e-PG Pathshala

Subject : Computer Science

Paper: Machine Learning

Module: Classification and Regression Trees - I

Module No: CS/ML/15

Quadrant I – e-text

Welcome to the e-PG Pathshala Lecture Series on Machine Learning. In this module we will be discussing in detail the Classification and Regression Tree (CART) approach for the construction of Decision Trees.

Learning Objectives:

The learning objectives of this module are as follows:

- To explain CART approach to decision tree building
- To outline the key features of CART
- To explain the various steps of the CART approach
- To discuss the recursive partitioning approach
- To outline the Gini Index

15.1 Decision Tree & CART

Any decision tree will successively split the data into smaller and smaller subsets. Ideally all the samples associated with a leaf node should be from the same class. Such a subset, or node, is considered *pure* in this case. A generic tree-growing methodology, known as CART, successively splits nodes until they are pure.

.

15.2 The CART approach

As we have discussed Classification And Regression Trees is a generic methodology for the construction of effective decision trees. CART was developed by Breiman, Friedman, Olshen, Stone in early 80's. They are the pioneers in bringing the tree-based modeling paradigm into the statistical fold. They proposed a rigorous approach involving cross-validation to select the optimal tree. In this context Quinlan developed C4.5 a machine learning approach for generating the decision tree which was essentially an extension of based on his earlier ID3 algorithm.

This is a non-parametric technique, using the methodology of tree building. CART are binary tree structured classifiers constructed by repeated splits of subsets (nodes) of the search space S into two descendant subsets. Each terminal subset is assigned a class label; the resulting partition of S corresponds to the classifier. The method classifies objects or predicts outcomes by selecting from among the large number of variables, that is by selecting the most important variables that affect the outcome variable. CART analysis is a form of binary recursive partitioning. We will discuss this aspect later on.

15.2.1 An Example from Clinical Research

Let us consider a dataset from University of California for the development of a reliable clinical decision rule to classify new patients into categories *after* admission for a heart attack. The dataset has 19 variables collected during the first 24 hours for 215 patients (for those who survived the 24 hours) after their admittance to San Diego Hospital. The goal of the work is to identify the patients with high risk that is those patients who would not survive 30 days.

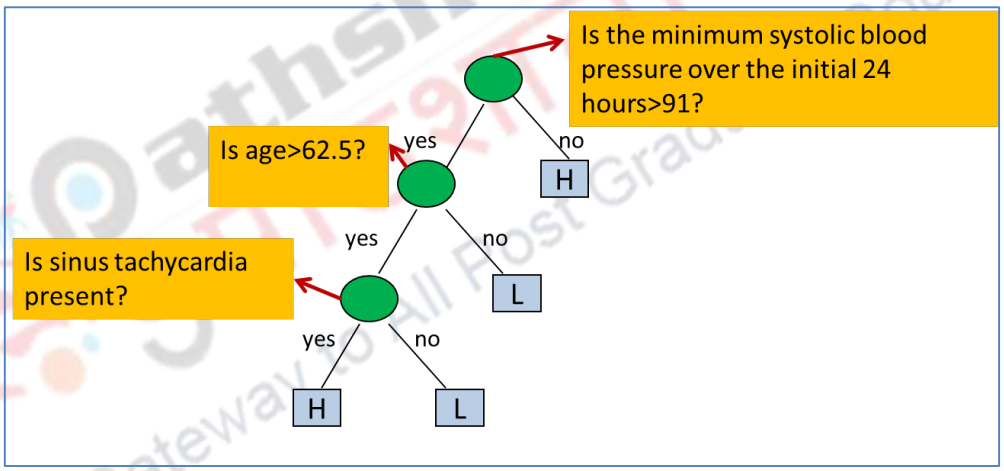

Figure 15.1 Classification of Patients as High or Low risk groups

Figure 15.1 shows a part of the decision tree where all patients are divided into two subsets (binary partitioning), one subset A of patients whose minimum systolic blood pressure over the initial 24 hour period > 91 and the other subset B of patients whose minimum systolic blood pressure over the initial 24 hour period ≤ 91. Now in the subset A, we further partition the set based on whether age > 62.5 and so on.

15.2.2 Definition of CART

Now let us define CART. CART builds classification or regression trees for categorical attributes (classification) or numeric attributes (regression). Tree models where the target variable can take a finite set of values are called classification trees. Regression tree analysis is where the target variable can take continuous values typically real numbers such as the price of a house,

or a patient's length of stay in a hospital. The term **C**lassification **A**nd **R**egression **T**ree (CART) analysis is used to refer to both types of attributes categorical and numeric. Trees used for classification and regression are similar but differ how they determine the splitting procedure.

15.2.3 Salient Aspects of CART

One of the key features of CART is automated attribute or feature selection. The process can automatically select relevant fields irrespective of the total number of fields. Moreover no data preprocessing or transformation of the variables is needed. The CART is is tolerant to missing values, and there is only a moderate loss of accuracy due to missing values.

15.3 CART- General Framework - The Six Questions

Associated with CART are six questions to be answered or six issues to be addressed. The following are the questions:

1 **How many splits will be there at each node?**
Should the questions to be asked for the split at each node be binary (e.g., is gender male or female) or numeric (e.g., height is 5'4") or multi-valued (e.g., caste)?

2 **Which properties should be tested at each node?**
The decision on which variable or feature to use for splitting is an important issue and in fact determines the efficiency of the procedure.

3 **When does a node become a leaf?**
We need a criteria to determine that the subset of examples associated with a node need not be partitioned further.

4 **How to prune a large tree?**

Simplify the tree by pruning peripheral branches to avoid overfitting. Some parts of the tree need not be explored or partitioned further since deep trees that fit the training data well, will not generalize well to new test data**.** How do we determine this?

5 **If a leaf node is impure, how to assign labels?**
In case the leaf node is impure that is results in more than one output variable value, how do we assign an output value for unknown samples.

6 **How to handle missing data?**
The issue here is how to account for missing data of some features when decisions regarding this data are taken.

15.4 CART Steps

The following are the steps used by CART for the construction of the decision tree.

1 **Initialization**: Initially a tree is created with a single root node containing all the training data.

2 **Tree growing:** Recursive Partitioning is the most important component of tree construction. This procedure consists of 5 sub steps which are described below:

   a. The first step here is the selection of the splitting criterion which in most cases is based on likelihood. For this purpose we need to find the best question for splitting each node. We then split the node that results in the greatest increase in the likelihood.
   b. We then rank all of the best splits and then the best split in the variable is selected in terms of the reduction in impurity (heterogeneity). This aspect we will discuss later.
   c. The Predicted classes are assigned to the nodes according to a rule that minimizes misclassification costs.
   d. The next step is the decision about the stopping criteria that is how far to grow? We generally stop expanding a node when all the records belong to the same class or when all the records have similar attribute values. Stopping criteria decides whether to continue splitting a node.
   e. Using surrogates program best available information is used to replace missing data, normally based on a variable that is relative to the outcome variable.

3 **Stop tree building**: When every aspect of the dataset has been taken into account while building the decision tree, the tree building process itself is stopped.

4 **Tree Pruning:** An independent test set or cross-validation is generally used to prune the tree. Pruning is carried out by trimming off parts of the tree that don't work. Another method is to first order the nodes of a large tree by contribution to tree accuracy and this ordering is used as a basis to prune nodes.

5 **Optimal Tree Selection:** This process is the selection of the best tree that fits dataset with a low percentage of error. The decision about the best tree is made after growing and pruning. This selection of optimal tree is also based on balancing simplicity against accuracy.

15.5 Recursive Partitioning

The key idea behind the CART technique is recursive partitioning. Let us understand this concept in detail. The recursive procedure is described as follows:

i. Consider all the data samples.

ii. Consider *all* possible values of *all* variables.

iii. Select the variable and the value ($X=t_1$) that produces the greatest "separation" in the target. This point of the variable X ($X=t_1$) is called a "split".

iv. If $X< t_1$ then send the data to the "left"; otherwise, send data point to the "right".

v. Now repeat same process on these two "nodes"

What you get as output is a "tree". Note that CART only uses *binary* splits.

The main step in the process is Recursive Partitioning (step iii) where we

- First we pick a value of x_i, say s_i, that divides the training data into two (not necessarily equal) portions
- Then we measure how "pure" or homogeneous each of the resulting portions are. In this context a partition being "Pure" means that they contain records of mostly one class.
- The algorithm tries different values of x_i, and s_i to maximize purity in initial split
- After we get a "maximum purity" split, we repeat the process for a second split, and so on

15.6 Features of CART

Some of the important features of the process of CART are as follows:

- Data is split into two partitions that is at every node the set is split into two subsets.
- Splits at any node are based only on one variable.
- Partitions can also be split into sub-partitions and hence procedure is recursive. CART tree is generated by repeated partitioning of data set where each decision node has two child nodes and so on.

15.7 Construction of a Tree

During the construction of the tree we need to first carry out the most important step that is the selection of the splits. We further need to make decisions as to whether a node is a terminal node (i.e. not to split it any further). We also need to assign a class to each terminal node.

The following are the steps of tree building

1. Start with splitting a variable at all of its split points. Pick one of the predictor variables X_i, and pick a value of the variable say S_i that divides the training data into two (not necessarily equal) portions. Measure how "pure" or homogeneous each of the resulting portions are. As we discussed a pure partition contains records of mostly one class. Idea is to select different values S_i of the variable X_i to form the partitions. These different values of the variables used for forming the partitions are called split points.
2. The sample set splits into two binary nodes at each split point.
3. Select the best split in the variable in terms of the reduction in impurity (heterogeneity)
4. Repeat steps 1,2 for all variables at root node.
5. Rank all of best splits & select the variable that achieves the highest purity at root.

15.7.1 Method of Splitting

Car type	Age	Chd	Sub
sedan	23	0	yes
sports	31	1	no
sedan	36	1	yes
truck	25	2	no
sports	30	0	no
sedan	36	0	no
sedan	25	0	yes
truck	36	1	no
sedan	30	2	yes
sedan	31	1	yes
sports	25	0	no
sedan	45	1	yes
sports	23	2	no
truck	45	0	yes

Figure 15.2 Insurance Example

We first order records according to one variable. Then we need to find midpoints between successive values. We then divide records. After evaluating that split, try the next midpoint. In the case of categorical variables, examine all possible ways in which the categories can be split. For example assume that the categories A, B, C can be split 3 ways as {A} and {B, C}, {B} and {A, C}, {C} and {A, B}. Please note that with many categories, the number of splits becomes huge. Many of the tools available support only binary categorical variables.

Now let us consider the example given in Figure 15.2. The decision variables in this example are car type, age and number of children. The output variable in this example is whether there is insurance claim or not. The decision tree is shown in Figure 15.3. In this example we start with age as the root node of the decision tree. The numerical value of age has been split as ≤ 30 and > 30. This divides the total number of 14 samples into two sets of 7 samples each. Now at the left we split the 7 examples into two sets of sedan on one side (with 3 as sedan) and 4 either sports or truck on the right. This leaves us with claim as all yes on left and all no on right. For the right side of the root we first use the number of children attribute for splitting. Here the split is taken as no children (2 samples) or having children (5 samples). At the next level we split on both sides based on car type. Now at the leaves we are left with no on one side and yes on the other. This decision helps to decide whether claim is possible or not.

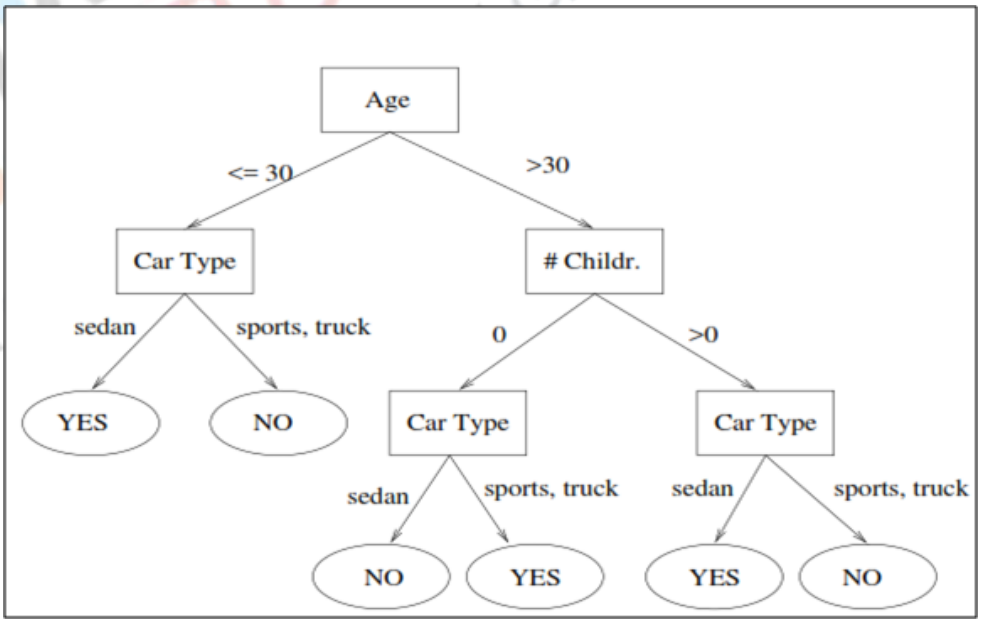

Figure 15.3 Decision Tree for Insurance Example

In the Insurance Example

- Suppose we have 3 variables
  Type of Car :{sedan, sports, truck},
  Age category:{ ≤30 &> 30},
  Number of Children :{0,1,2}

At each iteration, CART tests all splits.
Some of the splits for the variable

Type of car having three values are - Car = {sedan} & {truck & sports} or { sedan & truck} & {sports} or {sedan & sports} & {truck} .

For the Age variable as we can see the split point can be ≤30 &> 30, ≤35 &> 35 etc.

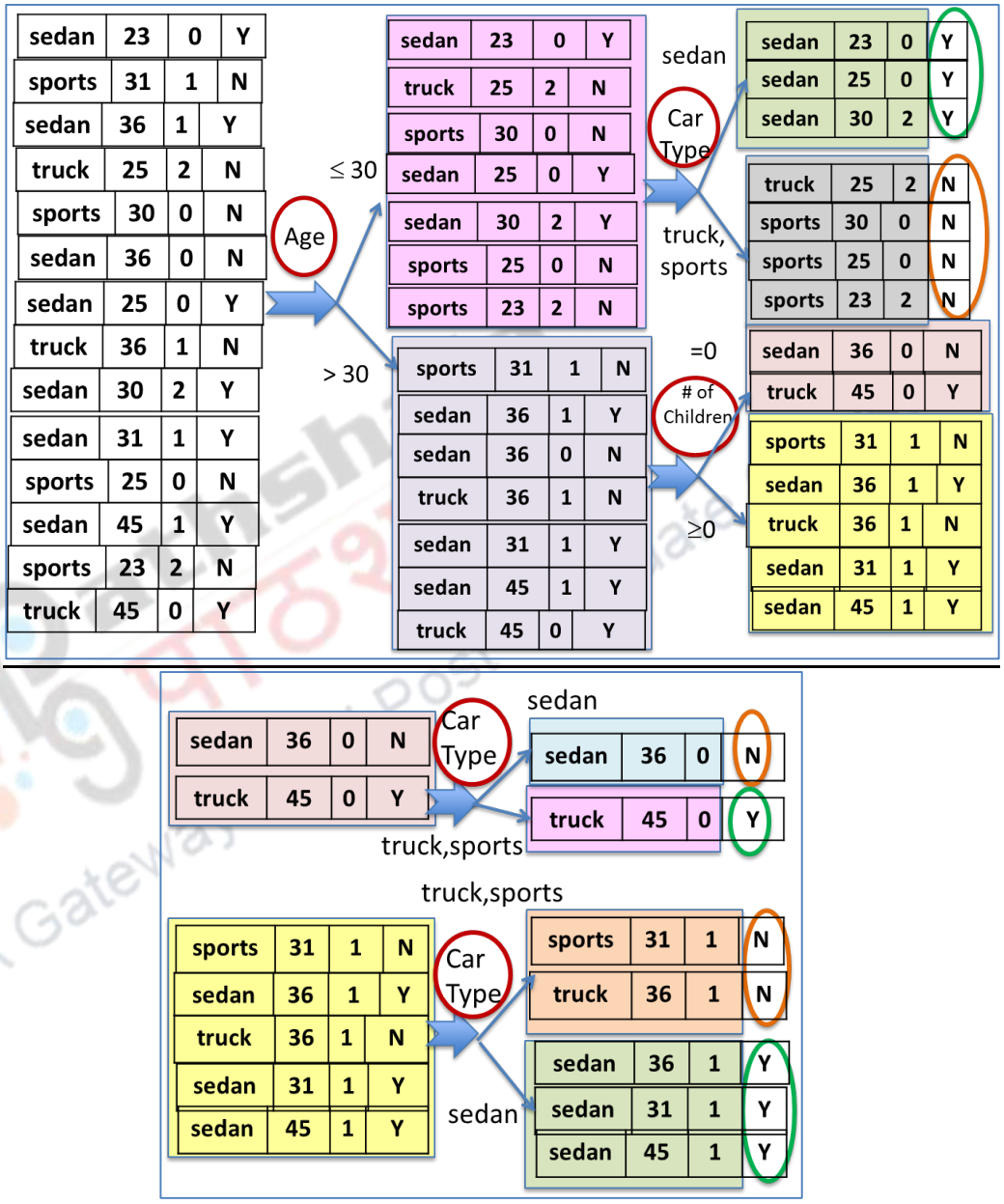

Figure 15.4 The Split of Samples based on Decision Tree in Figure 15.3

15.8 Selection of Split

The split has to be so selected that it results in greater increase in *purity*. In case of the example given in the previous section:

- Perfect purity: each split has either all claims or all no-claims.
- Perfect impurity: each split has same proportion of claims as overall population.

15.8.1 Splitting Rules

As discussed before we need to select the variable value ($X=t_1$) that produces the greatest "separation" in the target variable. "Separation" can be defined in many ways. In the case of **Regression Trees** (continuous target), we can use sum of squared errors. In the case of **Classification Trees** (categorical target) we can use either *entropy*, *Gini measure*, or a "*twoing*" splitting rule. In the tree-based modeling for *discrete* target variable, various measures of *purity* are used. The intuition is that an ideal model would produce nodes that contain either calims only or no-claims (example in Figure 15.2) only that is completely pure nodes.

The main issue in the choice of a measure is the balance between getting pure nodes and getting equal size nodes (that is partitioning into roughly equal sized sets).

The Gini purity of a node and Entropy of a node are discussed in the next section. The maximum entropy/Gini index occurs when records are equally distributed among all classes implying least information and minimum entropy/Gini index occurs when when all records belong to one class, implying most interesting information.

Gini might produce small but pure nodes. The "twoing" rule strikes a balance between purity and creating roughly equal-sized nodes so that we do not always choose a split that gives pure nodes but rather choose a split that gives an almost pure nodes that gives roughly equal size partitions.

15.8.2 Impurity and Recursive Partitioning

The idea is to select the split that decreases the Gini Index. We have to consider all possible places for a split. This essentially means that we have to select the variable and the possible splits in values of the variables appropriately. The chosen split points become nodes on the tree. We keep splitting until the terminal nodes have very few cases or are all pure. The best approach is to grow a larger tree than required and then to *prune* it. We will discuss pruning in the next module.

Now we need a measure of impurity of a node to help decide on how to split a node, or which node to split. The measure should be at a maximum when a

node is equally divided amongst all classes. The impurity should be zero if the node is all one class

There are various measures of impurity. Misclassification rate is one such measure, but this measure is not generally used since no split improves the misclassification rate, and can be equal when one option is clearly better for the next step. Other measures are Information, or Entropy and Gini Index.

15.8.1 Information or Entropy

If a node has a proportion of p_j of each of the classes then the information or entropy is:

$$i(p) = -\sum_j p_j \log p_j$$

where $p=(p_1,p_2,\ldots. p_n)$ is the relative frequency of class *k at the node.*

Entropy ranges between 0 (most pure) and $\log_2(m)$ (equal representation of classes) where m is total number of classes.

15.8.2 Gini Index

This the most widely used measure of impurity by CART

If a data set contains examples from n classes, gini index, is defined as

$$i(p) = \sum_{i \neq j} p_i p_j = 1 - \sum_j p_j^2$$

where p is the proportion of cases that belong to class *k*

i(p) = 0 when all cases belong to same class and has maximum value when all classes are equally represented (= 0.50 in binary case)

15.8.3 Tree Impurity

The impurity of a tree is the sum over all terminal nodes of the impurity of a node multiplied by the proportion of cases that reach that node of the tree. Impurity of a tree with one single node, with both A and B having 400 cases (Figure 15.5), using the Gini Index is as follows:

Proportions of the two cases
= number of cases of one attribute/total number of cases = 400/800 (for both attribute A and B)

Gini Index
= **$1- p^2_A- p^2_B$**
$=1-(0.5)^2- (0.5)^2 = 0.5$

Numbers of Cases		Proportion of Cases				Gini Index
A	B	A	B			
		p_A	p_B	p^2_A	p^2_B	$1- p^2_A- p^2_B$
400	400	0.5	0.5	0.25	0.25	0.5

Figure 15.5 Tree Impurity Calculations – Example 1

Number of Cases		Proportion of Cases				Gini Index	Contrib. To Tree
A	B	A	B				
		p_A	p_B	p^2_A	p^2_B	$1- p^2_A - p^2_B$	
300	100	0.75	0.25	0.5625	0.0625	0.375	0.1875
100	300	0.25	0.75	0.0625	0.5625	0.375	0.1875
						Total	0.375
200	400	0.33	0.67	0.1111	0.4444	0.4444	0.3333
200	0	1	0	1	0	0	0
						Total	0.3333

Figure 15.6 Tree Impurity Calculations – Example 2

Figure 15.6 shows another example where we show the nodes of the tree and its contribution to impurity is calculated based on Gini Index. Let us consider the calculation of Gini Index of row 1.

Proportions of the two cases
= number of cases of one attribute/total number of cases
= 300/400 = 0.75 (for both attribute A) and 100/400 =0.25 (for both attribute B)
Gini Index
= **$1- p^2_A- p^2_B$**
$=1-(0.75)^2- (0.25)^2 = 0.375$

15.9 Tree Structure

The decision tree is represented by using the split points to decide which are the nodes of the tree (circles with split value in center). Rectangles represent “leaves” (terminal points, no further splits, classification value noted). Numbers on lines (branches) between nodes indicate the number of cases. We read down the tree from the root to derive the rule.

15.9.1 Determining Leaf Node Label

Another important aspect in the construction of the tree is to assign a label to each leaf node. Each leaf node label is determined by “voting” of the records within it, and by the cutoff value. Records within each leaf node are from the training data. The leaf node’s label will be assigned the majority class when the default cutoff is 0.5. When the cutoff = 0.75, the requirement is that a majority of 75% or more “1” records in the leaf is needed to label it a “1” node.

Summary

- Explained the CART approach to decision tree building
- Outlined the key features of CART
- Explained the various steps of the CART approach
- Discussed the recursive partitioning approach
- Outlined the Gini Index