**CHAID AND CART**

CART :-

Classification and Regression Trees or CART for short is a term introduced by [Leo Breiman](https://en.wikipedia.org/wiki/Leo_Breiman) to refer to [Decision Tree](https://en.wikipedia.org/wiki/Decision_tree_learning) algorithms that can be used for classification or regression predictive modeling problems.

Classically, this algorithm is referred to as “decision trees”, but on some platforms like R they are referred to by the more modern term CART.

The CART algorithm provides a foundation for important algorithms like bagged decision trees, random forest and boosted decision trees

**CART Model Representation**

The representation for the CART model is a binary tree.

This is your binary tree from algorithms and data structures, nothing too fancy. Each root node represents a single input variable (x) and a split point on that variable (assuming the variable is numeric).

The leaf nodes of the tree contain an output variable (y) which is used to make a prediction.

Given a dataset with two inputs (x) of height in centimeters and weight in kilograms the output of sex as male or female, below is a crude example of a binary decision tree (completely fictitious for demonstration purposes only).

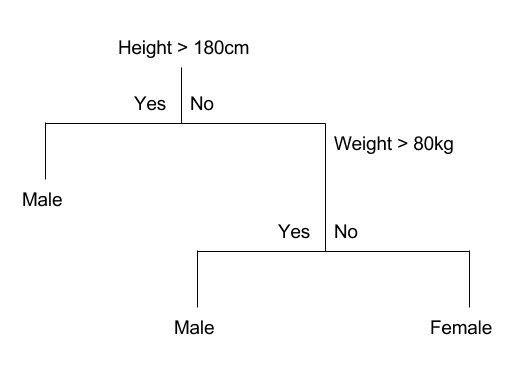
With the binary tree representation of the CART model described above, making predictions is relatively straightforward.

Given a new input, the tree is traversed by evaluating the specific input started at the root node of the tree.

A learned binary tree is actually a partitioning of the input space. You can think of each input variable as a dimension on a p-dimensional space. The decision tree split this up into rectangles (when p=2 input variables) or some kind of hyper-rectangles with more inputs.

New data is filtered through the tree and lands in one of the rectangles and the output value for that rectangle is the prediction made by the model. This gives you some feeling for the type of decisions that a CART model is capable of making, e.g. boxy decision boundaries.

For example, given the input of [height = 160 cm, weight = 65 kg], we would traverse the above tree as follows:



Example Decision Tree

The tree can be stored to file as a graph or a set of rules. For example, below is the above decision tree as a set of rules.

## Learn a CART Model From Data

Creating a CART model involves selecting input variables and split points on those variables until a suitable tree is constructed.

The selection of which input variable to use and the specific split or cut-point is chosen using a greedy algorithm to minimize a cost function. Tree construction ends using a predefined stopping criterion, such as a minimum number of training instances assigned to each leaf node of the tree.

### Greedy Splitting

Creating a binary decision tree is actually a process of dividing up the input space. A greedy approach is used to divide the space called [recursive binary splitting](https://en.wikipedia.org/wiki/Binary_splitting).

This is a numerical procedure where all the values are lined up and different split points are tried and tested using a cost function. The split with the best cost (lowest cost because we minimize cost) is selected.

All input variables and all possible split points are evaluated and chosen in a greedy manner (e.g. the very best split point is chosen each time).

For regression predictive modeling problems the cost function that is minimized to choose split points is the sum squared error across all training samples that fall within the rectangle:

sum(y – prediction)^2

Where y is the output for the training sample and prediction is the predicted output for the rectangle.

For classification the Gini index function is used which provides an indication of how “pure” the leaf nodes are (how mixed the training data assigned to each node is).

G = sum(pk \* (1 – pk))

Where G is the Gini index over all classes, pk are the proportion of training instances with class k in the rectangle of interest. A node that has all classes of the same type (perfect class purity) will have G=0, where as a G that has a 50-50 split of classes for a binary classification problem (worst purity) will have a G=0.5.

For a binary classification problem, this can be re-written as:

G = 2 \* p1 \* p2  
or  
G = 1 – (p1^2 + p2^2)

The Gini index calculation for each node is weighted by the total number of instances in the parent node. The Gini score for a chosen split point in a binary classification problem is therefore calculated as follows:

G = ((1 – (g1\_1^2 + g1\_2^2)) \* (ng1/n)) + ((1 – (g2\_1^2 + g2\_2^2)) \* (ng2/n))

Where G is the Gini index for the split point, g1\_1 is the proportion of instances in group 1 for class 1, g1\_2 for class 2, g2\_1 for group 2 and class 1, g2\_2 group 2 class 2, ng1 and ng2 are the total number of instances in group 1 and 2 and n are the total number of instances we are trying to group from the parent node.

### Stopping Criterion

The recursive binary splitting procedure described above needs to know when to stop splitting as it works its way down the tree with the training data.

The most common stopping procedure is to use a minimum count on the number of training instances assigned to each leaf node. If the count is less than some minimum then the split is not accepted and the node is taken as a final leaf node.

The count of training members is tuned to the dataset, e.g. 5 or 10. It defines how specific to the training data the tree will be. Too specific (e.g. a count of 1) and the tree will overfit the training data and likely have poor performance on the test set.

### Pruning The Tree

The stopping criterion is important as it strongly influences the performance of your tree. You can use [pruning](https://en.wikipedia.org/wiki/Pruning_(decision_trees)) after learning your tree to further lift performance.

The complexity of a decision tree is defined as the number of splits in the tree. Simpler trees are preferred. They are easy to understand (you can print them out and show them to subject matter experts), and they are less likely to overfit your data.

The fastest and simplest pruning method is to work through each leaf node in the tree and evaluate the effect of removing it using a hold-out test set. Leaf nodes are removed only if it results in a drop in the overall cost function on the entire test set. You stop removing nodes when no further improvements can be made.

More sophisticated pruning methods can be used such as cost complexity pruning (also called weakest link pruning) where a learning parameter (alpha) is used to weigh whether nodes can be removed based on the size of the sub-tree

Classification and regression trees are prediction models constructed by re-cursively partitioning a data set and fitting a simple model to each partition. Their name derives from the usual practice of describing the partitioning pro-cess by a decision tree. This article reviews some widely available algorithms and compares their capabilities, strengths and weaknesses in two examples. Classification and regression trees are machine learning methods for con-structing prediction models from data. The models are obtained by recursively partitioning the data space and fitting a simple prediction model within each partition. As a result, the partitioning can be represented graphically as a decision tree. Classification trees are designed for dependent variables that take a finite number of unordered values, with prediction error measured in terms of misclassification cost. Regression trees are for dependent variables that take continuous or ordered discrete values, with prediction error typically measured by the squared difference between the observed and predicted values.

# Math behind Decision Tree Algorithm

[MLMath.io](https://medium.com/@ankitnitjsr13?source=post_page-----2aa398561d6d--------------------------------)

[MLMath.io](https://medium.com/@ankitnitjsr13?source=post_page-----2aa398561d6d--------------------------------)

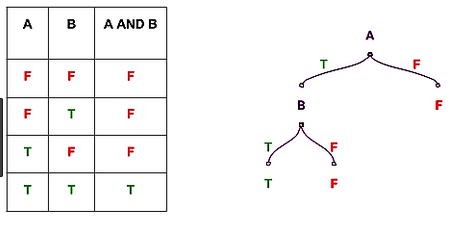
[Feb 20, 2019·9 min read](https://medium.com/@ankitnitjsr13/math-behind-decision-tree-algorithm-2aa398561d6d?source=post_page-----2aa398561d6d--------------------------------)

Decision tree algorithm is one of the most popular machine learning algorithm. It is a supervised machine learning algorithm, used for both classification and regression task. It is a model that uses set of rules to classify something.

This is the PART I of Decision Tree Tutorial.

[Link For PART II DECISION TREE TUTORIAL](https://medium.com/@ankitnitjsr13/decision-tree-algorithm-id3-d512db495c90)

Lets see decision tree with this simple example, It is normal “AND’ operation problem, where ‘A’, ‘B’ are features and “A and B” are corresponding labels.



Source Hackerearth

If A=F then result=F

If A=T and B=T, then result=T

If A=T and B=F, then result = F

This is a an example of binary classifier. It classify “And” operation is ‘False’ or ‘True’.

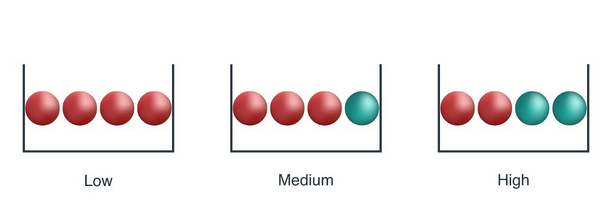
This story is about understanding the full mathematical concept of Decision tree algorithm. So,lets break it!!!

**Before diving deeper into the concept lets understand about impurity, types of measures of impurity. it will help us to understand the algorithm better.**

## Impurity

Let’s understand Impurity with the following toy example,





source Medium

From above image, a ball is randomly drawn from each bowl. So how much information you needed to accurately tells the color of ball. So, left bowl needed less information as all of the ball is red colored, central bowl needed more information than left bowl to tell it accurately, and right bowl needed maximum information as both number of both color ball are same.

As information is measure of purity, so we can say that left bowl is pure node, middle is less impure and right is more impure.

***So, how can we measure impurity in sample??***

There are couple of impurity measures are there, but in this story we will talk about only two such measure,

1. Entropy
2. Gini index/ Gini impurity

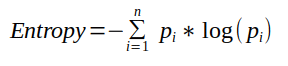
## Entropy

Entropy is amount of information is needed to accurately describe the some sample. So if sample is homogeneous, means all the element are similar than Entropy is 0, else if sample is equally divided than entropy is maximum 1.

So, left bowl has lowest entropy, middle bowl has more entropy and right bowl has highest entropy.

Mathematically it is written as ,

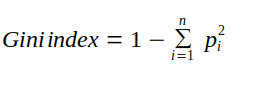




## Gini index / Gini impurity

Gini index is measure of inequality in sample. It has value between 0 and 1. Gini index of value 0 means sample are perfectly homogeneous and all element are similar, whereas, Gini index of value 1 means maximal inequality among elements. It is sum of the square of the probabilities of each class. It is illustrated as,





i is number of classes

***So what is the importance of impurity measure in decision tree??***

Impurity measures the homogeneity in the data sample. If the sample is homogeneous then sample are from same class.

## Decision Tree Algorithm

Decision tree algorithm is a tree where nodes represents features(attributes), branch represents decision(rule) and leaf nodes represents outcomes(discrete and continuous).

So how decision tree algorithm are built actually??

There are Various algorithm that are used to generate decision tree from data, some are as following,

1. Classification and regression tree CART
2. ID 3
3. CHAID
4. ID 4.5

In this tutorial we will only talk about CART and next tutorial will explain concept of ID3 algorithm. These are mostly used in the industry.

So lets start with a generating a classification tree with the help of CART algorithm*,*

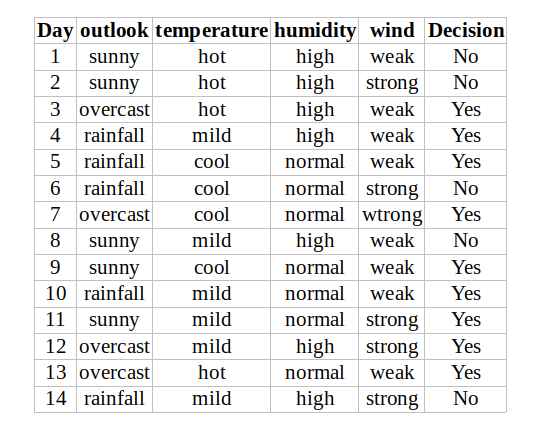
## CART

1. It is used for generating both classification tree and regression tree.
2. It uses Gini index as metric/cost function to evaluate split in feature selection in case of classification tree.
3. It is used for binary classification.
4. It use least square as a metric to select features in case of Regression tree.

*Lets start with generating classification tree.*

Lets start with weather data set, which is quite famous in explaining decision tree algorithm,where target is to predict play or not( Yes or No) based on weather condition.





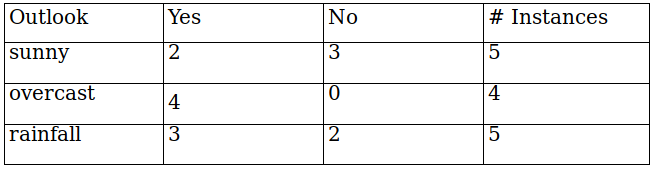
From data, outlook, temperature, humidity, wind are the features of data.

So, lets start building tree,

## Outlook

Outlook is a nominal feature. it can take three value, sunny, overcast and rain. Lets summarize the final decision for outlook features,





Gini index (outlook=sunny)= 1-(2/5)²-(3/5)² = 1- 0.16–0.36 = 0.48

Gini index(outlook=overcast)= 1- (4/4)²-(0/4)² = 1- 1- 0 = 0

Gini index(outlook=rainfall)= 1- (3/5)² -(2/5)² = 1- 0.36- 0.16 = 0.48

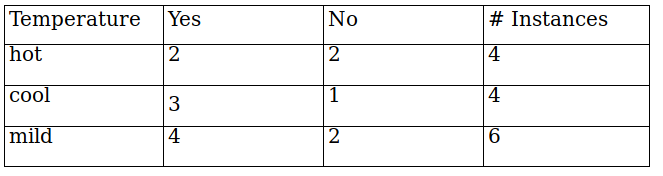
Now , we will calculate the weighted sum of Gini index for outlook features,

Gini(outlook) = (5/14)\*0.48 + (4/14) \*0 + (5/14)\*0.48 = 0.342

## Temperature

Similarly, temperature is also a nominal feature, it can take three values, hot,cold and mild. lets summarize the final decision of temperature feature,





Gini(temperature=hot) = 1-(2/4)²-(2/4)² = 0.5

Gini(temperature=cool) = 1-(3/4)²-(1/4)² = 0.375

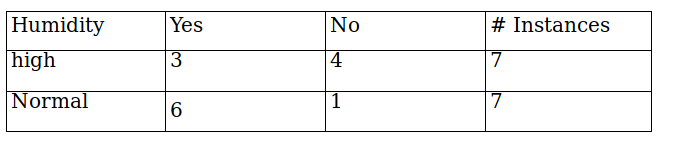
Gini(temperature=mild) = 1-(4/6)²-(2/6)² = 0.445

Now, the weighted sum of Gini index for temperature features can be calculated as,

Gini(temperature)= (4/14) \*0.5 + (4/14) \*0.375 + (6/14) \*0.445 =0.439

## Humidity





Humidity is a binary class feature , it can take two value high and normal.

Gini(humidity=high) = 1-(3/7)²-(4/7)² = 0.489

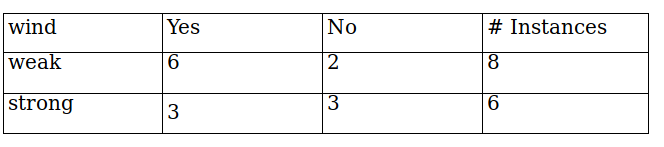
Gini(humidity=normal) = 1-(6/7)²-(1/7)² = 0.244

Now, the weighted sum of Gini index for humidity features can be calculated as,

Gini(humidity) = (7/14) \*0.489 + (7/14) \*0.244=0.367

## Wind





wind is a binary class feature , it can take two value weak and strong.

Gini(wind=weak)= 1-(6/8)²-(2/8)² = 0.375

Gini(wind=strong)= 1-(3/6)²-(3/6)²= 0.5

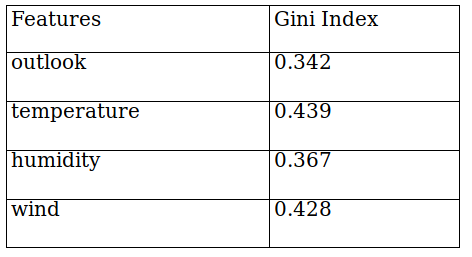
Now, the weighted sum of Gini index for wind features can be calculated as,

Gini(wind) = (8/14) \*0.375 + (6/14) \*0.5=0.428

## Decision for root node

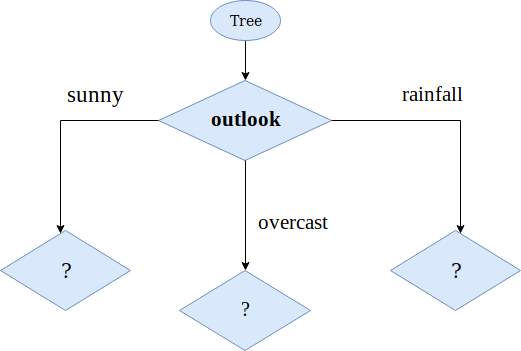
So,the final decision of all the features,





From table, you can seen that Gini index for outlook feature is lowest. So we get our root node.

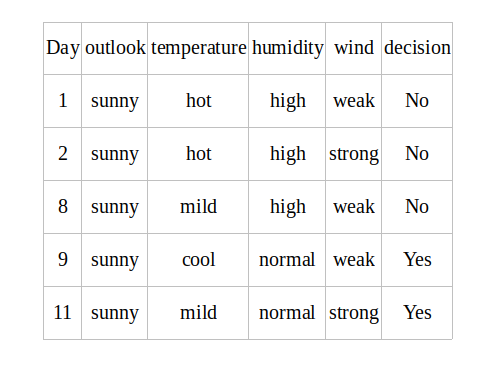




Lets calculate the Gini index on sub data set for outlook feature, as you can seen we have three sub data section, sunny, overcast, and rainfall of outlook feature. we will use same method as above to find next split.

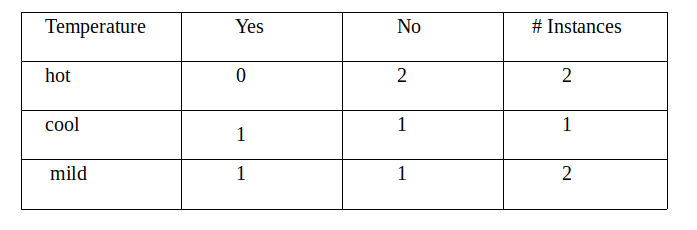
***So , lets focus on sub data on sunny outlook feature. we need to find the Gini index for temperature, humidity and wind feature respectively.***





## Gini index for temperature on sunny outlook





Gini(outlook=sunny & temperature=hot) = 1-(0/2)²-(2/2)² = 0

Gini(outlook=sunny & temperature=cool) = 1-(1/1)²-(0/1)² = 0

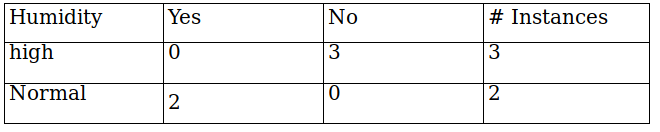
Gini(outlook=sunny & temperature=mild) = 1-(1/2)²-(1/2)² = 0.5

Now, the weighted sum of Gini index for temperature on sunny outlook features can be calculated as,

Gini(outlook=sunny & temperature)= (2/5) \*0 + (1/5) \*0+ (2/5) \*0.5 =0.2

## Gini Index for humidity on sunny outlook





Gini(outlook=sunny & humidity=high) = 1-(0/3)²-(3/3)² = 0

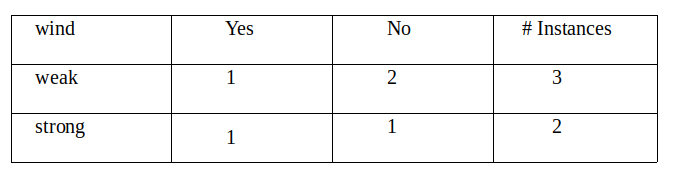
Gini(outlook=sunny & humidity=normal) = 1-(2/2)²-(0/2)² = 0

Now, the weighted sum of Gini index for humidity on sunny outlook features can be calculated as,

Gini(outlook = sunny & humidity) = (3/5) \*0 + (2/5) \*0=0

## Gini Index for wind on sunny outlook





Gini(outlook=sunny & wind=weak) = 1-(1/3)²-(2/3)² = 0.44

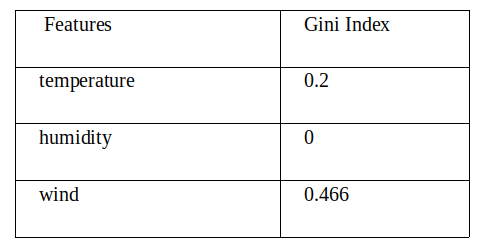
Gini(outlook=sunny & wind=strong) = 1-(1/2)²-(1/2)² = 0.5

Now, the weighted sum of Gini index for wind on sunny outlook features can be calculated as,

Gini(outlook = sunny and wind) = (3/5) \*0.44 + (2/5) \*0.5=0.266+0.2= 0.466

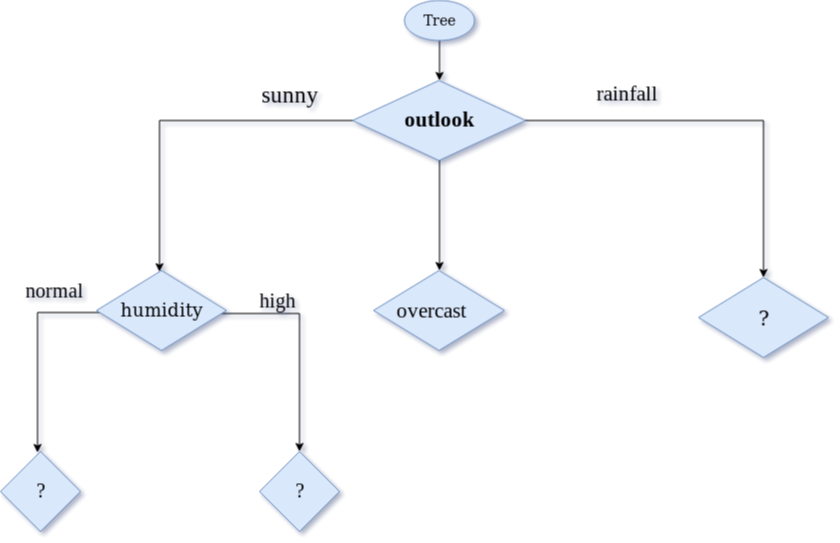
## Decision on sunny outlook factor





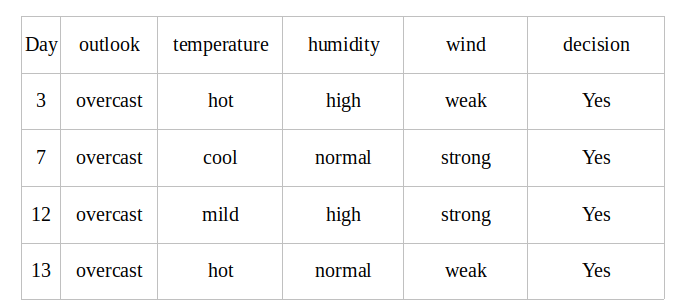
we have calculated the Gini index of all the features when the outlook is sunny. You can infer that humidity has lowest value. so next node will be humidity.





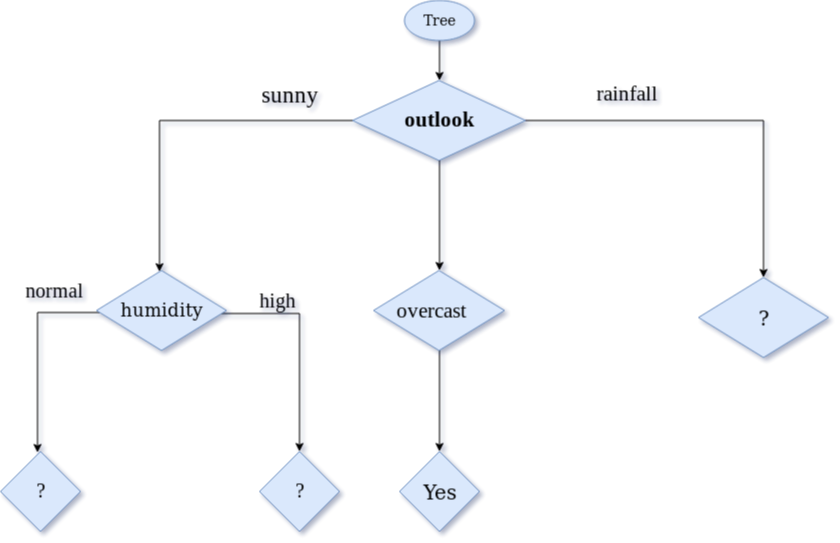
***Now,Lets focus on sub data for overcast outlook feature.***





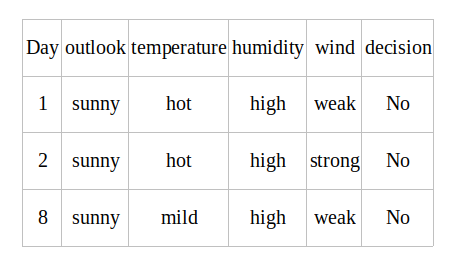
As, you can see from the above table all the decision for overcast outlook feature is always ‘Yes’. Then Gini index for each feature is 0, means it is a leaf nodes.



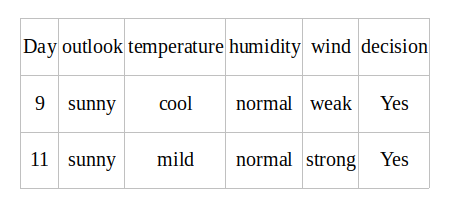


***Now,Lets focus on sub data for high and normal humidity feature.***



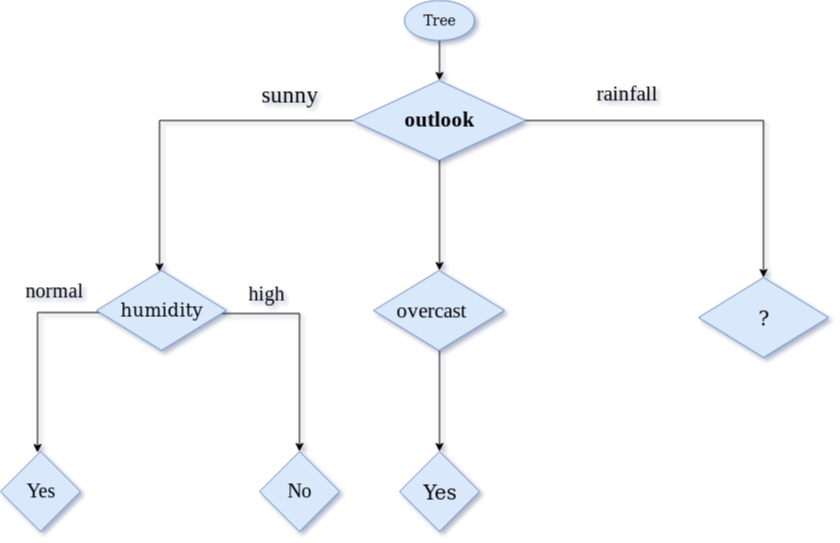






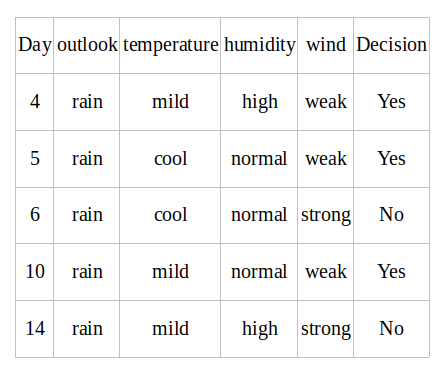
From the given two table, the decision is always ‘No’ when humidity is ‘high’ and decision is always ‘Yes’ when humidity is ‘normal’. So we got leaf node. now decision tree can be viewed as,





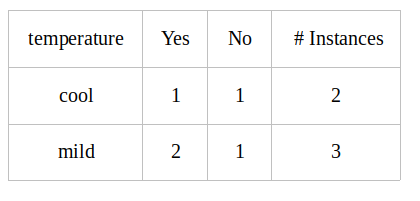
***Now,Lets focus on sub data for rainfall outlook feature. we need to find the Gini index for temperature,humidity and wind feature respectively.***





## Gini index for temperature for rainfall outlook





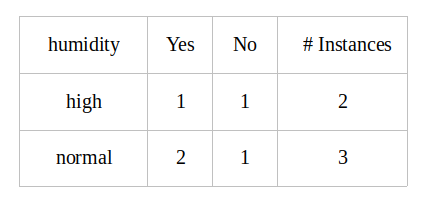
Gini(outlook=rainfall and temp.=Cool) = 1 — (1/2)2 — (1/2)2 = 0.5

Gini(outlook=rainfall and temp.=Mild) = 1 — (2/3)2 — (1/3)2 = 0.444

Gini(outlook=rainfall and temp.) = (2/5)\*0.5 + (3/5)\*0.444 = 0.466

## Gini index for humidity for rainfall outlook





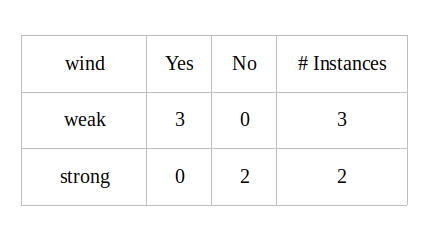
Gini(outlook=rainfall and humidity=high) = 1 — (1/2)2 — (1/2)2 = 0.5

Gini(outlook=rainfall and humidity=normal) = 1 — (2/3)2 — (1/3)2 = 0.444

Gini(Outlook=rainfall and humidity) = (2/5)\*(0.5 + (3/5)\*0.444 = 0.466

## Gini index for wind for rainfall outlook feature





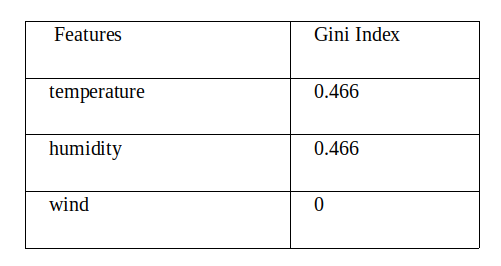
Gini(outlook=rainfall and wind=weak) = 1 — (3/3)2 — (0/3)2 = 0

Gini(outlook=rainfall and wind=strong) = 1 — (0/2)2 — (2/2)2 = 0

Gini(outlook=rainfall and wind) = (3/5)\*0 + (2/5)\*0 = 0

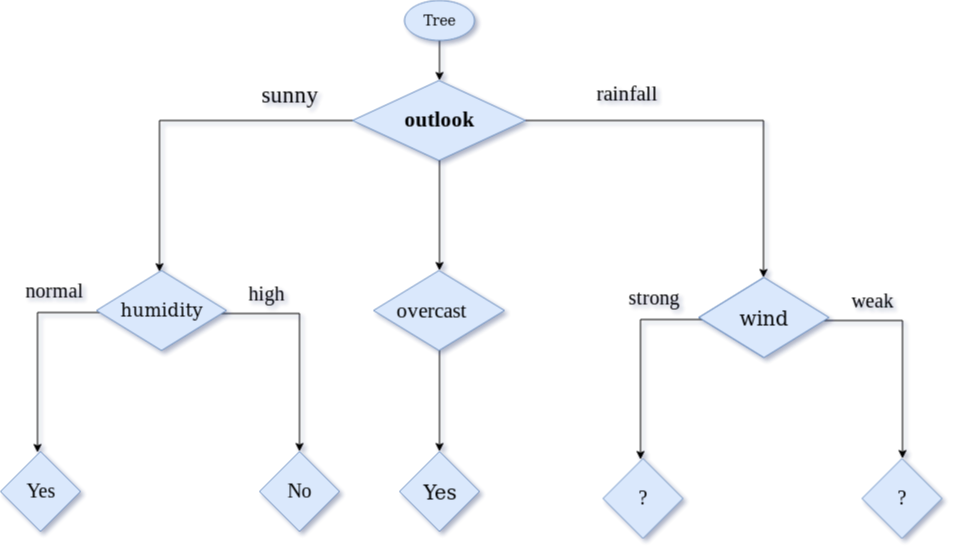
## Decision on rainfall outlook factor





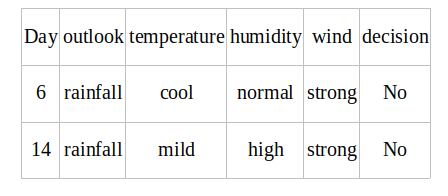
we have calculated the Gini index of all the features when the outlook is rainfall. You can infer that wind has lowest value. so next node will be wind.



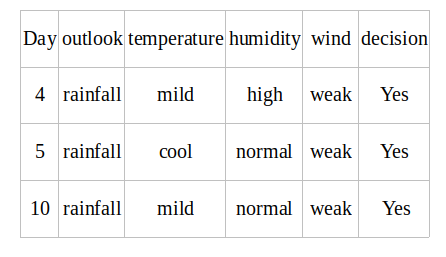


***Now,Lets focus on sub data strong and weak for wind rainfall feature.***



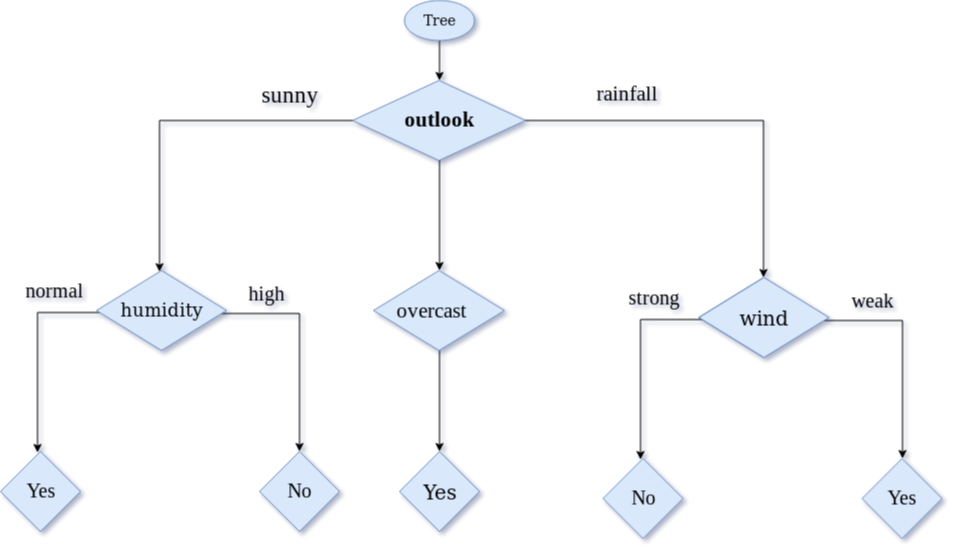






From the above two table, the decision is always ‘No’ when wind is ‘strong’ and decision is always ‘Yes’ when wind is ‘weak’. So we got leaf node.





So, we have explained the generation of decision tree in step by step manner.

**Chi-square Automatic Interaction Detector (CHAID)** was a technique created by Gordon V. Kass in 1980.  CHAID is a tool used to discover the relationship between variables.  CHAID analysis builds a predictive model, or tree, to help determine how variables best merge to explain the outcome in the given dependent variable. In  CHAID analysis, nominal, ordinal, and continuous data can be used, where continuous predictors are split into categories with approximately equal number of observations.  CHAID creates all possible cross tabulations for each categorical predictor until the best outcome is achieved and no further splitting can be performed.  In the CHAID technique, we can visually see the relationships between the split variables and the associated related factor within the tree.  The development of the decision, or classification tree, starts with identifying the target variable or dependent variable; which would be considered the root.  CHAID analysis splits the target into two or more categories that are called the initial, or parent nodes, and then the nodes are split using statistical algorithms into child nodes. Unlike in regression analysis, the CHAID technique does not require the data to be normally distributed.

**Merging:** In CHAID analysis, if the dependent variable is continuous, the*F*test is used and if the dependent variable is categorical, the [chi-square test](https://www.statisticssolutions.com/academic-solutions/membership-resources/member-profile/conducting-analyses-results/videos/chi-square/) is used.  Each pair of predictor categories are assessed to determine what is least significantly different with respect to the dependent variable.  Due to these steps of merging, a Bonferroni adjusted p-value is calculated for the merged cross tabulation.

**Decision tree components in CHAID analysis:**

In CHAID analysis, the following are the components of the decision tree:

1. **Root node:** Root node contains the dependent, or target, variable.  For example, CHAID is appropriate if a bank wants to predict the credit card risk based upon information like age, income, number of credit cards, etc.  In this example, credit card risk is the target variable and the remaining factors are the predictor variables.
2. **Parent’s node:** The algorithm splits the target variable into two or more categories.  These categories are called parent node or initial node.  For the bank example, high, medium and low categories are the parent’s nodes.
3. **Child node:** Independent variable categories which come below the parent’s categories in the CHAID analysis tree are called the child node.
4. **Terminal node:** The last categories of the CHAID analysis tree are called the terminal node.  In the CHAID analysis tree, the category that is a major influence on the dependent variable comes first and the less important category comes last.  Thus, it is called the terminal node.

### Classification Trees: CART vs. CHAID

When it comes to classification trees, there are three major algorithms used in practice. CART ("Classification and Regression Trees"), C4.5, and CHAID.  
  
All three algorithms create classification rules by constructing a tree-like structure of the data. However, they are different in a few important ways.  
  
The main difference is in the tree construction process. In order to avoid over-fitting the data, all methods try to limit the size of the resulting tree. CHAID (and variants of CHAID) achieve this by using a statistical stopping rule that discontinuous tree growth. In contrast, both CART and C4.5 first grow the full tree and then prune it back. 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. The tree is pruned until the performance is similar on both datasets (thereby indicating that there is no over-fitting of the training set). This highlights another difference between the methods: CHAID and C4.5 use a single dataset to arrive at the final tree, whereas CART uses a training set to build the tree and a holdout set to prune it.  
  
A difference between CART and the other two is that the CART splitting rule allows only binary splits (e.g., "if Income<$50K then X, else Y"), whereas C4.5 and CHAID allow multiple splits. In the latter, trees sometimes look more like bushes. CHAID has been especially popular in marketing research, in the context of market segmentation. In other areas, CART and C4.5 tend to be more popular. One important difference that came to my mind is in the goal that CHAID is most useful for, compared to the goal of CART. To clarify my point, let me first explain the CHAID mechanism in a bit more detail. At each split, the algorithm looks for the predictor variable that if split, most "explains" the category response variable. In order to decide whether to create a particular split based on this variable, the CHAID algorithm tests a hypothesis regarding dependence between the splitted variable and the categorical response(using the chi-squared test for independence). Using a pre-specified significance level, if the test shows that the splitted variable and the response are independent, the algorithm stops the tree growth. Otherwise the split is created, and the next best split is searched. In contrast, the CART algorithm decides on a split based on the amount of homogeneity within class that is achieved by the split. And later on, the split is reconsidered based on considerations of over-fitting. Now I get to my point: **It appears to me that CHAID is most useful for analysis, whereas CART is more suitable for prediction**. In other words, 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.

## DECISION TREE

Decisiontree learning or classification Trees are a collection of divide and conquer problem-solving strategies that use tree-like structures to predict the value of an outcome variable.

The tree starts with the root node consisting of the complete data and thereafter uses intelligent strategies to split the nodes into multiple branches.

The original dataset divided into subsets in this process.

To answer the fundamental inquiry, your oblivious brain makes a few computations (in light of the example questions recorded below) and you wind up purchasing the necessary amount of milk. Is it normal or weekday?

On weekdays days we require 1 Liter of Milk.

Is it a weekend? On weekends we require 1.5 Liter of Milk

Is it accurate to say that we are anticipating any guests today? We need to purchase 250 ML additional milk for every guest, and so on.

Before jumping into the hypothetical idea of decision trees how about we initially explain what are decision trees? what’s more, for what reason would it be a good idea for us to utilize them?

## Why use decision trees?

Outstanding amongst other supervised learning methods are tree-based algorithm. These are predictive models with higher accuracy, simple understanding.

How does the decision tree work?

There are different algorithm written to assemble a decision tree, which can be utilized by the problem

A few of the commonly used algorithms are listed below:

• CART

• ID3

• C4.5

• CHAID

Now we will explain about CHAID Algorithm step by step. Before that, we will discuss a little bit about chi\_square.

### chi\_square

Chi-Square is a statistical measure to find the difference between child and parent nodes. To calculate this we find the difference between observed and expected counts of target variable for each node and the squared sum of these standardized differences will give us the Chi-square value.

### Formula

To find the most dominant feature, chi-square tests will use that is also called CHAID whereas ID3 uses information gain, C4.5 uses gain ratio and CART uses the GINI index.

Today, most programming libraries (e.g. Pandas for Python) use Pearson metric for correlation by default.

The formula of chi-square:-

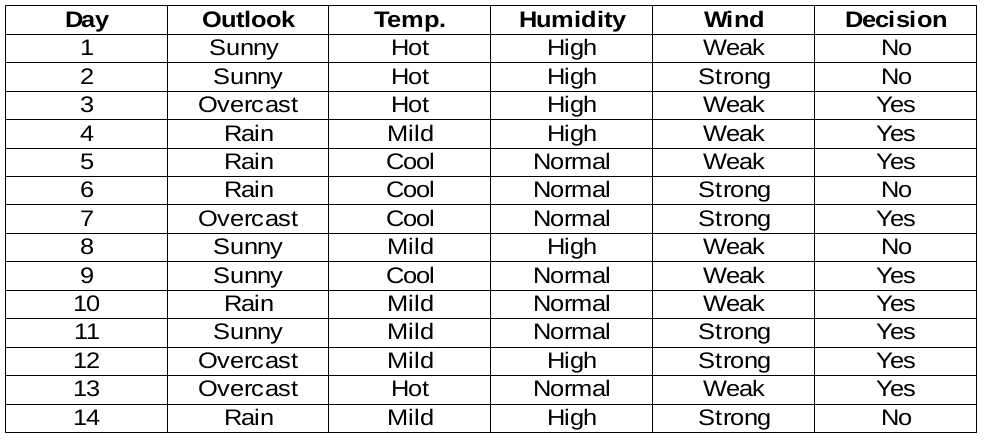
**√((y – y’)2 / y’)**

where y is actual and y’ is expected.

### Data set

We are going to build decision rules for the following data set. The decision column is the target we would like to find based on some features.

By The Way, we will ignore the day column because it just the row number.



to read dataset from CSV file python implementation below:-

import pandas as pd

data = pd.read\_csv("dataset.csv")

data.head()

We need to find the most important feature w.r.t target columns to choose the node to split data in this data set.

### Humidity feature

There are two types of the class present in humidity columns such that high and normal. Now we will calculate the chi\_square values for them.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | yes | No | Total | Expected | Chi-square Yes | Chi-square    No |
| High | 3 | 4 | 7 | 3.5 | 0.267 | 0.267 |
| low | 6 | 1 | 7 | 3.5 | 1.336 | 1.336 |

For each row, the total column is the sum of yes and no decisions.**Half of the total column is called Expected values**because there are 2 classes in the decision. It is easy to calculate the chi-squared values based on this table.

For example,

chi-square yes for high humidity is √(( 3– 3.5)2 / 3.5) = 0.267

whereas actual is 3 and expected is 3.5.

So, the chi-square value of the humidity feature is

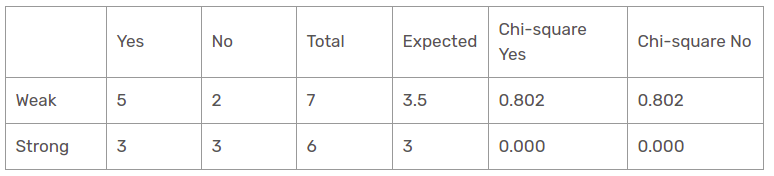
=  0.267 + 0.267 + 1.336 + 1.336

= 3.207

Now, we will find chi-square values for other features also. The feature having the maximum chi-square value will be the decision point. What about the wind feature?

### Wind feature

There are two types of the class present in wind columns such that weak and strong. The following table is the below table.



Herein, the chi-square test value of the wind feature is

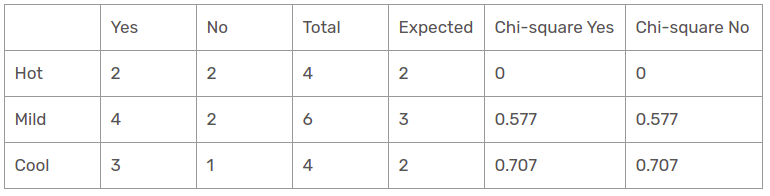
                                = 0.802 + 0.802 + 0 + 0

                                = 1.604

This is less value than the chi-square value of humidity as well. What about the temperature feature?

#### Temperature feature

There are three types of the class present in temperature columns such that hot, cool and mild. The following table is the below table.



Herein, the chi-square test value of the temperature feature is

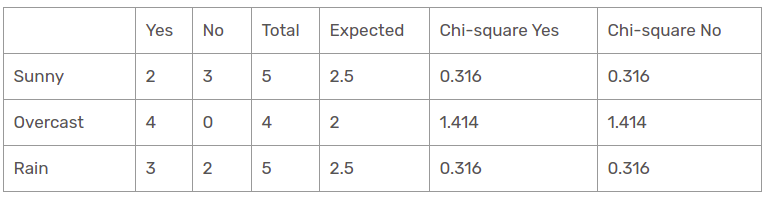
                                         = 0 + 0 + 0.577 + 0.577 + 0.707 + 0.707

                                         = 2.569

This is less value than the chi-square value of humidity and greater than the chi\_square value of wind as well. What about the outlook feature?

#### Outlook feature

There are three types of a class present in temperature columns such that sunny, rain, and overcast. The following table is the below table.

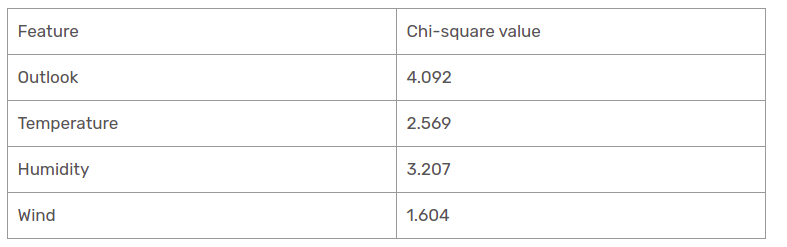


Herein, the chi-square test value of the outlook feature is

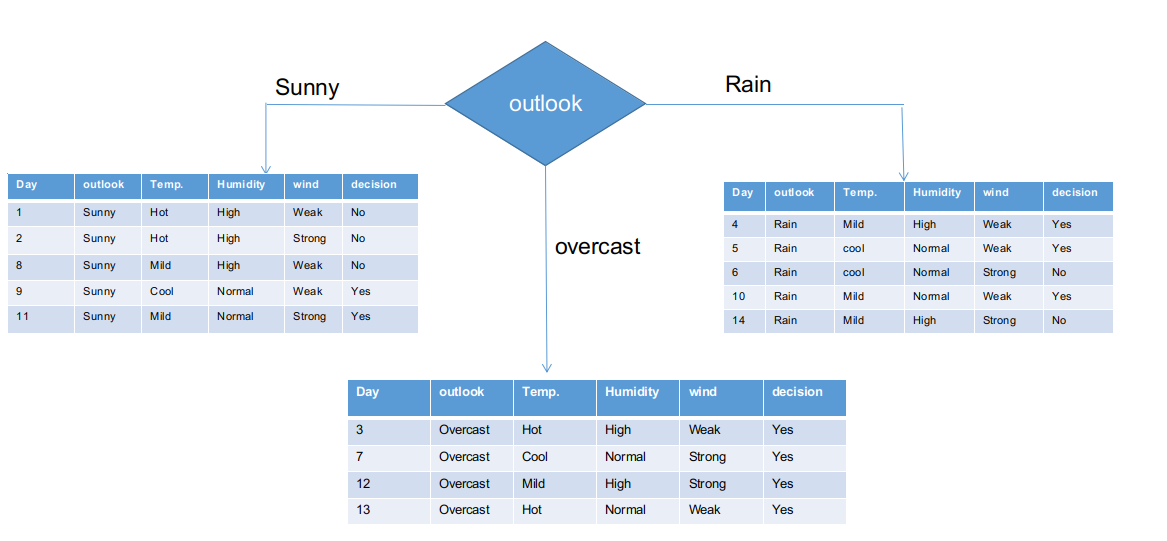
= 0.316 + 0.316 + 1.414 + 1.414 + 0.316 + 0.316

= 4.092

We have calculated the chi-square values of all features. Let’s see them all at one table.



As seen, the outlook column has the most elevated and highest chi-square value. This implies that it is the main component feature. Along with these values, we will put this feature to the root node.

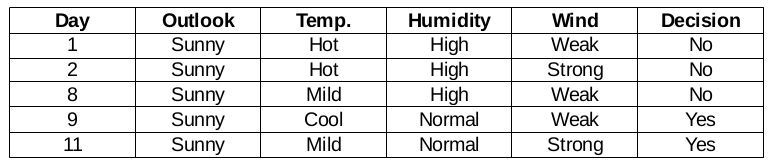


We’ve separated the raw information based on the outlook classes on the illustration above. For instance, the overcast branch simply has a yes decision in the sub informational dataset. This implies that the CHAID tree returns YES if the outlook is overcast.

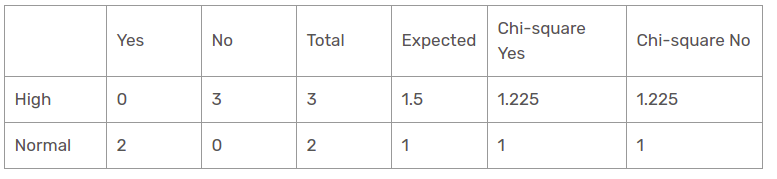
Both sunny and rain branches have yes and no decisions. We will apply chi-square tests for these sub informational datasets.

#### Outlook = Sunny branch

This branch has 5 examples. Presently, we search for the most predominant feature. By The Way, we will disregard the outlook feature now since they are altogether the same. At the end of the day, we will find out the most predominant columns among temperature, humidity, and wind.



#### Humidity feature for when the outlook is Sunny

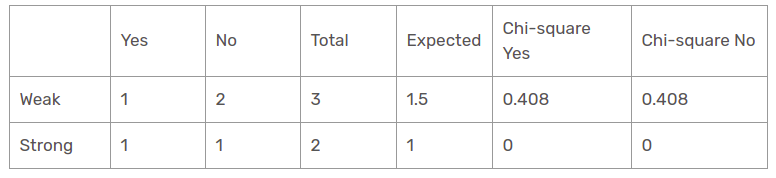


Chi-square value of humidity feature for sunny outlook is

=   1.225 + 1.225 + 1 + 1

= 4.449

#### Wind feature for when the outlook is Sunny

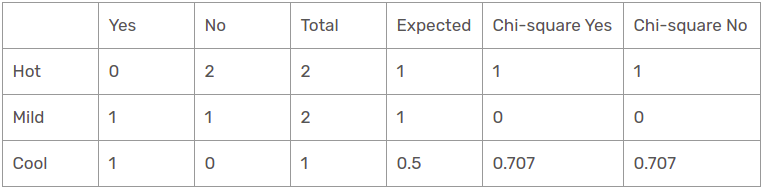


Chi-square value of wind feature for sunny outlook is

=     0.408 + 0.408 + 0 + 0

= 0.816

#### Temperature feature for when the outlook is Sunny

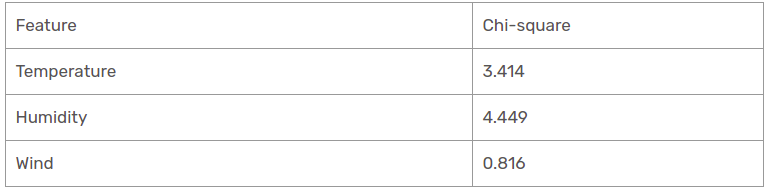


So, the chi-square value of temperature feature for sunny outlook is

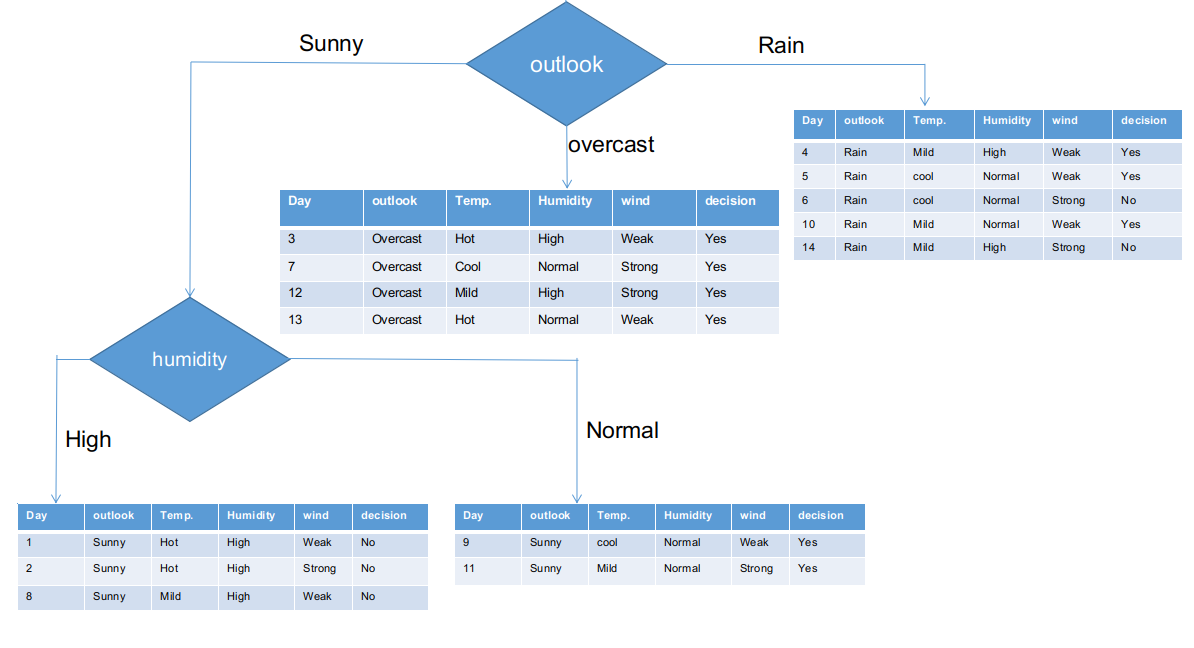
=     1 + 1 + 0 + 0 + 0.707 + 0.707

= 3.414

We have found chi-square values for sunny is outlook. Let’s see them all at a table.



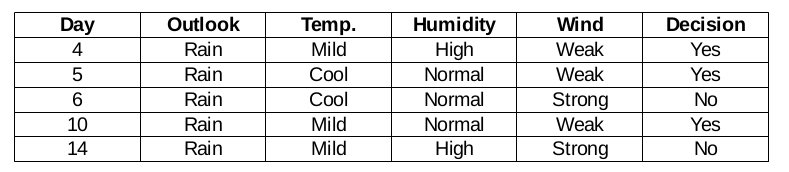
Presently, humidity is the most predominant feature for the sunny outlook branch. We will put this feature as a decision rule.



Presently, both humidity branches for sunny outlook have only one decision as delineated previously. CHAID tree will return NO for sunny outlook and high humidity and it will return YES for sunny outlook and normal humidity.

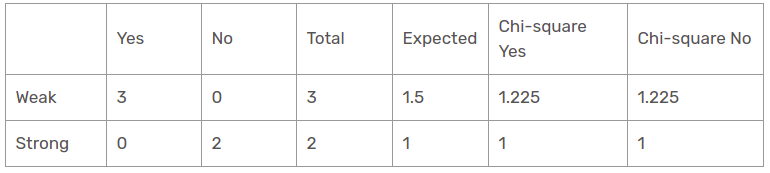
#### Rain outlook branch

This branch actually has both yes and no decisions. We need to apply the chi-square test for this branch to find out an accurate decision. This branch has 5 distinct instances as demonstrated in the accompanying sub informational collection dataset. How about we find out the most predominant feature among temperature, humidity and wind.



#### Wind feature for rain outlook

There are two types of a class present in wind feature for rain outlook such that weak and strong.



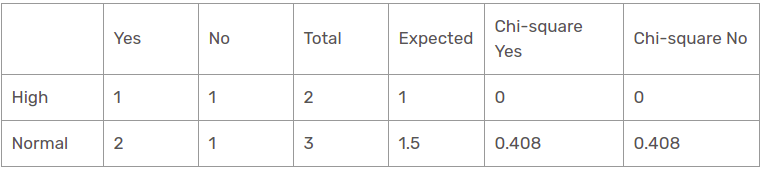
So, the chi-square value of wind feature for rain outlook is

=     1.225 + 1.225 + 1 + 1

= 4.449

#### Humidity feature for rain outlook

There are two types of a class present in humidity feature for rain outlook such that high and normal.



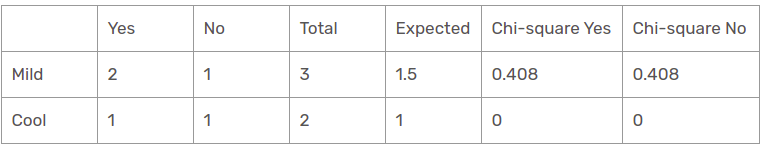
Chi-square value of humidity feature for rain outlook is

=      0 + 0 + 0.408 + 0.408

=    0.816

#### Temperature feature for rain outlook

There are two types of a class present in temperature features for rain outlook such that mild and cool.

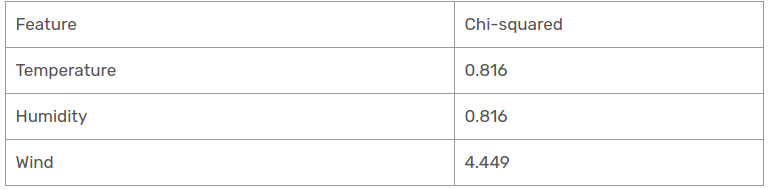


Chi-square value of temperature feature for rain outlook is

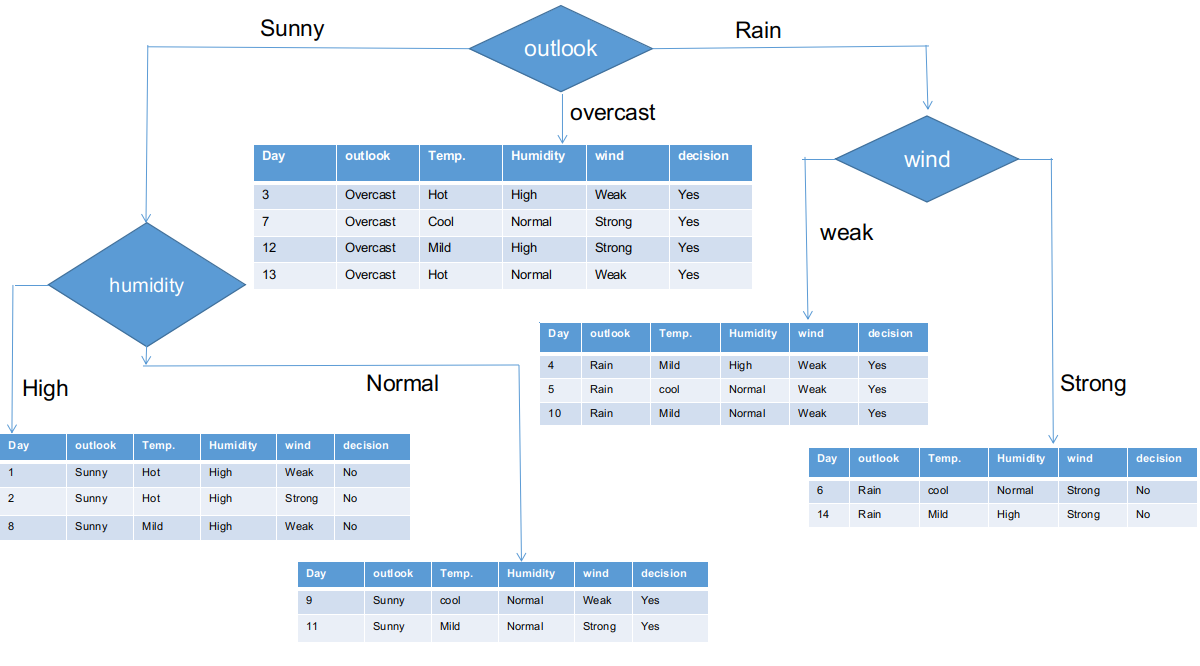
= 0 + 0 + 0.408 + 0.408

                                                   = 0.816

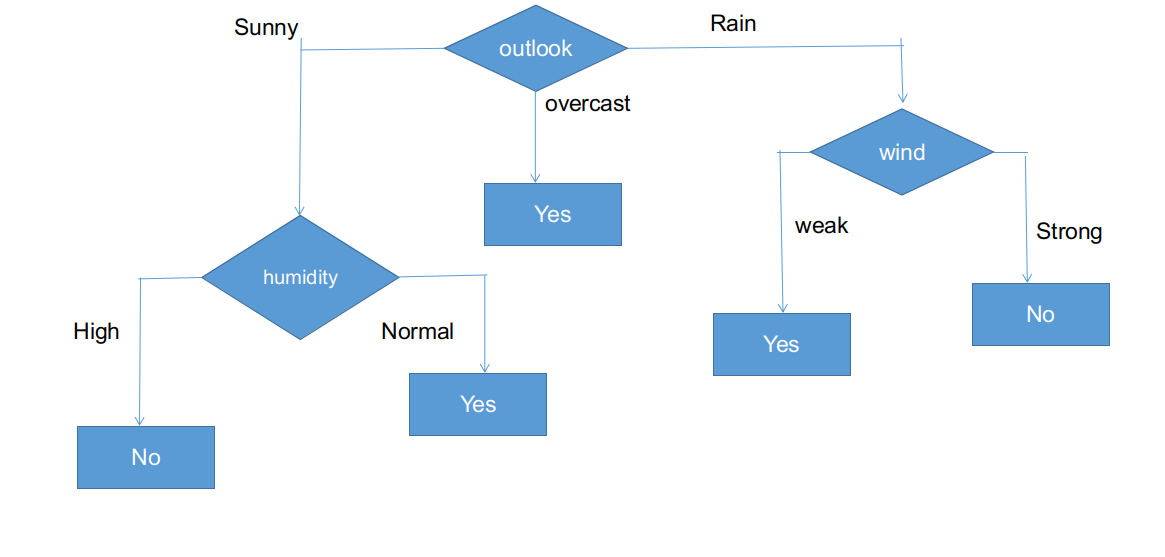
We have found all chi-square values for rain is outlook branch. Let’s see them all at a single table.



Thus, the wind feature is the victor for the rain is the outlook branch. Put this column in the connected branch and see the corresponding sub informational dataset.



As seen, all branches have sub informational datasets having a single decision such that yes or no. In this way, we can generate the CHAID tree as illustrated below.



The final form of the CHAID tree.

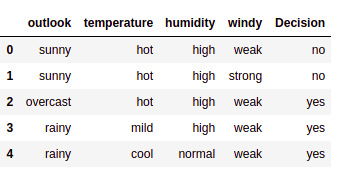
## Python implementation of a Decision tree using CHAID

from chefboost import Chefboost as cb

import pandas as pd

data = pd.read\_csv("/home/kajal/Downloads/weather.csv")

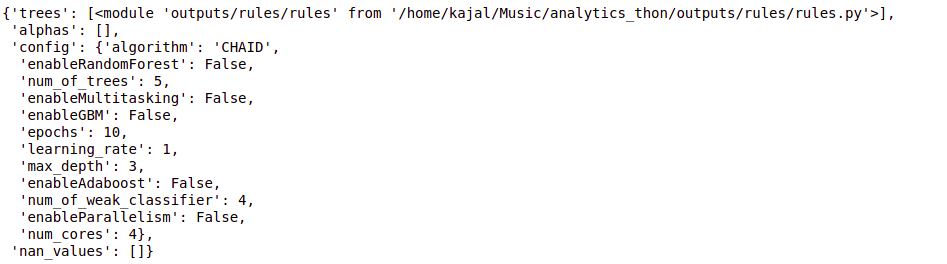
data.head()



config = {"algorithm": "CHAID"}

tree = cb.fit(data, config)

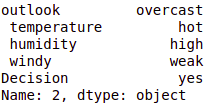
#### tree



# test\_instance = ['sunny','hot','high','weak','no']

test\_instance = data.iloc[2]

test\_instance



cb.predict(tree,test\_instance)

output:- 'Yes'

#obj[0]: outlook, obj[1]: temperature, obj[2]: humidity, obj[3]: windy

# {"feature": "outlook", "instances": 14, "metric\_value": 4.0933, "depth": 1}

def findDecision(obj):

          if obj[0] == 'rainy':

          # {"feature": " windy", "instances": 5, "metric\_value": 4.4495, "depth": 2}

                  if obj[3] == 'weak':

                         return 'yes'

                  elif obj[3] == 'strong':

                         return 'no'

                  else:

                          return 'no'

          elif obj[0] == 'sunny':

           # {"feature": " humidity", "instances": 5, "metric\_value": 4.4495, "depth": 2}

                 if obj[2] == 'high':

                        return 'no'

                 elif obj[2] == 'normal':

                         return 'yes'

                 else:

                         return 'yes'

         elif obj[0] == 'overcast':

                      return 'yes'

         else:

                    return 'yes'

## Conclusion

Thus, we have created a CHAID decision tree from scratch to end in this post. CHAID uses a chi-square measurement metric to find out the most important feature and apply this recursively until sub informational datasets have a single decision. Even though this is a legacy decision tree algorithm, it is as yet the same process for classification problems.

lassification and regression trees is a term used to describe decision tree algorithms that are used for classification and regression learning tasks.

**The Classification and Regression Tree methodology, also known as the CART were introduced in 1984 by Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone.**

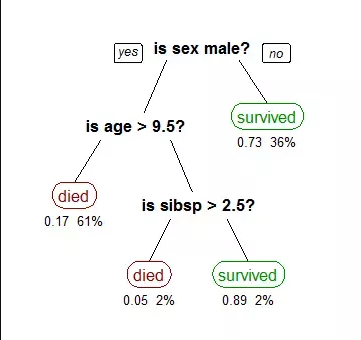
In order to understand classification and regression trees better, we need to first understand decision trees and how they are used.

While there are many classification and regression tree ppts and tutorials around, we need to start with the basics.

## ****What are Decision Trees?****

**Table of Contents**[[show](https://www.digitalvidya.com/blog/classification-and-regression-trees/)]

If you strip it down to the basics, decision tree algorithms are nothing but if-else statements that can be used to predict a result based on data. For instance, this is a simple decision tree that predicts whether a passenger on the Titanic survived.



Decision Trees

Machine learning algorithms can be classified into two types- supervised and unsupervised. A decision tree is a supervised machine learning algorithm. It has a tree-like structure with its root node at the top.

## ****Classification and Regression Trees Tutorial****

The CART or Classification & Regression Trees methodology refers to these two types of decision trees.

While there are many classification and regression trees tutorials and classification and regression trees ppts out there, here is a simple definition of the two kinds of decision trees. It also includes classification and regression tree examples.

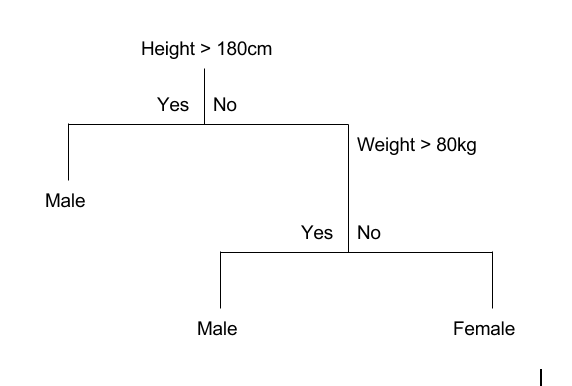
### (i) Classification Trees

A classification tree is an algorithm where the target variable is fixed or categorical. The algorithm is then used to identify the “class” within which a target variable would most likely fall.

An example of a classification-type problem would be determining who will or will not subscribe to a digital platform; or who will or will not graduate from high school.

These are examples of simple binary classifications where the categorical dependent variable can assume only one of two, mutually exclusive values. In other cases, you might have to predict among a number of different variables. For instance, you may have to predict which type of smartphone a consumer may decide to purchase.

In such cases, there are multiple values for the categorical dependent variable. Here’s what a classic classification tree looks like.

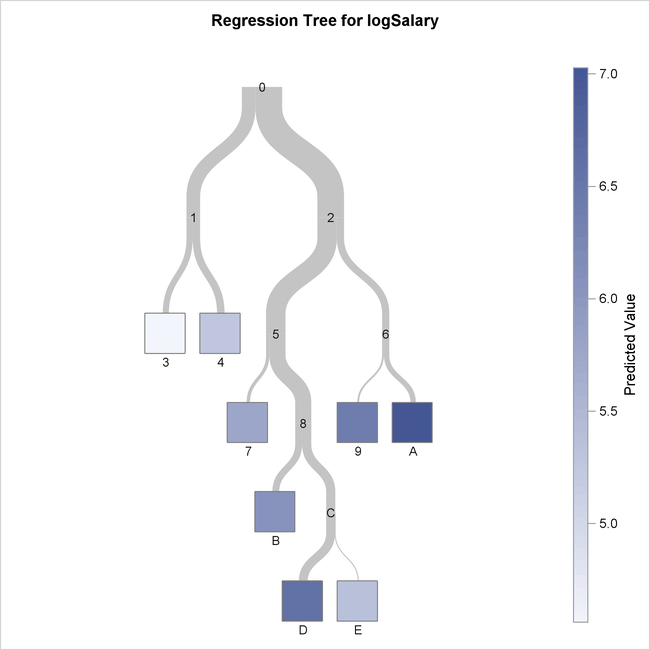


Classification Trees

### (ii) Regression Trees

A regression tree refers to an algorithm where the target variable is and the algorithm is used to predict its value. As an example of a regression type problem, you may want to predict the selling prices of a residential house, which is a continuous dependent variable.

This will depend on both continuous factors like square footage as well as categorical factors like the style of home, area in which the property is located, and so on.



## ****Difference Between Classification and Regression Trees****

[Decision trees](https://www.digitalvidya.com/blog/a-in-depth-decision-tree-learning-tutorial-to-get-you-started/) are easily understood and there are several classification and regression trees ppts to make things even simpler. However, it’s important to understand that there are some fundamental differences between classification and regression trees.

### When to use Classification and Regression Trees

Classification trees are used when the dataset needs to be split into classes that belong to the response variable. In many cases, the classes Yes or No.

In other words, they are just two and mutually exclusive. In some cases, there may be more than two classes in which case a variant of the classification tree algorithm is used.

Regression trees, on the other hand, are used when the response variable is continuous. For instance, if the response variable is something like the price of a property or the temperature of the day, a regression tree is used.

In other words, regression trees are used for prediction-type problems while classification trees are used for classification-type problems.

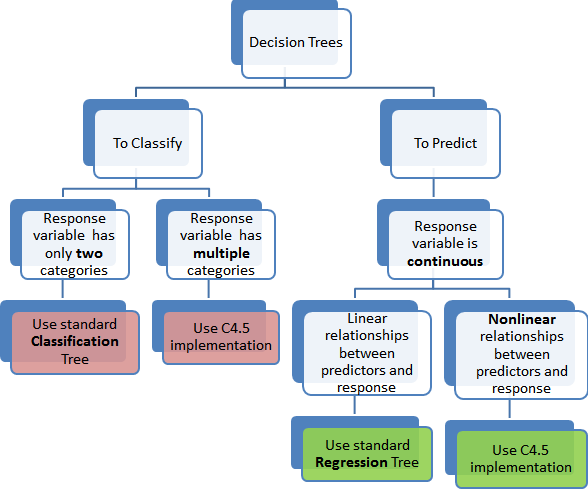
### How Classification and Regression Trees Work

A classification tree splits the dataset based on the homogeneity of data. Say, for instance, there are two variables; income and age; which determine whether or not a consumer will buy a particular kind of phone.

If the training data shows that 95% of people who are older than 30 bought the phone, the data gets split there and age becomes a top node in the tree. This split makes the data “95% pure”. Measures of impurity like entropy or Gini index are used to quantify the homogeneity of the data when it comes to classification trees.

In a regression tree, a regression model is fit to the target variable using each of the independent variables. After this, the data is split at several points for each independent variable.

At each such point, the error between the predicted values and actual values is squared to get “A Sum of Squared Errors”(SSE). The SSE is compared across the variables and the variable or point which has the lowest SSE is chosen as the split point. This process is continued recursively.



CART Working

### **Advantages of Classification and Regression Trees**

The purpose of the analysis conducted by any classification or regression tree is to create a set of if-else conditions that allow for the accurate prediction or classification of a case.

Classification and regression trees work to produce accurate predictions or predicted classifications, based on the set of if-else conditions. They usually have several advantages over regular decision trees.

#### (i) The Results are Simplistic

The interpretation of results summarized in classification or regression trees is usually fairly simple. The simplicity of results helps in the following ways.

* It allows for the rapid classification of new observations. That’s because it is much simpler to evaluate just one or two logical conditions than to compute scores using complex nonlinear equations for each group.
* It can often result in a simpler model which explains why the observations are either classified or predicted in a certain way. For instance, business problems are much easier to explain with if-then statements than with complex nonlinear equations.

#### (ii) Classification and Regression Trees are Nonparametric & Nonlinear

The results from classification and regression trees can be summarized in simplistic if-then conditions. This negates the need for the following implicit assumptions.

* The predictor variables and the dependent variable are linear.
* The predictor variables and the dependent variable follow some specific nonlinear link functions.
* The predictor variables and the dependent variable are monotonic.

Since there is no need for such implicit assumptions, classification and regression tree methods are well suited to data mining. This is because there is very little knowledge or assumptions that can be made beforehand about how the different variables are related.

As a result, classification and regression trees can actually reveal relationships between these variables that would not have been possible using other techniques.

#### (iii) Classification and Regression Trees Implicitly Perform Feature Selection

Feature selection or variable screening is an important part of analytics. When we use decision trees, the top few nodes on which the tree is split are the most important variables within the set. As a result, feature selection gets performed automatically and we don’t need to do it again.

### **Limitations of Classification and Regression Trees**

Classification and regression tree tutorials, as well as classification and regression tree ppts, exist in abundance. This is a testament to the popularity of these decision trees and how frequently they are used. However, these decision trees are not without their disadvantages.

There are many classification and regression tree examples where the use of a decision tree has not led to the optimal result. Here are some of the limitations of classification and regression trees.

#### (i) Overfitting

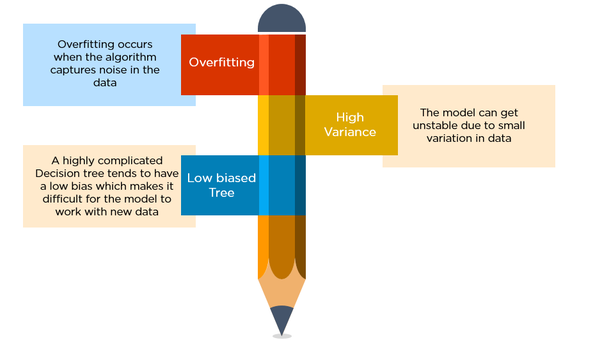
Overfitting occurs when the tree takes into account a lot of noise that exists in the data and comes up with an inaccurate result.

#### (ii) High variance

In this case, a small variance in the data can lead to a very high variance in the prediction, thereby affecting the stability of the outcome.

#### (iii) Low bias

A decision tree that is very complex usually has a low bias. This makes it very difficult for the model to incorporate any new data.



Limitations of Classification and Regression Trees

## ****What is a CART in Machine Learning?****

A Classification and Regression Tree(CART) is a predictive algorithm used in [machine learning](https://www.digitalvidya.com/blog/introduction-to-machine-learning/). It explains how a target variable’s values can be predicted based on other values.

It is a decision tree where each fork is split in a predictor variable and each node at the end has a prediction for the target variable.

The CART algorithm is an important [decision tree algorithm](https://www.digitalvidya.com/blog/decision-tree-algorithm/) that lies at the foundation of machine learning. Moreover, it is also the basis for other powerful machine learning algorithms like bagged decision trees, random forest, and boosted decision trees.

### Summing up

The Classification and regression tree(CART) methodology are one of the oldest and most fundamental algorithms. It is used to predict outcomes based on certain predictor variables.

# DIFFERENCE BETWEEN CHAID AND CART

**Classification and Regression Trees (CART)**  
  
**Regression Tree :** The outcome (dependent) variable is a continuous variable and predictor (independent) variables can be continuous or categorical variables (binary). It creates binary split.  
  
**Algorithm of Regression Tree:  Least-Squared Deviation or Least Absolute Deviation**  
  
The impurity of a node is measured by the Least-Squared Deviation (LSD), which is simply the within variance for the node.  
  
**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 conditional inference tree algorithm should be used.

**Algorithm of Classification Tree: Gini Index**  
  
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.  
  
**Process :**

1. Rules based on variables' values are selected to get the best split to differentiate observations based on the dependent variable
2. Once a rule is selected and splits a node into two, the same process is applied to each "child" node (i.e. it is a recursive procedure)
3. Splitting stops when CART detects no further gain can be made, or some pre-set stopping rules are met. (Alternatively, the data are split as much as possible and then the tree is later pruned.
4. **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).  
     
   When independent variables are continuous, they need to be transformed into categorical variables (bins/groups) before using CHAID.  
     
   **Algorithm :**  
     
   If dependent variable is categorical, Chi-Square test determines the best next split at each step.  
     
   If dependent variable is continuous, F test determines the best next split at each step.  
     
   **Process :**
5. Cycle through the predictors to determine for each predictor the pair of (predictor) categories that is least significantly different with respect to the dependent variable; for classification problems (where the dependent variable is categorical as well), it will compute a Chi-square test (Pearson Chi-square); for regression problems (where the dependent variable is continuous), F tests. If the respective test for a given pair of predictor categories is not statistically significant as defined by an alpha-to-merge value, then it will merge the respective predictor categories and repeat this step (i.e., find the next pair of categories, which now may include previously merged categories). If the statistical significance for the respective pair of predictor categories is significant (less than the respective alpha-to-merge value), then (optionally) it will compute a Bonferroni adjusted p-value for the set of categories for the respective predictor.  
     
   Selecting the split variable. The next step is to choose the split the predictor variable with the smallest adjusted p-value, i.e., the predictor variable that will yield the most significant split; if the smallest (Bonferroni) adjusted p-value for any predictor is greater than some alpha-to-split value, then no further splits will be performed, and the respective node is a terminal node.  
     
   Continue this process until no further splits can be performed (given the alpha-to-merge and alpha-to-split values). **(Source : Statsoft)**

|  |
| --- |
|  |
| Comparison of CHAID and CART |

1. **How CHAID is better than CART ?**  
     
   1. CHAID uses multiway splits by default (multiway splits means that the current node is splitted into more than two nodes). Whereas, CART does binary splits (each node is split into two daughter nodes) by default.  
   2. CHAID prevents overfitting problem. A node is only split if a significance criterion is fulfilled.

* In CART technique independent variable can be a binary(0/1,Yes/No, Married/Unmarried,Male/Female etc) or a continuous (ex: salary, age, height etc) but in CHAID it can be categorical (type of house :apartment,villa,bungalow / mode of transport : bus,car,train) variable.
* In CART dependent variables could be binary/continuous but in CHAID it can be more than 2 categories or continuous variables.
* In CART Gini index is the measure of classification and in CHAID it could be Chi-square or F test determines classification.

CHAID (Chi-squared Automatic Interaction Detection) and CART (Classification And Regression Trees)

* CHAID uses **multiway splits** by default (multiway splits means that the current node is splitted into more than two nodes). This may or may not be desired (it can lead to better segments or easier interpretation). When used for segmentation purposes this can backfire soon as CHAID needs a large sample sizes to work well. CART does binary splits (each node is split into two daughter nodes) by default.
* CHAID is intended to work with categorical targets. CART can definitely do regression and classification.
* CHAID uses a **pre-pruning idea**. A node is only split if a significance criterion is fulfilled. This ties in with the above problem of needing large sample sizes as the Chi-Square test has only little power in small samples. CART on the other hand grows a large tree and then **post-prunes** the tree back to a smaller version.
* Thus 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.
* The most important difference in my opinion is that split variable and split point selection in CHAID is less strongly confounded as in CART. This is largely irrelevant when the trees are used for prediction but is an important issue when trees are used for interpretation: A tree that has those two parts of the algorithm highly confounded is said to be "biased in variable selection". This means that split variable selection prefers variables with many possible splits . CART is highly "biased" in that sense, CHAID not so much.

A little whilst ago a colleague of mine in the digital analytics space asked me for help. The task at hand was to model both customer behaviour and selectively target potential new customers. However, she wanted increased efficacy than her predecessors. She described to me what the prior resource in her position had done, they had solely leveraged descriptive analytics. I explained the power of using descriptive, predictive and prescriptive analytics in conjunction, and she was sold. As a fairly new employee prescriptive was out of reach for now, so we focused on the descriptive and predictive disciplines.

The two techniques I identified as suitable solutions for her problem were CART 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 is a supervised model, where it has a sample of the population withheld. This subset will be used to train the proposed model in hopes of reducing over-fit data. On the other hand CHAID is an unsupervised technique, and it uses the entire model to build the tree.

In a CART model, the entire tree is grown, and then branches where data is deemed to be an over-fit are truncated by comparing the decision tree through the withheld subset. CHAID uses a statistical rule to stop tree growth called the Chi-Square test. The Chi-Square test qualifies the values observed, to those in theory. Values which are far off from theory, independent, are cut off and the tree stops at that branch. 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.

How did I use them in my work? CHAID is often a nice technique to use in marketing. If you want to decide on which customers you should target based on a campaign, and what is your likelihood of conversion. We used the 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 next touch-point 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.