THE OneR (1R) CLASSIFIER

```
For each attribute,
   For each value of the attribute, make a rule as follows:
      count how often each class appears
      find the most frequent class
      the rule assigns that class to this attribute-value
   Calculate the error rate of the rules
Choose the rules with the smallest error rate
```

OneR Example

	Age	Prescription	Astigmatic	Tears	Lenses
1	8	myope	no	reduced	NO
2	12	myope	yes	normal	HARD
3	10	hypermetrope	no	normal	SOFT
4	14	hypermetrope	yes	reduced	NO
5	15	myope	no	normal	SOFT
6	24	myope	yes	reduced	NO
7	35	hypermetrope	yes	reduced	NO
8	55	hypermetrope	yes	normal	NO
9	56	myope	no	normal	NO
10	60	myope	yes	normal	HARD
11	65	hypermetrope	no	reduced	NO
12	72	hypermetrope	no	normal	SOFT

```
Prescription = myope;
   Lenses = NO 3 times; = HARD 2 times; = SOFT 1 times
Prescription = hypermetrope;
   Lenses = NO 4 times; = HARD 0 times; = SOFT 2 times

Rule for Prescription Attribute

If Prescription = myope then Lens = NO
If Prescription = hypermetrope then Lenses = NO

This rule will be correct 7  times and
wrong 5 times on the training data.

Best Rule (Weka format)

Tears:
        reduced -> NO
        normal  -> SOFT
(8/12 instances correct)
```

MUSHROOM DATA

```
@relation mushroom
@attribute 'cap-shape' { 'b', 'c', 'f', 'k', 's', 'x'}
@attribute 'cap-surface' { 'f', 'g', 's', 'y'}
@attribute 'cap-color' { 'b', 'c', 'e', 'g', 'n', 'p', 'r', 'u', 'w', 'y'}
@attribute 'bruises?' { 'f', 't'}
@attribute 'odor' { 'a', 'c', 'f', 'l', 'm', 'n', 'p', 's', 'y'}
@attribute 'gill-attachment' { 'a', 'd', 'f', 'n'}
@attribute 'gill-spacing' { 'c', 'd', 'w'}
@attribute 'gill-size' { 'b', 'n'}
@attribute 'gill-color' { 'b', 'e', 'g', 'h', 'k', 'n', 'o', 'p', 'r', 'u', 'w',
 'y'}
@attribute 'stalk-shape' { 'e', 't'}
@attribute 'stalk-root' { 'b', 'c', 'e', 'r', 'u', 'z'}
@attribute 'stalk-surface-above-ring' { 'f', 'k', 's', 'y'}
@attribute 'stalk-surface-below-ring' { 'f', 'k', 's', 'y'}
@attribute 'stalk-color-above-ring' { 'b', 'c', 'e', 'g', 'n', 'o', 'p', 'w', 'y
'}
@attribute 'stalk-color-below-ring' { 'b', 'c', 'e', 'g', 'n', 'o', 'p', 'w', 'y
'}
@attribute 'veil-type' { 'p', 'u'}
@attribute 'veil-color' { 'n', 'o', 'w', 'y'}
@attribute 'ring-number' { 'n', 'o', 't'}
@attribute 'ring-type' { 'c', 'e', 'f', 'l', 'n', 'p', 's', 'z'}
@attribute 'spore-print-color' { 'b', 'h', 'k', 'n', 'o', 'r', 'u', 'w', 'y'}
@attribute 'population' { 'a', 'c', 'n', 's', 'v', 'y'}
@attribute 'habitat' { 'd', 'g', 'l', 'm', 'p', 'u', 'w'}
@attribute 'class' { 'e', 'p'}
@data
'x','s','n','t','p','f','c','n','k','e','e','s','s','w','w','p','w','o','p','k',
's','u','p'
'x','s','y','t','a','f','c','b','k','e','c','s','s','w','w','p','w','o','p','n',
'n','g','e'
'b','s','w','t','l','f','c','b','n','e','c','s','s','w','w','p','w','o','p','n',
'n','m','e'
.
.
8124 examples
```

OneR ON MUSHROOM DATA

```
=== Run information ===

Scheme:       weka.classifiers.rules.OneR -B 6
Relation:     mushroom
Instances:    8124
Attributes:   23

Test mode:    split 66.0% train, remainder test

=== Classifier model (full training set) ===

odor:
        a       -> e
        c       -> p
        f       -> p
        l       -> e
        m       -> p
        n       -> e
        p       -> p
        s       -> p
        y       -> p
(8004/8124 instances correct)


Time taken to build model: 0.14 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances        2724               98.6242 %
Incorrectly Classified Instances        38                1.3758 %

Total Number of Instances             2762


=== Confusion Matrix ===

    a    b   <-- classified as
 1410    0 |    a = e
   38 1314 |    b = p
```

HOW GOOD IS A CLASSIFIER? TRAIN AND TEST SPLIT

- We need to measure its accuracy on known data that WAS NOT used in its construction.

- Randomly divide all available data into two files: TRAINING and TEST. Use the TRAINING file to build the classifier, use the TEST file to measure its accuracy.

- Essentially statistical estimation: We take a random sample from the population of interest and compute a parameter.

- Heuristic: Use 2/3 for training and 1/3 for testing

- Use this method if there are more than 1,000 examples

- Technically speaking this is an estimate of the TRUE ERROR RATE, ie errors the deployed classifier will make

HOW GOOD IS A CLASSIFIER? CROSS VALIDATION

- n-fold cross validation.
  Randomly split all available data into n files (folds)
  Repeat n times:
    Learn a classifier using $n - 1$ folds as training data
    Measure error on the fold left out
  Error rate is the average of the n errors

- Ten (10) is a commonly used number of folds and is the WEKA default

- Some people do the 10 fold procedure 10 times

- Stratified: The split between training and test preserves class frequency

- Use for up to 1,000 examples

- For less than 50 examples use leave-one-out

OVER-FITTING

- Also called over-training
- A major problem in data mining/machine learning
- Classifier does not generalize well
- The classifier works well on the data from which it was derived but not on other data from the same population
- High accuracy on training data, low accuracy on test data

THE ZeroR CLASSIFIER

- Don't look at the attributes
- Find the most common class
- Assign this class to all unknown examples
- If a classifier can't beat ZeroR it hasn't learnt anything

WHAT IS WRONG WITH THIS PROCEDURE

- Take the first 100 examples in the `iris.arff` as `train.arff`
- Take the last 50 examples in the `iris.arff` as `test.arff`
- Run weka, and put in `train.arff` as training file and `test.arff` as the test file

TEST/TRAIN OUTCOMES

- What would you conclude from the following outcomes for a two class problem:
  - Train accuracy 99%, Test accuracy 80%
  - Train accuracy 75% Test accuracy 73%
  - Train accuracy 53% Test accuracy 52%

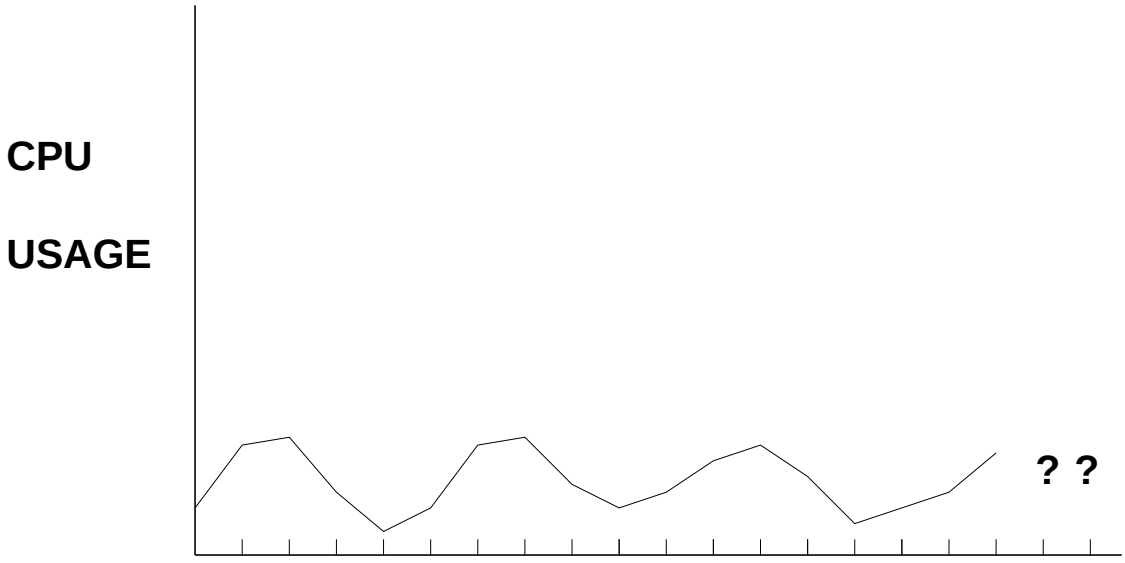
NUMERIC PREDICTION

- The class variable is numeric
  - Option 1: Discretise the class variable and use a standard classifier
    - Example: Age is numeric between 0 and 130
    - Baby = 0,2; Preschool = 3-5; Primary = 6-12; etc
    - Weka "[~ ,2]"; "[3,5]"; "[6-12]"; etc
    - There are a dozen methods
  - Option2: Use a classifier specifically designed for numeric prediction.

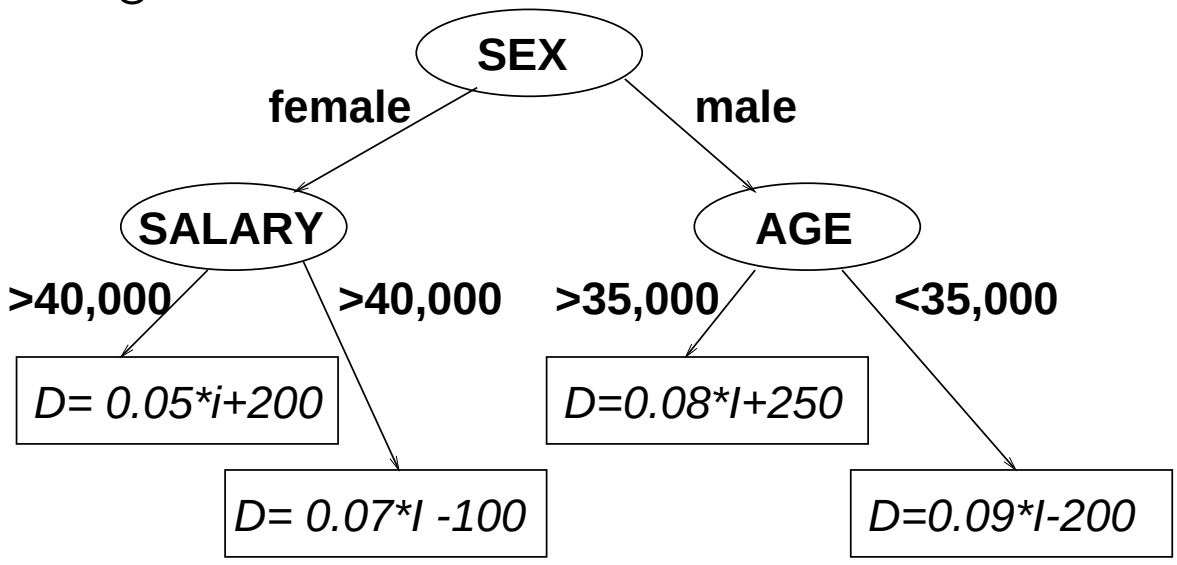
- Note, some classifiers only accept symbolic attribute values numeric values need to be discretised.

NUMERIC PREDICTION

- Time series forecasting



- Regression tree



D=Debt **I=Income**

- Root mean squared error

ACCURACY OF NUMERIC PREDICTION

- Mean Absolute Error

$$\frac{\sum_{i=1}^{n} |actual(i) - predicted(i)|}{n}$$

- Root Mean Squared Error

$$\sqrt{\sum_{i=1}^{n} (actual(i) - predicted(i))^2 / n}$$

n is the number of examples in the test set

- Why not just add up the errors?

NUMERIC PREDICTION EXAMPLE

Computer performance on benchmarks

Predict CPU time from other variables

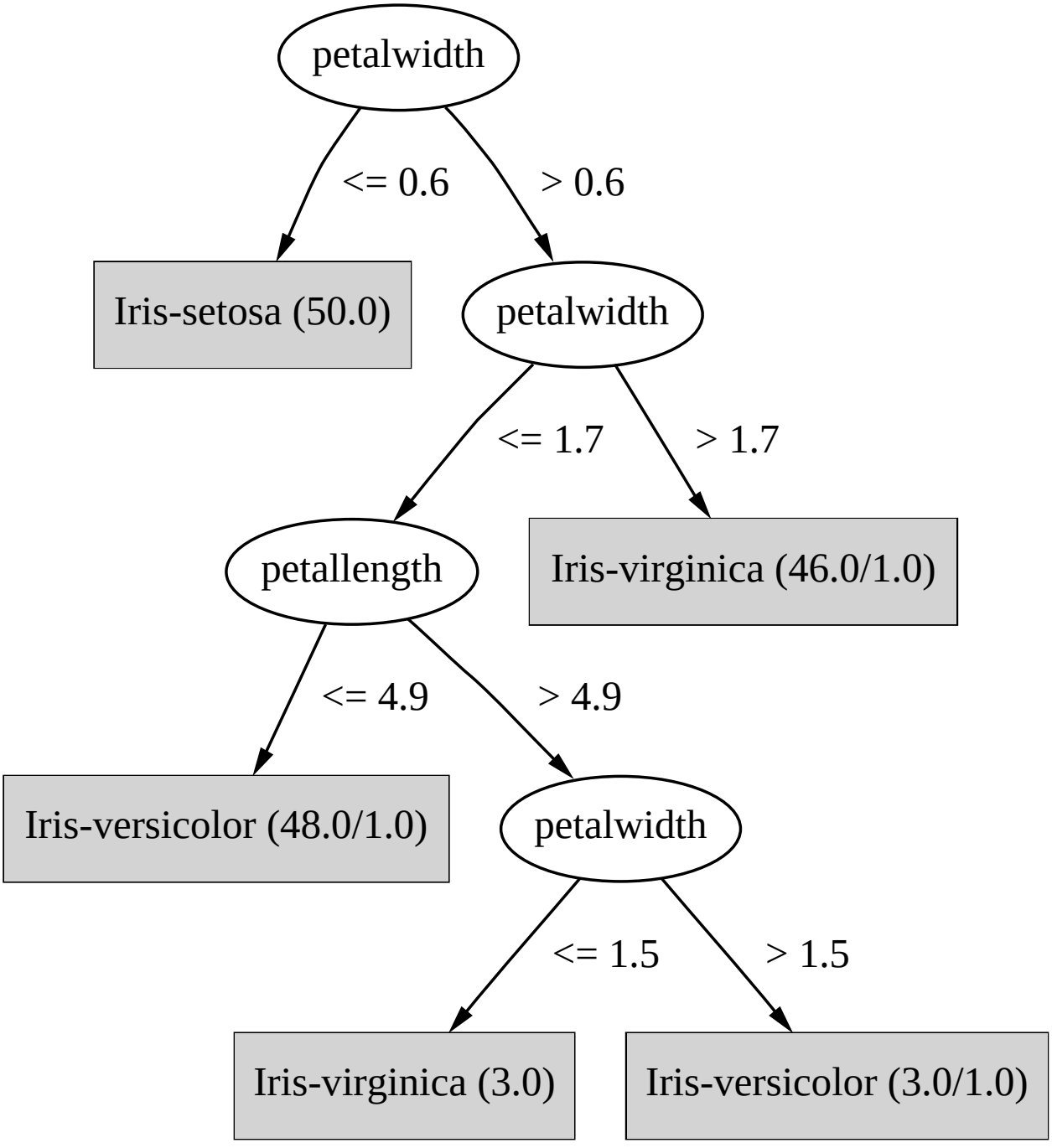
```
%
% As used by Kilpatrick, D. & Cameron-Jones, M. (1998). Numeric prediction
% using instance-based learning with encoding length selection. In Progress
% in Connectionist-Based Information Systems. Singapore: Springer-Verlag.
%

@relation 'cpu'
@attribute vendor { adviser, amdahl, apollo, basf, bti, burroughs, c.r.d, cdc}
@attribute MYCT real
@attribute MMIN real
@attribute MMAX real
@attribute CACH real
@attribute CHMIN real
@attribute CHMAX real
@attribute class real
@data
cdc,125,256,6000,256,16,128,199
amdahl,29,8000,32000,32,8,32,253
apollo,29,8000,32000,32,8,32,253
...
```

NUMERIC PREDICTION WITH M5P

```
=== Run information ===

Scheme:weka.classifiers.trees.M5P -M 4.0
Relation: cpu
Instances:209
Attributes:8
              vendor
              MYCT
              MMIN
              MMAX
              CACH
              CHMIN
              CHMAX
              class
Test mode:split 95.0% train, remainder test

=== Classifier model (full training set) ===

M5 pruned model tree:
(using smoothed linear models)

MMAX <= 14000 : LM1 (141/2.365%)
MMAX >  14000 : LM2 (68/28.342%)

LM num: 1
class =
        -2.0542 * vendor=honeywell,ipl,ibm,cdc,ncr,basf,gould,siemens,
        + 5.4303 * vendor=adviser,sperry,amdahl
        - 5.7791 * vendor=amdahl
        + 0.0064 * MYCT
        + 0.0016 * MMIN
        + 0.0034 * MMAX
        + 0.5524 * CACH
        + 1.1411 * CHMIN
        + 0.0945 * CHMAX
        + 4.1463

LM num: 2
class =
        -57.3649 * vendor=honeywell,ipl,ibm,cdc,ncr,basf,gould,siemens,
        + 46.1469 * vendor=adviser,sperry,amdahl
        - 58.0762 * vendor=amdahl
```

```
        + 0.012 * MYCT
        + 0.0162 * MMIN
        + 0.0086 * MMAX
        + 0.8332 * CACH
        - 1.2665 * CHMIN
        + 1.2741 * CHMAX
        - 107.243

Number of Rules : 2

Time taken to build model: 0.16seconds

=== Predictions ontest split===

inst#,    actual, predicted, error
     1    120        124.956      4.956
     2     27         25.865     -1.135
     3    102        100.884     -1.116
     4     42         44.458      2.458
     5     20         15.218     -4.782
     6     30         25.96      -4.04
     7    107         98.92      -8.08
     8    181        230.493     49.493
     9    603        573.598    -29.402
    10     21         12.618     -8.382

=== Evaluation on test split ===
=== Summary ===

Correlation coefficient                  0.9939
Mean absolute error                     11.3844
Root mean squared error                 18.7668
Relative absolute error                 11.7003 %
Root relative squared error             11.072  %
Total Number of Instances               10
```

DECISION TREE CLASSIFIER FOR IRIS DATA



DECISION TREE CLASSIFIER

Suppose we have the following data. How can we use it to decide whether some unknown person will get sunburnt?

name	Hair	Height	Weight	Lotion	Result
Sara	blonde	average	light	no	sunburnt
Dana	blonde	tall	average	yes	not burnt
Alex	brown	short	average	yes	not burnt
Annie	blonde	short	average	no	sunburnt
Emily	red	average	heavy	no	sunburnt
Peter	brown	tall	heavy	no	not burnt
John	brown	average	heavy	no	not burnt
Katie	blonde	short	light	yes	not burnt

1. We could use (K) Nearest Neighbour, OneR, ZeroR

2. Can we do better?

3. Most popular, most versatile classifier, the Decision Tree

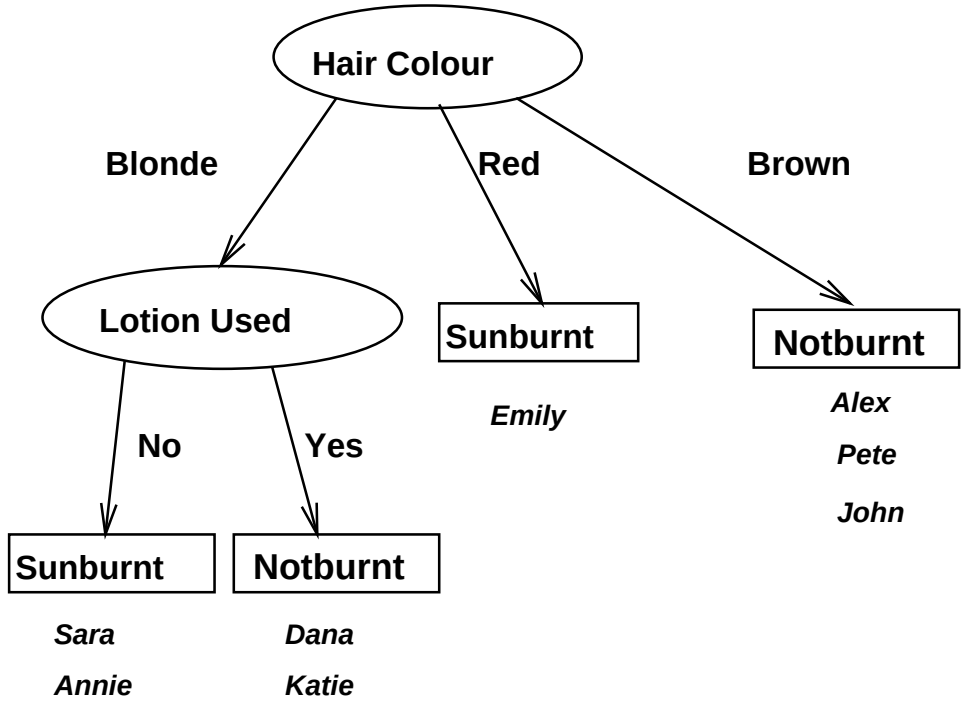
4. Ross Quinlan, Australia, 1980's, C4.5, then C4.8

5. Refined, implemented in Weka as J48

6. Winston identification tree = decision tree

DECISION TREE CLASSIFIER

- A leaf indicates a class
- An internal node is a decision node with a test to be carried out and a sub tree for each outcome.

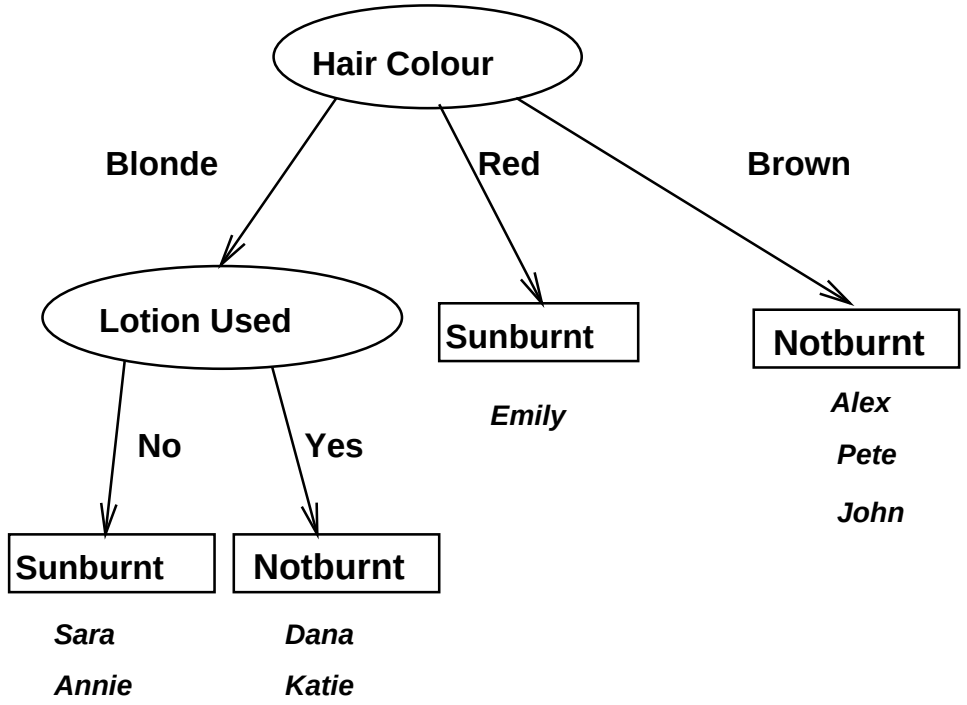


- Will Jack get sunburnt?

Name	Hair	Height	Weight	Lotion
Jack	blond	average	heavy	no

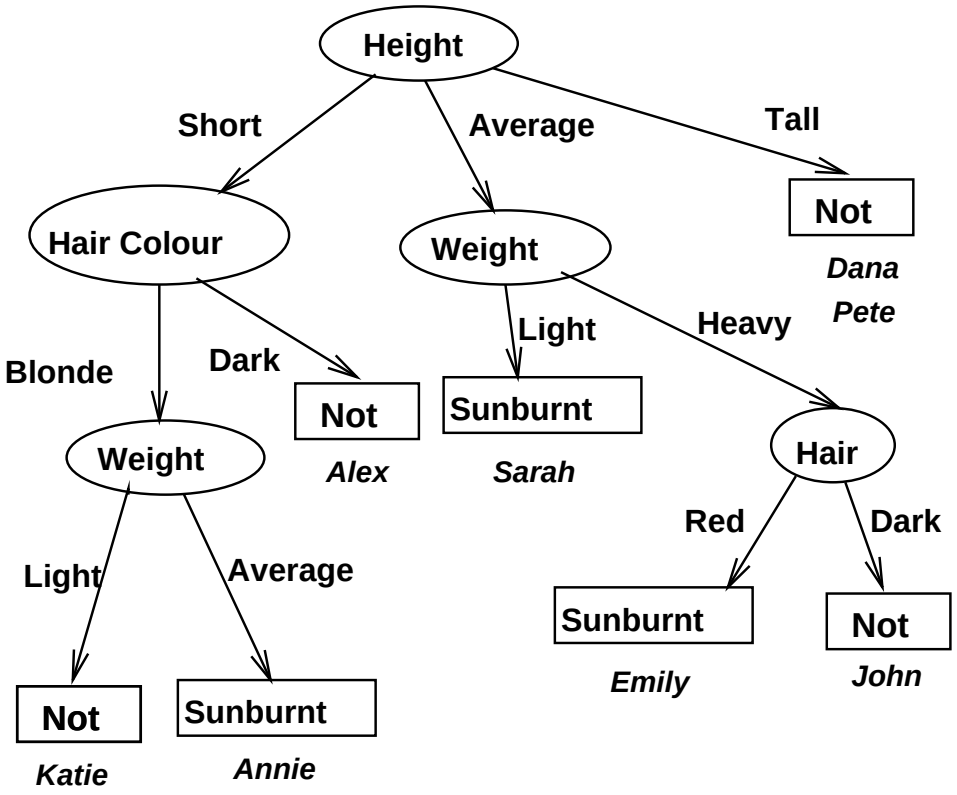
Start from root, carry out the test and follow indicated branch to a leaf.

LINE PRINTER FORM OF DECISION TREE




```
Hair Colour = Blonde:
|   Lotion Used = no: Sunburnt
|   Lotion Used = yes: Not burnt
Hair Colour = Red: Sunburnt
Hair Colour = Brown: Not Burnt
```

ALTERNATIVE DECISION TREE



- Does this tree give the same classifications as the previous one?
- What is its line printer form?
- Which tree is preferable? Why?
  - Tree 1 is smaller (Occam's Razor)
  - Knowledge in tree 1 is a much better fit to what is known in the domain.

BASIC DECISION TREE ALGORITHM

Subroutine Create tree(Examples)

```
IF      all examples belong to the same class
THEN    Return a leaf node with the class as label
ELSE    Attribute = Select an Attribute(Examples)
        Node = Create Test Node(Attribute)
        For each value of Attribute
          Partition = Subset of examples with
                      attribute value
          Subtree = Create tree(Partition)
        Attach(Subtree,Node,Value)
        Return Node
```

Main program

Create tree(Training Set)

- The basic idea is to keep splitting the training set into smaller pieces until each piece has only members of one class.

- Which attribute to select for splitting?

- Choose attribute split which minimizes disorder.

MEASURE OF DISORDER

Use a result from information theory:

n_b	number of examples in branch b
n_t	total number of examples in all branches
n_{bc}	total number of examples in branch b of class c

Disorder at the end of branch b is

$$\left(\sum_c -\frac{n_{bc}}{n_b} log_2 \frac{n_{bc}}{n_b} \right)$$

Disorder is low (close to 0) if almost all examples are of the same class.

Disorder is high (close to 1) if almost all examples are of different classes.

Example if we have two classes, 1 and 2, with 50 examples each:

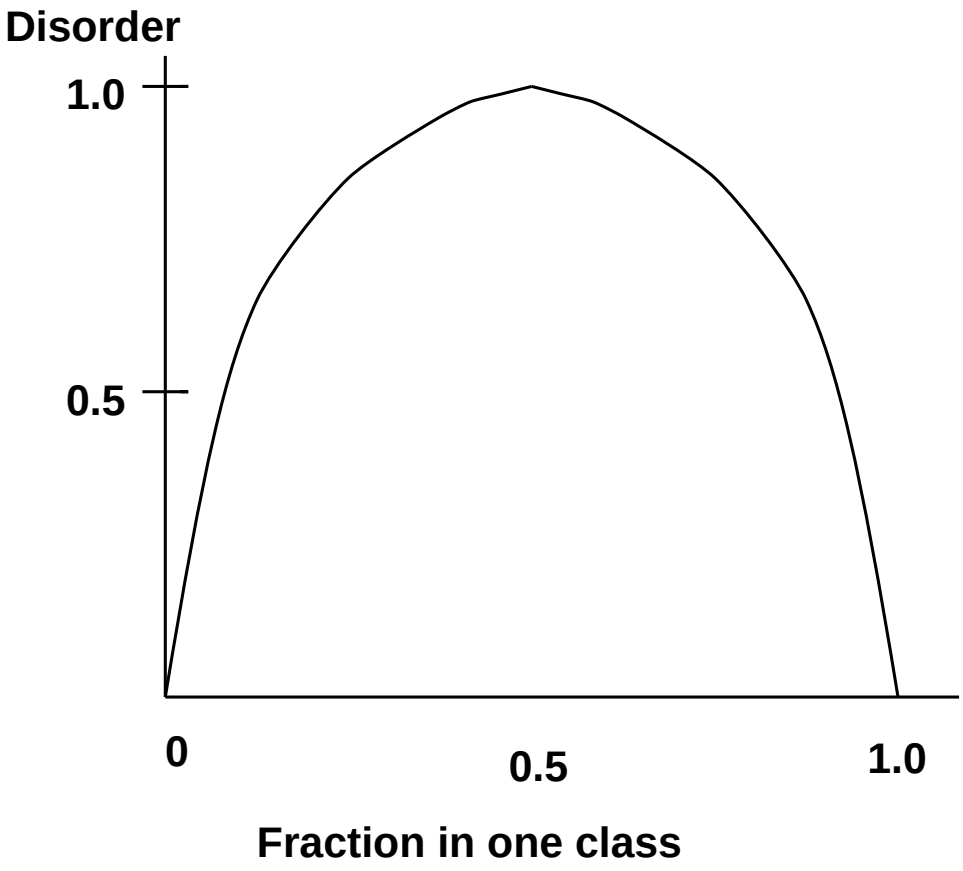
$$n_{b1} = 50 \qquad n_{b2} = 50 \qquad n_b = 100$$

$$disorder = -\frac{50}{100} log_2 \frac{50}{100} - \frac{50}{100} log_2 \frac{50}{100} = \frac{1}{2} + \frac{1}{2} = 1$$

$$(log_2 \frac{1}{2} = -1)$$

MEASURE OF DISORDER

Disorder vs fraction in one class for TWO classes



CHOOSING THE SPLIT ATTRIBUTE

For a split on any attribute

$$Average\,disorder = \sum_b \left(\frac{n_b}{n_t}\right) \times (Disorder\,in\,branch\,b)$$

$$Average\,disorder = \sum_b \left(\frac{n_b}{n_t}\right) \times \left(\sum_c -\frac{n_{bc}}{n_b} log_2 \frac{n_{bc}}{n_b}\right)$$

Work out the disorder for each attribute.

Choose the attribute with LOWEST disorder.

Test	Disorder
Hair	0.50
Height	0.69
Weight	0.94
Lotion	0.61

Simple option, but not the best

PROBLEMS WITH BASIC ALGORITHM

1. Overfitting – The decision tree works well on the data from which it was constructed but not on other data from the same population.

2. How can we get the most accurate tree? Examine all possible trees?

3. Which attribute to choose for splitting?

4. Should we split at all?

SPLITTING

How to choose the attribute for splitting?

disorder = entropy = information

Switch to C4.5/J48 terminology

1. Maximize information gain

2. Maximize information ratio

- For 8 equally probable messages the information in any one is

$$-log_2(1/8) = 3\ bits$$

- A message that a case in a set S is in class C_j has probability

$$\frac{freq(C_j, S)}{|S|}$$

- information in the message is

$$-log_2\left(\frac{freq(C_j, S)}{|S|}\right)$$

INFORMATION GAIN

- Information in a (training) data set T containing k classes is (Non equally probable messages need to be weighted by probability (frequency).)

$$info(T) = -\sum_{j=1}^{k} \frac{freq(C_j, T)}{|T|} \times log_2 \left(\frac{freq(C_j, T)}{|T|} \right) \quad bits$$

- Assume we have example set T and are making a decision to split.

- Current information is $info(T)$

- After splitting on attribute X with k values the information will be

$$info_X(T) = \sum_{i=1}^{k} \frac{|T_i|}{|T|} \times info(T_i)$$

$$gain(X) = info(T) - info_X(T)$$

- gain criterion – select attribute split to maximize information gain (mutual information)

- Used by ID3

INFORMATION RATIO

- Gain criterion is biased toward attributes with many values
  - If each example has a unique record number $info_X(T)$ will be 0 so $gain(X)$ will be maximum
  - This split is useless
- Put in a normalizing factor for this bias

$$split\ info(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times log_2 \left(\frac{|T_i|}{|T|} \right)$$

$$gain\ ratio = \frac{gain(X)}{split\ info(X)}$$

- Choose attribute split that maximizes this ratio

PREVENTING OVERFITTING

- How to get most accurate tree (on WITHHELD data)?
- Prevent the tree from becoming too complex
- Pre-pruning or Stopping
  - Only split if the information gain will be significant. ID3 used (χ^2) to test for significance
  - Not as good in practice as post pruning
- Post pruning
  - Build complete tree first
  - Make the tree less complex and hopefully more general by
  - replacing subtrees with leaves
  - replacing subtrees with most common branch

PRUNING OF DECISION TREES

- If a branch is replaced by a leaf, the leaf will correspond to several classes. Its label will be the most common class.

- Error on training set will increase

- Outline of the approach

  1. Assume that one can (magically) predict error rate of (sub) tree and leaf
  2. Start from bottom, examine each non leaf subtree
  3. if predicted error would be lower if this sub tree was a leaf then replace with a leaf
  4. if predicted error would be lower if this sub tree replaced with most common branch, then replace

PREDICTING ERROR RATE OF A SUBTREE

- Can't use error rate on training set directly
- Approach 1. Apply tree to a separate set of cases, validation set [Weka Reduced Error Pruning]
  - Break the original training set randomly into
  - Actual training set
  - Validation set
- Approach 2. Use some smart statistical theory
  1. We know that training set error will always be lower than test set error
  2. Build a statistical model of this difference
  3. Use the model to adjust (raise) the error on training set for pruning.
  4. C4.5/J48 Pruning confidence

CONTINUOUS ATTRIBUTES

- Much more straight forward than it might appear
- Sort on values of attribute A, get $v_1, v_2 ... v_m$
- Can select threshold:

$$\frac{v_i + v_{i+1}}{2}$$

- $m - 1$ possible splits to examine
- C4.5 selects largest value of A that does not exceed midpoint
- Decision trees and rules have numbers actually in the data

RUNNING J48

- Important Parameters
  - Minimum number of objects. Do not perform a split if the node has fewer than this number of examples.
  - Pruning Confidence. Smaller values cause more heavy pruning. Adjust if error rate on pruned trees for test cases is much higher than estimated error rate.
  - Reduced error pruning
- Output
  - Visualise the tree
  - Visualize the errors

OTHER INDUCTION PROGRAMS

- There are dozens of other programs for generating decision trees from data based on different algorithms.

- There are also many programs for generating classification rules directly from data.

- Decision trees or rules are often preferred for data mining over nearest neighbour and neural network methods because they can:
  - Give insight into the data
  - Be used to explain how a decision (eg fraud/not fraud) has been made
  - What do you conclude if
    * If the accuracy of OneR and J48 are very similar, eg 94% vs 98%?
    * If the accuracy of OneR and J48 are far apart, eg 64% vs 90%?