Implications of PAC Learnability

- "Nothing is more practical than a good theory" [1]
- Can we conceptually/mathematically quantify learning?
- Main theory: PAC Learning
- The world of all possible classifiers over a dataset:
  - Discrete data
  - Induced classifier's error rate
  - Expected accuracy defined by user
- Continuous data?
  - VC-dimension

Probably Approximately Correct (PAC) Learning

- Assumptions:
  - Use terms "example" and "sample" indistinctly
  - Use terms "attribute" and "feature" indistinctly
  - $m =$ number of samples
  - n = number of attributes
  - The data is noise-free
    - Implies that ∃ at least one classifier that correctly labels all the samples
  - 2 classes: positive and negative
  - Classifier is a Boolean function that outputs
    - true for positive samples and
    - false for negative samples
  - Each logical expression is a hypothesis about the classifier
  - All hypotheses form a hypothesis space H, whose size is $|H|$
  - For any dataset with a finite number of samples, $|H|$ is a finite number

PAC Learning

- First bound…
- Imperfect classifier:
  - The one that correctly classifies all training samples but may err in the future
- Key question:
  - How many training samples do we need for a chance of success in the future?
- Consider:
  - The error rate of a hypothetical classifier on the entire H is greater than ϵ
- Or equivalently:
  - The probability that this classifier will correctly classify a random sample is less than $1 - \epsilon$
- Then,
  - The probability P that the imperfect classifier will correctly classify m random samples is bounded by:

$$P \leq (1 - \epsilon)^m \qquad (1)$$

Interpretation:

- With prob P, an entire dataset of m samples will be correctly classified by a classifier whose error rate exceeds ϵ

Examples

- $\epsilon = 0.1$ and $m = 10 \Rightarrow P < 0.348$
- $\epsilon = 0.1$ and $m = 20 \Rightarrow P < 0.122$
- …
- $\epsilon = 0.1$ and $m = 100 \Rightarrow P < 10^{-4}$
- $\epsilon = 0.1$ and $m = 1{,}000 \Rightarrow P < 10^{-45}$
- $\epsilon = 0.1$ and $m = 10{,}000 \Rightarrow P < 10^{-458}$

Drop the error:

- $\epsilon = 0.05$ and $m = 100 \Rightarrow P < 0.0059$

PAC Learning

- Second bound…
- Eliminate poor classifiers:
  - Those classifiers that are error free on the training dataset
  - But their error on the entire space exceeds ϵ
  - Say, there are k such classifiers
- It's impossible to count all of them!
- But we can say $k < |H|$
- Rewriting the bound (1):

Prob that at least one of the k classifiers is error free on m samples:

$$P \leq k(1-\epsilon)^m \leq |H|(1-\epsilon)^m \quad (2)$$

Or equivalently

$$P \leq |H|e^{-m\epsilon} \quad (3)$$

- The user may set an error as follows:

$$|H|e^{-m\epsilon} \leq \delta \quad (4)$$

- Taking log on both sides and rearranging:

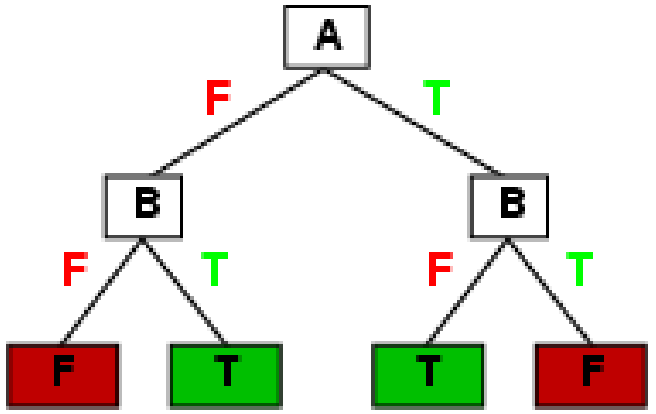
$$m > \frac{1}{\epsilon}\left(\ln|H| + \ln\frac{1}{\delta}\right) \quad (5)$$

- which means
  - How many training samples m needed
  - If with prob at least δ
  - A classifier with error rate lower than ϵ is derived
- Note:
  - m grows linearly in $1/\epsilon$
  - and grows logarithmically in $1/\delta$
- Bound has theoretical importance
- Practically, it would not really tell how many samples
- If $k = |H|$, it would be impractical

PAC Learning and Decision Trees

- Decision trees can express any function of the input attributes
- E.g., for Boolean functions, truth table row → path to leaf:

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



- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless *f* nondeterministic in *x*)
- But it probably won't generalize to new examples
- Prefer to find more compact decision trees

Hypothesis Space

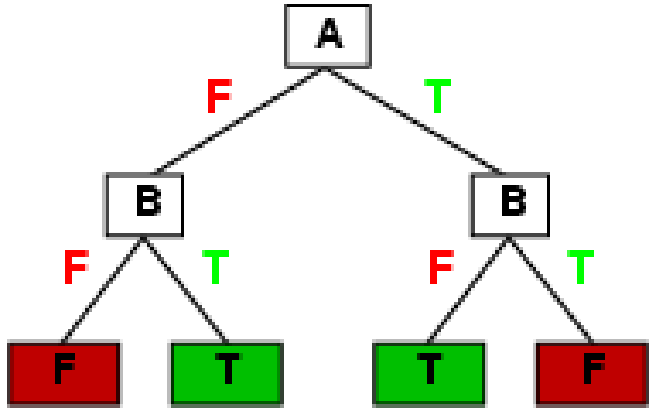
How many distinct decision trees with *n* Boolean attributes?

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

- E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

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



Hypothesis Space

PAC learnability: Any Boolean function

How many distinct decision trees with *n* Boolean attributes?

= number of Boolean functions

= number of distinct truth tables with 2^n rows

= 2^{2^n}

- E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

- Since $\ln|H| = \ln 2^{2^n} = 2^n \ln 2$

$$m > \frac{1}{\epsilon}\left(2^n \ln 2 + \ln\frac{1}{\delta}\right)$$

where the number of samples m grows exponentially with the number of attributes n

PAC learnability: Conjunction of Boolean attributes

Ex: att1 = true AND att3 = false

Size of the hypothesis space?

- Each attribute can be in (positive), in (negative), or out

  $\Rightarrow |H| = 3^n$ distinct conjunctive hypotheses

- Since $|H| = 3^n$, we have $\ln|H| = \ln 3^n = n \ln 3$
- The bound of Eq. (5) results in:

$$m > \frac{1}{\epsilon}\left(n \ln 3 + \ln\frac{1}{\delta}\right)$$

Computational Learning Theory

Occam's Razor:

- If there exist two hypotheses, both consistent with the same outcome, choose the shortest one
- That is if using conjunction of Boolean attributes, 2 cases:
  - Some attributes can be absent:
    $\ln |H| = n \ln 3$
  - All attributes are present:
    $\ln |H| = n \ln 2$ (which is smaller)

Irrelevant and redundant attributes

- Corresponds to the first case
  - Some attributes can be absent:
    $\ln |H| = n \ln 3$

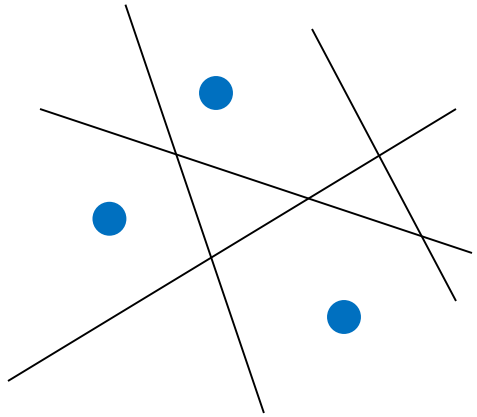
VC Dimension

- PAC learnability is for discrete attributes
- We now consider continuous attributes

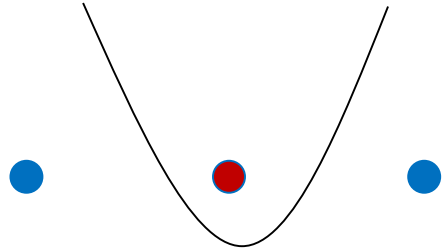
Shattered training set:

- Given a specific classifier g, e.g., linear
  and a training set D
- For any labeling of D, g will always be able to correctly classify all samples

Always classified (shattered) by a linear function



Not classified (shattered) by a linear function



Classified by a parabola
2nd degree polynomial

VC Dimension

- Called Vapnik-Chervonenkis Dimension or VC-dimension

Definition

- The VC-dimension of a given class of classifiers is the size of the largest set of examples shattered by that class
- In example of 2D space: $VC_L = 3$
- VC-dimension makes it possible to deal with continuous features
- Some VC-dimensions in $\mathbb{R}^n$:

Hypothesis class	VC-dimension
Hyperplane	$n+1$
Hypersphere	$n+2$
Quadratic	$\frac{(n+1)(n+2)}{2}$
r-order polynomial	$\binom{n+r}{r}$

- If VC-dimension d of classifier class H is finite
- Then, the error rate below ϵ can be achieved
- with confidence $1-\delta$ if
  - the target class is identical with some hypothesis $h \in H$
  - and the number of training samples m satisfies:

$$m \geq \max\left(\frac{4}{\epsilon}\log\frac{2}{\delta}, \frac{8d}{\epsilon}\log\frac{13}{\epsilon}\right)$$

The number of examples grows linearly in the VC-dimension

But very fast with the number of attributes

Example:

2nd order polynomial in 100 dimensions

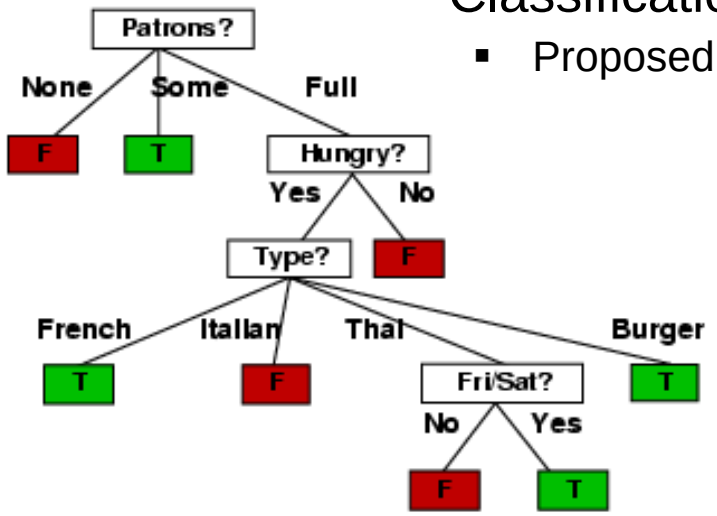
$$d = \frac{102 \cdot 101}{2 \cdot 1} = 5{,}050$$

4th order polynomial in 100 dimensions

$$d = \frac{104 \cdot 103 \cdot 102 \cdot 101}{4 \cdot 3 \cdot 2 \cdot 1} = 4{,}598{,}126$$

Decision trees

- A decision tree (DT) is a graph representation of classification rules
- Each internal node represents a decision
  - Binary: if then else
  - or more than two choices
- External nodes correspond to outcomes (classes)
- Decision trees are well-suited for nominal features (attributes)
- Samples are now “examples”
- Real-valued features can be used but have to be discretized though



Different DT learning algorithms:

- Iterative dichomotizer (ID3): Initially proposed by R. Quinlan in 1986. Works only on categorical features
- C4.5: Quinlan’s algorithm extended to discrete sets of intervals.
  - Can deal with numerical features
  - It’s the most popular DT algorithm
  - Called J48 in Weka
  - We focus on this algorithm
  - C5.0 is more recent but proprietary
- Classification and regression Trees (CART):
  - Proposed by Breiman et al in 1984

Decision trees

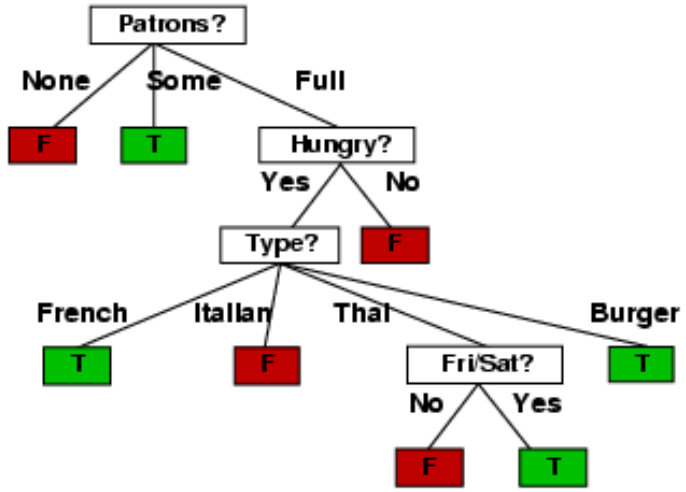
Decision trees can be used for:

- Classification (two or more classes)
- Multi-output: e.g., for face recognition/reconstruction
- Regression: e.g., prediction in time series models, stock markets, etc.

- We focus on classification

Terminology used in this chapter:

- attribute = feature = variable
- outcome = class
- example = sample



Decision trees - example

Problem: decide whether to wait for a table at a restaurant, based on the following attributes (aka features):

1. Alternative: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Attribute-based representations

- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

class

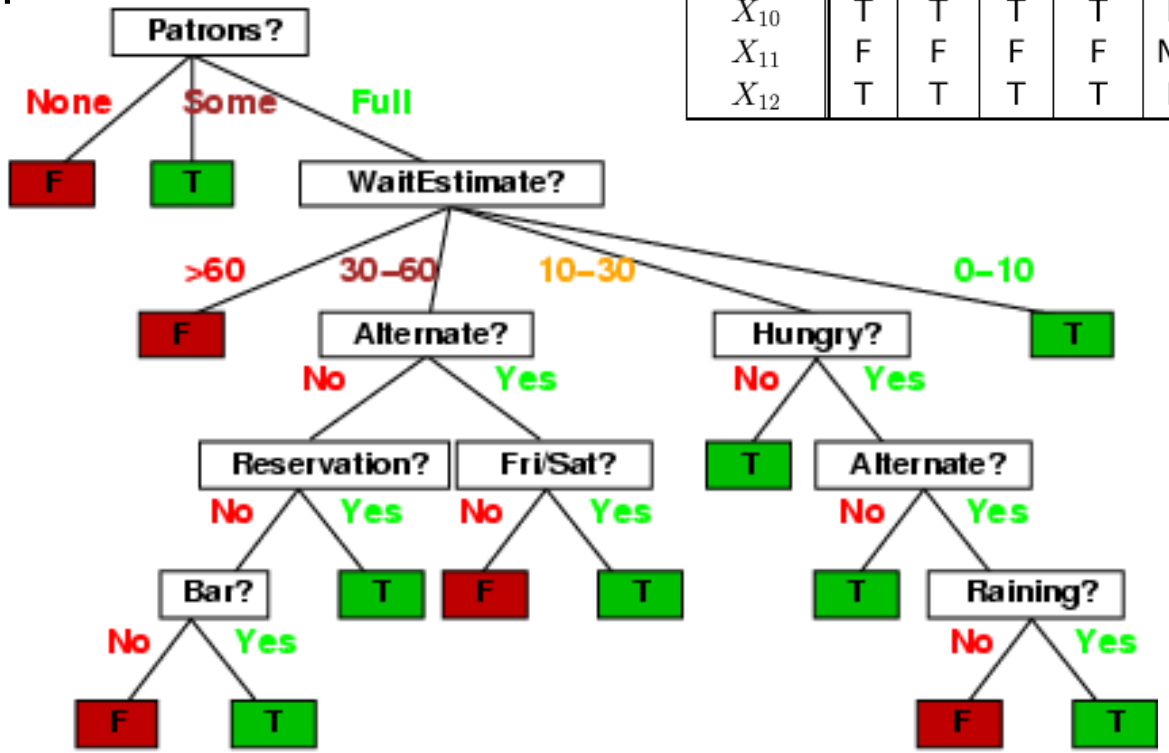
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Classification of examples is positive (T) or negative (F)

Decision trees

- One possible representation for hypotheses
- E.g., here is the “true” tree for deciding whether to wait:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T



Decision tree learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose “most significant” attribute as root of (sub)tree

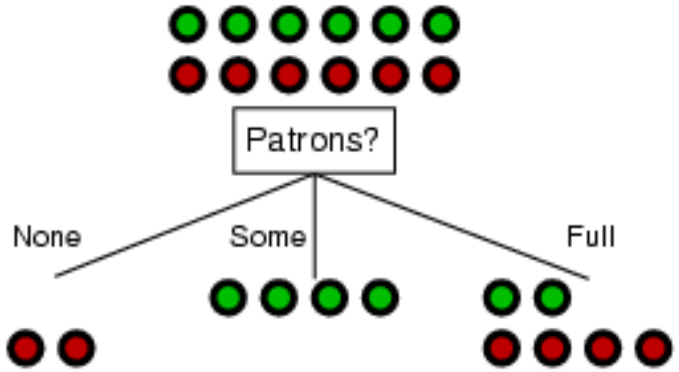
```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

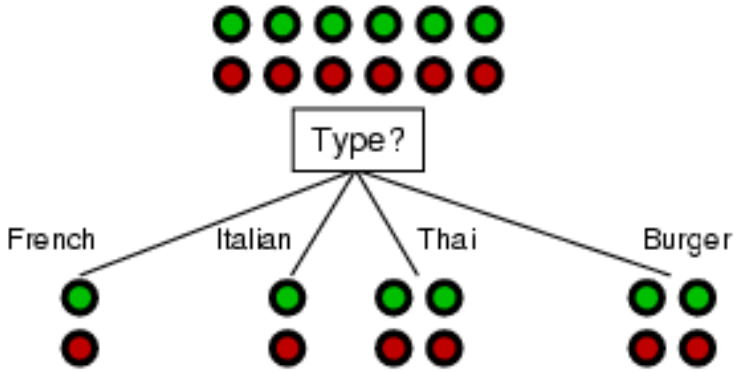
- The DTL algorithm always outputs a decision tree that is consistent with the training examples

Choosing an attribute

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

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T





- *Patrons?* is a better choice

Using information theory

- To implement `Choose-Attribute` in the DTL algorithm
- Information Content (Entropy):

$$I\big(P(v_1), \ldots, P(v_n)\big) = \sum_{i=1}^{n} -P(v_i)\log_2 P(v_i)$$

- For a training set containing *p* positive examples and *n* negative examples:

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Information gain

- A chosen attribute A divides the training set E into subsets $E_1, \ldots, E_v$ according to their values for A, where A has v distinct values

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

p_i and n_i are the number of positives and negatives for the i^{th} value

- Information Gain (IG) or reduction in entropy from the attribute test:

$$IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - remainder(A)$$

- Choose the attribute with the largest IG

Information gain

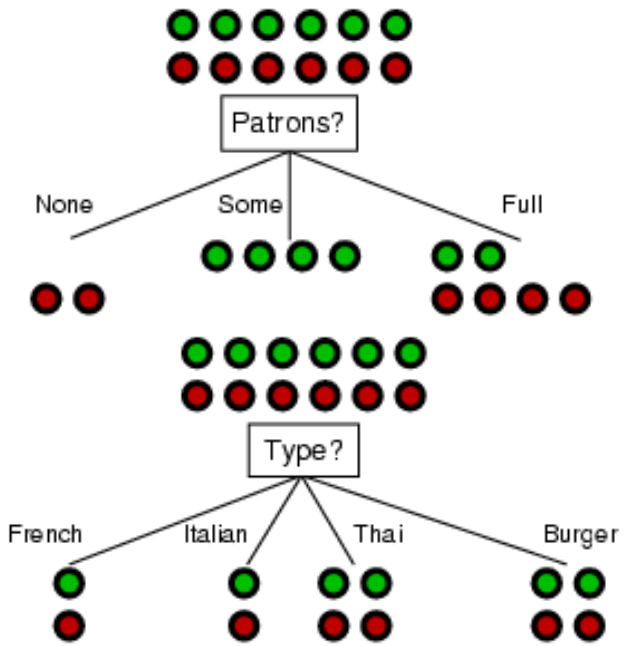
For the training set, $p = n = 6$, *I(6/12, 6/12) = 1* bit

Consider the attributes *Patrons* and *Type* (and others too):

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

$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = .0541 \text{ bits}$$

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



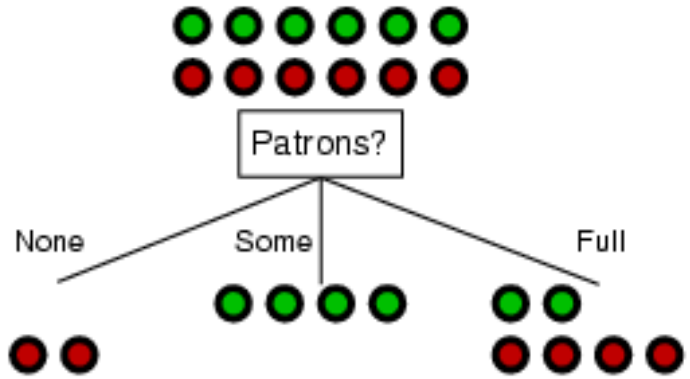
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Next: Choose Patrons? and continue in the same way for each subtree.

Information gain – next step

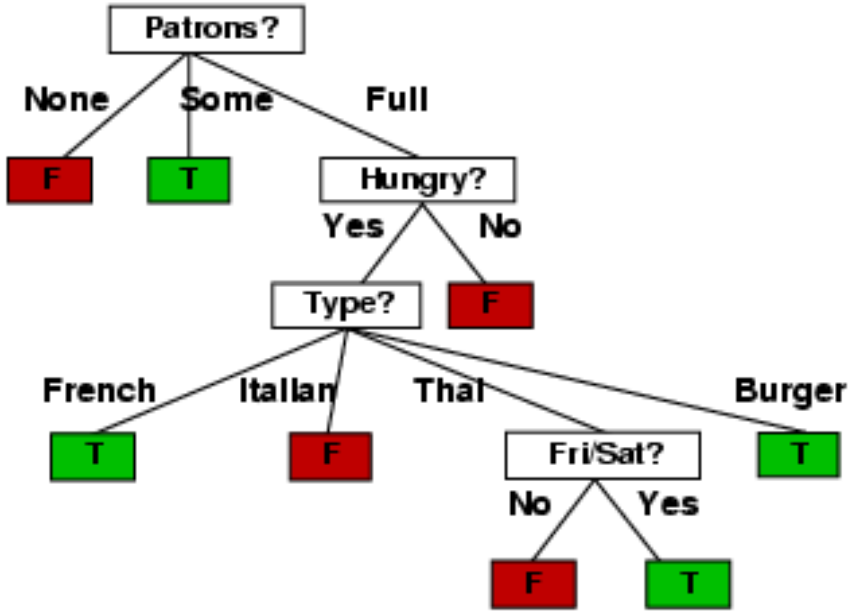
Next: Choose Patrons? and continue in the same way for each subtree.

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T



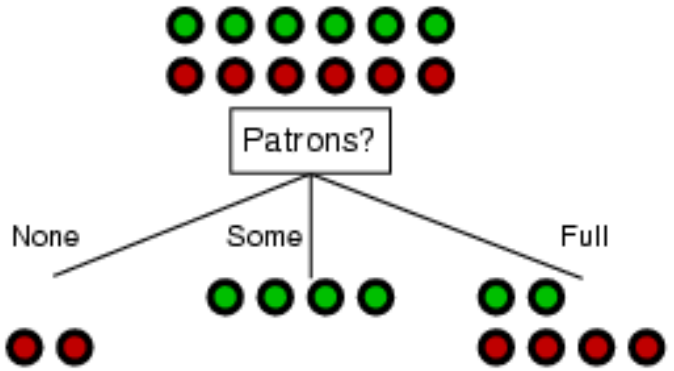
Example contd.

- Decision tree learned from the 12 examples:



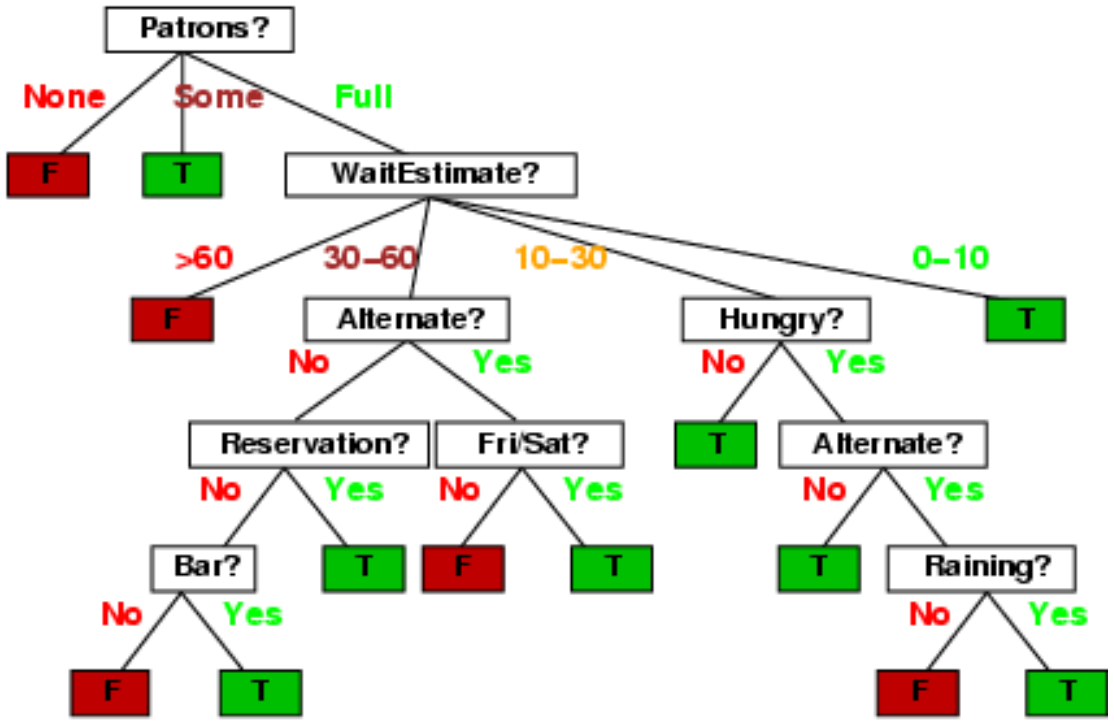
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Substantially simpler than “true” tree… a more complex hypothesis isn’t justified by a small amount of data

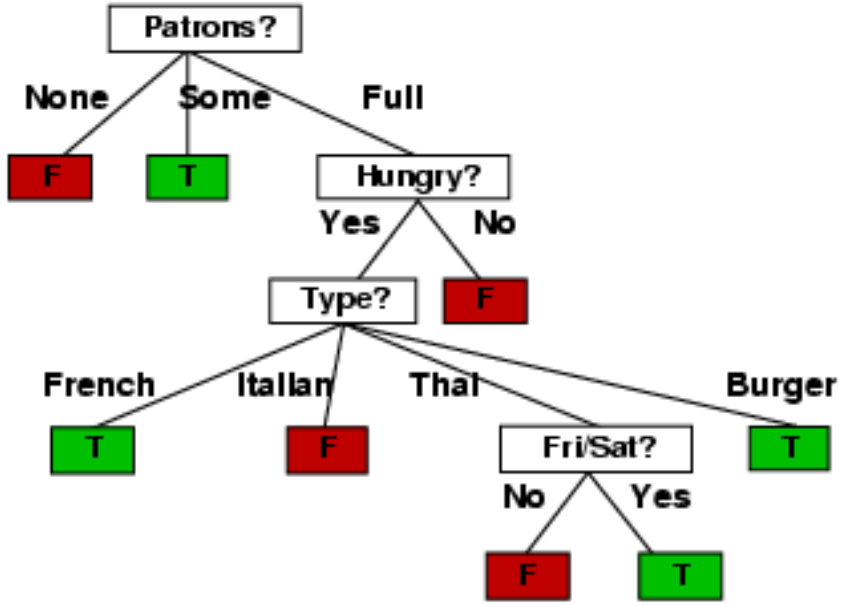


Comparison of Decision Trees

Original:



Learned from IG:



Non-nominal Attributes – Multiple classes

- Attributes can be nominal, discrete, binary, alphanumeric, real, integers, etc.
- Nominal, discrete and binary are easy to deal with (see previous examples)
- Problems arise when dealing with numeric attributes such as integer or real
- Example: wait time of previous example
- Even more complex is the case of > 2 classes
- Solution: divide space into rectangular (hypercubic) regions using thresholds

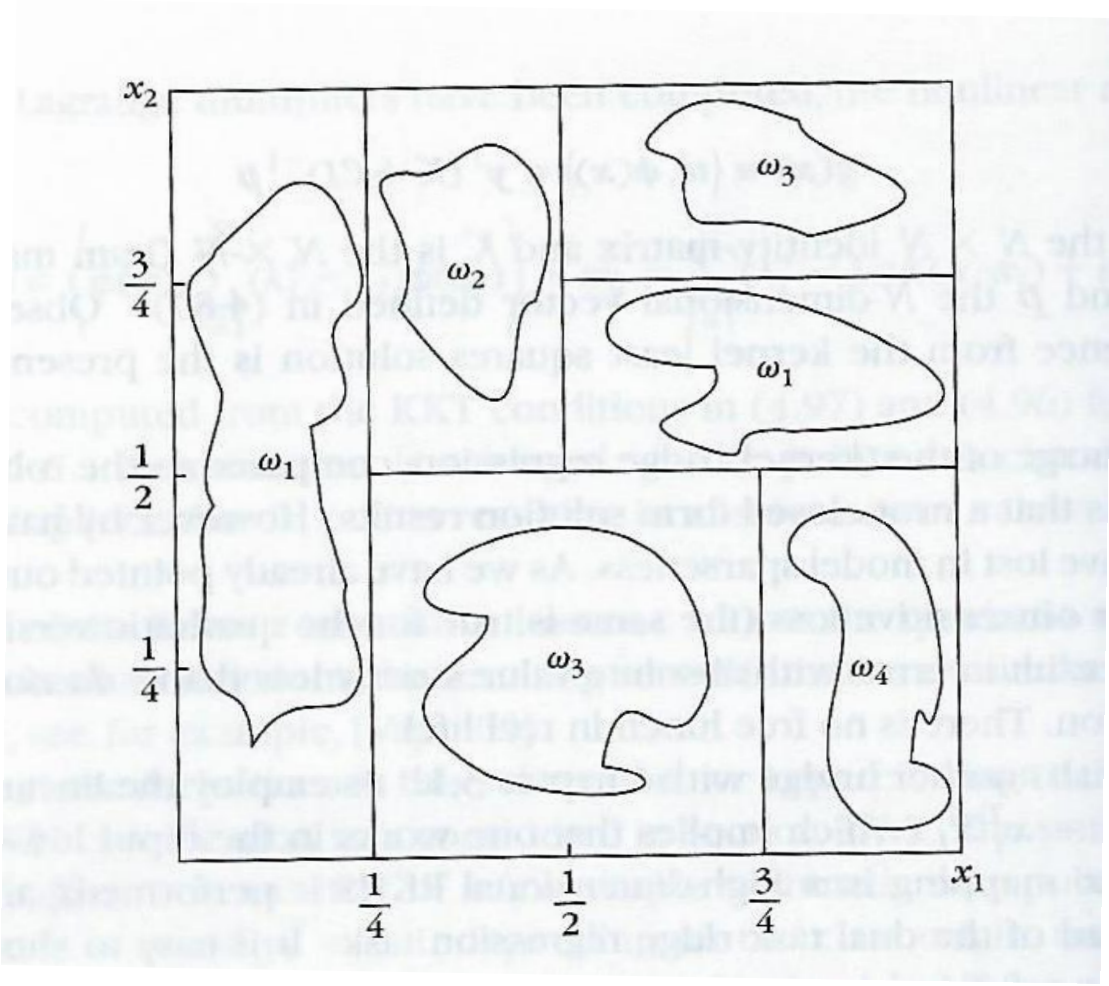
Example:

- Two real variables x_1 and x_2
- 4 classes: ω_1, ω_2, ω_3, ω_4
- Thresholds are used for binary splits of the 2D space
- Internal nodes in DT compare a variable against a threshold
- Order of nodes/attributes to be tested depend on compactness of the tree (e.g., based on Information Gain)

Example – cont'd

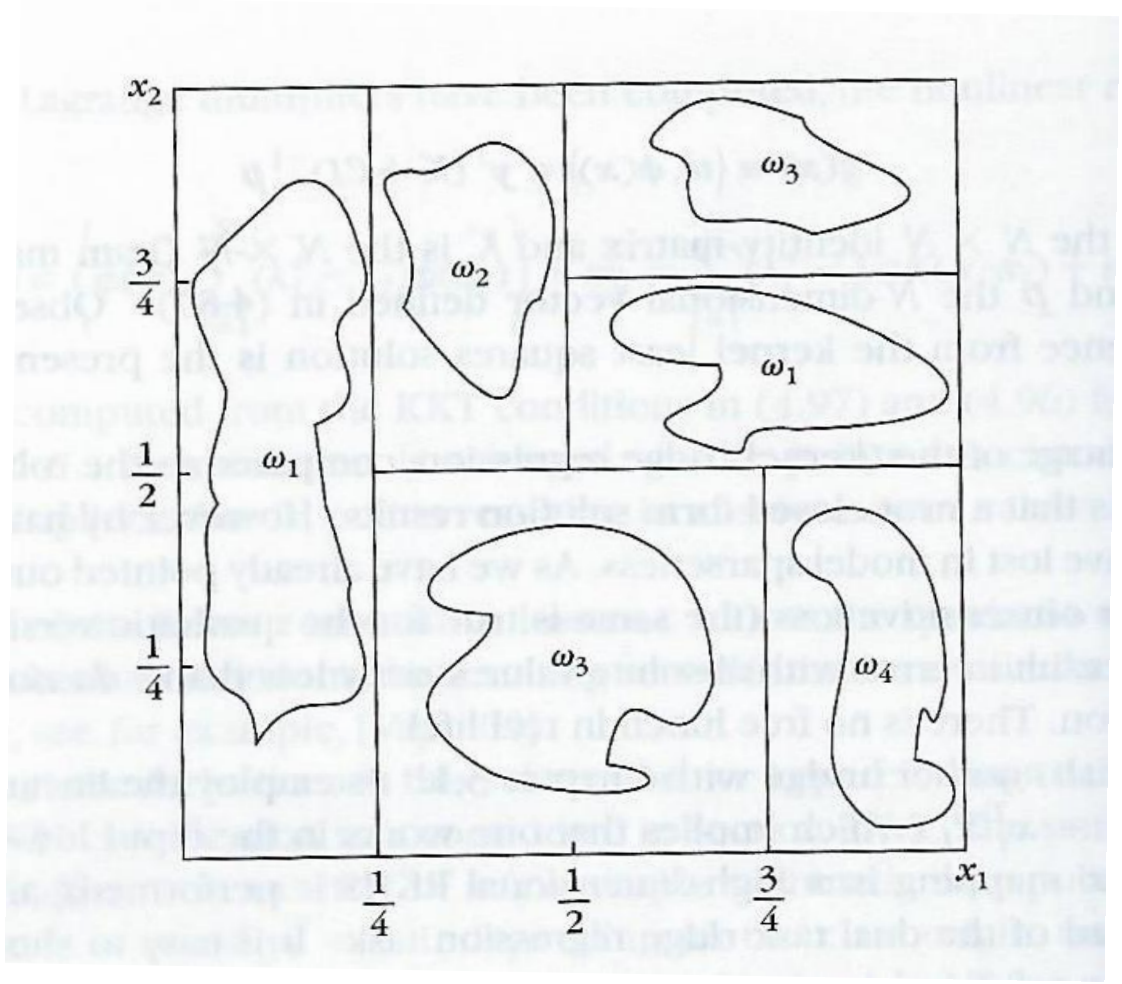
- Consider 4 classes: ω_1, ω_2, ω_3, ω_4
- 2 features: x_1 and x_2
- Features are real numbers in [0,1]
- One-against-all yields nonlinearly separable problems
- Space is partitioned via hyperplanes parallel to the axes

Partition of the space

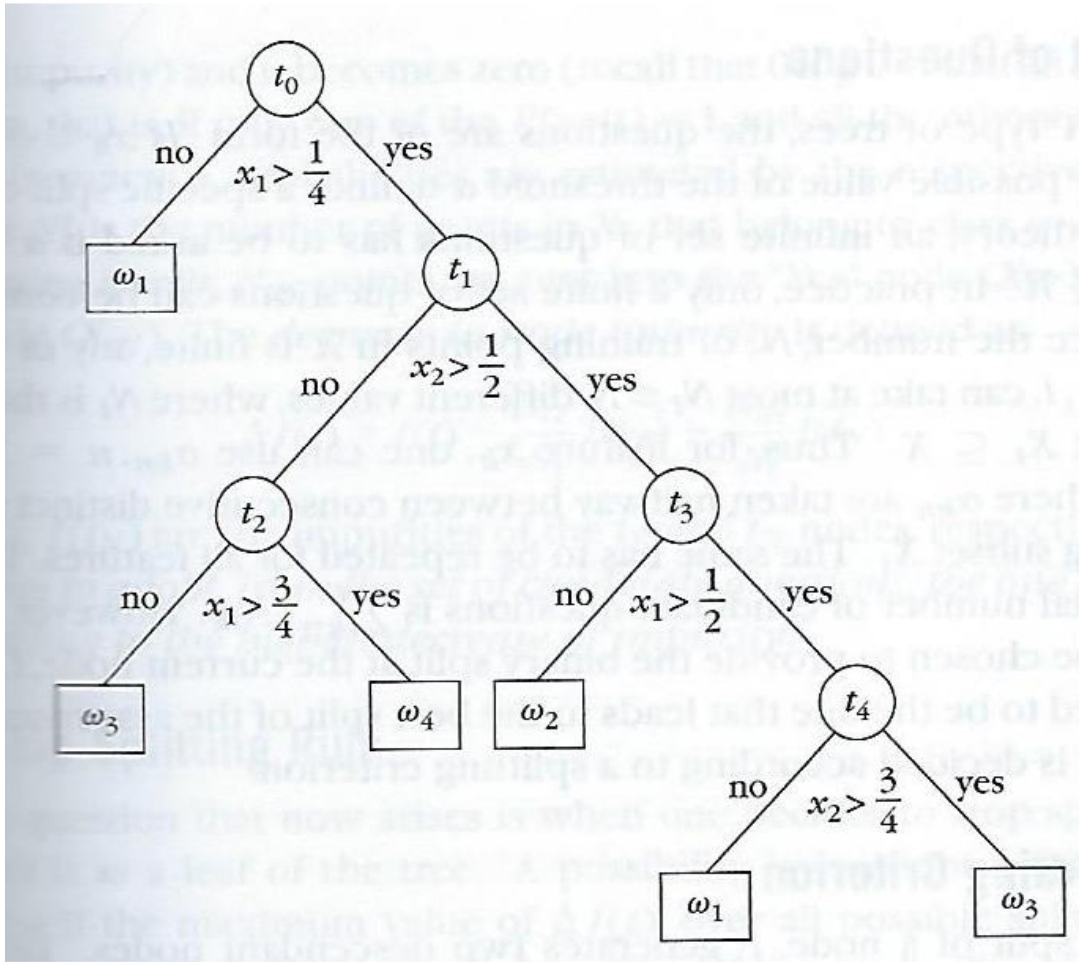


Example

Partition of the space



Decision tree



See pages 216-17 of [6]

Combining Classifiers

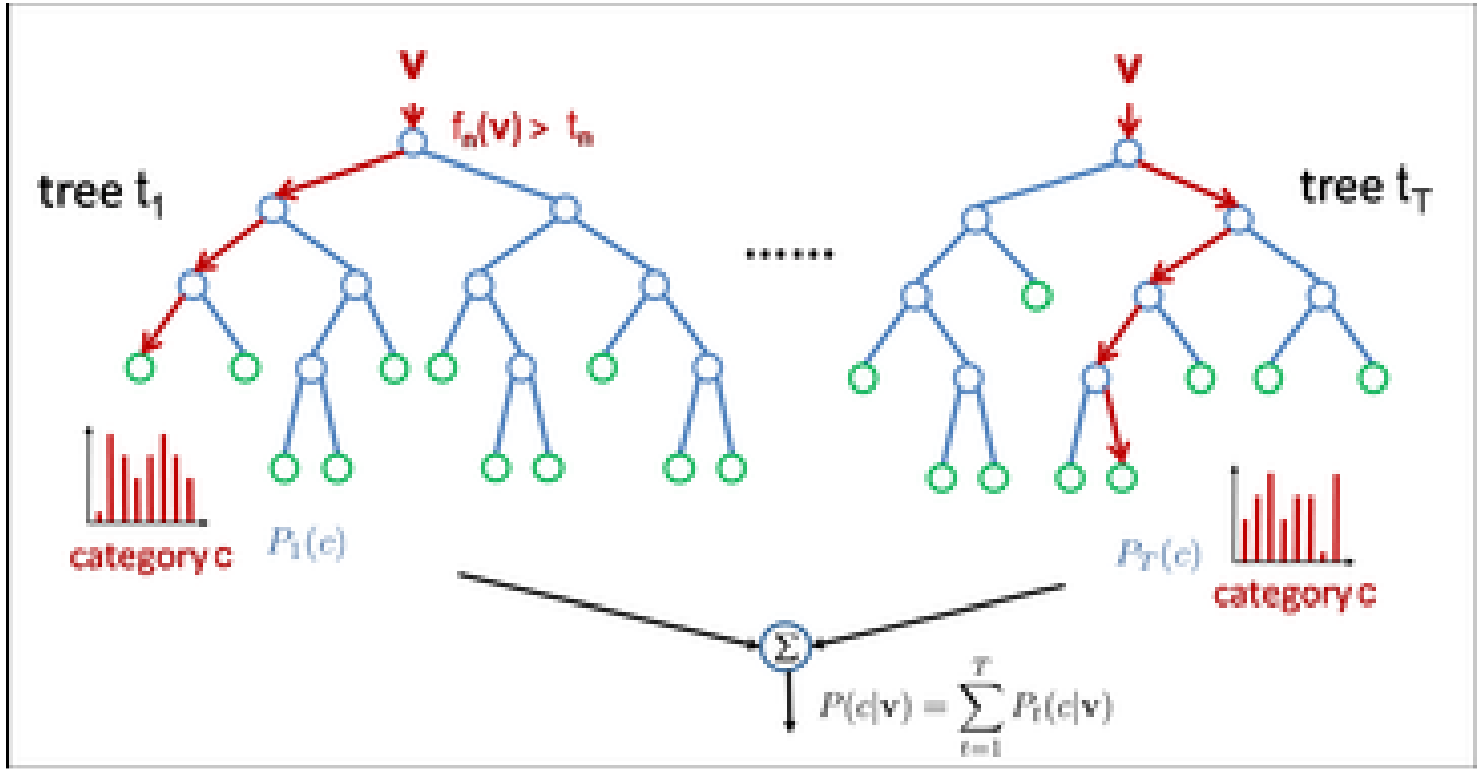
- We've seen single classifiers, so far
- Well, indeed, we have combined linear classifiers to solve multi-class problems
- In many cases, classification performance can be boosted by combining different models
- Can combine L different models and then make predictions using weighted average of individual predictions
- Such models are usually called committees or voting schemes

Known models:

- Bayesian model averaging
- Committees
  - Aka voting schemes
- Boosting
- Tree-based models
- Conditional mixture models
  - Mixture of linear regression models
  - Mixture of logistic models
  - Mixture of experts

Combining Decision Trees

- We can extend the notion of combining classifiers to more than one tree
  - We build different trees, and then
  - combine individual decisions from each tree into a single decision, for all trees
  - Decisions can be made via voting, stacking, averaging, etc.



Random Forest (RF)

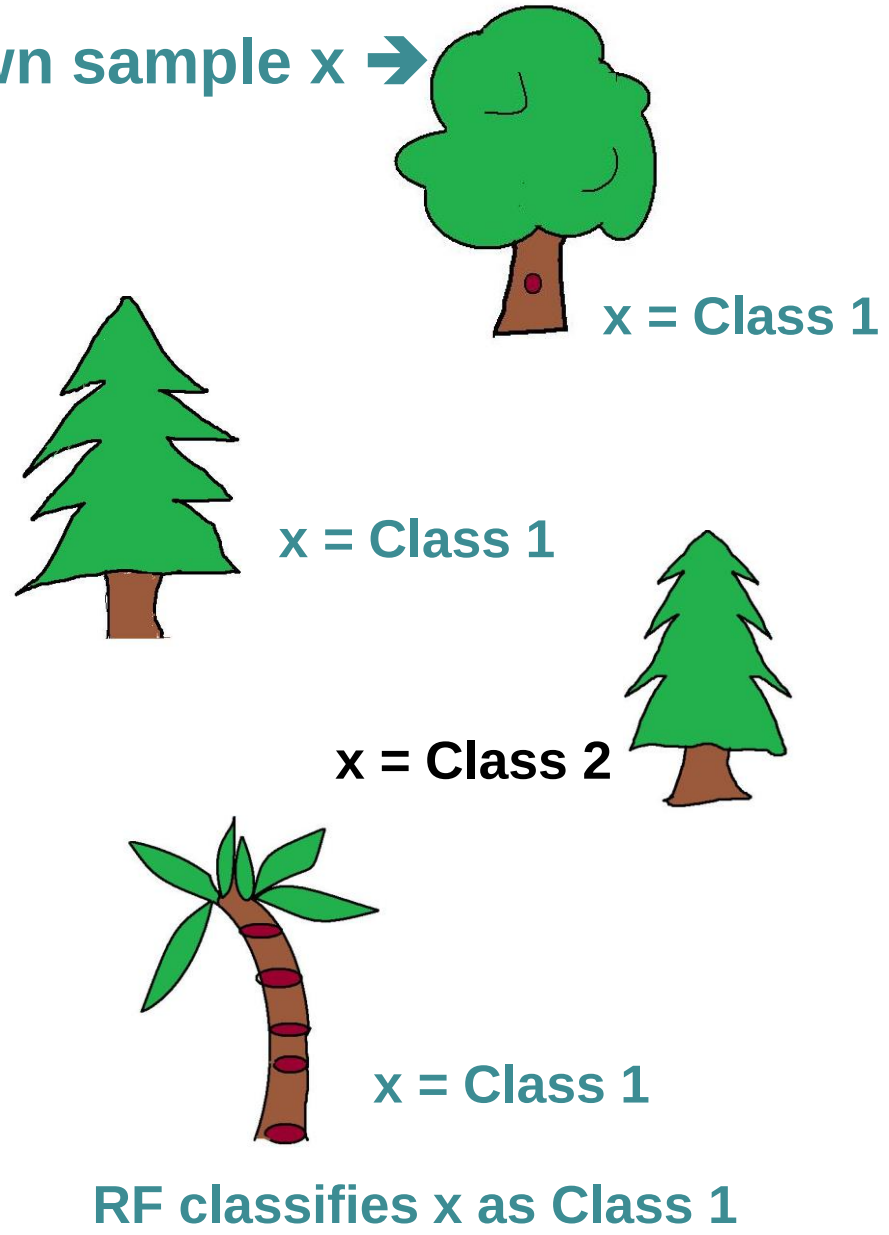
- Proposed by L. Breiman from UC at Berkeley [2]
- Random forest is an ensemble of decision trees
  - Each tree depends on values of a random vector sampled independently
  - Each vector occurs with the same distribution for all trees in the forest
- **Ancestors of the model**:
  - Classification and Regression Trees - CART
  - Learning ensembles, committee of experts, combining models
  - Bootstrap Aggregation (Bagging)

- **Comprehensive approach**:
  - Data exploration
  - Data analysis
  - Classification
  - Regression
  - Feature selection

- Generalization error of RF converges asymptotically
  - Provided that the number of trees is large
  - It depends on the individual trees in the forest and the correlation among them

Definition

- **Random Forest**: grows **many different decision trees** to classify sample x
- Each tree contributes to the decision rule to classify x
- Forest chooses the class having the **majority** of the votes
- **Gini importance** is the splitter of each tree
- **Internal error estimation**: No cross-validation or separate test set is needed for error estimation



Motivation

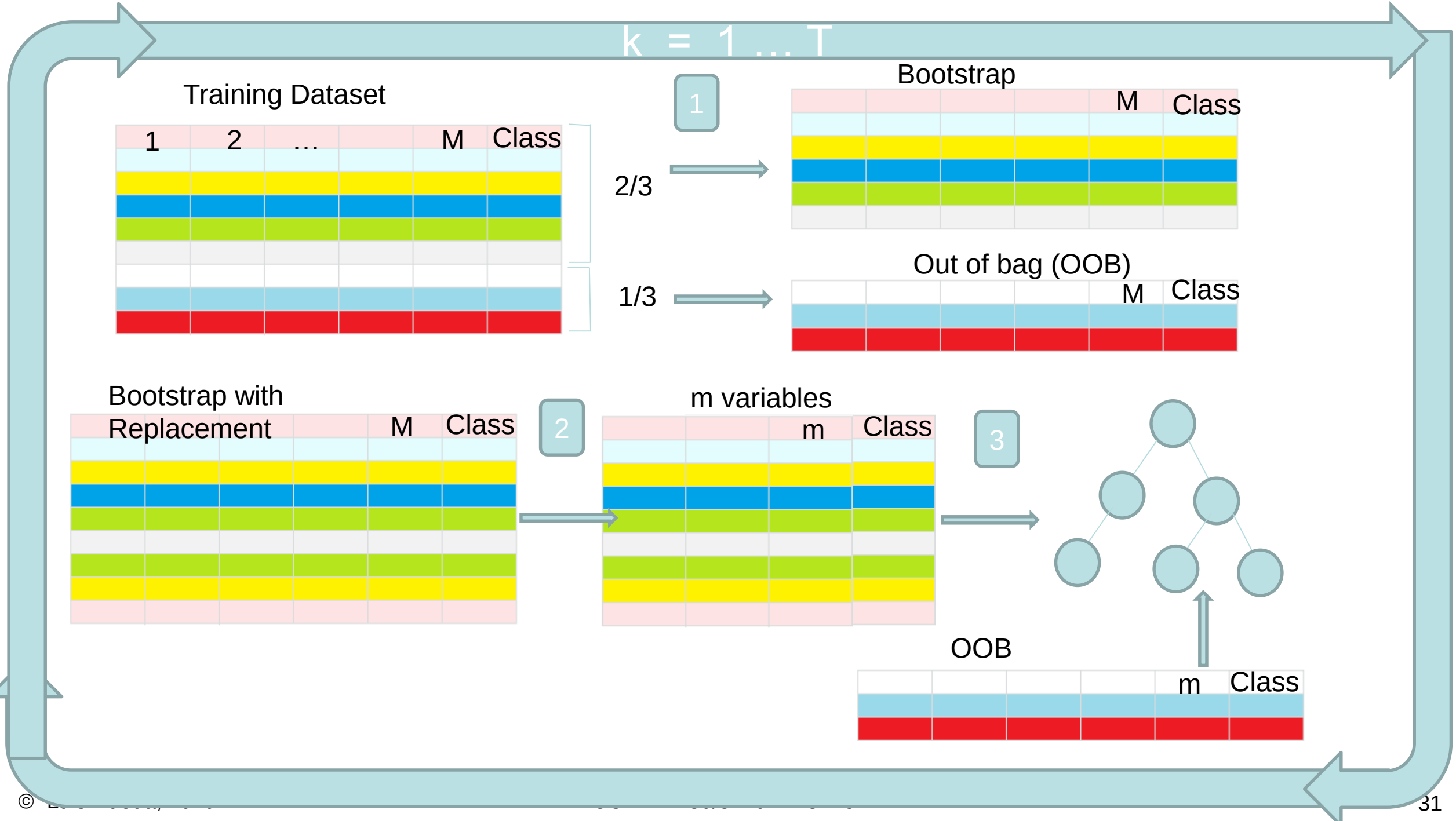
- **Task**

  Generating multiple models randomly

- **Aim**

  Combining models results will be better than relying on a single model

- ***Randomness*** is injected at each tree construction:
  1. Rows, random subset of samples with replacement
  2. Columns, random subset of variables
- *No **overfitting** and No **pruning***


k = 1 … T
Training Dataset
1
2
…
M
Class
1
2/3
Bootstrap
M
Class
1/3
Out of bag (OOB)
M
Class
Bootstrap with Replacement
M
Class
2
m variables
m
Class
3
OOB
m
Class

Algorithm

T is the number of trees	N is number of samples in training dataset
M is number of variables	m is the number of selected variables, where m << M

For tree k (1..T) repeat:

- Select bootstrap samples (2/3 of N)
  - Top it to be N with replacement
  - Select m variables randomly from M
  - On N$\times$m$^{(th)}$ bootstrap, grow the kth tree
  - Gini importance to split the kth tree at each node
- The rest (1/3 N) is out of bag (OOB)
  - For each sample **x** in OOB

    classify **x** using kth tree to class j

end repeat

- For each sample n in Training dataset repeat
  - Every time n appears in OOB, find the class j with maximum number of votes
  - n is classified as j
  - If j != real class of n

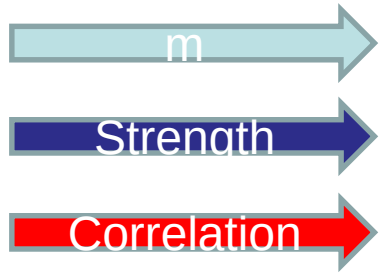
    e $\leftarrow$ e + 1 (error)

  end repeat
- Error estimation (Accuracy)

Random Forest Parameters

Breiman suggested the following number of splitting variables in each tree (m)

- m = ½ $\sqrt{M}$
- m = $\sqrt{M}$
- m = $2\sqrt{M}$
- For M=100 test values for m: 5, 10, 20
- For M=400 test values for m: 10, 20, 40



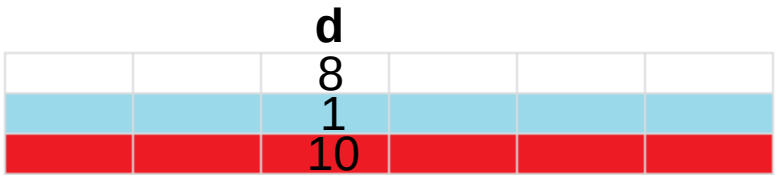
Remark

- A high value of m will increase the classification strength and the correlation among trees
- The number of trees decided by system's developer
  - It depends on the dataset

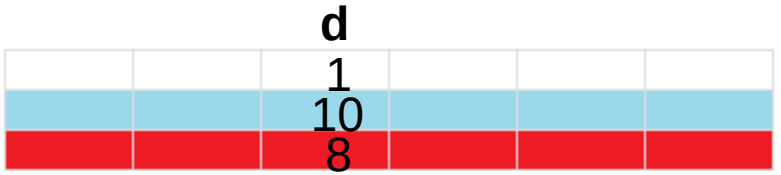
Variable Importance

A wrapping-like feature selection method that considers interactions between variables

- Calculate the error using OOB (discussed earlier).
- Randomly permute the values in variable d
- Calculate the Error using OOB after permuting d
- I (d) the importance of d equals:
- I (d) = $\{Error\ after\ permutation\}$ - $\{Error\ before\ permutation\}$



a) OOB of the k[th] tree before permuting d



b) OOB of the k[th] tree after permuting d

Other Applications for RF

- **Balancing Prediction Error:**
  For unbalanced data, RF can take a weight to the class that has a small number of samples
- **Detecting Novelties:**
  Outliers can fit into a new class
- **Regression:**
  Input takes on numerical values rather than class labels
- **Unsupervised Learning:**
  1. generate a synthetic data set from the original data set
  2. Label original data set as class 1, synthetic data set as class 2
  3. Run Random Forest on them , if error > 40% ignore this model, otherwise calculate , proximity matrix, replace missing values

  ✘ Duplicate data required

Main Features of RF

- RF are good for prediction
- Can do feature selection, while doing classification
- Law of large numbers:
  - RF do not overfit large datasets
- Injecting randomness:
  - Good for classification and regression
- Results are difficult to understand:
  - e.g., in terms of rules of classification
- RF give results competitive with boosting and adaptive bagging
- RF may be as powerful as using kernels
  - e.g., SVM, Fisher's, PCA

DT and RF in Scikit

- Included in the sklearn.ensemble module
- Includes two averaging algorithms of randomized decision trees
- Creates forests of randomized trees
- http://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees
- Trees are combined through an ensemble
- Scikit implementation is different from Brieman's original approach
- Each tree is built from a bootstrap sample (drawing with replacement)
- Many trees are created
- Combined by averaging their probabilistic prediction
  - Instead of simple voting

References

1. M. Kubat. An Introduction to Machine Learning, 2nd Edition, Springer, 2017.
2. *Artificial Intelligence: A Modern Approach*, 3rd Edition, S. Russell and P. Norvig, Prentice Hall, ISBN: 0136067387, 2010.
3. Random Forest by L. Brieman and A. Cutler. Machine Learning, 45(1):5–32, 2001
4. A Brief Overview to Random Forest by D. Steinberg et al., Salford Systems, http://www.salford-systems.com
5. Understanding Random Forests from theory to practice by Gilles Louppe, PhD dissertation, University of Liège
6. Limiting the Number of Trees in Random Forests by P. Latinne et al., MCS'2001, Cambridge, UK
7. Pattern Recognition by Theodoridis/Koutroumbas, 4th edition 2009.
8. Rattle: A Data Mining GUI for R PDF download
9. G. Williams (2009). Rattle: A Data Mining GUI for R. The R Journal, Vol. 1/2.