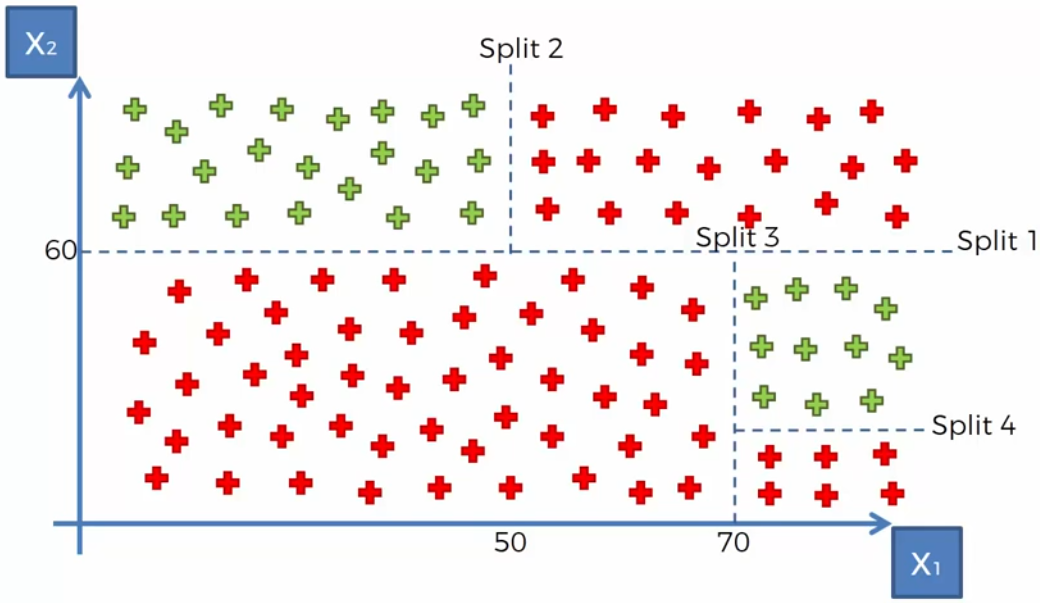
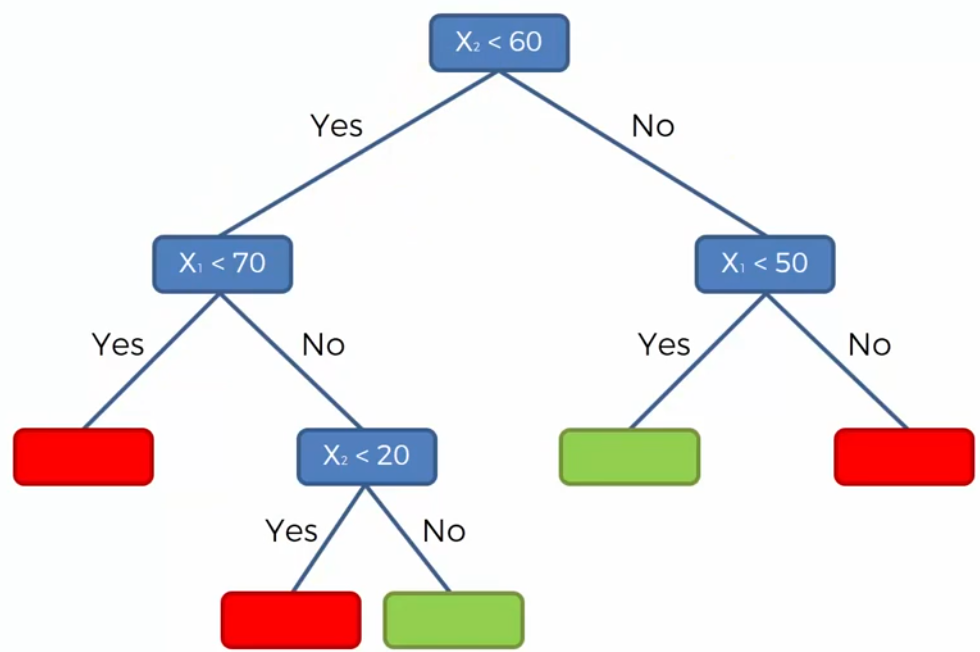
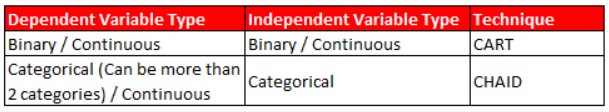
# INTRODUCTION

A decision tree is a graphical representation of possible solutions to a decision based on certain conditions. It's called a decision tree because it starts with a single box (or root), which then branches off into a number of solutions, just like a tree. A decision tree is drawn upside down with its root at the top.







# ALGORITHM TYPES

When it comes to decision trees, there are three major algorithms used in practice. CART ("Classification and Regression Trees"), C4.5, and CHAID.

CART stands for classification and regression trees where as CHAID represents Chi-Square automatic interaction detector. Both algorithms, create tree like structures to model data, however they differ in their attempt to stop tree growth. CART overgrows a tree and then prunes unnecessary branches.

All three algorithms create classification rules by constructing a tree-like structure of the data. However, they are different in a few important ways.

## CART

CART stands for Classification and Regression Trees.

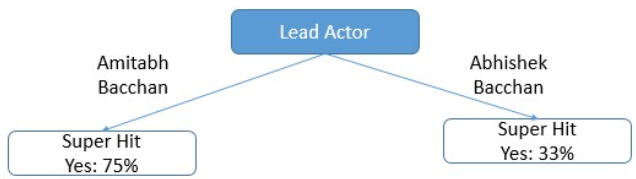
### REGRESSION TREE

The outcome (dependent) variable is a continuous variable and predictor (independent) variables can be continuous or categorical variables (binary).

### CLASSIFICATION TREE

The outcome (dependent) variable is a categorical variable (binary) and predictor (independent) variables can be continuous or categorical variables (binary). It creates binary split.

Note: If the dependent variable has more than 2 categories, then C4.5 algorithm or CHAID tree algorithm should be used.

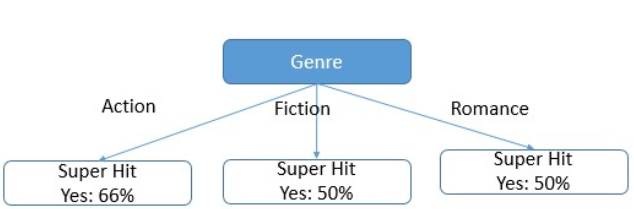


## CHAID

CHAID stands for Chi-square Automated Interaction Detection. The outcome (dependent) variable can be continuous and categorical. But, predictor (independent) variables are categorical variables only (can be more than 2 categories). It can create multiple splits (more than 2).

### CLASSIFICATION TREES

When independent variables are continuous, they need to be transformed into categorical variables (bins/groups) before using CHAID.



## CARRT VS CHAID

CART is a supervised model on the other hand, CHAID is an unsupervised technique, and it uses the entire model to build the tree.

A key difference between the two models, is that CART produces binary splits, one out of two possible outcomes, whereas CHAID can produce multiple branches of a single root/parent node.

CHAID is most frequently used for descriptive analysis whereas CART is frequently used in predictive analysis.

Example: If you want to decide on which customers you should target based on a campaign, and what is your likelihood of conversion.

## USE CASE

CHAID should be used when the goal is to describe or understand the relationship between a response variable and a set of explanatory variables, whereas CART is better suited for creating a model that has high prediction accuracy of new cases.

For example, we can use CHAID model to predict which customers would respond best to a given campaign.

CART on the other hand can be useful in analyzing and understanding customer behaviour. For example, 80% of my customers who landed on my web site through an ad campaign, was the products page. You could use this to redirect future customers coming through an ad, directly to the products page in hopes of increasing your conversions. This technique allowed us to identify patterns of previous customers.

CHAID is intended to work with categorical/discretized targets whereas CART can definitely do regression and classification.

CART will always yield binary trees, which sometimes cannot be summarized as efficiently for interpretation and/or presentation". In other words, if the goal is explanatory, CHAID is better suited for the task.

# COST FUNCTION FOR SPLIT

Let’s take a look at cost functions used for classification and regression. In both cases the cost functions try to find most homogeneous branches, or branches having groups with similar responses.

## FOR REGRESSION

The CART algorithm decides on a split based on the amount of homogeneity within class that is achieved by the split.

For CART regression tress we can use below metrics / cost functions for split criteria:

* Feature and value with minimum error (i.e. max gain).
* Feature with minimum std (homogenous group has 0 std).

**EXAMPLE:** Let’s say, we are predicting the price of houses. Now the decision tree will start splitting by considering each feature in training data. The mean of responses of the training data inputs of particular group is considered as prediction for that group. The above function is applied to all data points and cost is calculated for all candidate splits. Again, the split with lowest cost is chosen.

## FOR CLASSIFICATION

For CART classification tress we can use below cost functions to decide on split:

* The split with min GINI (max decrease in impurity).
* The split with min entropy (max decrease in uncertainty).
* The split with maximum information gain.

For CHAID classification tress we can use below cost functions to decide on split:

* If dependent variable is categorical, Chi-Square test determines the best next split at each step. Feature and category with least p values is selected (i.e. high association between independent and response).
* If dependent variable is continuous, F test determines the best next split at each step (equality of variance of two distributions).

## GINI VS ENTROPY

Gini calculation is faster than entropy which is why it’s the default measure in most of the algorithms. They are calculated using below formulas:



Gini Index measures impurity in node. It varies between 0 and (1-1/n) where n is the number of categories in a dependent variable.

Few important points:

* Gini is intended for continuous attributes and Entropy is for attributes that occur in classes
* Gini is to minimize misclassification
* Entropy is a little slower to compute

Reading: <https://stackoverflow.com/questions/1859554/what-is-entropy-and-information-gain>

# PRUNING

The performance of a tree can be further increased by pruning. It involves removing the branches that make use of features having low importance. This way, we reduce the complexity of tree, and thus increasing its predictive power by reducing overfitting.

Pruning can start at either root or the leaves. The simplest method of pruning starts at leaves and removes each node with most popular class in that leaf, this change is kept if it doesn't deteriorate accuracy. It’s also called reduced error pruning. More sophisticated pruning methods can be used such as cost complexity pruning where a learning parameter (alpha) is used to weigh whether nodes can be removed based on the size of the sub-tree. This is also known as weakest link pruning.

CHAID tries to prevent overfitting right from the start (only split is there is significant association), whereas CART may easily overfit unless the tree is pruned back. On the other hand, this allows CART to perform better than CHAID in and out-of-sample (for a given tuning parameter combination).

The tree pruning is done by examining the performance of the tree on a holdout dataset, and comparing it to the performance on the training set. CHAID uses a statistical rule to stop tree growth called the Chi-Square test.

Pruning helps us to avoid overfitting. Generally, it is preferred to have a simple model, it avoids overfitting issue. Any additional split that does not add significant value is not worthwhile. We can avoid overfitting by changing the parameters like

* **max\_leaf\_nodes:** Reduce the number of leaf nodes.
* **min\_samples\_leaf:** Restrict the size of sample leaf.

Minimum sample size in terminal nodes can be fixed to 30, 100, 300 or 5% of total.

* **max\_depth:** Reduce the depth of the tree to build a generalized tree.

Set the depth of the tree to 3, 5, 10 depending after verification on test data.

# TIPS ON PRACTICAL USE

* Decision trees tend to overfit on data with a large number of features. Getting the right ratio of samples to number of features is important, since a tree with few samples in high dimensional space is very likely to overfit.
* Consider performing dimensionality reduction (PCA, ICA, or Feature selection) beforehand to give your tree a better chance of finding features that are discriminative.
* Visualise your tree as you are training by using the export function. Use max\_depth=3 as an initial tree depth to get a feel for how the tree is fitting to your data, and then increase the depth.
* Remember that the number of samples required to populate the tree doubles for each additional level the tree grows to. Use max\_depth to control the size of the tree to prevent overfitting.
* Use min\_samples\_split or min\_samples\_leaf to ensure that multiple samples inform every decision in the tree, by controlling which splits will be considered. A very small number will usually mean the tree will overfit, whereas a large number will prevent the tree from learning the data. Try min\_samples\_leaf=5 as an initial value. If the sample size varies greatly, a float number can be used as percentage in these two parameters. While min\_samples\_split can create arbitrarily small leaves, min\_samples\_leaf guarantees that each leaf has a minimum size, avoiding low-variance, over-fit leaf nodes in regression problems. For classification with few classes, min\_samples\_leaf=1 is often the best choice.
* Balance your dataset before training to prevent the tree from being biased toward the classes that are dominant. Class balancing can be done by sampling an equal number of samples from each class, or preferably by normalizing the sum of the sample weights (sample\_weight) for each class to the same value. Also note that weight-based pre-pruning criteria, such as min\_weight\_fraction\_leaf, will then be less biased toward dominant classes than criteria that are not aware of the sample weights, like min\_samples\_leaf.
* If the samples are weighted, it will be easier to optimize the tree structure using weight-based pre-pruning criterion such as min\_weight\_fraction\_leaf, which ensure that leaf nodes contain at least a fraction of the overall sum of the sample weights.
* All decision trees use np. float32 arrays internally. If training data is not in this format, a copy of the dataset will be made.
* If the input matrix X is very sparse, it is recommended to convert to sparse csc\_matrix before calling fit and sparse csr\_matrix before calling predict. Training time can be orders of magnitude faster for a sparse matrix input compared to a dense matrix when features have zero values in most of the samples.

# TREE BUILDING PROCESS

Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop.

## REGRESSION ALGORITHM

We can follow below steps for creating decision tree using ID3 algorithm (C4.5 is improved version of this).

ID3 Regression Algorithm: <http://www.saedsayad.com/decision_tree_reg.htm>

## CLASSIFCATION ALGORITHM

Below are examples:

Three splits from root node

<https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/>

Two splits from root node

<http://ucanalytics.com/blogs/decision-tree-cart-retail-case-example-part-5/>