000	0.6667	55.	8.	29.0
95.	6	0.5000	0.6667	55.	8.	30.0

96.	4	0.5000	0.5000	56.	1.	20.0
97.	4	0.5000	1.0000	56.	1.	22.0
98.	4	0.5000	0.6667	56.	1.	25.0
99.	3	0.0806	0.6667	56.	2.	18.0
100.	4	0.5000	1.0000	56.	2.	20.0
101.	6	0.5000	0.6667	56.	3.	27.0
102.	6	0.5000	0.6667	56.	3.	28.0
103.	6	0.5000	0.8000	56.	3.	30.0
104.	3	0.0806	0.0345	56.	4.	1.0
105.	1	0.2632	0.0571	56.	5.	3.0
106.	1	0.2632	0.2278	56.	5.	4.0
107.	1	0.2632	0.0964	56.	5.	5.0
108.	1	0.2632	0.2807	56.	5.	6.0
109.	1	0.2632	0.4103	56.	5.	7.0
110.	2	0.1250	1.0000	56.	5.	11.0
111.	6	0.5000	1.0000	56.	5.	28.0
112.	3	0.0806	0.7500	56.	6.	5.0
113.	3	0.0806	0.0952	56.	6.	6.0
114.	3	0.0806	0.0870	56.	6.	7.0
115.	3	0.0806	0.0556	56.	6.	8.0
116.	3	0.0806	0.1250	56.	6.	9.0
117.	3	0.0806	0.3333	56.	6.	13.0
118.	3	0.0806	0.1818	56.	6.	14.0
119.	3	0.0806	0.1538	56.	6.	15.0
120.	3	0.0806	0.2000	56.	6.	16.0
121.	5	0.2609	0.3529	56.	6.	20.0
122.	6	0.5000	0.2857	56.	6.	30.0
123.	5	0.2609	0.5000	56.	7.	20.0
124.	5	0.2609	0.6667	56.	7.	22.0
125.	5	0.2609	0.5000	56.	7.	25.0

```
OUTPUT OF THE PROGRAM : CART-TEST

PROGRAM PURPOSE : PROVIDES A SUMMARY OF RESULTS
                  OBTAINED FROM BUILD AND CASE
                  PROGRAMS.
INPUT FILE :
             NAME     : TEST2
             CONTENTS : GENERAL MOS CAT. 2 DATA
```

1 WELCOME TO CART (TM) Version 1.0 November 20, 1984

FILE: APPENDIX B A1

Summary statistics for a test sample with 125 cases.

Relative error based on tree = 0.190 +/- 0.444
Initial mean absolute deviation = 0.241
Initial median = 0.286

1 6 TERMINAL NODES

NODE	LEARNING SAMPLE CASES	MEDIAN	MEAN AD	TEST SAMPLE CASES	MEDIAN	MEAN AD	MEAN ERROR
1	23	0.263	0.97E-01	19	0.228	0.11	0.12
2	8	0.125	0.33E-01	6	0.286	0.24	0.28
3	63	0.08	0.80E-01	45	0.105	0.13	0.13
4	14	0.500	0.13	6	1.00	0.14	0.36
5	55	0.261	0.14	34	0.400	0.20	0.24
6	28	0.500	0.17	15	0.667	0.17	0.22

APPENDIX C

OUTPUTS OF THE CART PROGRAMS USING
3RD GENERAL CATEGORIZATION DATA AS INPUT

```
OUTPUT OF THE PROGRAM : CART-BUILD

PROGRAM PURPOSE : BUILDING A REGRESSION TREE

INPUT FILE :
             NAME     : BUILD3
             CONTENTS : GENERAL MOS CAT. 3 DATA
```

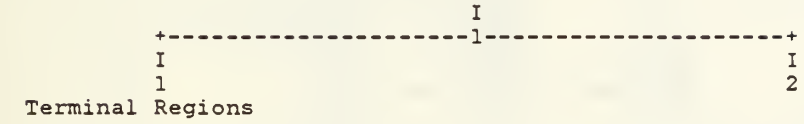
TREE SEQUENCE

Tree	Terminal Nodes	Cross-Validated Relative Error	Resubstitution Relative Error	Complexity Parameter
1	33	0.65 +/- 0.000	0.25	0.000
2	32	0.65 +/- 0.000	0.25	0.001
3	30	0.65 +/- 0.000	0.25	0.004
4	29	0.65 +/- 0.000	0.26	0.011
5	28	0.65 +/- 0.000	0.26	0.019
6	27	0.65 +/- 0.000	0.26	0.020
7	26	0.66 +/- 0.000	0.26	0.022
8	25	0.65 +/- 0.000	0.26	0.026
9	24	0.64 +/- 0.000	0.27	0.029
10	23	0.64 +/- 0.000	0.27	0.032
11	22	0.64 +/- 0.000	0.27	0.037
12	18	0.64 +/- 0.000	0.29	0.050
13	14	0.63 +/- 0.000	0.32	0.080
14	13	0.63 +/- 0.000	0.33	0.105
15	12	0.63 +/- 0.000	0.34	0.137
16	11	0.64 +/- 0.000	0.35	0.137
17	10	0.64 +/- 0.000	0.36	0.157
18	9	0.65 +/- 0.000	0.38	0.204
19	6	0.64 +/- 0.000	0.44	0.231
20	5	0.65 +/- 0.000	0.46	0.263
21	4	0.68 +/- 0.000	0.51	0.546
22	3	0.70 +/- 0.000	0.58	0.896
23*	2	0.68 +/- 0.000	0.67	1.09
24	1	1.00 +/- 0.000	1.00	3.91

Initial median = 0.127
Initial mean absolute deviation = 0.153

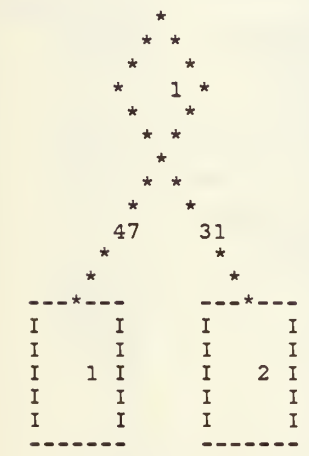
FILE: APPENDIX C A1

CLASSIFICATION/REGRESSION TREE



Terminal Regions

1 PARTITIONING TREE



Node 1 was split on variable LOS
A case goes left if variable LOS .le. 1.95E+01
Improvement = 5.0E-02 (C. T. = 3.9E+00)

Node	Cases	Median	Mean Abs Dev.
1	78	0.13	0.15
-1	47	0.56E-01	0.52E-01
-2	31	0.33	0.18

	Surrogate	Split	Assoc.	Improve.
1	GRADE r	0, 1, 2, 3, 7, 8, 9	0.54	1.5E-02

	Competitor	Split	Improve.
1	GRADE	4, 5, 7	2.2E-02

1 2 TERMINAL NODES

Node	Cases	Median	Mean Ad
1	47	0.562E-01	0.52E-01
2	31	0.329	0.18

	Relative Importance	Number Of Categories	Minimum Category
LOS	100.	numerical	
GRADE	82.	10	0
MOS	0.	numerical	

Number of cases in the learning sample = 78

FILE: APPENDIX C A1

PRIMARY OPTION SETTINGS

construction rule	least absolute deviation
estimation method	5-fold cross-validation
tree selection	1.0 se rule
variables used	See variable importance list above.
	response is variable RATE
linear combinations	no

SECONDARY OPTION SETTINGS

1	Minimum node size	=	1
2	Minimum size below which node will not be split	=	5
3	Number of surrogate splits printed	=	2
4	Number of competing splits printed	=	2
5	Maximum number of trees for which errors are printed	=	100
6	Initial value of the complexity parameter	=	0.0000E+00
7	Maximum number of cases to put into learning sample	=	20000
8	Maximum number of cases to put into test sample	=	20000
9	Maximum node size without sub-sampling the node	=	79
10	Maximum number of surrogates used	=	2
11	Maximum number of nodes allowed for in large tree	=	750
	(Actual maximum number of nodes	=	34)
12	Max. categorical primary + surrogate splits in a tree	=	1000
13	Max. linear combination splits in a tree	=	0
	(Actual number cat. + linear combination splits	=	9)
	(Actual number categorical competitor splits	=	7)
14	Maximum height of tree	=	750
	(Actual maximum height of tree	=	10)
	Maximum size of memory available	=	70000
	(Actual size of memory used in run	=	43126)

OUTPUT OF THE PROGRAM : CART-CASE

PROGRAM PURPOSE : VALIDATION OF RESULTS

INPUT FILE :
NAME : TEST3
CONTENTS : GENERAL MOS CAT. 3 DATA

CASE	NODE	PREDICTED	RATE	MOS	GRADE	LOS
1.	2	0.3288	0.6667	57.00	3.	26.0
2.	2	0.3288	0.5000	57.00	3.	28.0
3.	2	0.3288	1.0000	57.00	3.	29.0
4.	2	0.3288	0.5000	57.00	3.	30.0
5.	1	0.0562	0.0119	57.00	4.	0.0
6.	1	0.0562	0.0152	57.00	4.	1.0

FILE: APPENDIX C A1

7.	1	0.0562	0.0081	57.00	4.	2.0
8.	1	0.0562	0.0107	57.00	4.	3.0
9.	1	0.0562	0.0089	57.00	4.	4.0
10.	1	0.0562	0.0126	57.00	4.	5.0
11.	1	0.0562	0.0357	57.00	4.	6.0
12.	1	0.0562	0.0339	57.00	4.	7.0
13.	1	0.0562	0.0137	57.00	5.	3.0
14.	1	0.0562	0.0207	57.00	5.	4.0
15.	1	0.0562	0.0251	57.00	5.	5.0
16.	1	0.0562	0.0363	57.00	5.	6.0
17.	1	0.0562	0.0351	57.00	5.	7.0
18.	1	0.0562	0.0580	57.00	5.	8.0
19.	1	0.0562	0.0396	57.00	5.	9.0
20.	1	0.0562	0.0247	57.00	5.	10.0
21.	1	0.0562	0.1538	57.00	5.	12.0
22.	1	0.0562	0.1667	57.00	5.	13.0
23.	1	0.0562	0.1996	57.00	6.	5.0
24.	1	0.0562	0.1975	57.00	6.	6.0
25.	1	0.0562	0.1049	57.00	6.	7.0
26.	1	0.0562	0.0896	57.00	6.	8.0
27.	1	0.0562	0.0981	57.00	6.	9.0
28.	1	0.0562	0.0800	57.00	6.	10.0
29.	1	0.0562	0.1436	57.00	6.	11.0
30.	1	0.0562	0.1902	57.00	6.	12.0
31.	1	0.0562	0.1736	57.00	6.	13.0
32.	1	0.0562	0.0909	57.00	6.	14.0
33.	1	0.0562	0.0165	57.00	6.	15.0
34.	1	0.0562	0.0741	57.00	6.	16.0
35.	1	0.0562	0.4000	57.00	6.	17.0
36.	2	0.3288	1.0000	57.00	6.	20.0
37.	1	0.0562	0.0194	57.00	7.	10.0
38.	1	0.0562	0.0280	57.00	7.	11.0
39.	1	0.0562	0.0240	57.00	7.	12.0
40.	1	0.0562	0.0198	57.00	7.	13.0
41.	1	0.0562	0.0056	57.00	7.	14.0
42.	1	0.0562	0.0109	57.00	7.	15.0
43.	1	0.0562	0.0284	57.00	7.	17.0
44.	1	0.0562	0.0108	57.00	7.	18.0
45.	2	0.3288	0.7027	57.00	7.	20.0
46.	2	0.3288	0.2667	57.00	7.	21.0
47.	2	0.3288	0.4000	57.00	7.	22.0
48.	2	0.3288	0.6316	57.00	7.	23.0
49.	2	0.3288	0.7000	57.00	7.	24.0
50.	2	0.3288	0.9091	57.00	7.	25.0
51.	2	0.3288	0.2500	57.00	7.	26.0
52.	2	0.3288	0.6667	57.00	7.	27.0
53.	2	0.3288	1.0000	57.00	7.	29.0
54.	1	0.0562	0.0109	57.00	8.	18.0
55.	1	0.0562	0.0087	57.00	8.	19.0
56.	2	0.3288	0.1014	57.00	8.	20.0
57.	2	0.3288	0.1941	57.00	8.	21.0
58.	2	0.3288	0.1707	57.00	8.	22.0
59.	2	0.3288	0.2833	57.00	8.	23.0
60.	2	0.3288	0.2292	57.00	8.	24.0
61.	2	0.3288	0.3200	57.00	8.	25.0

62.	2	0.3288	0.2759	57.00	8.	26.0
63.	2	0.3288	0.3500	57.00	8.	27.0
64.	2	0.3288	0.5455	57.00	8.	28.0
65.	2	0.3288	1.0000	57.00	8.	29.0
66.	2	0.3288	0.5000	57.00	8.	30.0

OUTPUT OF THE PROGRAM : CART-TEST

PROGRAM PURPOSE : PROVIDES A SUMMARY OF RESULTS OBTAINED FROM BUILD AND CASE PROGRAMS.

1 WELCOME TO CART (TM) Version 1.0 November 20, 1984

961 Yorkshire Ct. Lafayette, California 94549
(415) 283-3392

Summary statistics for a test sample with 66 cases.

Relative error based on tree = 0.415E-01 +/- 0.597
Initial variance = 0.843E-01
Initial mean = 0.241

1 2 TERMINAL NODES

LEARNING SAMPLE				TEST SAMPLE			
NODE	CASES	AVERAGE	SD	CASES	AVERAGE	SD	STD ERROR
1	47	0.078	0.072	41	0.066	0.80E-01	0.08
2	31	0.391	0.24	25	0.527	0.28	0.31

APPENDIX D

OUTPUTS OF THE CART PROGRAMS USING
4TH GENERAL CATEGORIZATION DATA AS INPUT

```
 OUTPUT OF THE PROGRAM : CART-BUILD

PROGRAM PURPOSE : BUILDING A REGRESSION TREE

INPUT FILE :
             NAME     : BUILD4
             CONTENTS : GENERAL MOS CAT. 4 DATA
```

TREE SEQUENCE

Tree	Terminal Nodes	Cross-Validated Relative Error	Resubstitution Relative Error	Complexity Parameter
1	141	0.89 +/- 0.000	0.29	0.000
2	140	0.88 +/- 0.000	0.29	0.005
3	139	0.88 +/- 0.000	0.29	0.009
4	138	0.88 +/- 0.000	0.29	0.011
5	137	0.88 +/- 0.000	0.29	0.013
6	136	0.88 +/- 0.000	0.29	0.019
7	135	0.88 +/- 0.000	0.29	0.028
8	134	0.88 +/- 0.000	0.29	0.028
9	132	0.88 +/- 0.000	0.29	0.029
10	131	0.88 +/- 0.000	0.29	0.030
11	129	0.88 +/- 0.000	0.30	0.030
12	128	0.88 +/- 0.000	0.30	0.031
13	127	0.88 +/- 0.000	0.30	0.036
14	124	0.88 +/- 0.000	0.30	0.037
15	123	0.88 +/- 0.000	0.30	0.044
16	122	0.88 +/- 0.000	0.30	0.047
17	121	0.88 +/- 0.000	0.30	0.055
18	119	0.88 +/- 0.000	0.30	0.056
19	118	0.88 +/- 0.000	0.30	0.056
20	117	0.89 +/- 0.000	0.30	0.057
21	115	0.89 +/- 0.000	0.30	0.065
22	112	0.88 +/- 0.000	0.31	0.066
23	111	0.88 +/- 0.000	0.31	0.073
24	110	0.88 +/- 0.000	0.31	0.076
25	109	0.88 +/- 0.000	0.31	0.077
26	107	0.88 +/- 0.000	0.31	0.078
27	106	0.88 +/- 0.000	0.31	0.083
28	105	0.88 +/- 0.000	0.31	0.089
29	104	0.88 +/- 0.000	0.32	0.095
30	102	0.88 +/- 0.000	0.32	0.097
31	100	0.88 +/- 0.000	0.32	0.101
32	99	0.88 +/- 0.000	0.32	0.114

FILE: APPENDIX D A1

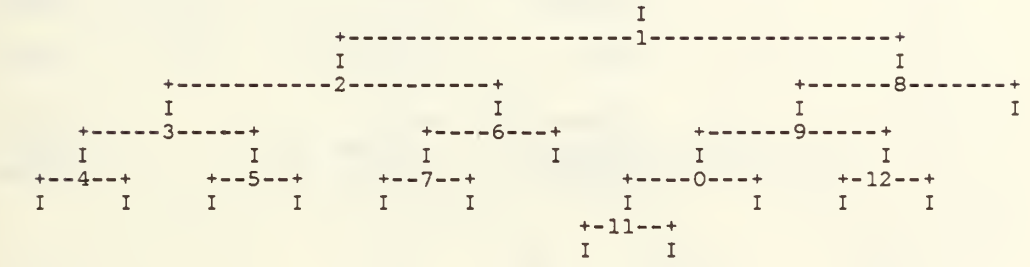
33	98	0.88	+/-	0.000	0.32	0.114
34	97	0.87	+/-	0.000	0.33	0.122
35	94	0.88	+/-	0.000	0.33	0.124
36	93	0.89	+/-	0.000	0.33	0.132
37	91	0.88	+/-	0.000	0.34	0.133
38	90	0.88	+/-	0.000	0.34	0.138
39	89	0.88	+/-	0.000	0.34	0.140
40	87	0.88	+/-	0.000	0.34	0.143
41	85	0.88	+/-	0.000	0.35	0.149
42	81	0.88	+/-	0.000	0.36	0.151
43	76	0.88	+/-	0.000	0.37	0.157
44	75	0.87	+/-	0.000	0.37	0.159
45	74	0.88	+/-	0.000	0.37	0.167
46	72	0.88	+/-	0.000	0.38	0.167
47	71	0.87	+/-	0.000	0.38	0.169
48	70	0.88	+/-	0.000	0.38	0.178
49	69	0.88	+/-	0.000	0.38	0.186
50	67	0.88	+/-	0.000	0.39	0.190
51	65	0.87	+/-	0.000	0.39	0.192
52	64	0.87	+/-	0.000	0.40	0.200
53	63	0.86	+/-	0.000	0.40	0.210
54	58	0.86	+/-	0.000	0.41	0.223
55	57	0.86	+/-	0.000	0.42	0.226
56	56	0.86	+/-	0.000	0.42	0.229
57	55	0.86	+/-	0.000	0.42	0.239
58	54	0.84	+/-	0.000	0.43	0.244
59	52	0.83	+/-	0.000	0.43	0.250
60	51	0.85	+/-	0.000	0.44	0.276
61	50	0.84	+/-	0.000	0.44	0.333
62	49	0.84	+/-	0.000	0.45	0.335
63	48	0.84	+/-	0.000	0.45	0.341
64	44	0.85	+/-	0.000	0.47	0.358
65	42	0.85	+/-	0.000	0.48	0.381
66	37	0.85	+/-	0.000	0.50	0.389
67	34	0.85	+/-	0.000	0.52	0.393
68	32	0.86	+/-	0.000	0.53	0.394
69	31	0.85	+/-	0.000	0.54	0.435
70	30	0.86	+/-	0.000	0.54	0.460
71	26	0.85	+/-	0.000	0.57	0.495
72	25	0.85	+/-	0.000	0.58	0.500
73	22	0.85	+/-	0.000	0.60	0.510
74	21	0.86	+/-	0.000	0.60	0.516
75	20	0.86	+/-	0.000	0.61	0.524
76	17	0.85	+/-	0.000	0.63	0.534
77	16	0.85	+/-	0.000	0.64	0.542
78	15	0.84	+/-	0.000	0.65	0.620
79	14	0.86	+/-	0.000	0.66	0.639
80*	13	0.86	+/-	0.000	0.67	0.772
81	11	0.89	+/-	0.000	0.69	0.827
82	10	0.89	+/-	0.000	0.70	0.829
83	9	0.88	+/-	0.000	0.71	0.873
84	8	0.89	+/-	0.000	0.73	1.12
85	7	0.89	+/-	0.000	0.74	1.28
86	6	0.91	+/-	0.000	0.77	1.94
87	3	0.91	+/-	0.000	0.85	2.11

FILE: APPENDIX D A1

88	2	0.90	+/-	0.000	0.90	3.19
89	1	1.00	+/-	0.000	1.00	7.66

Initial median = 0.263
Initial mean absolute deviation = 0.185

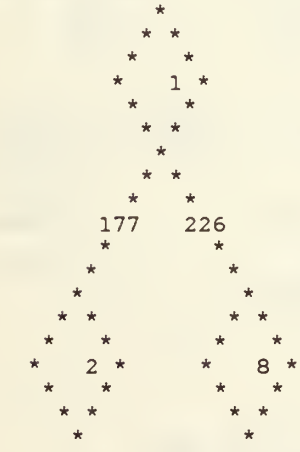
CLASSIFICATION/REGRESSION TREE



Terminal Regions

1 2 3 4 5 6 7 8 9 10 11 12 13

PARTITIONING TREE



Node 1 was split on variable LOS
A case goes left if variable LOS .le. 1.95E+01
Improvement = 1.9E-02 (C. T. = 7.6E+00)

Node	Cases	Median	Mean Abs Dev.
1	403	0.26	0.19
2	177	0.15	0.15
8	226	0.33	0.18

	Surrogate		Split	Assoc.	Improve.
1	GRADE	r	2, 3, 7, 8, 9	0.50	1.0E-02
2	MOS	r	58, 59, 60, 61, 62, 64	0.13	5.6E-04

	Competitor	Split	Improve.
1	GRADE	0, 1, 5, 6	1.2E-02
2	MOS	62	7.2E-03

Node 2 was split on variable LOS
A case goes left if variable LOS .le. 8.50E+00
Improvement = 3.6E-03 (C. T. = 2.1E+00)

Node	Cases	Median	Mean Abs Dev.
2	177	0.15	0.15
3	84	0.22	0.19
6	93	0.11	0.92E-01

	Surrogate		Split	Assoc.	Improve.
1	GRADE	r	0, 1, 2, 3, 6, 7, 8, 9	0.38	2.3E-03
2	MOS	r	58, 59, 60, 61, 64	0.14	1.5E-03

	Competitor	Split	Improve.
1	MOS	58, 59, 60	3.0E-03
2	GRADE	4	2.8E-03

Node 3 was split on variable GRADE
A case goes left if variable GRADE is in (1, 4)
Improvement = 2.2E-03 (C. T. = 2.4E+00)

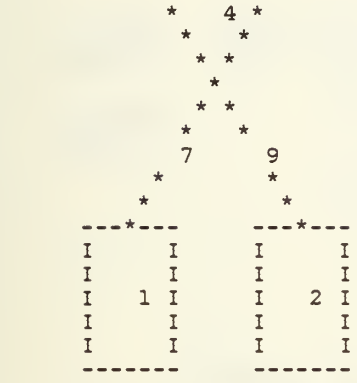
Node	Cases	Median	Mean Abs Dev.
3	84	0.22	0.19
4	16	0.67	0.34
5	68	0.22	0.15

	Surrogate		Split	Assoc.	Improve.
1	LOS	s	1.50E+00	0.25	-4.2E-08

	Competitor	Split	Improve.
1	MOS	58, 59, 60, 61	1.8E-03
2	LOS	5.00E-01	1.5E-03

Node 4 was split on variable MOS
A CASE GOES LEFT IF VARIABLE MOS IS IN (58, 59, 60)
Improvement = 9.8E-03 (C. T. = 3.9E+00)

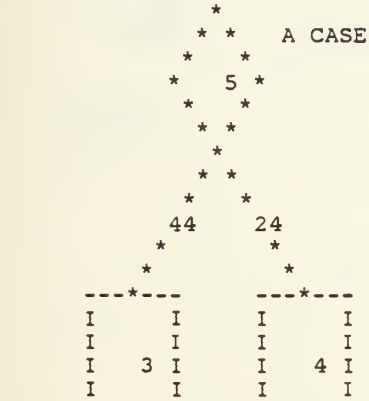
FILE: APPENDIX D A1



Node	Cases	Median	Mean Abs Dev.
4	16	0.67	0.34
-1	7	0.23E-01	0.64E-01
-2	9	0.80	0.12

	Surrogate	Split	Assoc.	Improve.
1	LOS s	2.50E+00	0.14	2.1E-03

	Competitor	Split	Improve.
1	LOS	2.50E+00	2.1E-03



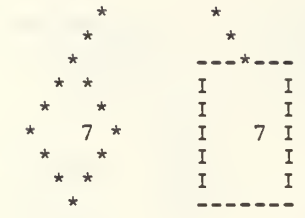
Node 5 was split on variable MOS
A CASE GOES LEFT IF VARIABLE MOS IS IN (58, 59, 61, 63
Improvement = 2.1E-03 (C. T. = 8.7E-01)

Node	Cases	Median	Mean Abs Dev.
5	68	0.22	0.15
-3	44	0.18	0.12
-4	24	0.29	0.16

	Competitor	Split	Improve.
1	GRADE	0	3.0E-04
2	LOS	5.50E+00	1.3E-04

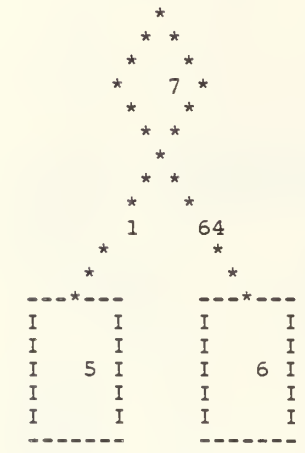
Node 6 was split on variable MOS
A CASE GOES LEFT IF VARIABLE MOS IS IN (58, 59, 60, 61
Improvement = 1.8E-03 (C. T. = 8.2E-01)

Node	Cases	Median	Mean Abs Dev.
6	93	0.11	0.92E-01
7	65	0.80E-01	0.92E-01
-7	28	0.18	0.67E-01



	Surrogate		Split	Assoc.	Improve.
1	GRADE	r	3, 4, 7, 8, 9	0.03	7.9E-06

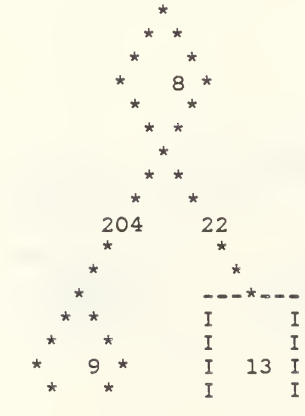
	Competitor	Split	Improve.
1	GRADE	4	1.1E-03
2	LOS	1.55E+01	7.6E-04



Node 7 was split on variable GRADE
A case goes left if variable GRADE is in (4)
Improvement = 2.2E-03 (C. T. = 9.2E-01)

Node	Cases	Median	Mean Abs Dev.
7	65	0.80E-01	0.92E-01
-5	1	1.0	0.00E+00
-6	64	0.80E-01	0.79E-01

	Competitor	Split	Improve.
1	MOS	60	6.3E-04
2	LOS	1.55E+01	2.2E-04



Node 8 was split on variable LOS
A case goes left if variable LOS .le. 2.95E+01
Improvement = 7.9E-03 (C. T. = 3.2E+00)

Node	Cases	Median	Mean Abs Dev.
8	226	0.33	0.18
9	204	0.32	0.16
-13	22	0.67	0.18

	Competitor	Split	Improve.
1	MOS	58, 59, 60, 61, 63	5.7E-03
2	GRADE	5, 6	3.3E-03

Node 9 was split on variable MOS
A CASE GOES LEFT IF VARIABLE MOS IS IN (58, 59, 60, 61, 63)
Improvement = 4.8E-03 (C. T. = 1.9E+00)

Node	Cases	Median	Mean Abs Dev.
9	204	0.32	0.16
10	173	0.29	0.15
12	31	0.50	0.20

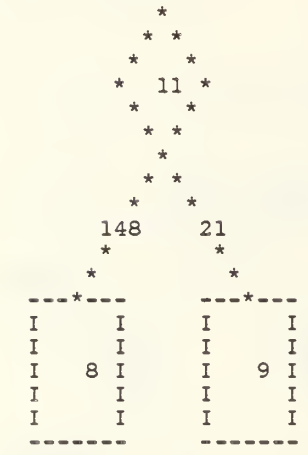
	Competitor	Split	Improve.
1	GRADE	2, 3, 5, 6, 7, 8	2.5E-03
2	LOS	2.05E+01	2.0E-03

Node 10 was split on variable GRADE
A CASE GOES LEFT IF VARIABLE GRADE IS IN (2, 3, 5, 6, 7, 8)
Improvement = 2.7E-03 (C. T. = 1.1E+00)

Node	Cases	Median	Mean Abs Dev.
10	173	0.29	0.15
11	169	0.29	0.14
-10	4	1.0	0.22

	Competitor	Split	Improve.
1	LOS	2.75E+01	2.0E-03
2	MOS	58	4.8E-04

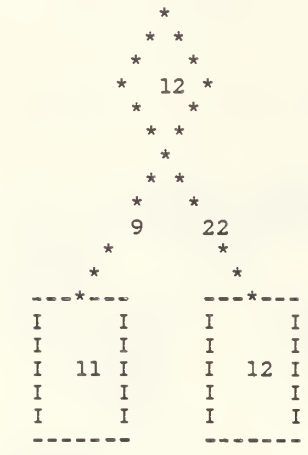
FILE: APPENDIX D A1



Node 11 was split on variable LOS
A case goes left if variable LOS .le. 2.75E+01
Improvement = 2.0E-03 (C. T. = 8.2E-01)

Node	Cases	Median	Mean Abs Dev.
11	169	0.29	0.14
-8	148	0.29	0.13
-9	21	0.43	0.14

	Competitor	Split	Improve.
1	GRADE	5	7.9E-04
2	MOS	58	7.0E-04



Node 12 was split on variable GRADE
A case goes left if variable GRADE is in (6)
Improvement = 3.1E-03 (C. T. = 1.2E+00)

Node	Cases	Median	Mean Abs Dev.
12	31	0.50	0.20
-11	9	0.22	0.15
-12	22	0.67	0.16

	Competitor	Split	Improve.
1	LOS	2.55E+01	1.6E-03

1 13 TERMINAL NODES

Node	Cases	Median	Mean Ad
1	7	0.227E-01	0.64E-01
2	9	0.800	0.12
3	44	0.177	0.12

FILE: APPENDIX D A1

4	24	0.286	0.16
5	1	1.00	0.00E+00
6	64	0.800E-01	0.79E-01
7	28	0.182	0.67E-01
8	148	0.286	0.13
9	21	0.429	0.14
10	4	1.00	0.22
11	9	0.222	0.15
12	22	0.667	0.16
13	22	0.667	0.18

	Relative Importance	Number Of Categories	Minimum Category
LOS	100.	numerical	
GRADE	76.	10	0
MOS	62.	7	58

Number of cases in the learning sample = 403

PRIMARY OPTION SETTINGS

construction rule	least absolute deviation
estimation method	5-fold cross-validation
tree selection	1.0 se rule
variables used	See variable importance list above. response is variable RATE
linear combinations	no

SECONDARY OPTION SETTINGS

1	Minimum node size	=	1
2	Minimum size below which node will not be split	=	5
3	Number of surrogate splits printed	=	2
4	Number of competing splits printed	=	2
5	Maximum number of trees for which errors are printed	=	100
6	Initial value of the complexity parameter	=	0.0000E+00
7	Maximum number of cases to put into learning sample	=	20000
8	Maximum number of cases to put into test sample	=	20000
9	Maximum node size without sub-sampling the node	=	404
10	Maximum number of surrogates used	=	2
11	Maximum number of nodes allowed for in large tree	=	750
	(Actual maximum number of nodes	=	142)
12	Max. categorical primary + surrogate splits in a tree	=	1000
13	Max. linear combination splits in a tree	=	0
	(Actual number cat. + linear combination splits	=	104)
	(Actual number categorical competitor splits	=	156)
14	Maximum height of tree	=	750
	(Actual maximum height of tree	=	16)
	Maximum size of memory available	=	70000
	(Actual size of memory used in run	=	48762)

* *

FILE: APPENDIX D A1

OUTPUT OF THE PROGRAM : CART-CASE

PROGRAM PURPOSE : VALIDATION OF RESULTS

INPUT FILE :
NAME : TEST4
CONTENTS : GENERAL MOS CAT. 4 DATA

CASE	NODE	PREDICTED	RATE	MOS	GRADE	LOS
1.	1	0.0227	0.2857	58.	1.	8.0
2.	6	0.0800	0.0952	58.	1.	9.0
3.	6	0.0800	0.0870	58.	1.	10.0
4.	6	0.0800	0.1429	58.	1.	11.0
5.	10	1.0000	0.5000	58.	1.	21.0
6.	8	0.2857	1.0000	58.	2.	23.0
7.	8	0.2857	0.5000	58.	3.	20.0
8.	8	0.2857	0.6667	58.	3.	22.0
9.	8	0.2857	0.1818	58.	3.	25.0
10.	8	0.2857	0.2857	58.	3.	26.0
11.	13	0.6667	0.2000	58.	3.	30.0
12.	1	0.0227	1.0000	58.	4.	0.0
13.	1	0.0227	0.5000	58.	4.	6.0
14.	3	0.1772	0.0417	58.	5.	2.0
15.	3	0.1772	0.1739	58.	5.	3.0
16.	3	0.1772	0.3448	58.	5.	4.0
17.	3	0.1772	0.1351	58.	5.	5.0
18.	3	0.1772	0.3590	58.	5.	6.0
19.	3	0.1772	0.0714	58.	5.	7.0
20.	3	0.1772	0.1176	58.	5.	8.0
21.	6	0.0800	0.1333	58.	5.	9.0
22.	6	0.0800	0.0714	58.	5.	10.0
23.	6	0.0800	0.0488	58.	5.	12.0
24.	6	0.0800	0.1667	58.	5.	13.0
25.	8	0.2857	0.1667	58.	5.	20.0
26.	8	0.2857	1.0000	58.	5.	21.0
27.	3	0.1772	0.5000	58.	6.	5.0
28.	3	0.1772	0.1034	58.	6.	6.0
29.	3	0.1772	0.0377	58.	6.	7.0
30.	3	0.1772	0.0370	58.	6.	8.0
31.	6	0.0800	0.0333	58.	6.	9.0
32.	6	0.0800	0.1333	58.	6.	11.0
33.	6	0.0800	0.5714	58.	6.	12.0
34.	6	0.0800	0.0952	58.	6.	14.0
35.	8	0.2857	0.3448	58.	6.	20.0
36.	8	0.2857	0.0833	58.	6.	21.0
37.	8	0.2857	0.1053	58.	6.	23.0
38.	8	0.2857	0.1111	58.	6.	24.0
39.	8	0.2857	0.2857	58.	6.	26.0
40.	6	0.0800	0.2000	58.	7.	10.0
41.	6	0.0800	0.2667	58.	7.	12.0
42.	6	0.0800	0.0645	58.	7.	14.0
43.	8	0.2857	0.7500	58.	7.	20.0

FILE: APPENDIX D A1

44.	8	0.2857	0.6667	58.	7.	21.0
45.	8	0.2857	0.7500	58.	7.	22.0
46.	8	0.2857	0.4000	58.	7.	23.0
47.	8	0.2857	0.4444	58.	7.	24.0
48.	8	0.2857	0.6667	58.	7.	25.0
49.	8	0.2857	0.2857	58.	7.	27.0
50.	9	0.4286	0.5000	58.	7.	29.0
51.	8	0.2857	0.6667	58.	8.	21.0
52.	9	0.4286	0.6000	58.	8.	28.0
53.	9	0.4286	0.5000	58.	8.	29.0
54.	13	0.6667	0.3333	58.	8.	30.0
55.	6	0.0800	0.1429	59.	0.	14.0
56.	6	0.0800	0.3333	59.	1.	10.0
57.	6	0.0800	0.2500	59.	1.	15.0
58.	10	1.0000	1.0000	59.	1.	22.0
59.	10	1.0000	1.0000	59.	1.	25.0
60.	8	0.2857	0.8000	59.	2.	21.0
61.	8	0.2857	0.6667	59.	3.	23.0
62.	8	0.2857	0.2500	59.	3.	27.0
63.	9	0.4286	0.5000	59.	3.	28.0
64.	13	0.6667	0.4000	59.	3.	30.0
65.	5	1.0000	1.0000	59.	4.	11.0
66.	3	0.1772	0.3750	59.	5.	3.0
67.	3	0.1772	0.1818	59.	5.	4.0
68.	3	0.1772	0.1765	59.	5.	5.0
69.	3	0.1772	0.0769	59.	5.	6.0
70.	3	0.1772	0.1429	59.	5.	7.0
71.	6	0.0800	0.2857	59.	5.	10.0
72.	6	0.0800	0.2222	59.	5.	11.0
73.	3	0.1772	0.1429	59.	6.	5.0
74.	3	0.1772	0.1667	59.	6.	6.0
75.	3	0.1772	0.1333	59.	6.	7.0
76.	3	0.1772	0.0435	59.	6.	8.0
77.	6	0.0800	0.0488	59.	6.	9.0
78.	6	0.0800	0.0952	59.	6.	11.0
79.	6	0.0800	0.1333	59.	6.	12.0
80.	6	0.0800	0.2500	59.	6.	13.0
81.	6	0.0800	0.0833	59.	6.	14.0
82.	6	0.0800	0.2500	59.	6.	17.0
83.	8	0.2857	0.3478	59.	6.	20.0
84.	8	0.2857	0.1176	59.	6.	21.0
85.	8	0.2857	0.3077	59.	6.	22.0
86.	8	0.2857	0.3636	59.	6.	23.0
87.	8	0.2857	0.4000	59.	6.	24.0
88.	8	0.2857	0.3333	59.	6.	26.0
89.	9	0.4286	1.0000	59.	6.	29.0
90.	6	0.0800	0.3333	59.	7.	10.0
91.	8	0.2857	0.6667	59.	7.	20.0
92.	8	0.2857	0.1818	59.	7.	21.0
93.	8	0.2857	0.5000	59.	7.	24.0
94.	8	0.2857	0.2857	59.	7.	25.0
95.	8	0.2857	0.1818	59.	7.	26.0
96.	8	0.2857	0.5000	59.	7.	27.0
97.	13	0.6667	0.8000	59.	7.	30.0
98.	8	0.2857	0.6667	59.	8.	24.0

FILE: APPENDIX D A1

99.	8	0.2857	0.6667	59.	8.	26.0
100.	8	0.2857	0.2000	59.	8.	27.0
101.	9	0.4286	0.2857	59.	8.	28.0
102.	6	0.0800	0.0769	60.	1.	12.0
103.	6	0.0800	0.0952	60.	1.	13.0
104.	6	0.0800	0.0909	60.	1.	14.0
105.	8	0.2857	0.6667	60.	3.	20.0
106.	8	0.2857	1.0000	60.	3.	22.0
107.	8	0.2857	0.5000	60.	3.	25.0
108.	9	0.4286	0.4000	60.	3.	28.0
109.	13	0.6667	0.3333	60.	3.	30.0
110.	1	0.0227	0.0227	60.	4.	1.0
111.	1	0.0227	0.0769	60.	4.	2.0
112.	4	0.2857	0.0138	60.	5.	2.0
113.	4	0.2857	0.2476	60.	5.	3.0
114.	4	0.2857	0.2311	60.	5.	4.0
115.	4	0.2857	0.1667	60.	5.	5.0
116.	4	0.2857	0.3636	60.	5.	6.0
117.	4	0.2857	0.3390	60.	5.	7.0
118.	4	0.2857	0.1081	60.	5.	8.0
119.	6	0.0800	0.2857	60.	5.	9.0
120.	8	0.2857	1.0000	60.	5.	22.0
121.	4	0.2857	0.3729	60.	6.	5.0
122.	4	0.2857	0.1754	60.	6.	6.0
123.	4	0.2857	0.0593	60.	6.	7.0
124.	4	0.2857	0.0240	60.	6.	8.0
125.	6	0.0800	0.0874	60.	6.	9.0
126.	6	0.0800	0.0429	60.	6.	10.0
127.	6	0.0800	0.1333	60.	6.	11.0
128.	6	0.0800	0.1429	60.	6.	12.0
129.	6	0.0800	0.0345	60.	6.	13.0
130.	6	0.0800	0.0597	60.	6.	14.0
131.	6	0.0800	0.0645	60.	6.	15.0
132.	6	0.0800	0.0513	60.	6.	16.0
133.	6	0.0800	0.2857	60.	6.	18.0
134.	8	0.2857	0.2857	60.	6.	20.0
135.	8	0.2857	0.1429	60.	6.	21.0
136.	8	0.2857	0.1538	60.	6.	22.0
137.	8	0.2857	0.2857	60.	6.	23.0
138.	8	0.2857	0.4000	60.	6.	25.0
139.	6	0.0800	0.0444	60.	7.	11.0
140.	6	0.0800	0.0339	60.	7.	12.0
141.	6	0.0800	0.0690	60.	7.	16.0
142.	8	0.2857	0.4444	60.	7.	20.0
143.	8	0.2857	0.2500	60.	7.	21.0
144.	8	0.2857	0.7500	60.	7.	22.0
145.	8	0.2857	0.2500	60.	7.	23.0
146.	8	0.2857	0.4000	60.	7.	24.0
147.	8	0.2857	0.2500	60.	7.	25.0
148.	8	0.2857	0.5000	60.	7.	26.0
149.	9	0.4286	0.5000	60.	7.	29.0
150.	8	0.2857	0.1481	60.	8.	20.0
151.	8	0.2857	0.1538	60.	8.	21.0
152.	8	0.2857	0.4167	60.	8.	22.0
153.	8	0.2857	0.0870	60.	8.	23.0

FILE: APPENDIX D A1

154.	8	0.2857	0.4706	60.	8.	24.0
155.	8	0.2857	0.1429	60.	8.	25.0
156.	8	0.2857	0.5000	60.	8.	27.0
157.	9	0.4286	0.3333	60.	8.	28.0
158.	9	0.4286	0.1818	60.	8.	29.0
159.	13	0.6667	0.8000	60.	8.	30.0
160.	6	0.0800	0.3077	61.	1.	10.0
161.	10	1.0000	0.6667	61.	1.	20.0
162.	8	0.2857	0.8000	61.	3.	20.0
163.	8	0.2857	0.6667	61.	3.	23.0
164.	8	0.2857	0.6667	61.	3.	26.0
165.	13	0.6667	0.6667	61.	3.	30.0
166.	2	0.8000	0.0526	61.	4.	1.0
167.	3	0.1772	0.1000	61.	5.	3.0
168.	3	0.1772	0.1446	61.	5.	4.0
169.	3	0.1772	0.1000	61.	5.	5.0
170.	3	0.1772	0.2979	61.	5.	6.0
171.	3	0.1772	0.3478	61.	5.	7.0
172.	6	0.0800	0.1333	61.	5.	10.0
173.	8	0.2857	0.4000	61.	5.	20.0
174.	3	0.1772	0.4615	61.	6.	5.0
175.	3	0.1772	0.0976	61.	6.	6.0
176.	3	0.1772	0.0526	61.	6.	8.0
177.	6	0.0800	0.1538	61.	6.	9.0
178.	6	0.0800	0.1111	61.	6.	11.0
179.	6	0.0800	0.1538	61.	6.	16.0
180.	8	0.2857	0.4000	61.	6.	20.0
181.	8	0.2857	0.2500	61.	6.	23.0
182.	6	0.0800	0.1667	61.	7.	12.0
183.	8	0.2857	0.8000	61.	7.	20.0
184.	8	0.2857	0.5714	61.	7.	22.0
185.	8	0.2857	1.0000	61.	7.	24.0
186.	9	0.4286	0.8571	61.	7.	28.0
187.	8	0.2857	0.2000	61.	8.	21.0
188.	8	0.2857	0.2857	61.	8.	22.0
189.	8	0.2857	0.8000	61.	8.	23.0
190.	8	0.2857	0.5714	61.	8.	26.0
191.	8	0.2857	0.5000	61.	8.	27.0
192.	9	0.4286	0.6667	61.	8.	29.0
193.	4	0.2857	0.4615	62.	5.	3.0
194.	4	0.2857	0.2500	62.	5.	4.0
195.	4	0.2857	0.6667	62.	5.	5.0
196.	4	0.2857	0.5000	62.	5.	6.0
197.	7	0.1818	0.2857	62.	6.	10.0
198.	7	0.1818	0.8000	62.	6.	13.0
199.	7	0.1818	1.0000	62.	6.	17.0
200.	11	0.2222	0.6667	62.	6.	21.0
201.	13	0.6667	0.5000	62.	6.	30.0
202.	12	0.6667	1.0000	62.	7.	22.0
203.	12	0.6667	1.0000	62.	8.	23.0
204.	13	0.6667	1.0000	63.	3.	30.0
205.	2	0.8000	0.2069	63.	4.	0.0
206.	2	0.8000	0.7059	63.	4.	1.0
207.	2	0.8000	0.8571	63.	4.	2.0
208.	2	0.8000	1.0000	63.	4.	3.0

FILE: APPENDIX D A1

209.	2	0.8000	1.0000	63.	4.	4.0
210.	2	0.8000	1.0000	63.	4.	5.0
211.	2	0.8000	1.0000	63.	4.	7.0
212.	2	0.8000	1.0000	63.	4.	8.0
213.	7	0.1818	1.0000	63.	4.	9.0
214.	3	0.1772	0.0488	63.	5.	3.0
215.	3	0.1772	0.0385	63.	5.	4.0
216.	3	0.1772	0.1765	63.	5.	5.0
217.	3	0.1772	0.3571	63.	5.	6.0
218.	3	0.1772	0.1290	63.	5.	7.0
219.	7	0.1818	0.5000	63.	5.	10.0
220.	3	0.1772	0.5263	63.	6.	3.0
221.	3	0.1772	0.1667	63.	6.	4.0
222.	3	0.1772	0.1918	63.	6.	5.0
223.	3	0.1772	0.1852	63.	6.	6.0
224.	3	0.1772	0.1839	63.	6.	7.0
225.	3	0.1772	0.1250	63.	6.	8.0
226.	7	0.1818	0.1702	63.	6.	9.0
227.	7	0.1818	0.1892	63.	6.	10.0
228.	7	0.1818	0.1290	63.	6.	11.0
229.	7	0.1818	0.1429	63.	6.	12.0
230.	7	0.1818	0.1250	63.	6.	13.0
231.	7	0.1818	0.6667	63.	6.	15.0
232.	8	0.2857	0.5000	63.	6.	21.0
233.	7	0.1818	0.1111	63.	7.	10.0
234.	7	0.1818	0.0308	63.	7.	11.0
235.	7	0.1818	0.0800	63.	7.	12.0
236.	7	0.1818	0.0571	63.	7.	13.0
237.	7	0.1818	0.0408	63.	7.	14.0
238.	8	0.2857	0.6667	63.	7.	20.0
239.	8	0.2857	1.0000	63.	7.	22.0
240.	8	0.2857	0.6667	63.	7.	27.0
241.	13	0.6667	1.0000	63.	7.	30.0
242.	8	0.2857	0.0571	63.	8.	20.0
243.	9	0.4286	0.4000	63.	8.	28.0
244.	12	0.6667	1.0000	64.	0.	22.0
245.	7	0.1818	0.2222	64.	1.	17.0
246.	4	0.2857	0.3810	64.	5.	3.0
247.	4	0.2857	0.2609	64.	5.	4.0
248.	4	0.2857	0.0800	64.	5.	5.0
249.	4	0.2857	0.2000	64.	5.	7.0
250.	4	0.2857	0.2857	64.	5.	8.0
251.	7	0.1818	0.6667	64.	5.	11.0
252.	4	0.2857	1.0000	64.	6.	4.0
253.	7	0.1818	0.3636	64.	6.	10.0
254.	7	0.1818	0.2000	64.	6.	11.0
255.	11	0.2222	0.2857	64.	6.	20.0
256.	11	0.2222	0.1429	64.	6.	22.0
257.	11	0.2222	0.1429	64.	6.	23.0
258.	11	0.2222	0.1053	64.	6.	24.0
259.	11	0.2222	0.4615	64.	6.	25.0
260.	12	0.6667	0.3636	64.	7.	21.0
261.	12	0.6667	0.2500	64.	7.	22.0
262.	12	0.6667	0.2222	64.	7.	23.0
263.	12	0.6667	0.6667	64.	7.	26.0

FILE: APPENDIX D A1

264.	12	0.6667	0.4000	64.	7.	27.0
265.	12	0.6667	1.0000	64.	7.	28.0
266.	13	0.6667	0.4000	64.	7.	30.0
267.	12	0.6667	1.0000	64.	8.	25.0

OUTPUT OF THE PROGRAM : CART-TEST

PROGRAM PURPOSE : PROVIDES A SUMMARY OF RESULTS OBTAINED FROM BUILD AND CASE PROGRAMS.

1 WELCOME TO CART (TM) Version 1.0 November 20, 1984

Summary statistics for a test sample with 267 cases.

Relative error based on tree = 0.185 +/- 0.727
Initial mean absolute deviation = 0.232
Initial median = 0.286

1 13 TERMINAL NODES

	LEARNING SAMPLE			TEST SAMPLE			
NODE	CASES	MEDIAN	MEAN AD	CASES	MEDIAN	MEAN AD	MEAN ERROR
1	7	0.227E-01	0.64E-01	5	0.286	0.28	0.35
2	9	0.800	0.12	9	1.00	0.24	0.28
3	44	0.177	0.12	39	0.143	0.94E-01	0.09
4	24	0.286	0.16	21	0.250	0.16	0.16
5	1	1.00	0.00E+00	1	1.00	0.00E+00	0.00
6	64	0.800E-01	0.79E-01	48	0.133	0.78E-01	0.08
7	28	0.182	0.67E-01	20	0.200	0.23	0.23
8	148	0.286	0.13	80	0.417	0.22	0.24
9	21	0.429	0.14	13	0.500	0.16	0.17
10	4	1.00	0.22	4	1.00	0.21	0.21
11	9	0.222	0.15	6	0.286	0.17	0.17
12	22	0.667	0.16	10	1.00	0.31	0.31
13	22	0.667	0.18	11	0.500	0.24	0.25

APPENDIX E

OUTPUTS OF THE CART PROGRAMS USING
5TH GENERAL CATEGORIZATION DATA AS INPUT

```
OUTPUT OF THE PROGRAM : CART-BUILD

PROGRAM PURPOSE : BUILDING A REGRESSION TREE

INPUT FILE :
             NAME     : BUILD5
             CONTENTS : GENERAL MOS CAT. 5 DATA
```

TREE SEQUENCE

Tree	Terminal Nodes	Cross-Validated Relative Error	Resubstitution Relative Error	Complexity Parameter
1	50	1.01 +/- 0.000	0.34	0.000
2	49	1.01 +/- 0.000	0.34	0.006
3	48	1.01 +/- 0.000	0.34	0.033
4	47	1.00 +/- 0.000	0.35	0.040
5	46	1.00 +/- 0.000	0.35	0.062
6	45	1.00 +/- 0.000	0.35	0.066
7	44	1.00 +/- 0.000	0.35	0.068
8	43	1.00 +/- 0.000	0.36	0.071
9	42	1.00 +/- 0.000	0.36	0.083
10	39	1.01 +/- 0.000	0.37	0.081
11	36	1.01 +/- 0.000	0.38	0.088
12	33	1.01 +/- 0.000	0.39	0.097
13	32	1.01 +/- 0.000	0.39	0.099
14	30	1.00 +/- 0.000	0.40	0.126
15	29	0.99 +/- 0.000	0.41	0.157
16	28	0.98 +/- 0.000	0.42	0.200
17	26	0.97 +/- 0.000	0.43	0.219
18	25	0.97 +/- 0.000	0.44	0.222
19	22	0.98 +/- 0.000	0.47	0.247
20	21	0.98 +/- 0.000	0.48	0.261
21	20	0.98 +/- 0.000	0.49	0.271
22	19	0.96 +/- 0.000	0.50	0.333
23	14	0.98 +/- 0.000	0.56	0.336
24	13	0.97 +/- 0.000	0.58	0.396
25	11	0.98 +/- 0.000	0.62	0.517
26	9	1.05 +/- 0.000	0.67	0.667
27	8	1.06 +/- 0.000	0.69	0.687
28	7	1.03 +/- 0.000	0.72	0.762
29	5	1.02 +/- 0.000	0.78	0.813
30	4	1.04 +/- 0.000	0.81	0.842
31	2	1.02 +/- 0.000	0.91	1.31

FILE: APPENDIX E A1

32* 1 1.00 +/- 0.000 1.00 2.31

Initial median = 0.286
Initial mean absolute deviation = 0.196

No tree created.
1 1 TERMINAL NODES

Node	Cases	Median	Mean Ad
1	59	0.286	0.45

	Relative Importance	Number Of Categories	Minimum Category
LOS	100.	numerical	
GRADE	76.	9	0
MOS	0.	3	65

Number of cases in the learning sample = 136

PRIMARY OPTION SETTINGS

construction rule	least absolute deviation
estimation method	5-fold cross-validation
tree selection	1.0 se rule
variables used	See variable importance list above. response is variable RATE
linear combinations	no

SECONDARY OPTION SETTINGS

1	Minimum node size	=	1
2	Minimum size below which node will not be split	=	5
3	Number of surrogate splits printed	=	2
4	Number of competing splits printed	=	2
5	Maximum number of trees for which errors are printed	=	100
6	Initial value of the complexity parameter	=	0.0000E+00
7	Maximum number of cases to put into learning sample	=	20000
8	Maximum number of cases to put into test sample	=	20000
9	Maximum node size without sub-sampling the node	=	137
10	Maximum number of surrogates used	=	2
11	Maximum number of nodes allowed for in large tree	=	750
	(Actual maximum number of nodes	=	51)
12	Max. categorical primary + surrogate splits in a tree	=	1000
13	Max. linear combination splits in a tree	=	0
	(Actual number cat. + linear combination splits	=	37)
	(Actual number categorical competitor splits	=	6)
14	Maximum height of tree	=	750
	(Actual maximum height of tree	=	11)
	Maximum size of memory available	=	70000
	(Actual size of memory used in run	=	42053)

FILE: APPENDIX E A1

OUTPUT OF THE PROGRAM : CART-CASE

PROGRAM PURPOSE : VALIDATION OF RESULTS

INPUT FILE :
NAME : TEST5
CONTENTS : GENERAL MOS CAT. 5 DATA

CASE	NODE	PREDICTED	RATE	MOS	GRADE	LOS
1.	1	0.2857	1.0000	65.	0.	20.0
2.	1	0.2857	0.1538	65.	1.	10.0
3.	1	0.2857	0.6667	65.	3.	25.0
4.	1	0.2857	1.0000	65.	3.	28.0
5.	1	0.2857	1.0000	65.	3.	29.0
6.	1	0.2857	0.4000	65.	3.	30.0
7.	1	0.2857	0.4000	65.	4.	5.0
8.	1	0.2857	0.1538	65.	5.	3.0
9.	1	0.2857	0.3529	65.	5.	4.0
10.	1	0.2857	0.1212	65.	5.	5.0
11.	1	0.2857	0.1935	65.	5.	6.0
12.	1	0.2857	0.3750	65.	5.	7.0
13.	1	0.2857	0.4000	65.	5.	8.0
14.	1	0.2857	0.4000	65.	5.	9.0
15.	1	0.2857	0.1176	65.	6.	6.0
16.	1	0.2857	0.8571	65.	6.	9.0
17.	1	0.2857	1.0000	65.	6.	20.0
18.	1	0.2857	0.4615	65.	6.	21.0
19.	1	0.2857	0.1333	65.	6.	22.0
20.	1	0.2857	0.1429	65.	6.	23.0
21.	1	0.2857	0.5455	65.	6.	24.0
22.	1	0.2857	0.1111	65.	6.	25.0
23.	1	0.2857	0.2222	65.	6.	26.0
24.	1	0.2857	0.3333	65.	6.	27.0
25.	1	0.2857	1.0000	65.	7.	20.0
26.	1	0.2857	0.2000	65.	7.	26.0
27.	1	0.2857	0.5000	65.	7.	29.0
28.	1	0.2857	1.0000	65.	8.	22.0
29.	1	0.2857	0.6667	65.	8.	27.0
30.	1	0.2857	0.4000	65.	8.	30.0
31.	1	0.2857	0.2222	66.	1.	14.0
32.	1	0.2857	0.4000	66.	1.	20.0
33.	1	0.2857	0.6667	66.	1.	21.0
34.	1	0.2857	1.0000	66.	1.	22.0
35.	1	0.2857	0.6667	66.	3.	23.0
36.	1	0.2857	0.0588	66.	4.	4.0
37.	1	0.2857	0.2273	66.	5.	3.0
38.	1	0.2857	0.1284	66.	5.	4.0
39.	1	0.2857	0.1633	66.	5.	5.0
40.	1	0.2857	0.1290	66.	5.	6.0
41.	1	0.2857	0.3158	66.	5.	7.0

FILE: APPENDIX E A1

42.	1	0.2857	0.3158	66.	5.	10.0
43.	1	0.2857	1.0000	66.	5.	11.0
44.	1	0.2857	0.4828	66.	6.	5.0
45.	1	0.2857	0.0357	66.	6.	6.0
46.	1	0.2857	0.1282	66.	6.	7.0
47.	1	0.2857	0.0274	66.	6.	8.0
48.	1	0.2857	0.0299	66.	6.	9.0
49.	1	0.2857	0.0282	66.	6.	10.0
50.	1	0.2857	0.0930	66.	6.	11.0
51.	1	0.2857	0.1250	66.	6.	12.0
52.	1	0.2857	0.0769	66.	6.	13.0
53.	1	0.2857	0.0870	66.	6.	14.0
54.	1	0.2857	0.0800	66.	6.	15.0
55.	1	0.2857	0.3077	66.	6.	16.0
56.	1	0.2857	0.4000	66.	6.	18.0
57.	1	0.2857	0.5000	66.	6.	20.0
58.	1	0.2857	0.0645	66.	7.	11.0
59.	1	0.2857	0.0571	66.	7.	13.0
60.	1	0.2857	0.8387	66.	7.	20.0
61.	1	0.2857	0.1538	66.	7.	21.0
62.	1	0.2857	0.0833	66.	8.	20.0
63.	1	0.2857	0.1000	66.	8.	21.0
64.	1	0.2857	0.1000	66.	8.	22.0
65.	1	0.2857	0.3077	66.	8.	23.0
66.	1	0.2857	0.6667	66.	8.	24.0
67.	1	0.2857	0.8000	66.	8.	26.0
68.	1	0.2857	0.3333	66.	8.	30.0
69.	1	0.2857	0.1538	67.	1.	10.0
70.	1	0.2857	0.2222	67.	1.	12.0
71.	1	0.2857	0.1000	67.	5.	3.0
72.	1	0.2857	0.3846	67.	5.	4.0
73.	1	0.2857	0.2222	67.	5.	7.0
74.	1	0.2857	0.6667	67.	5.	10.0
75.	1	0.2857	0.2857	67.	5.	16.0
76.	1	0.2857	0.2000	67.	6.	8.0
77.	1	0.2857	0.4000	67.	6.	12.0
78.	1	0.2857	0.4000	67.	6.	20.0
79.	1	0.2857	1.0000	67.	6.	26.0

```
OUTPUT OF THE PROGRAM : CART-TEST

PROGRAM PURPOSE : PROVIDES A SUMMARY OF RESULTS
                  OBTAINED FROM BUILD AND CASE
                  PROGRAMS.
```

1 WELCOME TO CART (TM) Version 1.0 November 20, 1984

FILE: APPENDIX E A1

(415) 283-3392

Summary statistics for a test sample with 79 cases.

Relative error based on tree	=	0.238	+/- 0.836E-02
Initial mean absolute deviation	=	0.237	
Initial median	=	0.308	

1 1 TERMINAL NODES

NODE	LEARNING SAMPLE CASES	MEDIAN	MEAN AD	TEST SAMPLE CASES	MEDIAN	MEAN AD	MEAN ERROR
1	0	0.286	0.00E+00	*****	0.308	0.17E-07	0.0

APPENDIX F

OUTPUTS OF THE CART PROGRAMS USING
6TH GENERAL CATEGORIZATION DATA AS INPUT

```
OUTPUT OF THE PROGRAM : CART-BUILD

PROGRAM PURPOSE : BUILDING A REGRESSION TREE

INPUT FILE :
             NAME     : BUILD6
             CONTENTS : GENERAL MOS CAT. 6 DATA
```

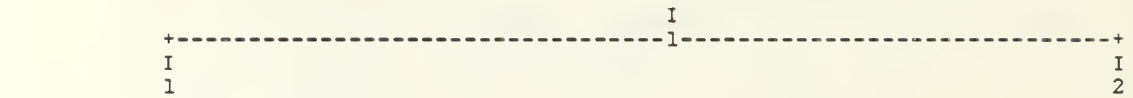
TREE SEQUENCE

TREE	TERMINAL NODES	CROSS-VALIDATED RELATIVE ERROR	RESUBSTITUTION RELATIVE ERROR	COMPLEXITY PARAMETER
1	44	1.21 +/- 0.000	0.43	0.0
2	43	1.21 +/- 0.000	0.43	0.004
3	42	1.21 +/- 0.000	0.43	0.012
4	41	1.21 +/- 0.000	0.43	0.016
5	40	1.21 +/- 0.000	0.43	0.026
6	39	1.21 +/- 0.000	0.43	0.026
7	38	1.21 +/- 0.000	0.43	0.037
8	37	1.21 +/- 0.000	0.44	0.039
9	36	1.21 +/- 0.000	0.44	0.045
10	35	1.21 +/- 0.000	0.44	0.045
11	34	1.21 +/- 0.000	0.44	0.060
12	33	1.21 +/- 0.000	0.45	0.063
13	32	1.21 +/- 0.000	0.45	0.066
14	31	1.20 +/- 0.000	0.45	0.068
15	30	1.18 +/- 0.000	0.46	0.114
16	24	1.18 +/- 0.000	0.50	0.138
17	23	1.17 +/- 0.000	0.50	0.152
18	22	1.17 +/- 0.000	0.51	0.167
19	19	1.14 +/- 0.000	0.53	0.169
20	18	1.13 +/- 0.000	0.54	0.202
21	17	1.14 +/- 0.000	0.55	0.214
22	16	1.14 +/- 0.000	0.56	0.226
23	15	1.14 +/- 0.000	0.58	0.250
24	13	1.06 +/- 0.000	0.60	0.260
25	12	1.05 +/- 0.000	0.62	0.346
26	9	0.94 +/- 0.000	0.67	0.379
27	5	0.94 +/- 0.000	0.75	0.448
28	4	0.94 +/- 0.000	0.78	0.500
29	3	0.92 +/- 0.000	0.82	0.988
30*	2	0.97 +/- 0.000	0.89	1.42
31	1	1.00 +/- 0.000	1.00	2.39

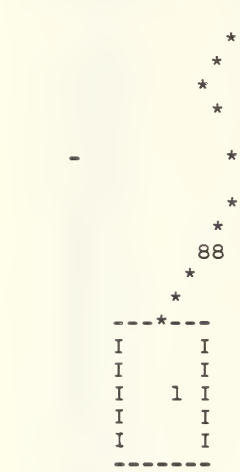
FILE: APPENDIX F A1

Initial median = 0.313
Initial mean absolute deviation = 0.185

CLASSIFICATION/REGRESSION TREE



1 PARTITIONING TREE



Node 1 was split on variable LOS
A case goes left if variable LOS .le. 2.35E+01
Improvement = 2.0E-02 (C. T. = 2.3E+00)

Node	Cases	Median	Mean Abs Dev.
1	116	0.31	0.18
-1	88	0.25	0.16
-2	28	0.50	0.19

	Competitor	Split	Improve.
1	GRADE	1, 2, 3, 7, 8	2.0E-02
2	MOS	68, 69	1.8E-03

1 2 TERMINAL NODES

Node	Cases	Median	Mean Ad
1	88	0.250	0.16
2	28	0.500	0.19

Relative Number Of Minimum

FILE: APPENDIX F A1

	Importance	Categories	Category
LOS	100.	numerical	
GRADE	49.	10	0
MOS	25.	3	68

Number of cases in the learning sample = 116

PRIMARY OPTION SETTINGS

construction rule	least absolute deviation
estimation method	3-fold cross-validation
tree selection	1.0 se rule
variables used	See variable importance list above. response is variable RATE
linear combinations	no

SECONDARY OPTION SETTINGS

1	Minimum node size	= 1
2	Minimum size below which node will not be split	= 5
3	Number of surrogate splits printed	= 2
4	Number of competing splits printed	= 2
5	Maximum number of trees for which errors are printed	= 100
6	Initial value of the complexity parameter	= 0.0000E+00
7	Maximum number of cases to put into learning sample	= 20000
8	Maximum number of cases to put into test sample	= 20000
9	Maximum node size without sub-sampling the node	= 117
10	Maximum number of surrogates used	= 2
11	Maximum number of nodes allowed for in large tree	= 750
	(Actual maximum number of nodes	= 45)
12	Max. categorical primary + surrogate splits in a tree	= 1000
13	Max. linear combination splits in a tree	= 0
	(Actual number cat. + linear combination splits	= 30)
	(Actual number categorical competitor splits	= 35)
14	Maximum height of tree	= 750
	(Actual maximum height of tree	= 12)
	Maximum size of memory available	= 70000
	(Actual size of memory used in run	= 43747)

OUTPUT OF THE PROGRAM : CART-CASE

PROGRAM PURPOSE : VALIDATION OF RESULTS

INPUT FILE :
NAME : TEST6
CONTENTS : GENERAL MOS CAT. 6 DATA

FILE: APPENDIX F A1

CASE	NODE	PREDICTED	RATE	MOS	GRADE	LOS
1.	1	0.2500	0.2857	68.	1.	9.0
2.	1	0.2500	0.3333	68.	1.	12.0
3.	1	0.2500	0.3333	68.	5.	3.0
4.	1	0.2500	0.2093	68.	5.	4.0
5.	1	0.2500	0.2353	68.	5.	5.0
6.	1	0.2500	0.1667	68.	5.	6.0
7.	1	0.2500	0.1905	68.	5.	7.0
8.	1	0.2500	0.3333	68.	5.	8.0
9.	1	0.2500	0.5000	68.	5.	11.0
10.	1	0.2500	0.5000	68.	5.	21.0
11.	1	0.2500	0.4444	68.	6.	5.0
12.	1	0.2500	0.0952	68.	6.	6.0
13.	1	0.2500	0.0952	68.	6.	11.0
14.	1	0.2500	0.2222	68.	6.	12.0
15.	1	0.2500	0.1818	68.	6.	15.0
16.	1	0.2500	0.5000	68.	6.	20.0
17.	1	0.2500	0.5000	68.	6.	21.0
18.	2	0.5000	1.0000	68.	6.	30.0
19.	1	0.2500	0.4444	68.	7.	20.0
20.	1	0.2500	0.2222	68.	7.	21.0
21.	1	0.2500	0.6667	68.	7.	22.0
22.	2	0.5000	0.2857	68.	7.	25.0
23.	2	0.5000	0.8000	68.	7.	26.0
24.	1	0.2500	0.3333	68.	8.	20.0
25.	1	0.2500	0.2000	68.	8.	22.0
26.	2	0.5000	0.5000	68.	8.	24.0
27.	2	0.5000	1.0000	68.	8.	25.0
28.	2	0.5000	1.0000	68.	8.	26.0
29.	1	0.2500	1.0000	69.	0.	16.0
30.	1	0.2500	0.4000	69.	1.	13.0
31.	1	0.2500	0.1538	69.	4.	2.0
32.	1	0.2500	0.1000	69.	5.	3.0
33.	1	0.2500	0.2857	69.	5.	4.0
34.	1	0.2500	0.1905	69.	5.	5.0
35.	1	0.2500	0.2353	69.	5.	6.0
36.	1	0.2500	0.4444	69.	5.	7.0
37.	1	0.2500	0.2222	69.	5.	13.0
38.	1	0.2500	0.6667	69.	5.	21.0
39.	1	0.2500	0.2500	69.	6.	5.0
40.	1	0.2500	0.1176	69.	6.	9.0
41.	1	0.2500	0.1818	69.	6.	12.0
42.	1	0.2500	0.1333	69.	6.	21.0
43.	2	0.5000	0.2222	69.	6.	25.0
44.	2	0.5000	0.5000	69.	6.	26.0
45.	1	0.2500	0.6667	69.	7.	22.0
46.	2	0.5000	0.2222	69.	7.	30.0
47.	1	0.2500	0.5000	69.	8.	21.0
48.	1	0.2500	0.5000	70.	0.	9.0
49.	2	0.5000	1.0000	70.	1.	26.0
50.	1	0.2500	0.0769	70.	4.	1.0
51.	1	0.2500	0.1818	70.	5.	3.0
52.	1	0.2500	0.1739	70.	5.	4.0
53.	1	0.2500	0.2963	70.	5.	5.0
54.	1	0.2500	0.3750	70.	5.	6.0

FILE: APPENDIX F A1

55.	1	0.2500	0.2500	70.	5.	7.0
56.	1	0.2500	0.1818	70.	5.	10.0
57.	1	0.2500	0.1538	70.	5.	11.0
58.	1	0.2500	0.2500	70.	5.	18.0
59.	1	0.2500	0.1538	70.	6.	7.0
60.	1	0.2500	0.0625	70.	6.	8.0
61.	1	0.2500	0.1212	70.	6.	9.0
62.	1	0.2500	0.0800	70.	6.	10.0
63.	1	0.2500	0.4286	70.	6.	12.0
64.	1	0.2500	0.4000	70.	6.	13.0
65.	1	0.2500	0.2500	70.	6.	18.0
66.	1	0.2500	0.6667	70.	6.	20.0
67.	1	0.2500	0.1250	70.	7.	19.0
68.	1	0.2500	0.9231	70.	7.	20.0
69.	1	0.2500	0.6667	70.	7.	21.0
70.	1	0.2500	0.6667	70.	7.	23.0
71.	2	0.5000	1.0000	70.	7.	24.0
72.	2	0.5000	0.3333	70.	7.	26.0
73.	1	0.2500	0.6667	70.	8.	20.0
74.	1	0.2500	0.1667	70.	8.	22.0
75.	1	0.2500	0.5000	70.	8.	23.0

OUTPUT OF THE PROGRAM : CART-TEST

PROGRAM PURPOSE : PROVIDES A SUMMARY OF RESULTS OBTAINED FROM BUILD AND CASE PROGRAMS.

1 WELCOME TO CART (TM) Version 1.0 November 20, 1984

Summary statistics for a test sample with 75 cases.

Relative error based on tree = 0.188 +/- 0.193
Initial mean absolute deviation = 0.202
Initial median = 0.286

1 2 TERMINAL NODES

	LEARNING SAMPLE			TEST SAMPLE			
NODE	CASES	MEDIAN	MEAN AD	CASES	MEDIAN	MEAN AD	MEAN ERROR
1	88	0.250	0.16	63	0.250	0.16	0.16
2	28	0.500	0.19	12	0.800	0.31	0.31

APPENDIX G

OUTPUTS OF THE CART PROGRAMS USING
7TH GENERAL CATEGORIZATION DATA AS INPUT

OUTPUT OF THE PROGRAM : CART-BUILD

PROGRAM PURPOSE : BUILDING A REGRESSION TREE

INPUT FILE :
NAME : BUILD7
CONTENTS : GENERAL MOS CAT. 7 DATA

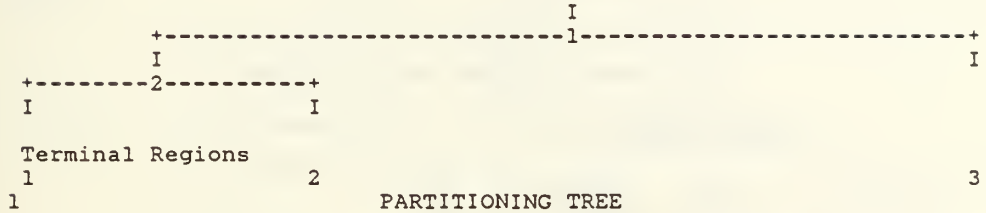
TREE SEQUENCE

Tree	Terminal Nodes	Cross-Validated Relative Error	Resubstitution Relative Error	Complexity Parameter
1	23	0.76 +/- 0.000	0.35	0.000
2	21	0.76 +/- 0.000	0.35	0.026
3	20	0.76 +/- 0.000	0.35	0.038
4	19	0.75 +/- 0.000	0.35	0.060
5	17	0.76 +/- 0.000	0.36	0.083
6	14	0.76 +/- 0.000	0.37	0.088
7	13	0.76 +/- 0.000	0.39	0.300
8	9	0.75 +/- 0.000	0.45	0.304
9	7	0.68 +/- 0.000	0.48	0.333
10	6	0.68 +/- 0.000	0.50	0.400
11	4	0.64 +/- 0.000	0.54	0.429
12*	3	0.61 +/- 0.000	0.57	0.643
13	2	0.71 +/- 0.000	0.68	2.31
14	1	1.01 +/- 0.000	1.00	6.86

Initial median = 0.500
Initial mean absolute deviation = 0.240

FILE: APPENDIX G A1

CLASSIFICATION/REGRESSION TREE



Node 1 was split on variable GRADE
A CASE GOES LEFT IF VARIABLE GRADE IS IN (1, 2, 3, 5, 6, 7, 8)
Improvement = 7.8E-02 (C. T. = 6.8E+00)

Node	Cases	Median	Mean Abs Dev.
1	88	0.50	0.24
2	69	0.57	0.19
-3	19	0.52E-01	0.63E-01

	Surrogate		Split	Assoc.	Improve.
1	MOS	r	77	0.63	7.1E-02
2	LOS	r	1.50E+00	0.10	9.5E-03

	Competitor	Split	Improve.
1	MOS	71, 72, 73, 74, 75, 76	7.1E-02
2	LOS	6.50E+00	1.6E-02

Node 2 was split on variable MOS
A CASE GOES LEFT IF VARIABLE MOS IS IN (71, 72, 73, 74
Improvement = 2.6E-02 (C. T. = 2.3E+00)

Node	Cases	Median	Mean Abs Dev.
2	69	0.57	0.19
-1	41	0.67	0.15
-2	28	0.40	0.16

	Surrogate		Split	Assoc.	Improve.
1	GRADE	s	0, 1, 2, 4, 5,	0.07	-3.2E-08

FILE: APPENDIX G A1

```
   ---*---      ---*---                          6, 7, 8, 9
   I     I      I     I     2 LOS          s     2.95E+01          0.03     2.7E-03
   I     I      I     I
   I   1 I      I   2 I       Competitor        Split                       Improve.
   I     I      I     I     1 LOS               4.50E+00                    8.6E-03
   I     I      I     I     2 GRADE              2, 3, 5, 6                 6.7E-03
   -------      -------
```

1 3 TERMINAL NODES

Node	Cases	Median	Mean Ad
1	41	0.667	0.15
2	28	0.400	0.16
3	19	0.522E-01	0.63E-01

	Relative Importance	Number Of Categories	Minimum Category
MOS	100.	7	71
GRADE	79.	10	0
LOS	43.	numerical	

Number of cases in the learning sample = 88

PRIMARY OPTION SETTINGS

construction rule	least absolute deviation
estimation method	10-fold cross-validation
tree selection	1.0 se rule
variables used	See variable importance list above. response is variable RATE
linear combinations	no

SECONDARY OPTION SETTINGS

1	Minimum node size	=	1
2	Minimum size below which node will not be split	=	5
3	Number of surrogate splits printed	=	2
4	Number of competing splits printed	=	2
5	Maximum number of trees for which errors are printed	=	100
6	Initial value of the complexity parameter	=	0.0000E+00
7	Maximum number of cases to put into learning sample	=	20000
8	Maximum number of cases to put into test sample	=	20000
9	Maximum node size without sub-sampling the node	=	89
10	Maximum number of surrogates used	=	2
11	Maximum number of nodes allowed for in large tree	=	750
	(Actual maximum number of nodes	=	32)
12	Max. categorical primary + surrogate splits in a tree	=	1000
13	Max. linear combination splits in a tree	=	0
	(Actual number cat. + linear combination splits	=	27)
	(Actual number categorical competitor splits	=	17)

FILE: APPENDIX G A1

```
 14 Maximum height of tree                              =    750
       (Actual maximum height of tree                   =     12)
    Maximum size of memory available                    =  70000
       (Actual size of memory used in run               =  43216)
```

OUTPUT OF THE PROGRAM : CART-CASE

PROGRAM PURPOSE : VALIDATION OF RESULTS

INPUT FILE :
NAME : TEST7
CONTENTS : GENERAL MOS CAT. 7 DATA

CASE	NODE	PREDICTED	RATE	MOS	GRADE	LOS
1.	2	0.4000	0.5000	76.	1.	10.0
2.	2	0.4000	0.6667	76.	5.	20.0
3.	2	0.4000	0.6667	76.	6.	25.0
4.	2	0.4000	0.6667	76.	6.	28.0
5.	2	0.4000	1.0000	76.	6.	29.0
6.	2	0.4000	0.6667	76.	7.	24.0
7.	2	0.4000	1.0000	76.	7.	30.0
9.	1	0.6667	1.0000	72.	2.	20.0
10.	1	0.6667	1.0000	72.	2.	23.0
11.	1	0.6667	1.0000	72.	3.	26.0
12.	1	0.6667	1.0000	72.	6.	14.0
13.	1	0.6667	0.6667	72.	6.	21.0
14.	1	0.6667	0.6667	72.	6.	22.0
15.	1	0.6667	0.6667	72.	6.	24.0
16.	1	0.6667	0.6667	72.	7.	12.0
17.	1	0.6667	1.0000	73.	7.	21.0
18.	1	0.6667	1.0000	73.	8.	21.0
19.	1	0.6667	1.0000	74.	6.	12.0
20.	1	0.6667	0.8571	74.	6.	20.0
21.	1	0.6667	1.0000	74.	6.	24.0
22.	1	0.6667	0.6667	74.	7.	28.0
23.	2	0.4000	1.0000	75.	3.	30.0
24.	2	0.4000	0.4000	75.	7.	30.0
25.	2	0.4000	1.0000	77.	3.	21.0
26.	2	0.4000	0.4000	77.	3.	23.0
27.	2	0.4000	0.4000	77.	3.	24.0
28.	2	0.4000	0.5000	77.	3.	25.0
29.	2	0.4000	1.0000	77.	3.	30.0
30.	3	0.0522	0.0191	77.	4.	0.0
31.	3	0.0522	0.0302	77.	4.	1.0
32.	3	0.0522	0.0214	77.	4.	2.0
33.	3	0.0522	0.0123	77.	4.	3.0
34.	3	0.0522	0.0855	77.	4.	4.0
35.	3	0.0522	0.1600	77.	4.	5.0

FILE: APPENDIX G A1

36.	3	0.0522	0.0290	77.	4.	6.0
37.	3	0.0522	0.0769	77.	4.	7.0
38.	3	0.0522	0.0556	77.	4.	8.0
39.	3	0.0522	0.1818	77.	4.	10.0
40.	3	0.0522	0.2500	77.	4.	30.0
41.	2	0.4000	0.6667	77.	5.	9.0
42.	3	0.0522	0.2857	77.	9.	20.0
43.	3	0.0522	0.0197	77.	9.	23.0
44.	3	0.0522	0.0286	77.	9.	24.0
45.	3	0.0522	0.0438	77.	9.	25.0
46.	3	0.0522	0.1318	77.	9.	26.0
47.	3	0.0522	0.2080	77.	9.	27.0
48.	3	0.0522	0.1685	77.	9.	28.0
49.	3	0.0522	0.1719	77.	9.	29.0

OUTPUT OF THE PROGRAM : CART-TEST

PROGRAM PURPOSE : PROVIDES A SUMMARY OF RESULTS OBTAINED FROM BUILD AND CASE PROGRAMS.

1 WELCOME TO CART (TM) Version 1.0 November 20, 1984

Summary statistics for a test sample with 49 cases.

Relative error based on tree = 0.180 +/- 1.72
Initial mean absolute deviation = 0.334
Initial median = 0.667

1 3 TERMINAL NODES

NODE	LEARNING SAMPLE CASES	MEDIAN	MEAN AD	TEST SAMPLE CASES	MEDIAN	MEAN AD	MEAN ERROR
1	41	0.667	0.15	15	1.00	0.14	0.19
2	28	0.400	0.16	15	0.667	0.19	0.30
3	19	0.522E-01	0.63E-01	19	0.769E-01	0.73E-01	0.74E-01

APPENDIX H

OUTPUTS OF THE CART PROGRAMS USING
8TH GENERAL CATEGORIZATION DATA AS INPUT

```
OUTPUT OF THE PROGRAM : CART-BUILD

PROGRAM PURPOSE : BUILDING A REGRESSION TREE

INPUT FILE :
             NAME     : BUILD8
             CONTENTS : GENERAL MOS CAT. 8 DATA
```

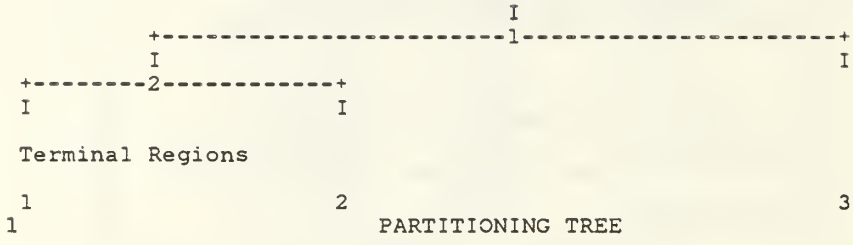
TREE SEQUENCE

Tree	Terminal Nodes	Cross-Validated Relative Error	Resubstitution Relative Error	Complexity Parameter
1	31	1.14 +/- 0.000	0.27	0.000
2	29	1.11 +/- 0.000	0.28	0.095
3	26	1.11 +/- 0.000	0.30	0.103
4	25	1.11 +/- 0.000	0.30	0.107
5	24	1.11 +/- 0.000	0.31	0.114
6	21	1.09 +/- 0.000	0.34	0.148
7	20	1.09 +/- 0.000	0.35	0.167
8	19	1.09 +/- 0.000	0.36	0.172
9	18	1.08 +/- 0.000	0.37	0.190
10	17	1.08 +/- 0.000	0.38	0.218
11	15	1.02 +/- 0.000	0.41	0.292
12	14	0.97 +/- 0.000	0.44	0.357
13	13	0.95 +/- 0.000	0.46	0.381
14	12	0.89 +/- 0.000	0.48	0.400
15	9	0.91 +/- 0.000	0.56	0.454
16	8	0.91 +/- 0.000	0.59	0.467
17	7	0.96 +/- 0.000	0.62	0.500
18	6	0.98 +/- 0.000	0.67	0.810
19*	3	0.98 +/- 0.000	0.81	0.823
20	2	1.00 +/- 0.000	0.88	1.14
21	1	1.00 +/- 0.000	1.00	1.98

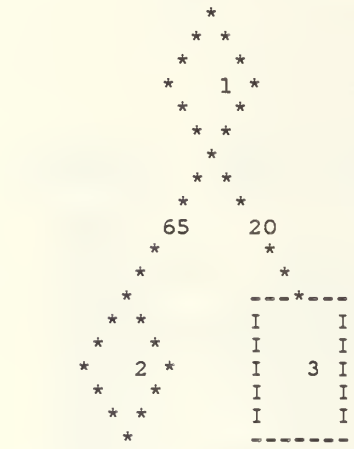
Initial median = 0.400
Initial mean absolute deviation = 0.198

CLASSIFICATION/REGRESSION TREE

FILE: APPENDIX H A1



PARTITIONING TREE

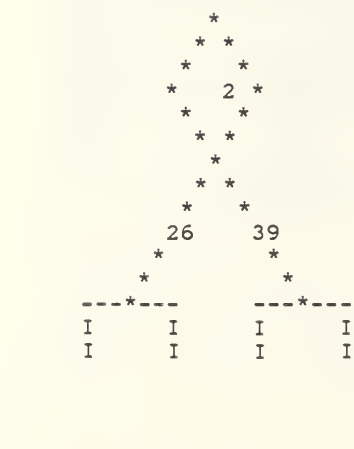


Node 1 was split on variable LOS
A case goes left if variable LOS .le. 2.75E+01
Improvement = 2.3E-02 (C. T. = 1.9E+00)

Node	Cases	Median	Mean Abs Dev.
1	85	0.40	0.20
2	65	0.33	0.16
-3	20	0.67	0.21

	Surrogate		Split	Assoc.	Improve.
1	GRADE	s	0, 1, 2, 3, 4, 5, 6, 7, 9	0.05	5.3E-03

	Competitor	Split	Improve.
1	GRADE	3, 6, 7, 8	1.9E-02
2	MOS	80, 81	2.9E-04



Node 2 was split on variable GRADE
A case goes left if variable GRADE is in (3, 7)
Improvement = 1.3E-02 (C. T. = 1.1E+00)

Node	Cases	Median	Mean Abs Dev.
2	65	0.33	0.16
-1	26	0.40	0.13
-2	39	0.27	0.16

	Surrogate		Split	Assoc.	Improve.
1	LOS	r	2.35E+01	0.27	4.6E-03

	Competitor	Split	Improve.
1	LOS	1.65E+01	1.1E-02

FILE: APPENDIX H A1

```
I   1 I     I   2 I    2 MOS          81                    3.2E-03
I     I     I     I
I     I     I     I
-------     -------
```

1 3 TERMINAL NODES

Node	Cases	Median	Mean Ad
1	26	0.400	0.13
2	39	0.267	0.16
3	20	0.667	0.21

	Relative Importance	Number Of Categories	Minimum Category
LOS	100.	numerical	
GRADE	89.	10	0
MOS	44.	5	78

Number of cases in the learning sample = 85

PRIMARY OPTION SETTINGS

construction rule	least absolute deviation
estimation method	10-fold cross-validation
tree selection	1.0 se rule
variables used	See variable importance list above. response is variable RATE
linear combinations	no

SECONDARY OPTION SETTINGS

1	Minimum node size	= 1
2	Minimum size below which node will not be split	= 5
3	Number of surrogate splits printed	= 2
4	Number of competing splits printed	= 2
5	Maximum number of trees for which errors are printed	= 100
6	Initial value of the complexity parameter	= 0.0000E+00
7	Maximum number of cases to put into learning sample	= 20000
8	Maximum number of cases to put into test sample	= 20000
9	Maximum node size without sub-sampling the node	= 86
10	Maximum number of surrogates used	= 2
11	Maximum number of nodes allowed for in large tree	= 750
	(Actual maximum number of nodes	= 32)
12	Max. categorical primary + surrogate splits in a tree	= 1000
13	Max. linear combination splits in a tree	= 0
	(Actual number cat. + linear combination splits	= 32)
	(Actual number categorical competitor splits	= 29)
14	Maximum height of tree	= 750
	(Actual maximum height of tree	= 12)
	Maximum size of memory available	= 70000

FILE: APPENDIX H A1

(Actual size of memory used in run = 43180)

OUTPUT OF THE PROGRAM : CART-CASE

PROGRAM PURPOSE : VALIDATION OF RESULTS

INPUT FILE :
NAME : TEST8
CONTENTS : GENERAL MOS CAT. 8 DATA

CASE	NODE	PREDICTED	RATE	MOS	GRADE	LOS
1.	2	0.2667	0.6667	78.	1.	21.0
2.	2	0.2667	0.5000	78.	2.	21.0
3.	2	0.2667	0.6667	78.	2.	22.0
4.	1	0.4000	0.6667	78.	3.	24.0
5.	1	0.4000	0.5000	78.	3.	26.0
6.	3	0.6667	0.6667	78.	3.	30.0
7.	2	0.2667	0.4444	79.	1.	10.0
8.	2	0.2667	1.0000	79.	1.	20.0
9.	2	0.2667	1.0000	79.	2.	20.0
10.	2	0.2667	0.5000	79.	2.	23.0
11.	3	0.6667	0.2857	79.	3.	28.0
12.	3	0.6667	0.8000	79.	3.	30.0
13.	2	0.2667	0.1429	79.	6.	20.0
14.	2	0.2667	0.5000	79.	6.	23.0
15.	2	0.2667	1.0000	79.	6.	27.0
16.	1	0.4000	1.0000	79.	7.	20.0
17.	1	0.4000	1.0000	79.	7.	23.0
18.	1	0.4000	0.6667	79.	7.	25.0
19.	3	0.6667	1.0000	79.	7.	29.0
20.	2	0.2667	0.6667	80.	1.	25.0
21.	1	0.4000	0.5000	80.	3.	24.0
22.	1	0.4000	0.6667	80.	3.	26.0
23.	2	0.2667	1.0000	80.	5.	4.0
24.	2	0.2667	1.0000	80.	5.	21.0
25.	2	0.2667	0.5000	80.	6.	20.0
26.	2	0.2667	0.6667	80.	6.	27.0
27.	1	0.4000	0.5000	80.	7.	20.0
28.	1	0.4000	0.5000	80.	7.	23.0
29.	2	0.2667	0.2500	81.	1.	13.0
30.	2	0.2667	0.1250	81.	1.	14.0
31.	2	0.2667	1.0000	81.	1.	22.0
32.	1	0.4000	0.6667	81.	3.	23.0
33.	1	0.4000	0.5000	81.	3.	26.0
34.	3	0.6667	0.3333	81.	3.	29.0
35.	3	0.6667	0.5000	81.	3.	30.0
36.	2	0.2667	1.0000	81.	5.	20.0
37.	2	0.2667	0.2857	81.	6.	21.0

38.	1	0.4000	0.3333	81.	7.	26.0
39.	3	0.6667	1.0000	81.	7.	29.0
40.	2	0.2667	1.0000	81.	8.	20.0
41.	3	0.6667	1.0000	81.	8.	30.0
42.	2	0.2667	0.5000	82.	1.	20.0
43.	2	0.2667	1.0000	82.	1.	22.0
44.	1	0.4000	0.8000	82.	3.	20.0
45.	1	0.4000	0.4000	82.	3.	22.0
46.	1	0.4000	1.0000	82.	3.	25.0

OUTPUT OF THE PROGRAM : CART-TEST

PROGRAM PURPOSE : PROVIDES A SUMMARY OF RESULTS OBTAINED FROM BUILD AND CASE PROGRAMS.

1 WELCOME TO CART (TM) Version 1.0 November 20, 1984

Summary statistics for a test sample with 46 cases.

Relative error based on tree = 0.341 +/- 0.688
Initial mean absolute deviation = 0.228
Initial median = 0.667

1 3 TERMINAL NODES

	LEARNING SAMPLE			TEST SAMPLE			
NODE	CASES	MEDIAN	MEAN AD	CASES	MEDIAN	MEAN AD	MEAN ERROR
1	26	0.400	0.13	15	0.667	0.17	0.26
2	39	0.267	0.16	23	0.667	0.26	0.43
3	20	0.667	0.21	8	0.800	0.25	0.25

APPENDIX I

OUTPUTS OF THE CART PROGRAMS USING
9TH GENERAL CATEGORIZATION DATA AS INPUT

```
OUTPUT OF THE PROGRAM : CART-BUILD

PROGRAM PURPOSE : BUILDING A REGRESSION TREE

INPUT FILE :
             NAME     : BUILD9
             CONTENTS : GENERAL MOS CAT. 9 DATA
```

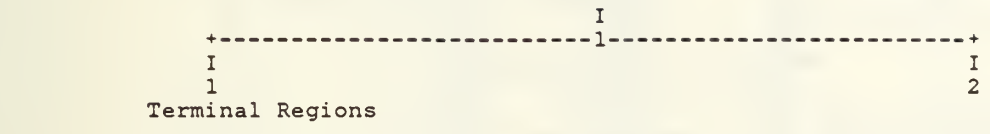
TREE SEQUENCE

Tree	Terminal Nodes	Cross-Validated Relative Error	Resubstitution Relative Error	Complexity Parameter
1	26	1.11 +/- 0.000	0.31	0.000
2	25	1.10 +/- 0.000	0.31	0.011
3	24	1.09 +/- 0.000	0.31	0.063
4	23	1.09 +/- 0.000	0.32	0.068
5	22	1.08 +/- 0.000	0.32	0.083
6	21	1.09 +/- 0.000	0.33	0.091
7	20	1.11 +/- 0.000	0.34	0.100
8	19	1.06 +/- 0.000	0.35	0.200
9	18	1.06 +/- 0.000	0.37	0.231
10	15	1.03 +/- 0.000	0.42	0.237
11	12	1.04 +/- 0.000	0.47	0.281
12	11	1.09 +/- 0.000	0.50	0.333
13	10	1.03 +/- 0.000	0.52	0.345
14	7	1.02 +/- 0.000	0.59	0.367
15	6	0.94 +/- 0.000	0.62	0.405
16	5	0.90 +/- 0.000	0.67	0.678
17	4	0.92 +/- 0.000	0.72	0.712
18	3	0.93 +/- 0.000	0.78	0.929
19*	2	0.98 +/- 0.000	0.87	1.27
20	1	1.00 +/- 0.000	1.00	1.97

Initial median = 0.400
Initial mean absolute deviation = 0.207

CLASSIFICATION/REGRESSION TREE

FILE: APPENDIX I A1



Terminal Regions

1 PARTITIONING TREE

Node 1 was split on variable GRADE
A CASE GOES LEFT IF VARIABLE GRADE IS IN (0, 1, 4)
Improvement = 2.7E-02 (C. T. = 1.9E+00)

Node	Cases	Median	Mean Abs Dev.
1	71	0.40	0.21
-1	7	0.00E+00	0.60E-01
-2	64	0.40	0.19

	Surrogate	Split	Assoc.	Improve.
1	LOS s	1.25E+01	0.71	1.7E-02
2	MOS r	83, 84, 85, 86, 89	0.28	1.1E-02

	Competitor	Split	Improve.
1	LOS	1.50E+00	2.2E-02
2	MOS	87, 88	1.1E-02

1 2 TERMINAL NODES

Node	Cases	Median	Mean Ad
1	7	0.000E+00	0.60E-01
2	64	0.400	0.19

	Relative Importance	Number Of Categories	Minimum Category
GRADE	100.	10	0
LOS	92.	numerical	
MOS	34.	7	83

Number of cases in the learning sample = 71

PRIMARY OPTION SETTINGS

construction rule	least absolute deviation
estimation method	5-fold cross-validation

tree selection	1.0 se rule
variables used	See variable importance list above.
	response is variable RATE
linear combinations	no

SECONDARY OPTION SETTINGS

1	Minimum node size	=	1
2	Minimum size below which node will not be split	=	5
3	Number of surrogate splits printed	=	2
4	Number of competing splits printed	=	2
5	Maximum number of trees for which errors are printed	=	100
6	Initial value of the complexity parameter	=	0.0000E+00
7	Maximum number of cases to put into learning sample	=	20000
8	Maximum number of cases to put into test sample	=	20000
9	Maximum node size without sub-sampling the node	=	72
10	Maximum number of surrogates used	=	2
11	Maximum number of nodes allowed for in large tree	=	750
	(Actual maximum number of nodes	=	27)
12	Max. categorical primary + surrogate splits in a tree	=	1000
13	Max. linear combination splits in a tree	=	0
	(Actual number cat. + linear combination splits	=	27)
	(Actual number categorical competitor splits	=	27)
14	Maximum height of tree	=	750
	(Actual maximum height of tree	=	9)
	Maximum size of memory available	=	70000
	(Actual size of memory used in run	=	42937)

OUTPUT OF THE PROGRAM : CART-CASE

PROGRAM PURPOSE : VALIDATION OF RESULTS

INPUT FILE :
NAME : TEST9
CONTENTS : GENERAL MOS CAT. 9 DATA

CASE	NODE	PREDICTED	RATE	MOS	GRADE	LOS
1.	1	0.0000	0.3333	84.	1.	15.0
2.	2	0.4000	0.2857	84.	3.	27.0
3.	2	0.4000	0.6667	84.	6.	20.0
4.	2	0.4000	1.0000	84.	6.	26.0
5.	2	0.4000	0.5000	84.	6.	27.0
6.	2	0.4000	1.0000	84.	7.	22.0
7.	2	0.4000	1.0000	84.	8.	29.0
8.	1	0.0000	0.5000	85.	1.	13.0
9.	1	0.0000	0.6667	85.	1.	20.0
10.	1	0.0000	0.1538	86.	1.	14.0
11.	2	0.4000	0.6667	86.	5.	10.0

FILE: APPENDIX I A1

12.	2	0.4000	0.6667	86.	6.	20.0
13.	2	0.4000	0.2353	86.	6.	21.0
14.	2	0.4000	0.2222	86.	6.	22.0
15.	2	0.4000	1.0000	86.	6.	29.0
16.	2	0.4000	0.6667	86.	7.	20.0
17.	2	0.4000	0.3333	86.	7.	21.0
18.	2	0.4000	0.4000	86.	7.	23.0
19.	2	0.4000	1.0000	86.	7.	28.0
20.	1	0.0000	1.0000	88.	0.	20.0
21.	2	0.4000	0.2857	88.	6.	18.0
22.	2	0.4000	0.5714	88.	6.	20.0
23.	2	0.4000	0.5000	88.	6.	21.0
24.	2	0.4000	0.6667	88.	6.	26.0
25.	2	0.4000	1.0000	89.	3.	30.0
26.	2	0.4000	0.6667	89.	6.	22.0
27.	2	0.4000	1.0000	89.	6.	23.0

OUTPUT OF THE PROGRAM : CART-TEST

PROGRAM PURPOSE : PROVIDES A SUMMARY OF RESULTS OBTAINED FROM BUILD AND CASE PROGRAMS.

1 WELCOME TO CART (TM) Version 1.0 November 20, 1984

Summary statistics for a test sample with 27 cases.

Relative error based on tree = 0.350 +/- 0.882
Initial mean absolute deviation = 0.235
Initial median = 0.667

1 2 TERMINAL NODES

------ LEARNING SAMPLE ------				-------- TEST SAMPLE ----------			
NODE	CASES	MEDIAN	MEAN AD	CASES	MEDIAN	MEAN AD	MEAN ERROR
1	7	0.000E+00	0.60E-01	5	0.500	0.24	0.53
2	64	0.400	0.19	22	0.667	0.23	0.31

APPENDIX J

CONSIDERED AND NEGLECTED RECORDS IN DATA FILES

NINE FILES WERE CREATED TO BE USED AS A LEARNING SAMPLES,EACH FILE IS CORRESPONDING TO ONE OF THE GENERAL MOS CATEGORIZATION GROUPS.SOME OF DATA AVAILABLE WERE NEGLECTED BECAUSE IT HAD ZERO LOSS RATES. THE FOLLOWING ARE THE NUMBER OF CASES CONSIDERED AND THOSE NEGLECTED IN EACH FILE.

GENERAL MOS CATEGORIZATION 1

MOS	GRADE	CONSIDERED	NEGLECTED
3	1	0	3
3	2	2	6
3	3	1	4
3	4	8	7
3	5	13	5
3	6	23	4
3	7	16	6
3	8	12	4
3	9	0	10
5	0	0	14
5	1	0	17
5	2	1	7
5	3	2	10
5	4	4	9
5	5	12	3
5	6	20	5
5	7	15	6
5	8	12	4
5	9	1	2
10	0	0	2
10	1	0	3
10	2	1	5
10	3	2	7
10	4	3	9
10	5	10	5
10	6	16	8
10	7	10	11
10	8	11	4
10	9	0	2
TOTAL		195	182

GENERAL MOS CATEGORIZATION 2

MOS	GRADE	CONSIDERED	NEGLECTED
7	0	2	12

7	1	3	14
7	2	3	12
7	3	5	7
7	4	3	16
7	5	13	12
7	6	19	8
7	7	12	9
7	8	9	7
7	9	0	3
13	0	0	6
13	1	0	8
13	2	1	6
13	3	2	11
13	4	4	10
13	5	12	5
13	6	21	3
13	7	14	7
13	8	10	6
13	9	0	2
20	0	1	18
20	1	8	12
20	2	4	14
20	3	4	7
20	4	2	12
20	5	9	20
20	6	16	11
20	7	10	11
20	8	4	9
TOTAL		191	278

GENERAL MOS CATEGORIZATION 3

MOS	GRADE	CONSIDERED	NEGLECTED
38	0	0	2
38	1	0	4
38	2	0	3
38	3	8	10
38	4	9	3
38	5	11	2
38	6	17	3
38	7	20	2
38	8	13	4
38	9	0	6
TOTAL		78	39

GENERAL MOS CATEGORIZATION 4

MOS	GRADE	CONSIDERED	NEGLECTED
1	0	3	16
1	1	6	15
1	2	7	11

FILE: APPENDIX J A1

1	3	9	5
1	4	2	20
1	5	12	14
1	6	20	7
1	7	14	8
1	8	9	8
1	9	0	1
2	0	1	15
2	1	1	16
2	2	5	11
2	3	10	3
2	4	1	20
2	5	12	12
2	6	19	8
2	7	12	9
2	8	7	8
16	0	2	15
16	1	4	14
16	2	4	12
16	3	10	5
16	4	4	19
16	5	14	12
16	6	23	4
16	7	12	9
16	8	10	6
16	9	0	4
19	0	1	17
19	1	3	14
19	2	2	12
19	3	7	6
19	4	1	15
19	5	7	16
19	6	20	7
19	7	11	9
19	8	5	11
23	0	0	9
23	1	0	10
23	2	1	3
23	3	2	9
23	4	1	9
23	5	5	13
23	6	7	20
23	7	5	16
23	8	3	15
24	0	1	10
24	1	1	10
24	2	0	4
24	3	0	5
24	4	7	3
24	5	7	16
24	6	17	11
24	7	13	11
24	8	6	12
24	9	0	3
28	0	0	18

28	1	0	16
28	2	2	8
28	3	4	8
28	4	1	18
28	5	11	14
28	6	13	14
28	7	4	15
28	8	2	14
TOTAL		403	728

GENERAL MOS CATEGORIZATION 5

MOS	GRADE	CONSIDERED	NEGLECTED
30	0	1	16
30	1	5	13
30	2	5	8
30	3	6	6
30	4	3	21
30	5	7	17
30	6	10	17
30	7	9	9
30	8	3	5
36	0	0	13
36	1	1	14
36	2	2	5
36	3	5	7
36	4	2	11
36	5	11	3
36	6	14	7
36	7	10	11
36	8	9	7
36	9	0	1
37	0	2	12
37	1	2	12
37	2	1	8
37	3	4	5
37	4	1	14
37	5	6	18
37	6	12	11
37	7	5	13
37	8	0	9
TOTAL		136	293

GENERAL MOS CATEGORIZATION 6

MOS	GRADE	CONSIDERED	NEGLECTED
4	0	0	14
4	1	0	13
4	3	1	0
4	4	2	11
4	5	7	16

4	6	10	16
4	7	7	14
4	8	4	10
4	9	0	1
14	0	1	13
14	1	1	14
14	2	1	3
14	3	3	9
14	4	0	17
14	5	8	15
14	6	12	14
14	7	6	15
14	8	0	11
14	9	0	1
21	0	0	13
21	1	3	11
21	2	1	6
21	3	1	6
21	4	2	17
21	5	12	10
21	6	16	10
21	7	9	12
21	8	8	6
TOTAL		115	298

GENERAL MOS CATEGORIZATION 7

MOS	GRADE	CONSIDERED	NEGLECTED
9	0	0	7
9	1	0	8
9	2	1	6
9	3	0	9
17	0	0	11
17	1	0	10
17	2	1	3
17	3	2	0
17	4	0	1
17	5	2	12
17	6	5	18
17	7	5	7
17	8	0	2
18	0	0	13
18	1	2	11
18	2	2	7
18	3	0	5
18	5	2	14
18	6	3	19
18	7	5	12
18	8	1	2
22	0	0	11
22	1	1	7
22	2	0	3
22	3	3	2

22	4	0	5
22	5	3	13
22	6	5	11
22	7	2	6
22	8	0	10
25	0	0	13
25	1	0	11
25	2	1	4
25	3	0	4
25	4	0	4
25	5	4	15
25	6	4	15
25	7	1	3
25	8	1	3
26	0	0	10
26	1	0	7
26	2	2	6
26	3	0	1
26	4	0	3
26	5	1	15
26	6	2	13
26	7	0	8
26	8	1	3
39	0	0	10
39	1	0	9
39	2	1	10
39	3	3	10
39	4	11	5
39	5	2	7
39	6	0	1
39	7	1	0
39	9	8	4
TOTAL		88	439

GENERAL MOS CATEGORIZATION 8

MOS	GRADE	CONSIDERED	NEGLECTED
6	0	1	13
6	1	2	14
6	2	2	8
6	3	6	6
6	4	0	1
6	5	0	3
6	6	0	1
11	0	3	13
11	1	4	16
11	2	2	10
11	3	5	9
11	4	0	4
11	5	0	16
11	6	4	14
11	7	6	7
11	8	1	3

12	0	0	14
12	1	1	16
12	2	0	5
12	3	6	6
12	4	0	8
12	5	3	16
12	6	8	11
12	7	7	6
12	8	0	1
33	0	1	13
33	1	1	17
33	2	0	11
33	3	2	7
33	5	0	14
33	6	6	11
33	7	4	9
33	8	0	9
35	0	0	14
35	1	1	15
35	2	1	8
35	3	4	10
35	5	0	4
35	6	2	0
TOTAL		83	363

GENERAL MOS CATEGORIZATION 9

MOS	GRADE	CONSIDERED	NEGLECTED
8	0	0	7
8	1	0	7
8	2	1	4
8	3	0	2
15	0	1	13
15	1	1	11
15	2	2	8
15	3	7	6
15	4	0	3
15	5	1	14
15	6	7	10
15	7	9	4
15	8	2	4
27	0	0	12
27	1	0	7
29	0	0	15
29	1	1	14
29	2	1	9
29	3	8	5
29	4	0	3
29	5	2	16
29	6	8	10
29	7	7	6
29	8	2	2
32	0	0	9

FILE: APPENDIX J A1

32	1	0	11
32	2	0	7
32	3	0	6
32	5	0	12
32	6	0	12
32	7	0	3
32	8	0	2
34	0	0	6
34	1	0	4
34	3	2	9
34	4	0	3
34	5	1	10
34	6	3	6
34	7	0	3
TOTAL		66	295

LIST OF REFERENCES

1. Breiman, Leo, et. al, Classification and Regression Trees, Graybill, F., Wadsworth, Inc., 1984.

2. Tucker, D. D., Loss Rate Estimation in Marine Corps Officer Manpower Models, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1985.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Professor Robert R. Read, Code 55 Naval Postgraduate School Monterey, California 93943-5100	1
4.	Professor Glenn F. Lindsay, Code 55 Naval Postgraduate School Monterey, California 93943-5100	1
5.	Professor A. R. Washburn, Code 55 Naval Postgraduate School Monterey, California 93943-5100	1
6.	Amin Elseramegy Hamdy 14B, Mona St., Apt. 10 Eldoky, Giza, Egypt	2
7.	Major D. D. Tucker USMC Headquarters MP 520 Washington, DC 20380	1

216084

Thesis
A4343
c.1

Amin Elseramegy
CART program: the implementation of the CART program and its application to estimating attrition rates.