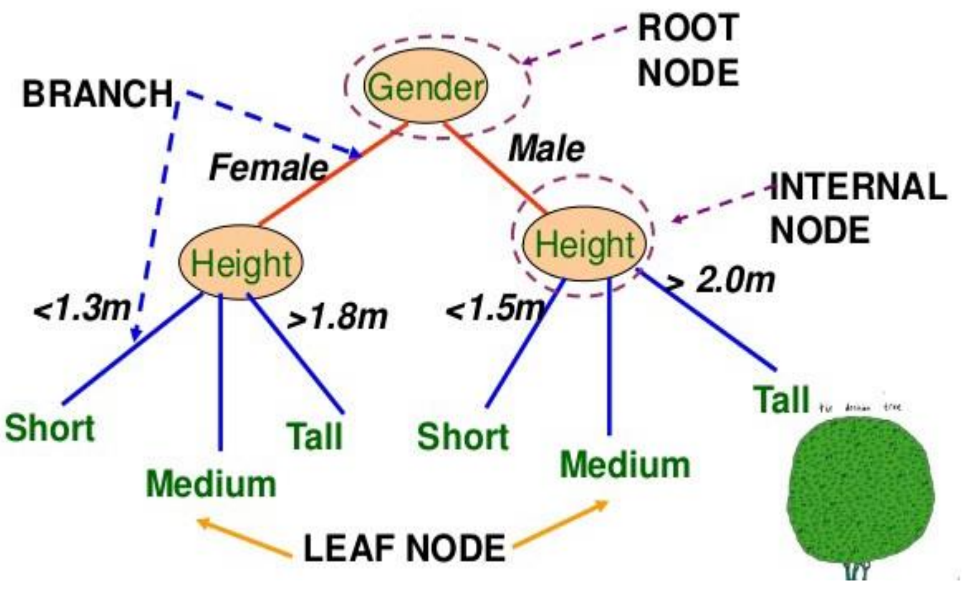
Chapter 9
Decision Trees

Classification is a two-step process, learning step and prediction step, in machine learning. In the learning step, the model is developed based on given training data. In the prediction step, the model is used to predict the response for given data. Decision Tree is one of the easiest and popular classification algorithms to understand and interpret.

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data (training data). In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

9.1 Decision Tree Representation

A decision tree is drawn upside down with its root at the top. The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.



Decision Trees follow Sum of Product (SOP) representation. The Sum of product (SOP) is also known as Disjunctive Normal Form. For a class, every branch from the root of the tree to a leaf node having the same class is conjunction (product) of values, different branches ending in that class form a disjunction (sum).

9.2 Advantages & Disadvantages of Decision Trees

Advantages

- Decision trees generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are capable of handling both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

Disadvantages

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems with many class and a relatively small number of training examples.
- Decision-tree learners can create over-complex trees that do not generalize the data well. This is called *overfitting*. Over fitting is one of the most practical difficulty for decision tree models. This problem gets solved by setting constraints on model parameters and pruning.
- Decision trees can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.

9.3 ID3 Algorithm

The ID3 algorithm builds decision trees using a top-down greedy search approach through the space of possible branches with no backtracking. A greedy algorithm, as the name suggests, always makes the choice that seems to be the best at that moment.

Steps in ID3 algorithm:

1. It begins with the original set S as the root node.
2. On each iteration of the algorithm, it iterates through the very unused attribute of the set S and calculates Entropy(H) and Information gain(IG) of this attribute.
3. It then selects the attribute which has the Largest Information gain.
4. The set S is then split by the selected attribute to produce a subset of the data.

5. The algorithm continues to recur on each subset, considering only attributes never selected before.

9.3.1 Attribute Selection Measures

If the dataset consists of N attributes, then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. By just randomly selecting any node to be the root can't solve the issue. If we follow a random approach, it may give us bad results with low accuracy. For solving this attribute selection problem, researchers worked and devised some solutions. They suggested using some criteria like:

- Entropy,
- Information gain,
- Gain Ratio
- Gini index
- Reduction in Variance
- Chi-Square

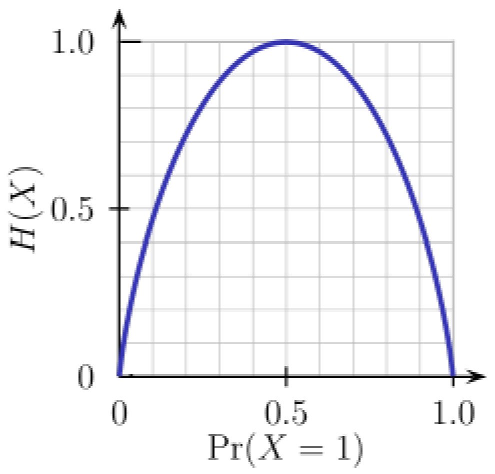
These criteria will calculate values for every attribute. The values are sorted, and attributes are placed in the tree by following the order i.e, the attribute with a high value (in case of information gain) is placed at the root. While using Information Gain as a criterion, we assume attributes to be categorical, and for the Gini index, attributes are assumed to be continuous.

Entropy

Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. Let x is our training set contains positive and negative examples, then the entropy of x relative to this classification is:

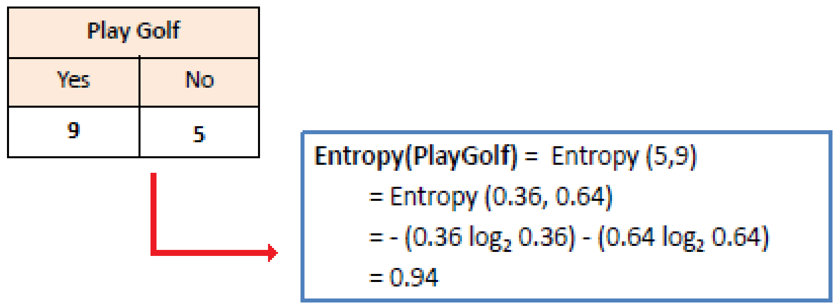
$$H(x) = -\ p_+\ log_2\ p_+ - p_-\ log_2\ p_-$$

OR



From the above graph, it is quite evident that the entropy H(X) is zero when the probability is either 0 or 1. The Entropy is maximum when the probability is 0.5 because it projects perfect randomness in the data and there is no chance if perfectly determining the outcome.

$$E(S)=\sum_{i=1}^{c}-p_i\log_2 p_i$$

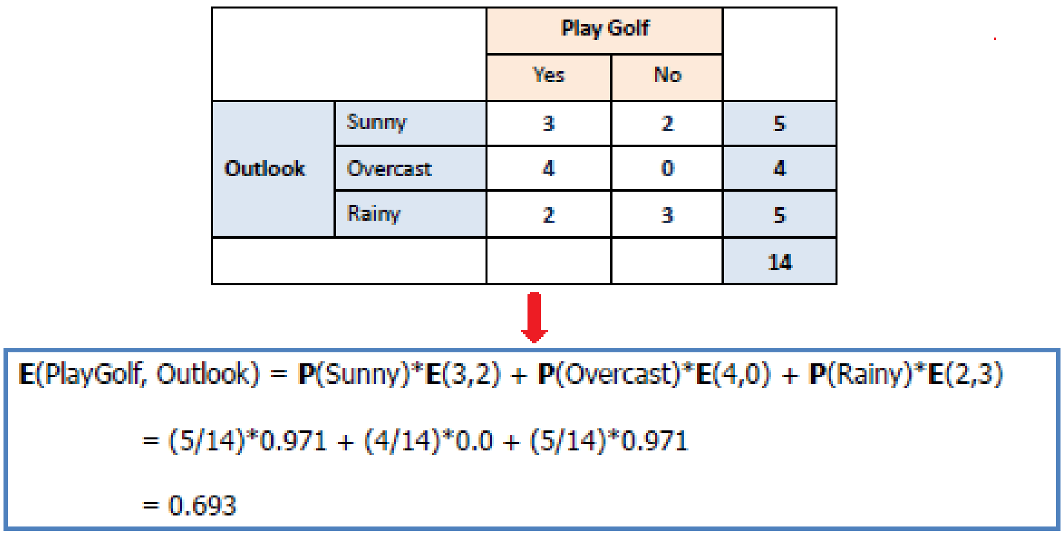


Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

Where **S → Current state, and Pi → Probability of an event *i* of state S or Percentage of class *i* in a node of state S.**
Mathematically Entropy for multiple attributes is represented as:

$$E(T,X)=\sum_{c\in X}P(c)E(c)$$



		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

Information Gain

Information gain or IG is a statistical property that measures how well a given attribute separates the training examples according to their target classification. Constructing a decision tree is all about finding an attribute that returns the highest information gain and the smallest entropy.

Information gain is a decrease in entropy. It computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values. ID3 (Iterative Dichotomiser) decision tree algorithm uses information gain.

Mathematically, IG is represented as:

$$Information\ Gain\ =\ Entropy(before) - \sum_{j=1}^{K} Entropy(j,\ after)$$

OR

$$Gain(S,A) = Entropy(S) - \sum \frac{|S_v|}{|S|} Entropy(S_v),\quad \text{v is values of atrribute A}$$

Where "before" is the dataset before the split, K is the number of subsets generated by the split, and (j, after) is subset j after the split.

9.3.2 An Intuitive Example: What the best day to Play Tennis?

Consider a table of data set below. Given the column "Play Tennis" as target attribute, and example of days which such conditions as attributes: Outlook; Temperature; Humidity; and Wind. We want to know what the best day to play tennis.

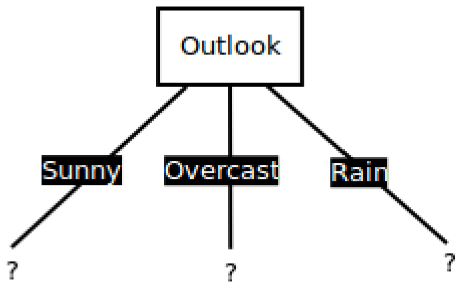
Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

We want to construct a decision tree of the training set above using ID3 algorithm.
For simplicity, we will give name of each attribute in data set:
- ✓ O is Outlook attribute
- ✓ T is Temperature attribute
- ✓ H is Humidity attribute
- ✓ W is Wind attribute

# First iteration
At first iteration, we need to know which is best attribute to be chosen as top root in our decision tree. To do that, ID3 will find the best attribute which is has maximum information gain. Given the information gain for each attribute:
attributes = [O, H, W, T]

G(x, O) = 0.246
G(x, H) = 0.151
G(x, W) = 0.048
G(x, T) = 0.029

So, based on the information gains calculated above, we choose attribute Outlook as first root node which has three branches Sunny, Rain, and Overcast. Our decision tree will look like an image below.



Initial decision tree

For the next iterations, we remove O from attribute list.

# Second iteration
We want examine which the best attribute for branch of Sunny. Remember, that new x is rows contains values of Sunny.

x = [x[0], x[1], x[7], x[8], x[10]]

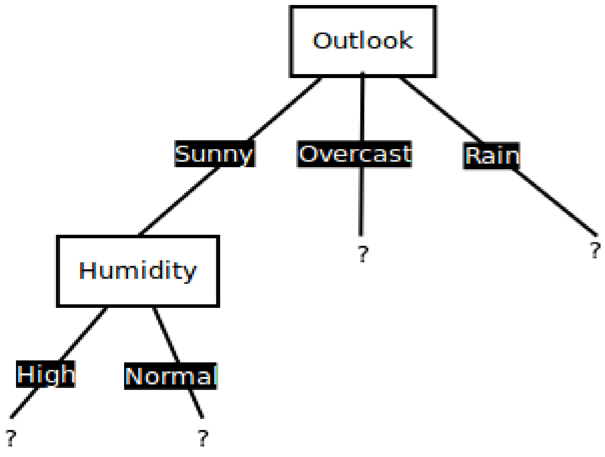
attribute= [H, W, T]

G(x, H) = .970 - (3/5) * 0 - (2/5) * 0 = .970

G(x, T) = .970 - (2/5) * 0 - (2/5) * 1 = .570

G(x, W) = .970 - (2/5) * 1 - (3/5) * .918 = .019

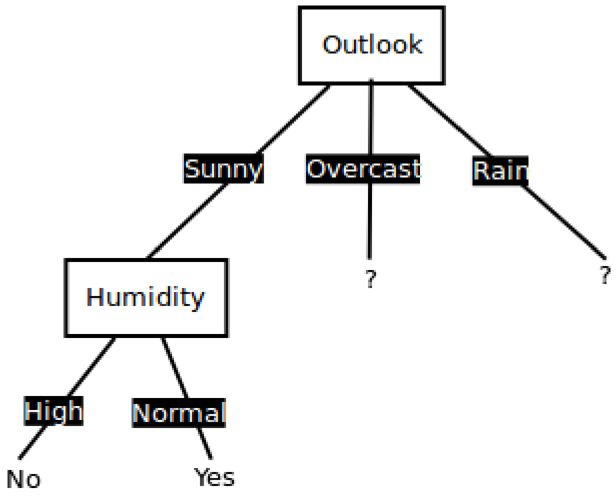
So, based on the information gains calculated above, we choose attribute Humidity as attribute in branch Sunny. Our decision tree will look like an image below.



Add new attribute Humidity

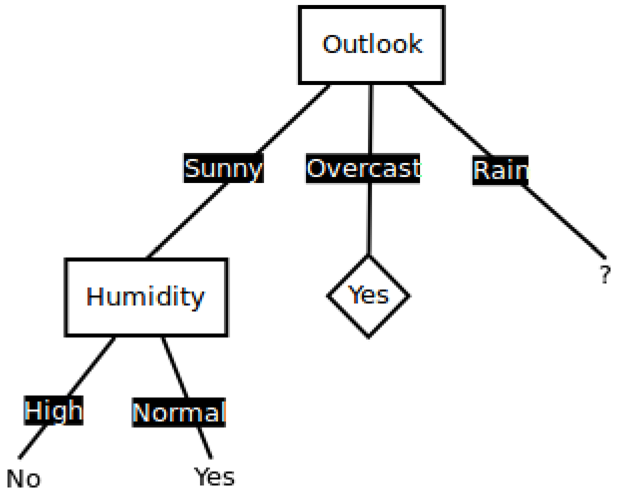
For the next iterations, we remove H from attribute list.

# Third iteration
Next node is an attribute Humidity which has two possible values {High, Normal}. A branch High dominated by single label which is No, caused this branch ended with a leaf contains label No. Same case with branch Normal which ended with a leaf contains label Yes.



Humidity’s Leafs

# Fourth iteration
All rows contain value Outcast are dominated by single label Yes, so branch of Overcast ended with a leaf contains label Yes.



Overcast's leaf

# Fifth iteration
We want examine which the best attribute for branch of Rain. Remember, that new x is rows contains values of Rain.

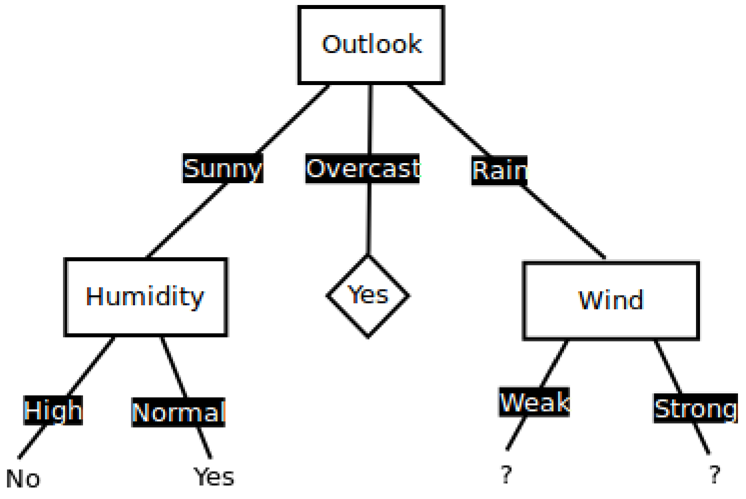
x = [x[3], x[4], x[5], x[9], x[13]]

attribute = [W, T]

G(x, W) = .970 - (2/5) * 0 - (3/5) * 0 = .970

G(x, T) = .970 - (0/5) * 0 - (3/5) * .918 - (2/5) * 1.0 = .019

So, based on the information gains calculated above, we choose attribute Wind as attribute in branch of Rain. Our decision tree will look like an image below.

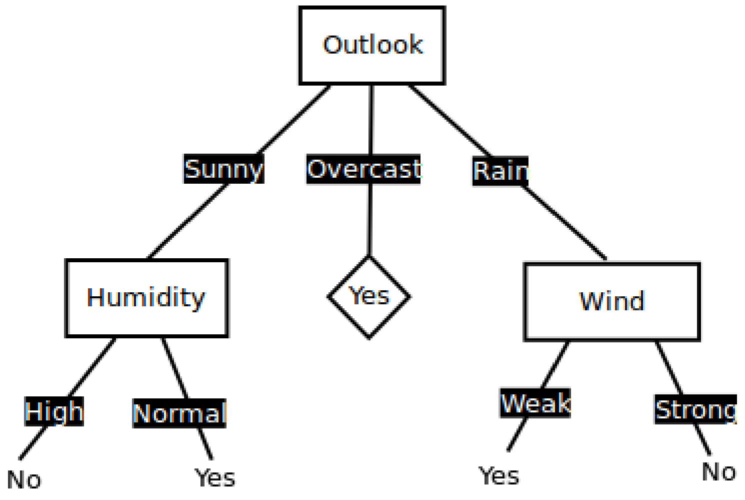


Add new attribute Wind

For next iteration, we remove attribute W from attribute list

# Sixth iteration

Next node is an attribute Wind which has two possible values {Weak, Strong}. A branch Strong dominated by single label which is No, caused this branch ended with a leaf contains label No. Same case with branch Weak which ended with a leaf contains label Yes.



# Seventh iteration
Actually, there is no iteration left, since all branches in our decision tree ended with leafs. In other word, we prune attribute Temperature from our decision tree. Ross Quinlan, inventor of ID3, made some improvements for these bottlenecks and created a new algorithm named C4.5. The C4.5 algorithms can create a more generalized models including continuous data and could handle missing data.

9.4 C4.5 Decision Tree

ID3 is the most common conventional decision tree algorithm but it has bottlenecks.

- Attributes must be nominal values,
- Dataset must not include missing data, and
- The algorithm tends to fall into overfitting.

Ross Quinlan, inventor of ID3, made some improvements for these bottlenecks and created a new algorithm named C4.5. Now, the algorithm can create a more generalized models including continuous data and could handle missing data.

ID3 uses information gain as a statistical property to measures how well a given attribute separates the training examples. However, Information gain is biased towards choosing attributes with a large number of values as root nodes. It means it prefers the attribute with a large number of distinct values.

C4.5, an improvement of ID3, uses Gain ratio which is a modification of Information gain that reduces its bias and is usually the best option. Gain ratio overcomes the problem with information gain by taking into account the number of branches that would result before making the split. It corrects information gain by taking the intrinsic information of a split into account. Gain ratio strategy, leads to better generalization (less overfitting) of DT models and it is better to use Gain ration in general.

$$Gain\ Ratio = \frac{Infomation\ Gain}{SplitInfo}$$

$$SplitInfo(S,A) = -\sum_{i=1}^{v} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

$$Gain(S,A) = Entropy(S) - \sum \frac{|S_v|}{|S|} Entropy(S_v), \quad \text{v is values of atrribute A}$$

S_i are the sets obtained by partitioning on value i of A.

Let us consider the following dataset. Here the temperature and humidity columns have continuous values instead of nominal ones.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

Firstly, we need to calculate global entropy. There are 14 examples; 9 instances refer to yes decision, and 5 instances refer to no decision.

Entropy(S) = -∑p(I) . log2p(I)
= – p(Yes) . log2p(Yes) – p(No) . log2p(No)
= – (9/14) . log2(9/14) – (5/14) . log2(5/14) = 0.940

Gain Raito for Wind Attribute

Let us consider first the Wind attribute. Wind is a nominal attribute. Its possible values are weak and strong.

Gain(S, Wind) = Entropy(S) – ∑ (p(S|Wind) . Entropy(S|Wind))
Gain(S, Wind) = Entropy(S) – [p(S|Wind=Weak) * Entropy(S|Wind=Weak)] + [p(S|Wind=Strong) . Entropy(S|Wind=Strong)]

There are 8 weak wind instances. 2 of them are concluded as no, 6 of them are concluded as yes.
Entropy(S|Wind=Weak) = – p(No) . log2p(No) – p(Yes) . log2p(Yes)
= – (2/8) . log2(2/8) – (6/8) . log2(6/8) = 0.811
Entropy(S|Wind=Strong) = – (3/6) . log2(3/6) – (3/6) . log2(3/6) = 1
Gain(S, Wind) = 0.940 – (8/14).(0.811) – (6/14).(1)
= 0.940 – 0.463 – 0.428 = ***0.049***
There are 8 decisions for weak wind, and 6 decisions for strong wind.
SplitInfo(S, Wind) = -(8/14).log2(8/14) – (6/14).log2(6/14)
= 0.461 + 0.524 = ***0.985***
GainRatio(S, Wind) = Gain(S, Wind) / SplitInfo(S, Wind)
= 0.049 / 0.985 = ***0.049***

Gain Raito for Outlook Attribute

Outlook is a nominal attribute, too. Its possible values are sunny, overcast and rain.
Gain(S, Outlook) = Entropy(S) – ∑ (p(S|Outlook) . Entropy(S|Outlook)) =
Gain(S, Outlook) = Entropy(S) – p(S|Outlook=Sunny)*
Entropy(S|Outlook=Sunny) – p(S|Outlook=Overcast)*
Entropy(S|Outlook=Overcast) – p(S|Outlook=Rain)*
Entropy(S|Outlook=Rain)
There are 5 sunny instances. 3 of them are concluded as no, 2 of them are concluded as yes.
Entropy(S|Outlook=Sunny) = – p(No) . log2p(No) – p(Yes) . log2p(Yes)
= -(3/5).log2(3/5) – (2/5).log2(2/5) = 0.441 + 0.528 = 0.970
Entropy(S|Outlook=Overcast) = – p(No) . log2p(No) – p(Yes) . log2p(Yes)
= -(0/4).log2(0/4) – (4/4).log2(4/4) = 0

Notice that log2(0) is actually equal to -∞ but assume that it is equal to 0.
Actually, lim (x->0) x.log2(x) = 0. If you wonder the proof, please look at this post.
Entropy(S|Outlook=Rain) = – p(No) . log2p(No) – p(Yes) . log2p(Yes)
= -(2/5).log2(2/5) – (3/5).log2(3/5) = 0.528 + 0.441 = 0.970
Gain(S, Outlook) = 0.940 – (5/14).(0.970) – (4/14).(0) – (5/14).(0.970) – (5/14).(0.970) = 0.246
There are 5 instances for sunny, 4 instances for overcast and 5 instances for rain
SplitInfo(S, Outlook) = -(5/14).log2(5/14) -(4/14).log2(4/14) -(5/14).log2(5/14) = 1.577

GainRatio(S, Outlook) = Gain(S, Outlook)/SplitInfo(S, Outlook) = 0.246/1.577 = 0.155

Gain Raito for Humidity Attribute

As an exception, humidity is a continuous attribute. We need to convert continuous values to nominal ones. C4.5 proposes to perform binary split based on a threshold value. Threshold should be a value which offers maximum gain for that attribute. Let's focus on humidity attribute. Firstly, we need to sort humidity values smallest to largest.

Now, we need to iterate on all humidity values and seperate dataset into two parts as instances less than or equal to current value, and instances greater than the current value. We would calculate the gain or gain ratio for every step. The value which maximizes the gain would be the threshold.

Day	Humidity	Decision
7	65	Yes
6	70	No
9	70	Yes
11	70	Yes
13	75	Yes
3	78	Yes
5	80	Yes
10	80	Yes
14	80	No
1	85	No
2	90	No
12	90	Yes
8	95	No
4	96	Yes

Check 65 as a threshold for humidity
Entropy(S|Humidity<=65) = – p(No) . log2p(No) – p(Yes) . log2p(Yes) = -(0/1).log2(0/1) – (1/1).log2(1/1) = 0
Entropy(S|Humidity>65) = -(5/13).log2(5/13) – (8/13).log2(8/13) =0.530 + 0.431 = 0.961
Gain(S, Humidity<> 65) = 0.940 – (1/14).0 – (13/14).(0.961) = 0.048
* The statement above refers to that what would branch of decision tree be for less than or equal to 65, and greater than 65. It does not refer to that humidity is not equal to 65!
SplitInfo(S, Humidity<> 65) = -(1/14).log2(1/14) -(13/14).log2(13/14) = 0.371
GainRatio(S, Humidity<> 65) = 0.126
Check 70 as a threshold for humidity
Entropy(S|Humidity<=70) = – (1/4).log2(1/4) – (3/4).log2(3/4) = 0.811
Entropy(S|Humidity>70) = – (4/10).log2(4/10) – (6/10).log2(6/10) = 0.970

Gain(S, Humidity<> 70) = 0.940 – (4/14).(0.811) – (10/14).(0.970) = 0.940 – 0.231 – 0.692 = 0.014
SplitInfo(S, Humidity<> 70) = -(4/14).log2(4/14) -(10/14).log2(10/14) = 0.863
GainRatio(S, Humidity<> 70) = 0.016
Check 75 as a threshold for humidity
Entropy(S|Humidity<=75) = – (1/5).log2(1/5) – (4/5).log2(4/5) = 0.721
Entropy(S|Humidity>75) = – (4/9).log2(4/9) – (5/9).log2(5/9) = 0.991
Gain(S, Humidity<> 75) = 0.940 – (5/14).(0.721) – (9/14).(0.991) = 0.940 – 0.2575 – 0.637 = 0.045
SplitInfo(S, Humidity<> 75) = -(5/14).log2(4/14) -(9/14).log2(10/14) = 0.940
GainRatio(S, Humidity<> 75) = 0.047
I think calculation demonstrations are enough. Now, I skip the calculations and write only results.
Gain(S, Humidity <> 78) =0.090, GainRatio(S, Humidity <> 78) =0.090
Gain(S, Humidity <> 80) = 0.101, GainRatio(S, Humidity <> 80) = 0.107
Gain(S, Humidity <> 85) = 0.024, GainRatio(S, Humidity <> 85) = 0.027
Gain(S, Humidity <> 90) = 0.010, GainRatio(S, Humidity <> 90) = 0.016
Gain(S, Humidity <> 95) = 0.048, GainRatio(S, Humidity <> 95) = 0.128
Here, I ignore the value 96 as threshold because humidity cannot be greater than this value.
As seen, gain maximizes when threshold is equal to 80 for humidity. This means that we need to compare other nominal attributes and comparison of humidity to 80 to create a branch in our tree.

Gain Raito for Temperature Attribute

Temperature feature is continuous as well. When we apply binary split to temperature for all possible split points, the following decision rule maximizes for both gain and gain ratio.

Gain(S, Temperature <> 83) = 0.113, GainRatio(S, Temperature<> 83) = 0.305

Attribute	**Gain**	**GainRatio**
Wind	0.049	0.049
Outlook	0.246	0.155
Humidity <> 80	0.101	0.107
Temperature <> 83	0.113	0.305

If we will use gain metric, then outlook will be the root node because it has the highest gain value. On the other hand, if we use gain ratio metric, then temperature will be the root node because it has the highest gain ratio value.

9.5 How to avoid/counter Overfitting in Decision Trees?

The common problem with Decision trees, especially having a table full of columns, they fit a lot. Sometimes it looks like the tree memorized the training data set. If there is no limit set on a decision tree, it will give you 100% accuracy on the training data set because in the worst case it will end up making 1 leaf for each observation. Thus this affects the accuracy when predicting samples that are not part of the training set.

Pruning Decision Trees

The splitting process results in fully grown trees until the stopping criteria are reached. But, the fully grown tree is likely to overfit the data, leading to poor accuracy on unseen data.

In pruning, you trim off the branches of the tree, i.e., remove the decision nodes starting from the leaf node such that the overall accuracy is not disturbed. This is done by segregating the actual training set into two sets: training data set, D and validation data set, V. Prepare the decision tree using the segregated training data set, D. Then continue trimming the tree accordingly to optimize the accuracy of the validation data set, V.

9.6 CART Decision tree

CART (**C**lassification **A**nd **R**egression **T**ree) is a decision tree algorithm variation. Decision Trees is the non-parametric supervised learning approach. CART can be applied to both regression and classification problems.

CART uses Gini Impurity in the process of splitting the dataset into a decision tree. Mathematically, we can write Gini Impurity as following:

$$I_{Gini} = 1 - \sum_{i=1}^{j} p_i^2$$

$$I_{Gini} = 1 - (\textit{the probability of target "No"})^2 - (\textit{the probability of target "Yes"})^2$$

In Gini Index, we do not require to compute logarithmic functions, which are computationally intensive.

CART Classification Tree)

Example (Play Game):

Gini index for Outlook Attribute

Gini(Outlook=Sunny)
= 1 – $(2/5)^2$ – $(3/5)^2$
= 1 – 0.16 – 0.36 = **0.48**

Gini(Outlook=Overcast)
= 1 – $(4/4)^2$ – $(0/4)^2$ = **0**

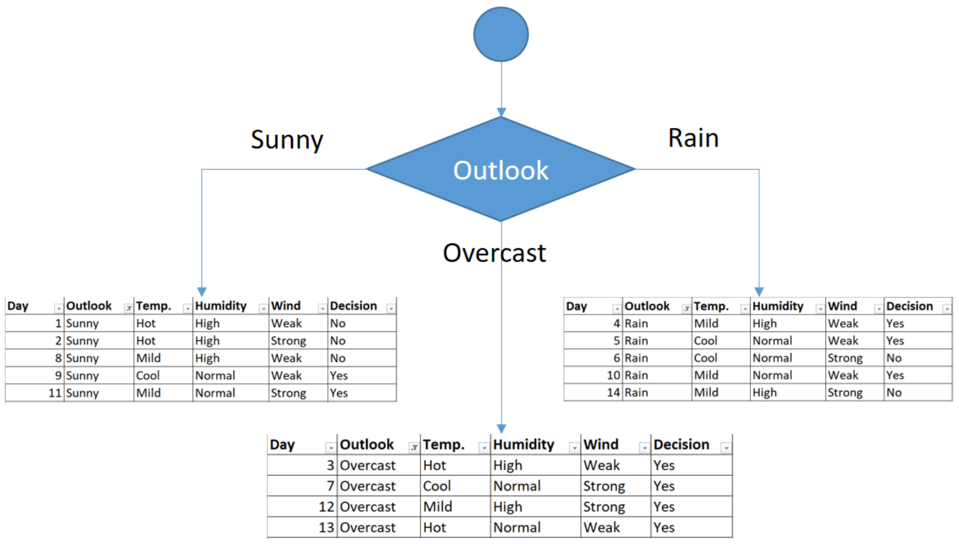
Gini(Outlook=Rain)
= 1 – $(3/5)^2$ – $(2/5)^2$
= 1 – 0.36 – 0.16 = **0.48**

Outlook	Yes	No	Number of instances
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5

Gini(Outlook) = (5/14) x 0.48 + (4/14) x 0 + (5/14) x 0.48 = 0.171 + 0 + 0.171
= **0.342**

Feature	Gini index
Outlook	0.342
Temperature	0.439
Humidity	0.367
Wind	0.428

Outlook feature will be the root of the tree because its cost is the lowest.



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

CART (Regression Tree)

Decision trees are powerful way to classify problems. On the other hand, they can be adapted into regression problems, too. Decision trees which built for a data set where the target column could be real number are called Regression Trees.

Sample Dataset

Day	Outlook	Temp.	Humidity	Wind	Golf Players
1	Sunny	Hot	High	Weak	25
2	Sunny	Hot	High	Strong	30
3	Overcast	Hot	High	Weak	46
4	Rain	Mild	High	Weak	45
5	Rain	Cool	Normal	Weak	52
6	Rain	Cool	Normal	Strong	23
7	Overcast	Cool	Normal	Strong	43
8	Sunny	Mild	High	Weak	35
9	Sunny	Cool	Normal	Weak	38
10	Rain	Mild	Normal	Weak	46
11	Sunny	Mild	Normal	Strong	48
12	Overcast	Mild	High	Strong	52
13	Overcast	Hot	Normal	Weak	44
14	Rain	Mild	High	Strong	30

Standard Deviation

Golf players = {25, 30, 46, 45, 52, 23, 43, 35, 38, 46, 48, 52, 44, 30}

Average of golf players = (25 + 30 + 46 + 45 + 52 + 23 + 43 + 35 + 38 + 46 + 48 + 52 + 44 + 30)/14 = **39.78**

Standard deviation of golf players = $\sqrt{[((25 - 39.78)^2 + (30 - 39.78)^2 + (46 - 39.78)^2 + \ldots + (30 - 39.78)^2)/14]}$ = **9.32**

We need to calculate standard deviation of golf players for all of Attribute values

Standard Deviation for Outlook attribute's value Sunny

Day	Outlook	Temp.	Humidity	Wind	Golf Players
1	Sunny	Hot	High	Weak	25
2	Sunny	Hot	High	Strong	30
8	Sunny	Mild	High	Weak	35
9	Sunny	Cool	Normal	Weak	38
11	Sunny	Mild	Normal	Strong	48

Golf players for sunny outlook = {25, 30, 35, 38, 48}

Average of golf players for sunny outlook = (25+30+35+38+48)/5 = **35.2**

Standard deviation of golf players for sunny outlook

= $\sqrt{(((25 - 35.2)^2 + (30 - 35.2)^2 + \ldots)/5)}$ = **7.78**

Standard Deviation for Outlook attribute's value Overcast

Day	Outlook	Temp.	Humidity	Wind	Golf Players
3	Overcast	Hot	High	Weak	46
7	Overcast	Cool	Normal	Strong	43
12	Overcast	Mild	High	Strong	52
13	Overcast	Hot	Normal	Weak	44

Golf players for overcast outlook = {46, 43, 52, 44}

Average of golf players for overcast outlook = (46 + 43 + 52 + 44)/4 = **46.25**

Standard deviation of golf players for overcast outlook

= √(((46-46.25)^2+(43-46.25)^2+…)= **3.49**

Standard Deviation for Outlook attribute's value Rain

Day	Outlook	Temp.	Humidity	Wind	Golf Players
4	Rain	Mild	High	Weak	45
5	Rain	Cool	Normal	Weak	52
6	Rain	Cool	Normal	Strong	23
10	Rain	Mild	Normal	Weak	46
14	Rain	Mild	High	Strong	30

Golf players for rainy outlook = {45, 52, 23, 46, 30}

Average of golf players for rainy outlook = (45+52+23+46+30)/5 = **39.2**

Standard deviation of golf players for rainy outlook

= √(((45 – 39.2)^2+(52 – 39.2)^2+…)/5)=**10.87**

Summarizing standard deviations for the outlook feature

Outlook	Stdev of Golf Players	Instances
Overcast	3.49	4
Rain	10.87	5
Sunny	7.78	5

Weighted standard deviation for outlook

= (4/14)x3.49 + (5/14)x10.87 + (5/14)x7.78 = **7.66**

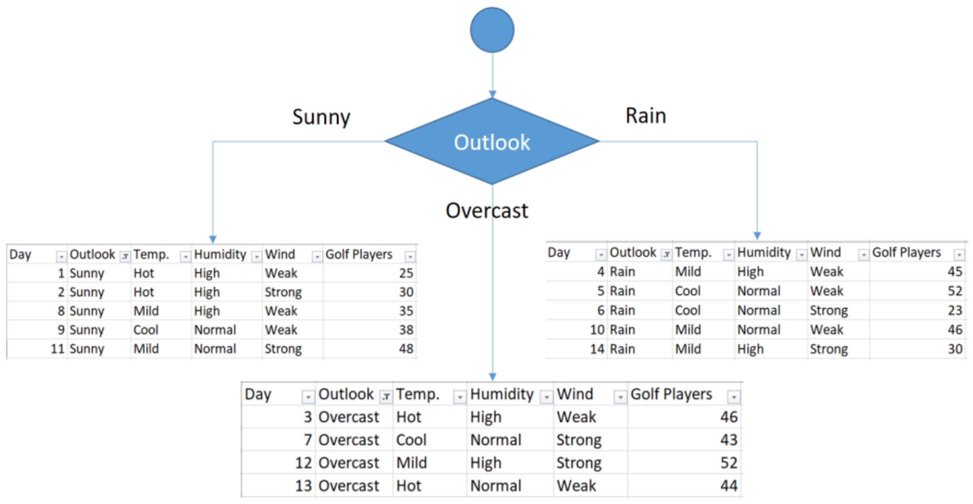
The global standard deviation of golf players = **9.32**

Standard deviation reduction for outlook = 9.32 – 7.66 = **1.66**

Summarizing standard deviations for different features

Feature	Standard Deviation Reduction
Outlook	1.66
Temperature	0.47
Humidity	0.27
Wind	0.29

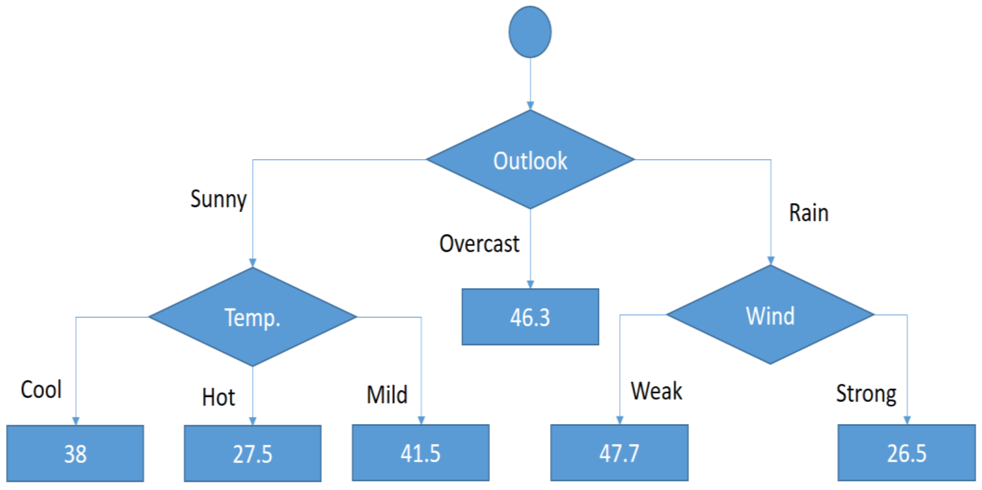
The **outlook attribute** will be at the top of decision tree, because it has the highest score.



Day	Outlook	Temp.	Humidity	Wind	Golf Players
1	Sunny	Hot	High	Weak	25
2	Sunny	Hot	High	Strong	30
8	Sunny	Mild	High	Weak	35
9	Sunny	Cool	Normal	Weak	38
11	Sunny	Mild	Normal	Strong	48

Day	Outlook	Temp.	Humidity	Wind	Golf Players
4	Rain	Mild	High	Weak	45
5	Rain	Cool	Normal	Weak	52
6	Rain	Cool	Normal	Strong	23
10	Rain	Mild	Normal	Weak	46
14	Rain	Mild	High	Strong	30

Day	Outlook	Temp.	Humidity	Wind	Golf Players
3	Overcast	Hot	High	Weak	46
7	Overcast	Cool	Normal	Strong	43
12	Overcast	Mild	High	Strong	52
13	Overcast	Hot	Normal	Weak	44



Final Tree