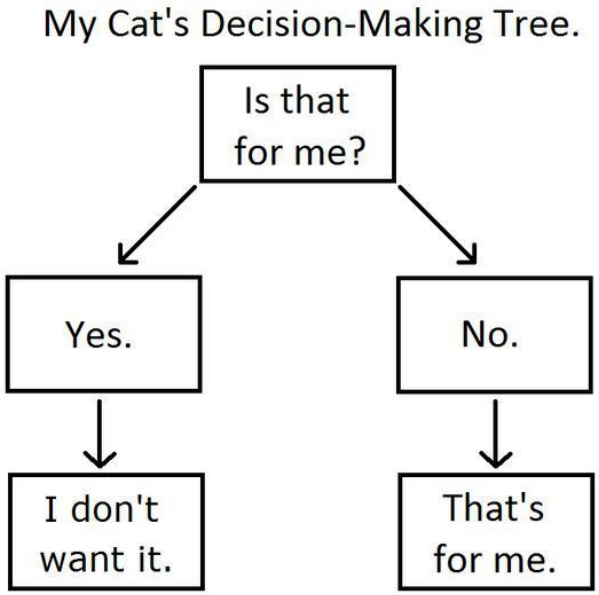
Decision Trees

ECS171 - Siena Saltzen

Trees

Regular Tree

Decision Tree

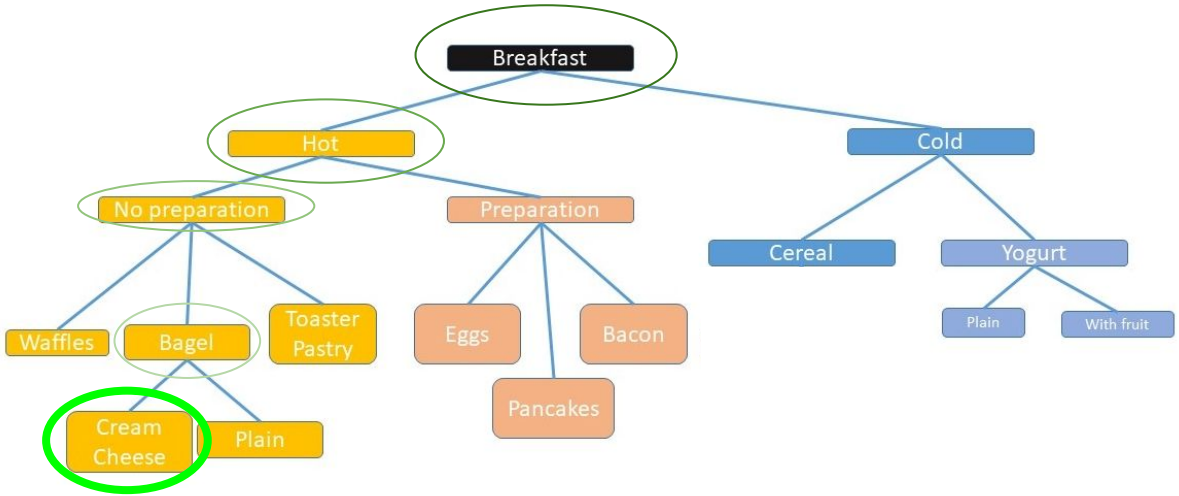


Intuition of the Decision Tree Classifier

For each attribute in the dataset, the Decision-Tree algorithm forms a node. The most important attribute is placed at the root node.

The root node splits on the attribute that “separates” the most data. In this case, hot or cold breakfast items.

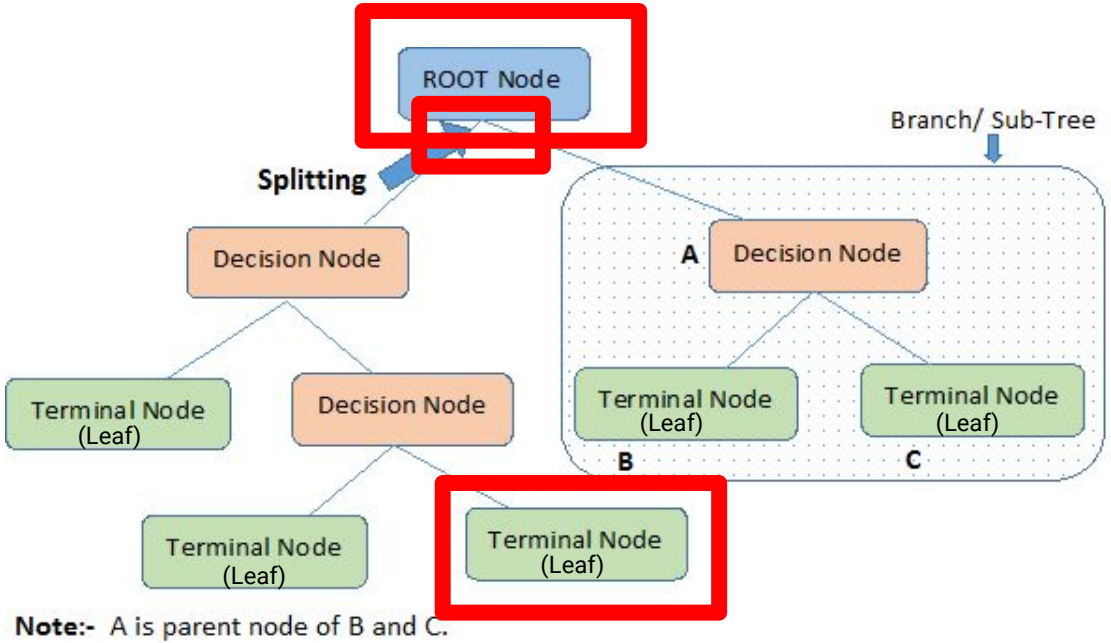
For evaluating the task in hand, we start at the root node and we work our way down the tree by following the corresponding node that meets our condition or decision.



This process continues until a leaf node is reached. It contains the prediction or the outcome of the Decision Tree.

A decision tree is trained on a set of data with labels/targets, which is used to inform the structure of the tree that can then be used for predictions.

Terminology



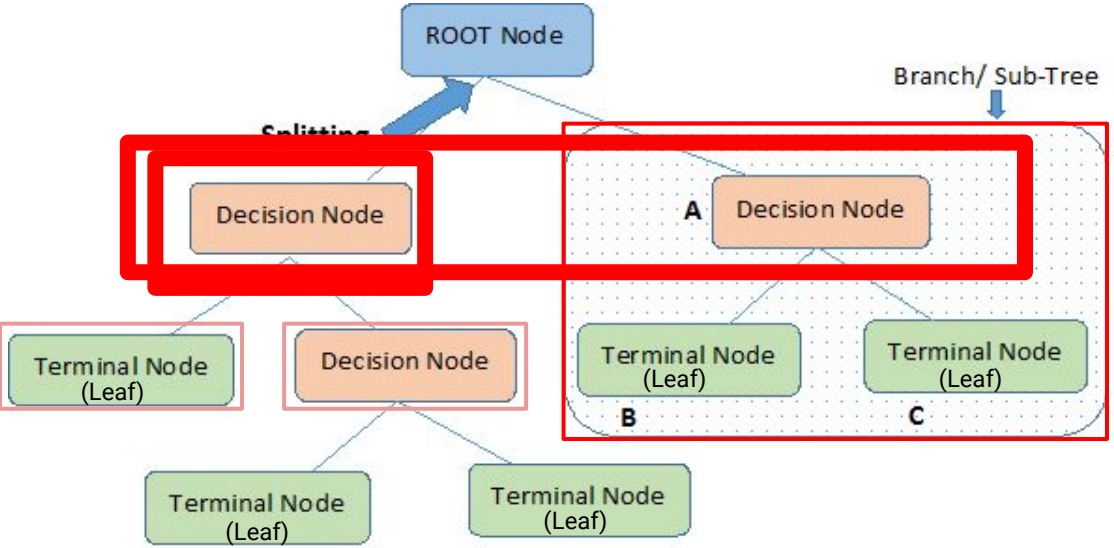
- Root Node
- It represents the entire population or sample. This further gets divided into two or more sets.

- Splitting
- It is a process of dividing a node into two or more sub-nodes.

- Decision Node
- When a sub-node splits into further sub-nodes, then it is called a decision node.

- Leaf/Terminal Node
- Nodes that do not split are called Leaf or Terminal nodes.

Terminology



Note:- A is parent node of B and C.

- Branch/Sub-Tree
  - A sub-section of an entire tree is called a branch or sub-tree.
- Parent and Child Node
  - A node, which is divided into sub-nodes is called the parent node of sub-nodes where sub-nodes are the children of a parent node.
- Level
  - Each successive split is sometimes referred to a level of the tree.
- Pruning
  - When we remove sub-nodes of a decision node, this process is called pruning.

Decision Tree Expressiveness

Discrete-input, discrete-output case: – Decision trees can express any boolean function of the input attributes. – E.g., for Boolean functions, **truth table row → path to leaf:**

(a) $A \wedge \neg B$

(b) $A \vee [B \wedge C]$

(c) A XOR B

(d) $[A \wedge B] \vee [C \wedge D]$

Every Variable in Boolean function such as A, B, C etc. has two possibilities that is True and False

Every Boolean function is either True or False

If the entire Boolean function is True we arrive at True at the leaf (T)

If the entire Boolean function is False we arrive at False at the leaf (F)

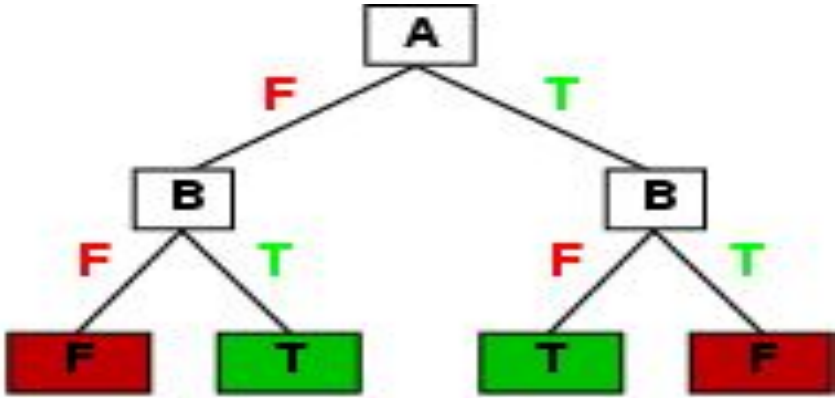
XOR - Only **ONE** Value is True

A	B	A ***XOR*** B
False	False	False
False	True	True
True	False	True
True	True	False

Decision Tree Representations

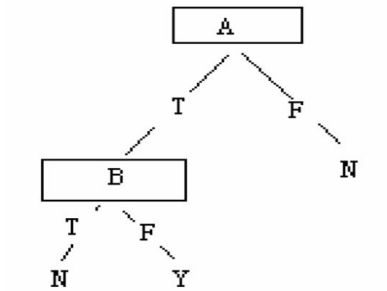
- Decision trees are fully expressive
  - —can represent any Boolean function
  - —Every path in the tree could represent 1 row in the truth table
  - —Yields an exponentially large tree
    - Truth table with 2^d rows, where d is the number of attributes

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F

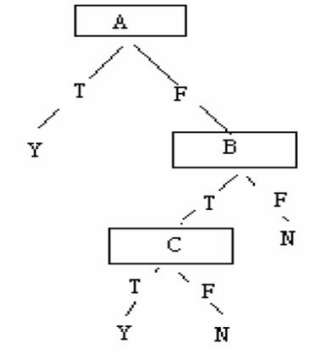


Decision Tree Representations - Examples

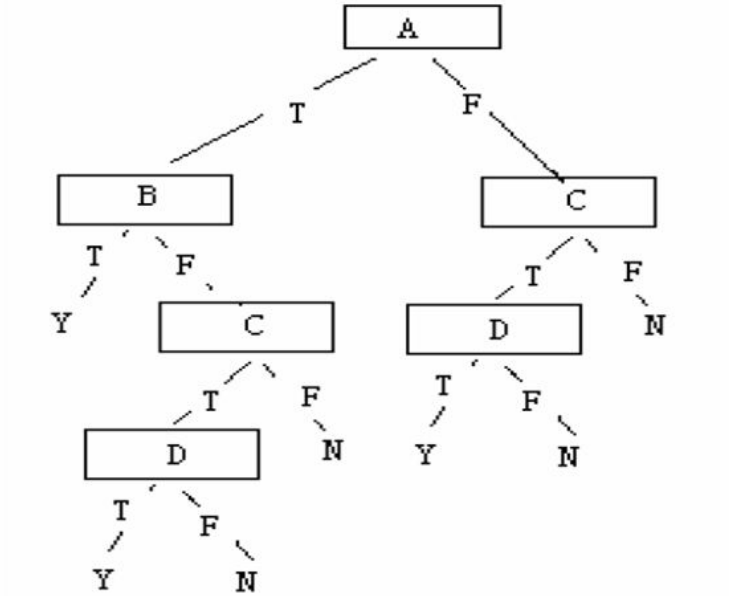
(a) $A \wedge \neg B$



(b) $A \vee [B \wedge C]$



(d) $[A \wedge B] \vee [C \wedge D]$



Pseudocode for Decision tree learning

```
DT(X, Y):
    If (stopping condition*) return decision for this node
    For each possible feature
        For each possible split
            Score the Split
    Pick the feature and split with the best score
    Split the data at that point
    Recurse on each Subset
```

Potential Stopping Conditions:
*# of data < k
*Depth > D
* All Same Label
*Prediction Sufficiently Accurate

*Decision Trees (2) - YouTube - Alex Ihler

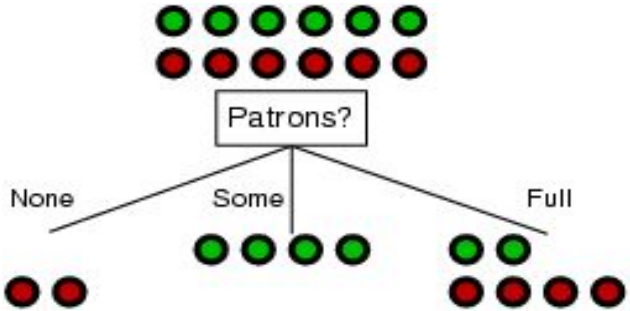
Example Training Data

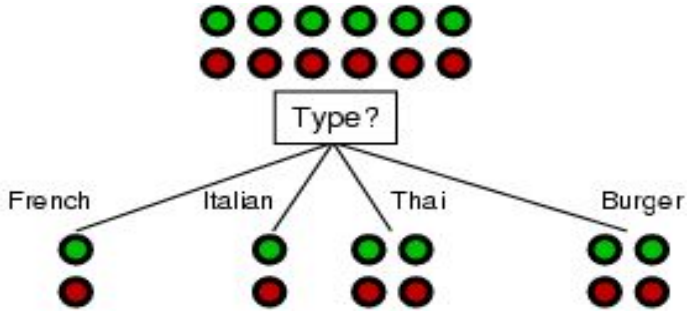
Attributes										Target
Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
T	F	F	T	Some	$$$	F	T	French	0–10	T
T	F	F	T	Full	$	F	F	Thai	30–60	F
F	T	F	F	Some	$	F	F	Burger	0–10	T
T	F	T	T	Full	$	F	F	Thai	10–30	T
T	F	T	F	Full	$$$	F	T	French	>60	F
F	T	F	T	Some	$$	T	T	Italian	0–10	T
F	T	F	F	None	$	T	F	Burger	0–10	F
F	F	F	T	Some	$$	T	T	Thai	0–10	T
F	T	T	F	Full	$	T	F	Burger	>60	F
T	T	T	T	Full	$$$	F	T	Italian	10–30	F
F	F	F	F	None	$	F	F	Thai	0–10	F
T	T	T	T	Full	$	F	F	Burger	30–60	T

- Note an implicit assumption:
  - For any set of attribute values there is a unique target value

Attribute Selection

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"





- *Patrons?* is a better choice
  - How can we quantify this?
  - One approach would be to use the classification error E directly (greedily)
    - Empirically it is found that this works poorly
  - Much better is to use information gain (next slides)

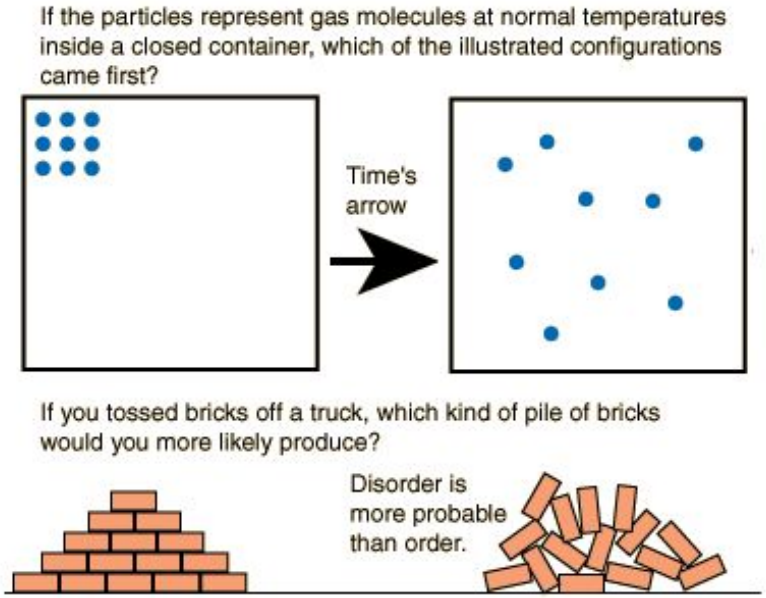
Attribute Selection

The primary challenge in the Decision Tree implementation is to identify the attributes which we consider as the root node and each level. This process is known as the **attributes selection**. There are different attributes selection measure to identify the attribute which can be considered as the root node at each level. The two most common are:

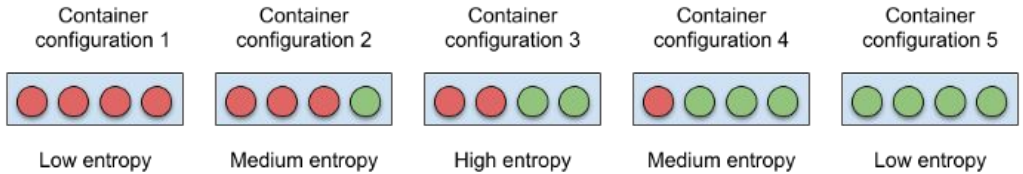
Information Gain & Gini Index

Entropy and Information

- "Entropy" is a measure of randomness
- In chemistry:



Consider you have a container in which you can only place four balls that can be either green or red.



Considering container configuration 1 the **entropy of the entire system is zero as there is no uncertainty associated with the event of extracting a ball** as it will always be red. In the container configuration 2 the entropy of the entire system is med/low as one is likely to extract a red ball from the container. The last important case is container configuration 3 as it provides maximum entropy because both events are equally likely.

https://www.youtube.com/watch?v=ZsY4WcQOrfk

Entropy and Information

- "Entropy" is a measure of randomness
  - How long a message does it take to communicate a result to you?
  - Depends on the probability of the outcomes; more predictable = shorter message

- Communicating fair coin tosses
  - Output: H H T H T T T H H H H T …
  - Sequence takes n bits – each outcome totally unpredictable

- Communicating my daily lottery results
  - Output: 0 0 0 0 0 0 …
  - Most likely to take one bit – I lost every day.
  - Small chance I'll have to send more bits (won & when)

Lost: 0
Won 1: 1(when)0
Won 2: 1(when)1(when)0

- More predictable takes less length to communicate because it's less random
  - Use a few bits for the most likely outcome, more for less likely ones

Entropy Formula

Entropy is a concept that stems from information theory, which measures the impurity of the sample values. It can also be thought of as a measurement of randomness. It is defined with by the following formula, where:

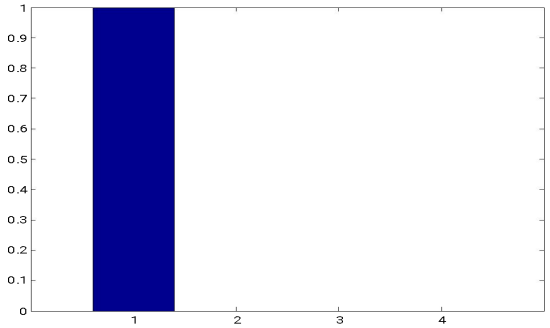
- Entropy $H(X) = E[\log 1/p(X)] = \sum_{x \in X} p(x) \log 1/p(x) = -\sum_{x \in X} p(x) \log p(x)$
  - Log base two, converts units of entropy into “bits”
  - If only two outcomes: H = - p log(p) - (1-p) log(1-p) or

$$Entropy(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

- **X** represents the data set that entropy is calculated
- **x** represents the classes in set, **X**
- **p(x)** represents the proportion of data points that belong to class **x,** to the number of total data points in set, **X** -> the probability of that class

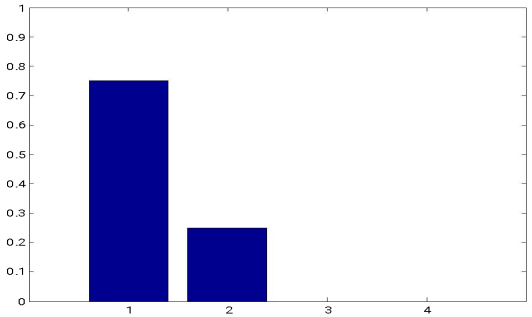
Entropy and Information

- Entropy H(X) = $= -\sum_{x \in X} p(x) \log p(x)$
  - Log base two, units of entropy are “bits”
  - If only two outcomes: H = - p log(p) - (1-p) log(1-p) or $E = -p\log_2(p) - q\log_2(q).$
- Examples:

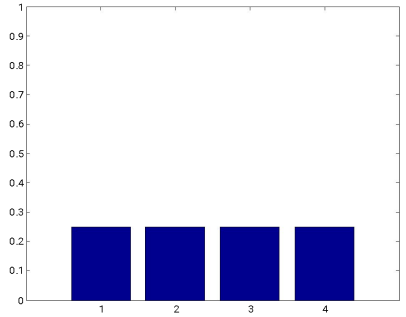


H(x) = 1 log 1
= 0 bits

Min entropy



H(x) = .75 log 4/3 + .25 log 4
= 0.8133 bits



H(x) = .25 log 4 + .25 log 4 +
.25 log 4 + .25 log 4
= log 4 = 2 bits

Max entropy for 4 outcomes

Information Gain

Information gain represents the difference in entropy before and after a split on a given attribute. The attribute with the highest information gain will produce the best split as it's doing the best job at classifying the training data according to its target classification. Information gain is usually represented with the following formula, where:

- ***A*** represents a specific attribute or class label
- ***Entropy(X)*** is the entropy of the dataset at that point, X
- **A/X** represents the proportion of the values in **A** to the number of values in dataset, **X**
- ***Entropy(X|A)*** is the entropy of dataset, **X** given **A - or the entropy after the split.**

$$InformationGain(X, A) = Entropy(X) - \sum_{A \in X} \frac{A}{X} Entropy(X|A)$$

Information Gain

- H(p) = entropy of class distribution at a particular node

- H(p | A) = conditional entropy
  - Weighted average entropy of conditional class distribution
  - Partitioned the data according to the values in A
  - The sum of each partition given the group/class

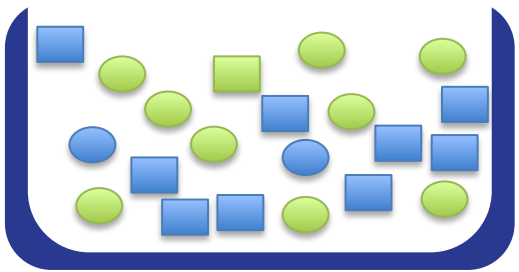
- Gain(A) = H(p) – H(p | A)
- Simple rule in decision tree learning
  - At each internal node, split on the node with the largest information gain (or equivalently, with smallest H(p|A))
- Note that by definition, conditional entropy can't be greater than the entropy

Information Gain - Example

Consider this data set:

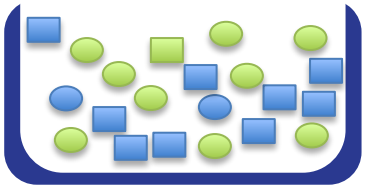
Squares = 10
Circles = 11

Green = 10
Blue = 11

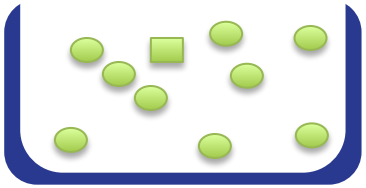


- Gain(A) = H(p) – H(p | A)
- We are going to calculate the Gain of splitting on **COLOR**.
- A = Color
- p = Shape

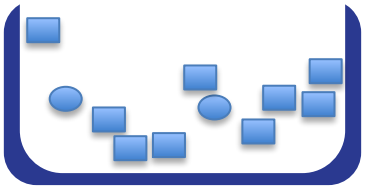
Entropy Example



10 Squares
11 Circles



1 Square
9 Circles



9 Square
2 Circles

Class = color

grn = green; blu = blue; sq = square

- $H(p_{sq}) = -\frac{10}{21}\log_2\frac{10}{21} - \frac{11}{21}\log_2\frac{11}{21}$
- $H(p_{sq}) = 0.998$

- $H(p_{sq}|color) = p_{blu}H(p_{sq})_{blu} + p_{grn}H(p_{sq})_{grn}$
- $H(p_{sq})_{blu} = -\frac{9}{11}\log_2\frac{9}{11} - \frac{2}{11}\log_2\frac{2}{11}$

- $H(p_{sq})_{blu} = 0.684$
- $H(p_{sq})_{grn} = 0.469$
- $Gain(color) = H(p_{sq}) \_ H(p_{sq}|color)$

Weighted average

- $H(p_{sq}|color) = \frac{10}{21} * 0.469 + \frac{11}{21} * 0.684$
- $H(p_{sq}|color) = 0.582$
- $Gain(color) = 0.998 - 0.582$
- $Gain(color) = 0.416$

Information Gain - Let's Put it Together

Day	Outlook	Temp	Humidity	Wind	Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Information Gain

For this dataset, the entropy is 0.94. Since Proportion of days where “Play Tennis” is “Yes”, which is 9/14, and the proportion of days where “Play Tennis” is “No”, which is 5/14. Then, these values can be plugged into the entropy formula.

Entropy (Tennis) = -(9/14) log2(9/14) – (5/14) log2 (5/14) = 0.94

We can then compute the information gain for each of the attributes individually. For example, the information gain for the attribute, “Humidity” would be the following:

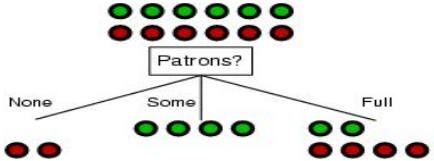
- 0.985 is the entropy when Humidity = “high”

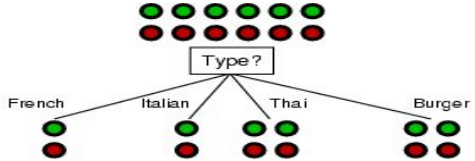
- 0.59 is the entropy when Humidity = “normal”

Gain (Tennis, Humidity) = (0.94)-(7/14)*(0.985) – (7/14)*(0.592) = 0.151

As you can see this is not a very informative split. So we then repeat the calculation for information gain for each attribute in the table. Once finished we **select the attribute with the highest information gain to be the first split point in the decision tree**. In this case, outlook produces the highest information gain. From there, the process is repeated for each subtree.

Root Node Example





For the training set, *6 positives, 6 negatives, H(6/12, 6/12) = 1* bit

positive (p)

negative (1-p)

Consider the attributes *Patrons* and *Type:*

$$IG(\text{Patrons}) = 1 - \left[\frac{2}{12}H(0,1) + \frac{4}{12}H(1,0) + \frac{6}{12}H(\frac{2}{6},\frac{4}{6})\right] = 0.541 \text{ bits}$$

$$IG(\text{Type}) = 1 - \left[\frac{2}{12}H(\frac{1}{2},\frac{1}{2}) + \frac{2}{12}H(\frac{1}{2},\frac{1}{2}) + \frac{4}{12}H(\frac{2}{4},\frac{2}{4}) + \frac{4}{12}H(\frac{2}{4},\frac{2}{4})\right] = 0 \text{ bits}$$

Conclude:

Patrons has the highest IG of all attributes and so is chosen by the learning algorithm as the root

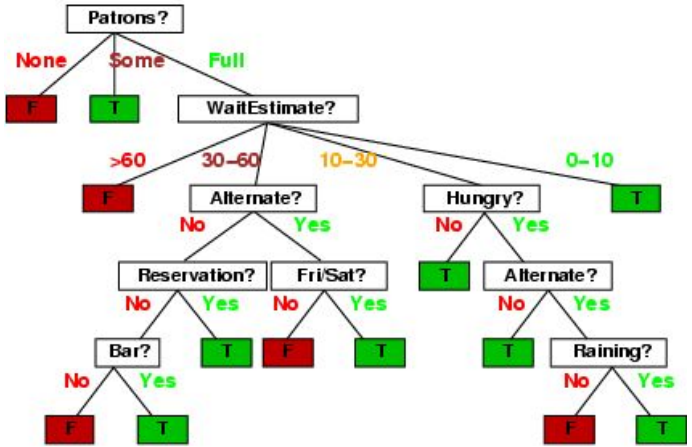
Information gain is then repeatedly applied at internal nodes until all leaves contain only examples from one class or the other

Example Training Data

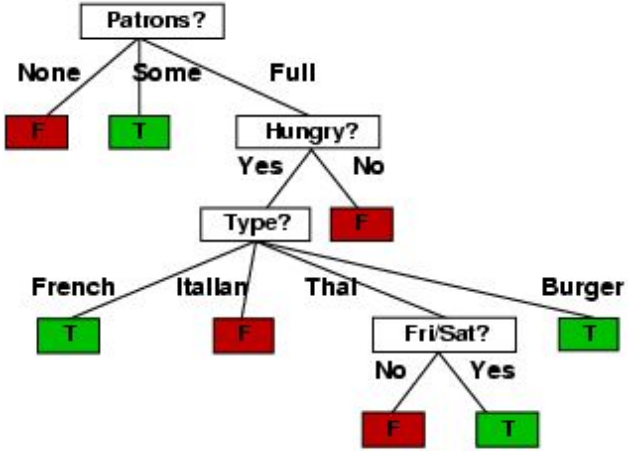
Attributes										Target
Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
T	F	F	T	Some	$$$	F	T	French	0–10	T
T	F	F	T	Full	$	F	F	Thai	30–60	F
F	T	F	F	Some	$	F	F	Burger	0–10	T
T	F	T	T	Full	$	F	F	Thai	10–30	T
T	F	T	F	Full	$$$	F	T	French	>60	F
F	T	F	T	Some	$$	T	T	Italian	0–10	T
F	T	F	F	None	$	T	F	Burger	0–10	F
F	F	F	T	Some	$$	T	T	Thai	0–10	T
F	T	T	F	Full	$	T	F	Burger	>60	F
T	T	T	T	Full	$$$	F	T	Italian	10–30	F
F	F	F	F	None	$	F	F	Thai	0–10	F
T	T	T	T	Full	$	F	F	Burger	30–60	T

Decision Tree Learned

Authors Created



Learned



Gini Impurity

Gini impurity, pronounced genie, is the probability of incorrectly classifying random data point in the dataset if it were labeled based on the class distribution of the dataset. The Gini Impurity of a dataset is a number between 0-0.5 and Similar to entropy, if set, S, is pure—i.e. belonging to one class) then, its impurity is zero. This is denoted by the following formula:

$$\text{Gini Impurity} = 1 - \sum_i (p_i)^2$$

Gini Contini-ued

Consider a dataset D that contains samples from k classes. The probability of samples belonging to class i at a given node can be denoted as p(i). Then the Gini Impurity of D is defined as:

$$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$

The node with uniform class distribution has the highest impurity. The minimum impurity is obtained when all records belong to the same class. Several examples are given in the following table to demonstrate the Gini Impurity computation.

	Count		Probability		Gini Impurity
	X_1	X_2	p_1	p_2	$1 - p_1^2 - p_2^2$
Node A	0	10	0	1	$1 - 0^2 - 1^2 = 0$
Node B	3	7	0.3	0.7	$1 - 0.3^2 - 0.7^2 = 0.42$
Node C	5	5	0.5	0.5	$1 - 0.5^2 - 0.5^2 = 0.5$

Information Gain & Gini Index

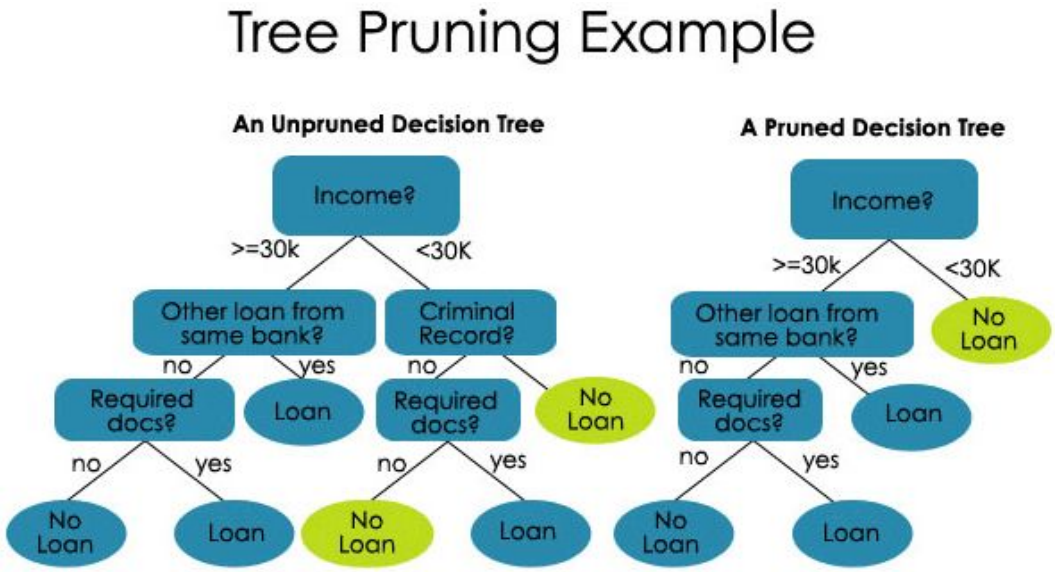
- The Gini Impurity favours bigger partitions (distributions) and is simple to implement, whereas information gains favour smaller partitions (distributions) with a variety of diverse values, necessitating a data and splitting criterion experiment.

- When working with categorical data variables, the Gini Impurity returns either “success” or “failure” and solely does binary splitting; in contrast, information gain evaluates the entropy differences before and after splitting and illustrates impurity in class variables.

- When building trees information gain and gini may produce different splits and therefore different tree structures. It is also possible to have the same decisions points. It depends on the data.

Overfitting

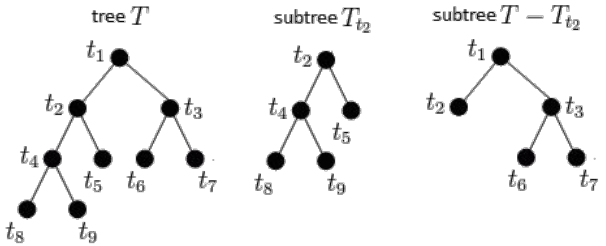
Overfitting is a practical problem while building a Decision-Tree model. The problem of overfitting is considered when the algorithm continues to go deeper and deeper to reduce the training-set error but results with an increased test-set error. So, accuracy of prediction for our model goes down. It generally happens when we build many branches due to outliers and irregularities in data. I tend to visualize decision tree becoming like a giant lookup table as they tend towards overfitting.



Overfitting Cont.

- **Pre-Pruning**
- In pre-pruning, we stop the tree construction a bit early. We prefer not to split a node if its goodness measure is below a threshold value. But it is difficult to choose an appropriate stopping point.
  - # of data < k
  - Depth > D
  - All Same Label
  - Prediction Sufficiently Accurate

- **Post-Pruning**
- In post-pruning, we allow the tree to continue growth. If the tree shows overfitting, then pruning is done after the models creation. In one method, using cross-validation data we test whether expanding a node will result in improve or not. If it shows an improvement, then we can continue by expanding that node. But if it shows a reduction in accuracy then it should not be expanded and should be converted to a leaf node.



Overfitting Cont. Part 2

Cost-complexity pruning

This falls under the post-pruning category. It works on calculating a Tree Score based on the Residual Sum of Squares (RSS) for the subtree, and a Tree Complexity Penalty.

Here, Tree Complexity Penalty is the function of the number of leaves in the subtree. Tree Score is defined as RSS+aT where a is alpha which is a hyperparameter we find using cross-validation, and T is the number of leaves in the subtree. Calculate the Tree Score for all the subtrees in the decision tree and pick the lowest tree score.

Therefore we can also observe from the equation that the value of alpha also determines the choice of the subtree because alpha value uses cross-validation. This process gets repeated until the different values of alpha gives us the sequence of trees. The value of alpha that on average gives the lowest Tree Score is the final value of alpha. Finally, our pruned decision tree will be a tree corresponding to the final value of alpha.

Bagging and Random Forests

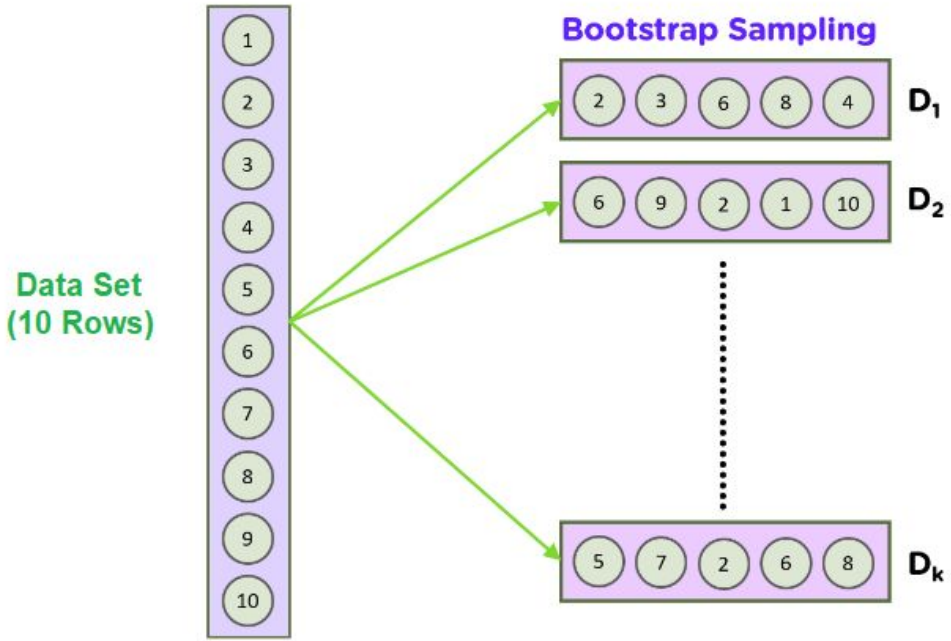
To address with that DT can be sensitive to such as over fitting, model instability, and their sensitivity to imbalance data, ensemble methods are used which improve the accuracy of DTs.

Bootstrap Sampling Approach

Estimate an unknown data distribution, given a limited dataset size.

Bootstrapping works by randomly forming some samples of data from the dataset with replacement.

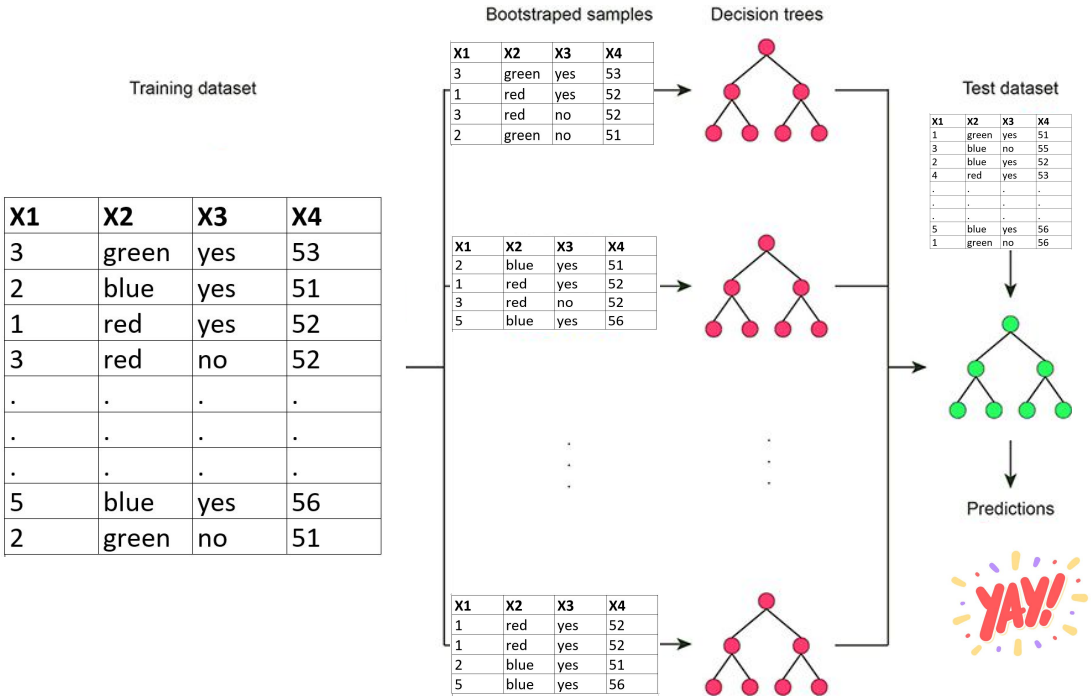
This process is repeated a large number of times (1,000-10,000 times), and by taking the mean or any other statistic during each iteration, we can gain a sense of the data distribution.



Bagging

Bagging can be also called bootstrap aggregation

1. Bootstrap a random training set subsamples
2. For each set, train a Tree
3. Given the test set, make prediction from each model
4. For a classification task, the majority vote across all models for a given class is the final predicted class.



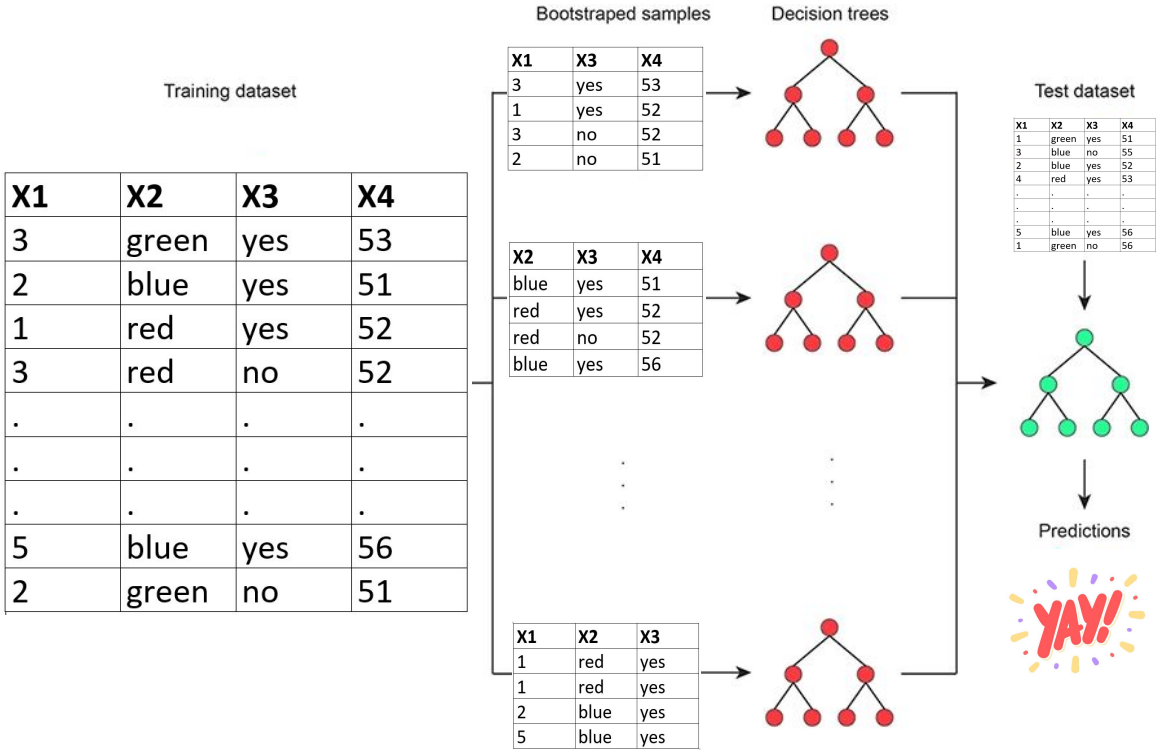
X1	X2	X3	X4
3	green	yes	53
2	blue	yes	51
1	red	yes	52
3	red	no	52
.	.	.	.
.	.	.	.
.	.	.	.
5	blue	yes	56
2	green	no	51

This procedure stabilizes the final model answer because it chooses paths that are shown in a majority of trees.

Random Forest

To address the issue with high variance features or collinearity, a better method would be to **omit highly variable features** or to remove some correlated features periodically during the bagging process. This would reveal important tree structures that are typically hidden by these features.

Random forests address this issue by making one minor tweak to the bagging process: instead of using all features while training each tree, a random and smaller subset of features is used. This eventually removes problematic features from the data, and hidden tree structures can begin to emerge after repeating this process many times.



X1	X2	X3	X4
3	green	yes	53
2	blue	yes	51
1	red	yes	52
3	red	no	52
.	.	.	.
.	.	.	.
.	.	.	.
5	blue	yes	56
2	green	no	51

X1	X3	X4
3	yes	53
1	yes	52
3	no	52
2	no	51

X2	X3	X4
blue	yes	51
red	yes	52
red	no	52
blue	yes	56

X1	X2	X3
1	red	yes
1	red	yes
2	blue	yes
5	blue	yes

X1	X2	X3	X4
1	green	yes	51
3	blue	no	55
2	blue	yes	52
4	red	yes	53
.	.	.	.
.	.	.	.
.	.	.	.
5	blue	yes	56
1	green	no	56

When are Decision Trees Good?

Advantages

- **Easy to interpret**
  - The hierarchical nature of a decision tree also makes it easy to see which attributes are most important, which isn't always clear with other algorithms, like neural networks.

- **Little to no data preparation required:**
  - It can handle various data types—i.e. discrete or continuous values, and continuous values can be converted into categorical values through the use of thresholds.

- **More flexible:**
  - Decision trees can be leveraged for both classification and regression tasks. It's also insensitive to underlying relationships between attributes; this means that if two variables are highly correlated, the algorithm will only choose one of the features to split on.

Disadvantage

- **Prone to overfitting:**
  - Complex decision trees tend to overfit and do not generalize well to new data.

- **High variance estimators:**
  - Small variations within data can produce a very different decision tree. Bagging, or the averaging of estimates, can be a method of reducing variance of decision trees.

- **More costly:**
  - Given that decision trees take a greedy search approach during construction, they can be more expensive to train compared to other algorithms.

Decision Tree Learning - Summary

- Find the smallest decision tree consistent that classifies the data
- Big Idea:
  - Select root node that is “best” in some sense
    - Gini or Info Gain
  - Partition data into subsets, depending on root attribute value
  - Recursively grow subtrees
  - Different termination criteria
    - Eg, if all examples at a node have the same label then declare it a leaf and backup
      - Or a depth-bound on the tree (or go to max depth) and use majority vote
- We have talked about binary variables up until now, but we can trivially extend to multi-valued variables and also used for regression

Sources

Pruning in Decision Tree (programsbuzz.com)

Introduction to Bootstrapping in Statistics with an Example - Statistics By Jim

Decision Tree Pruning Techniques In Python (cloudyml.com)

Cost-Complexity Pruning - ML Wiki

What is a Decision Tree | IBM

Decision-Tree Classifier Tutorial | Kaggle

chap_18_IntroLearning.pptx - Google Slides

Bagging and Random Forest in Machine Learning (learnetutorials.com)

Decision Tree for Boolean Functions Machine Learning - VTUPulse

Time for Some Code!

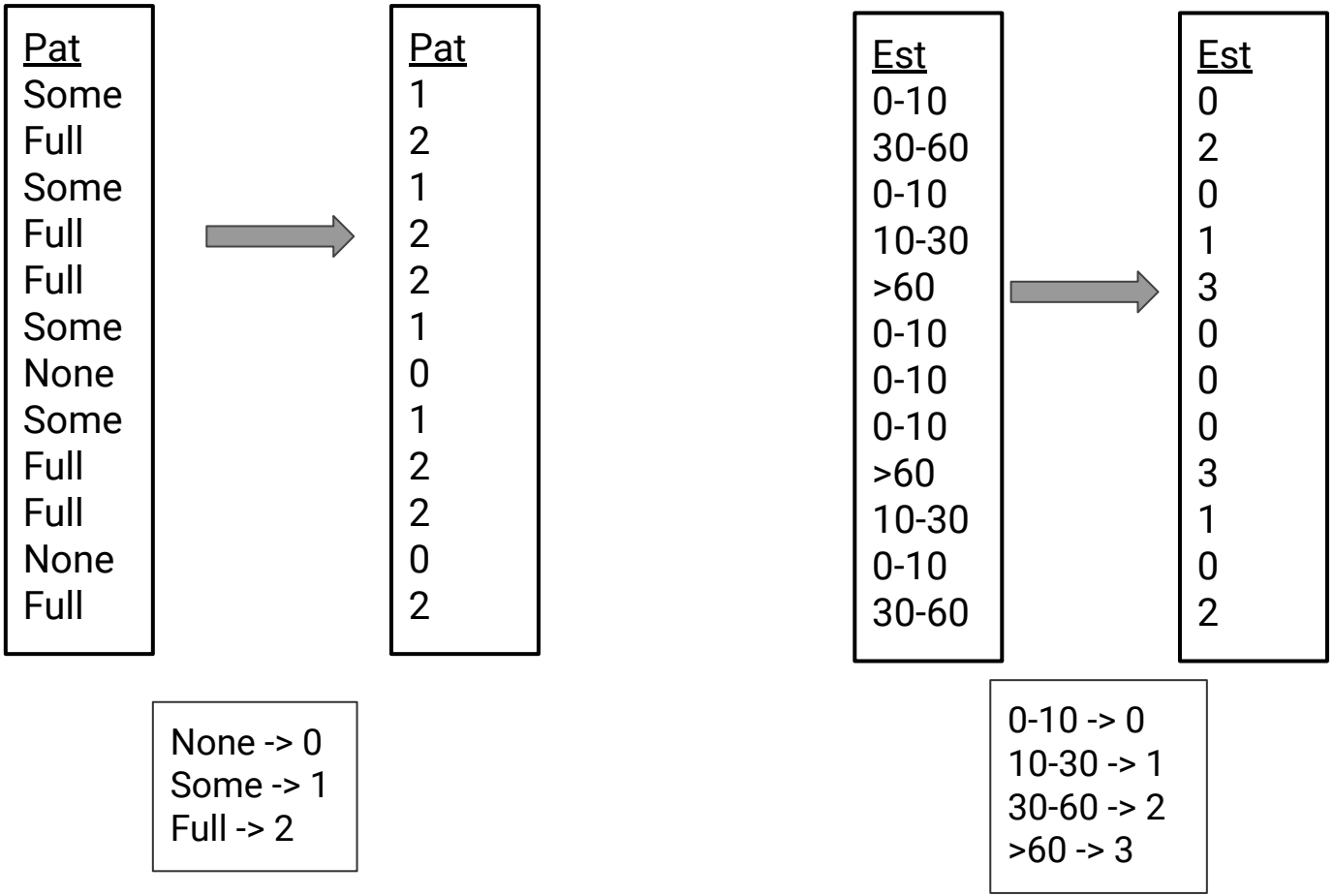
Fortunately for everyone, SciKit has a built in Decision Tree Implementation!

Unfortunately for everyone, we are going to build one from scratch.

Example Training Data

Attributes										Target
Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
T	F	F	T	Some	$$$	F	T	French	0–10	T
T	F	F	T	Full	$	F	F	Thai	30–60	F
F	T	F	F	Some	$	F	F	Burger	0–10	T
T	F	T	T	Full	$	F	F	Thai	10–30	T
T	F	T	F	Full	$$$	F	T	French	>60	F
F	T	F	T	Some	$$	T	T	Italian	0–10	T
F	T	F	F	None	$	T	F	Burger	0–10	F
F	F	F	T	Some	$$	T	T	Thai	0–10	T
F	T	T	F	Full	$	T	F	Burger	>60	F
T	T	T	T	Full	$$$	F	T	Italian	10–30	F
F	F	F	F	None	$	F	F	Thai	0–10	F
T	T	T	T	Full	$	F	F	Burger	30–60	T

Encoding


Pat
Some
Full
Some
Full
Full
Some
None
Some
Full
Full
None
Full
Pat
1
2
1
2
2
1
0
1
2
2
0
2
None -> 0
Some -> 1
Full -> 2
Est
0-10
30-60
0-10
10-30
>60
0-10
0-10
0-10
>60
10-30
0-10
30-60
Est
0
2
0
1
3
0
0
0
3
1
0
2
0-10 -> 0
10-30 -> 1
30-60 -> 2
>60 -> 3