Data Engineering

Data preprocessing and transformation

Just apply a learner? NO!

- Algorithms are ***biased***
  - No free lunch theorem: considering all possible data distributions, no algorithm is better than another
- Algorithms make ***assumptions*** about data
  - Conditionally independent features (naive Bayes)
  - All features relevant (e.g., kNN, C4.5)
  - All features discrete (e.g., 1R)
  - Little/no noise (many regression algorithms)
  - Little/no missing values (e.g., PCA)
- Given data:
  - Choose/adapt algorithm to data (selection/parameter tuning)
  - Adapt data to algorithm (data engineering)

Data Engineering

- **Attribute selection (feature selection)**
  - **Remove features with little/no predictive information**
- Attribute discretization
  - Convert numerical attributes to nominal ones
- Data transformations (feature generation)
  - Transform data to another representation
- Dirty data
  - Remove missing values or outliers

Irrelevant features can 'confuse' algorithms

- kNN: curse of dimensionality
  - # training instances required increases exponentially with # (irrelevant) attributes
  - Distance between neighbors increases with every new dimension
- C4.5: data fragmentation problem
  - select attributes on less and less data after every split
  - Even random attributes can look good on small samples
  - Partially corrected by pruning
- Naive Bayes: redundant (very similar) features
  - Features clearly not independent, probabilities likely incorrect
  - But, Naive Bayes is insensitive to irrelevant features (ignored)

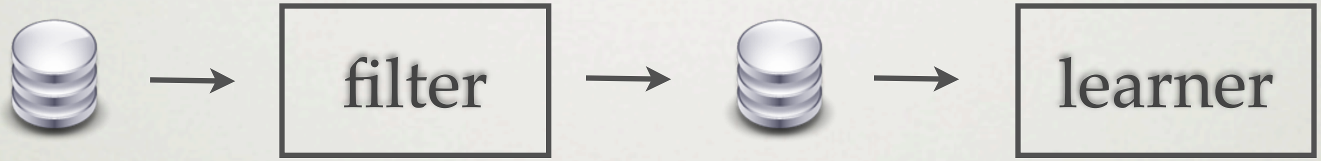
Attribute selection

- Other benefits
  - Speed: irrelevant attributes often slow down algorithms
  - Interpretability: e.g. avoids huge decision trees
- 2 types:
  - Feature Ranking: rank by relevancy metric, cut off
  - Feature Selection: search for optimal subset

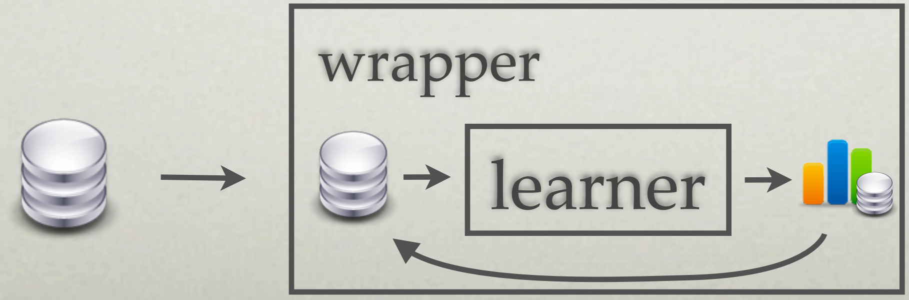
Attribute selection

2 approaches (besides manual removal):

- **Filter** approach: Learner independent, based on data properties or simple models built by other learners



- **Wrapper** approach: Learner dependent, rerun learner with different attributes, select based on performance



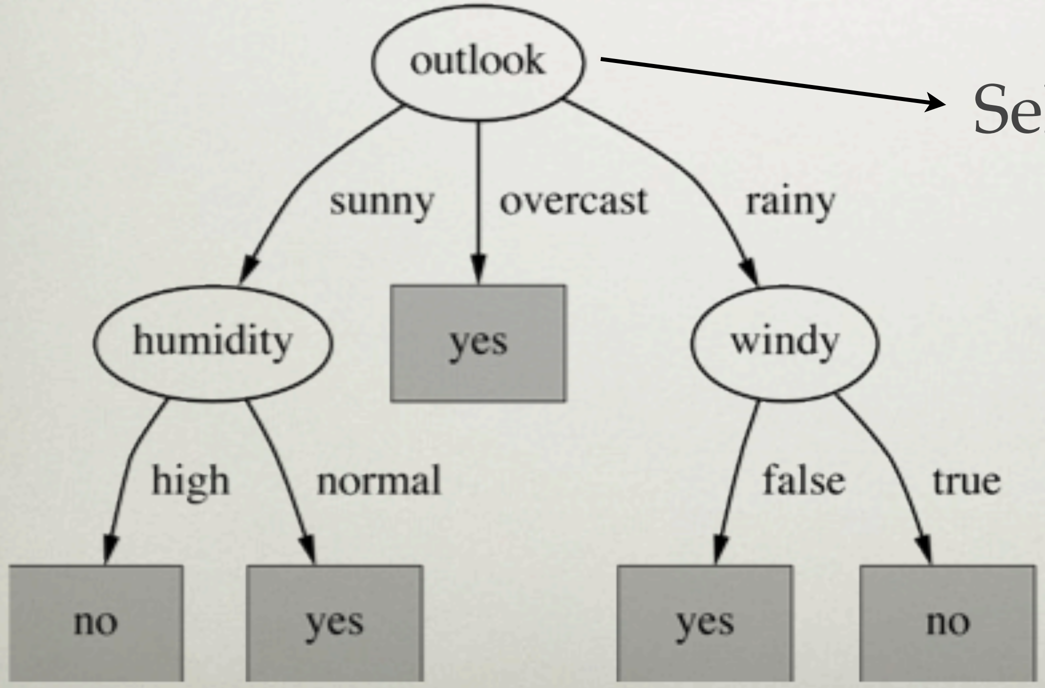
Filters

- Basic: find smallest feature set that separates data
  - Expensive, often causes overfitting
- Better: use another learner as filter
  - Many models show importance of features
    - e.g. C4.5, 1R, kNN, ...
  - Recursive: select 1 attribute, remove, repeat
  - Produces ranking: cut-off defined by user

Filters

Using C4.5

- select feature(s) tested in top-level node(s)
- `Decision stump' (1 node) sufficient



Select feature 'outlook', remove, repeat

Filters

Using 1R

- select the 1R feature, repeat

Rule:

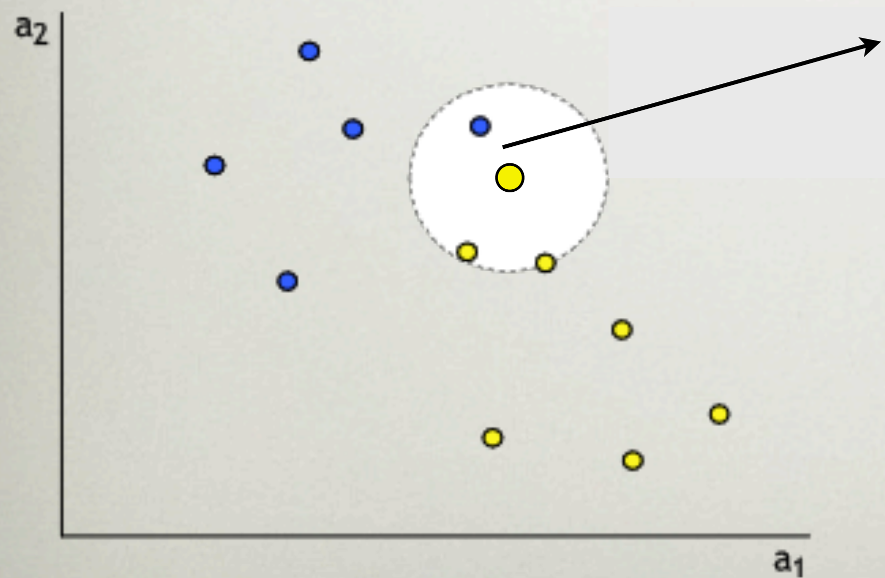
If(outlook=sunny) play=no, else play=yes

Select feature 'outlook', remove, repeat

Filters

Using kNN: weigh features by capability to separate classes

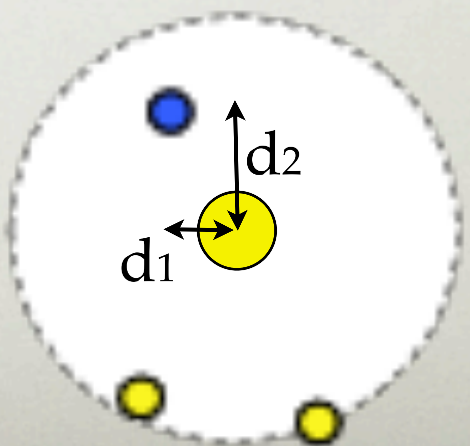
- same class: reduce weight of features with ≠ value (irrelevant)
- other class: increase weight of features with ≠ value (relevant)



Different classes:

increase weight of $a_1 \propto d_1$

increase weight of $a_2 \propto d_2$

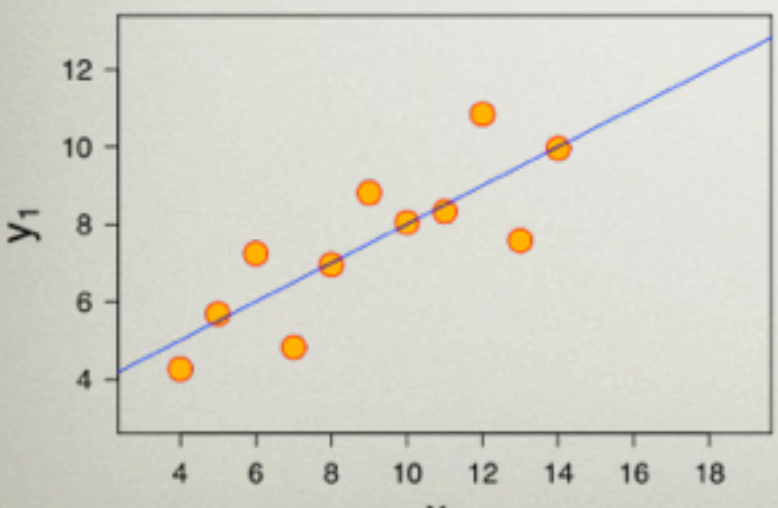


Filters

Using Linear regression (simple or logistic)

- Select features with highest weights

$$x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$$



Select w_i, so that $w_i \geq w_j$, $i \neq j$

remove, repeat

Filters

- Direct filtering: use data properties
  - Correlation-based Feature Selection (CFS)

$$U(A,B) = 2\frac{H(A)+H(B)-H(A,B)}{H(A)+H(B)} \in [0,1]$$

H(): Entropy
A: any attribute
B: class attribute

  - Select attributes with high class correlation, little intercorrelation
  - Select subset by aggregating over attributes A_j for class C
    - Ties broken in favor of smaller subsets

$$\sum U(A_j, C) / \sqrt{\left(\sum\sum U(A_i, A_j)\right)}$$

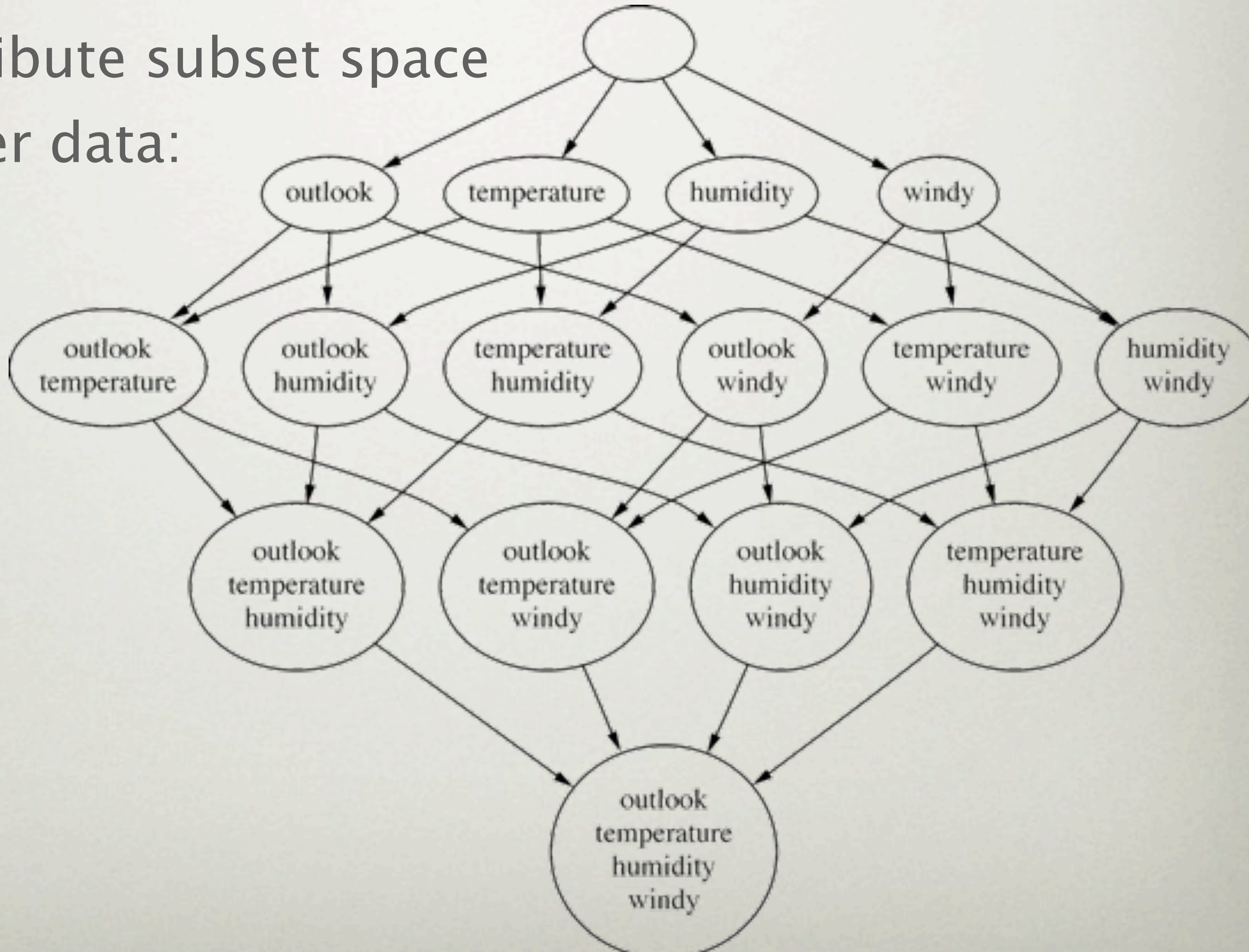
  - Fast, default in WEKA

Wrappers

- Learner-dependent (selection for specific learner)
- Wrapper around learner
  - Select features, evaluate learner (e.g., cross-validation)
- Expensive
  - Greedy search: $O(k^2)$ for k attributes
  - When using a prior ranking (only find cut-off): $O(k)$

Wrappers: search

- Search attribute subset space
- E.g. weather data:



Wrappers: search

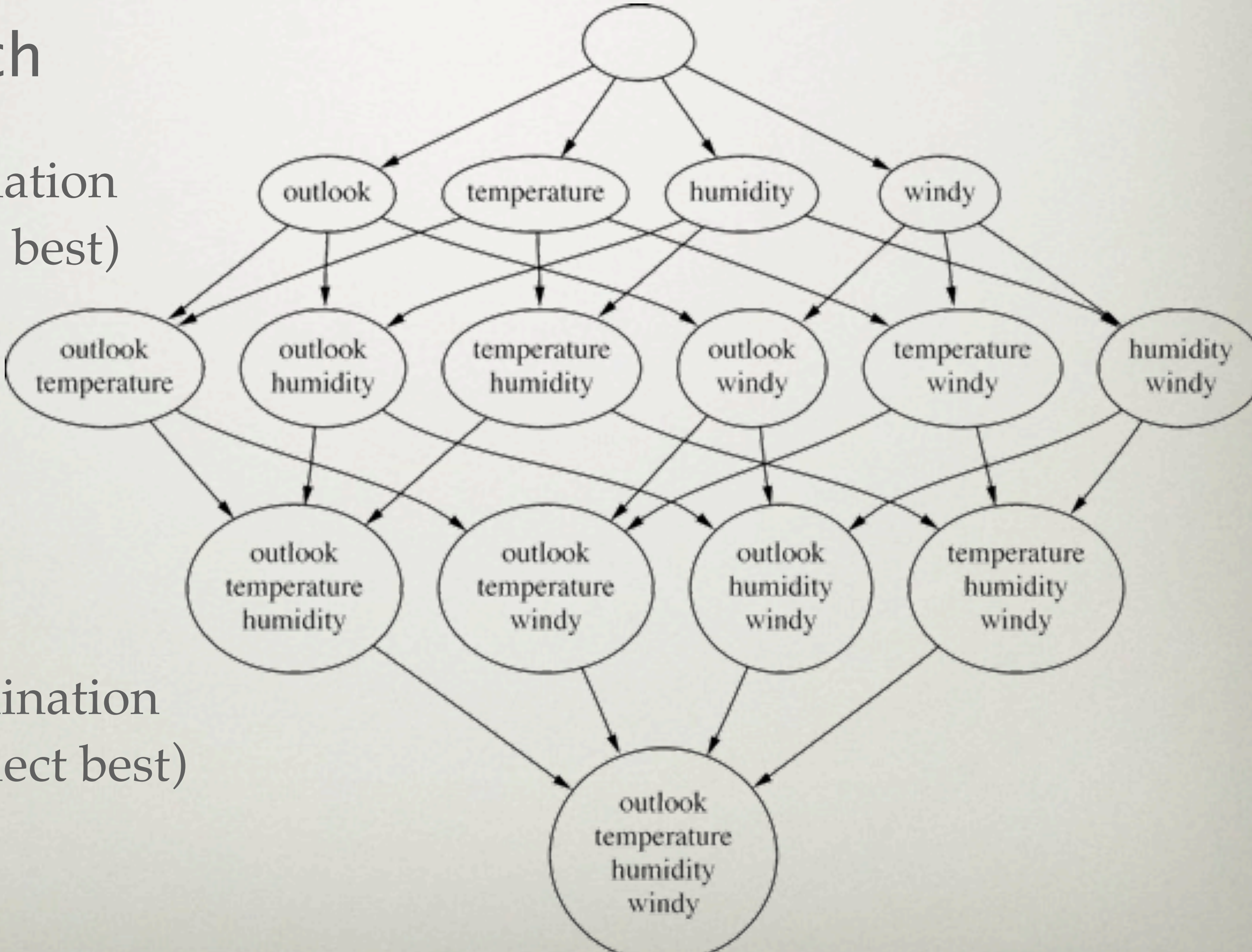
Greedy search

Forward elimination
(add one, select best)

↓

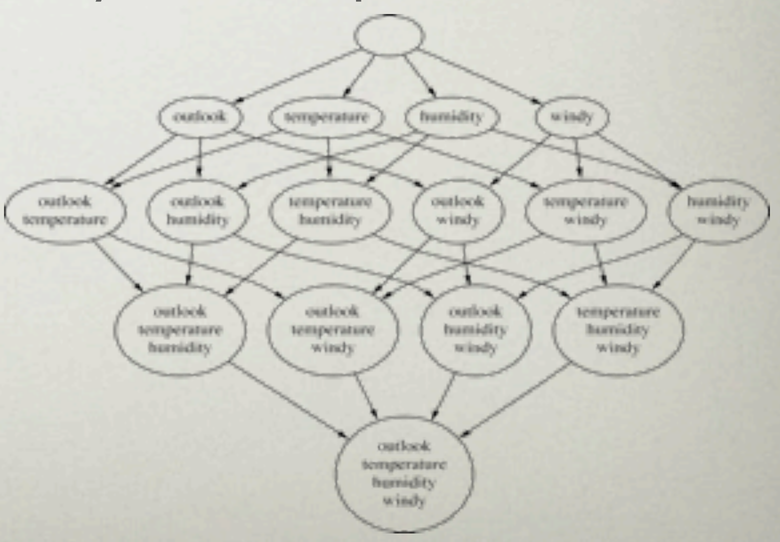
↑

Backward elimination
(remove one, select best)



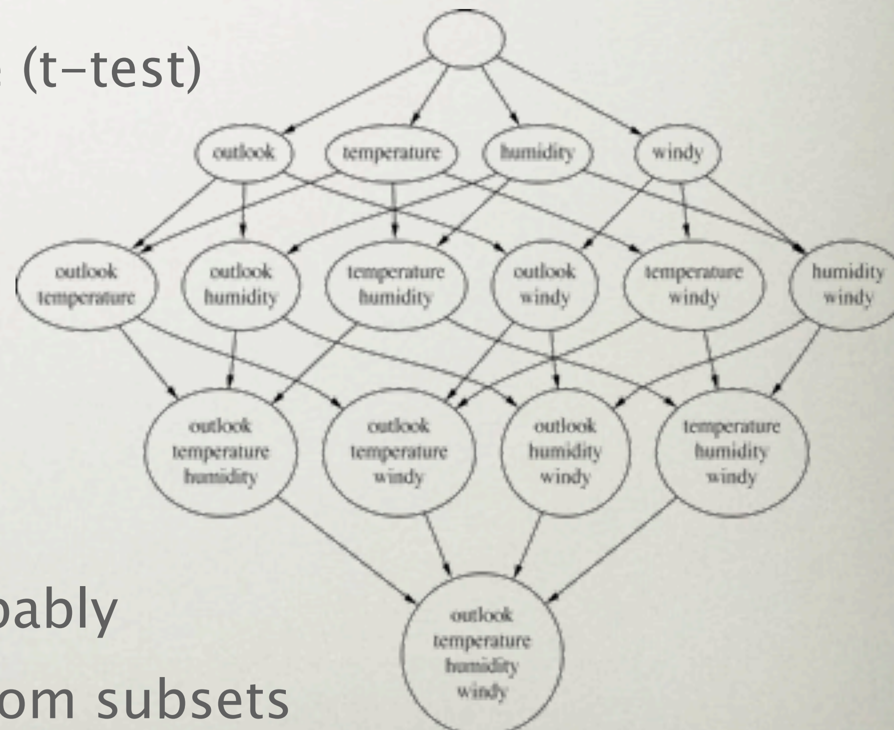
Wrappers: search

- Other search techniques (besides greedy search):
  - Bidirectional search
  - Best-first search: keep sorted list of subsets, backtrack until optimum solution found
  - Beam search: Best-first search keeping only k best nodes
  - Genetic algorithms: ‘evolve’ good subset by random perturbations in list of candidate subsets
  - Still expensive...



Wrappers: search

- Race search
  - Stop cross-validation as soon as it is clear that feature subset is not better than currently best one
  - Label winning subset per instance (t-test)



	outlook	temp	humid	windy
$inst_1$	-1	0	1	-1
$inst_2$	0	-1	1	-1

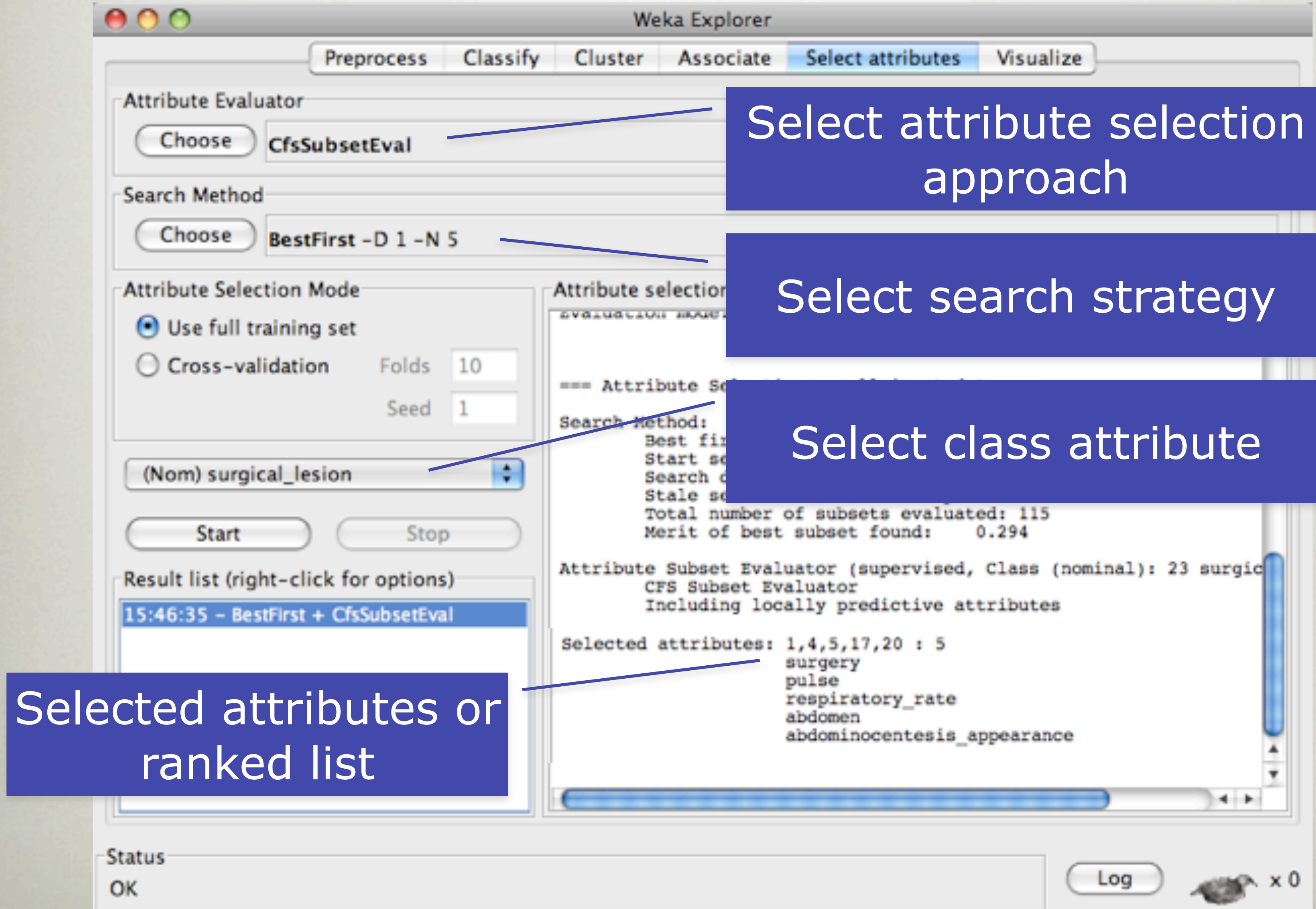
Selecting humid results in significantly better prediction for $inst_2$

  - Stop when one subset is better
    - better: significantly, or probably
  - Schemata-search: idem with random subsets
    - if one better: stop all races, continue with winner

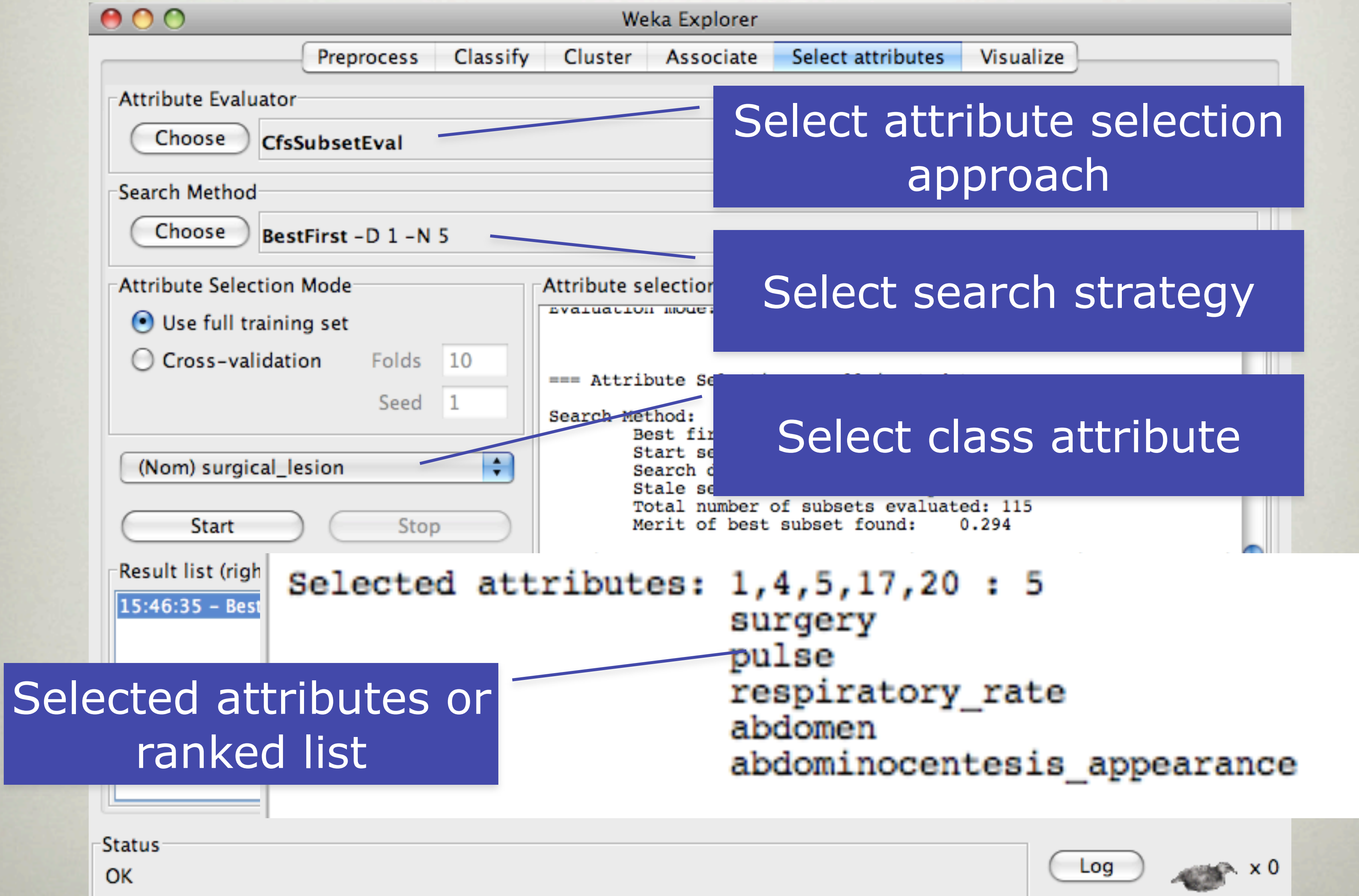
Preprocessing with WEKA

- Attribute subset selection:
  - **ClassifierSubsetEval:** Use another learner as filter
  - **CfsSubsetEval:** Correlation-based Feature Selection
  - **WrapperSubsetEval:** Choose learner to be wrapped (with search)
- Attribute ranking approaches (with ranker):
  - **GainRatioAttributeEval, InfoGainAttributeEval**
    - C4.5-based: rank attributes by gain ratio/information gain
  - **ReliefFAttributeEval:** kNN-based: attribute weighting
  - **OneRAttributeEval, SVMAttributeEval**
    - Use 1R or SVM as filter for attributes, with recursive feat. elim.

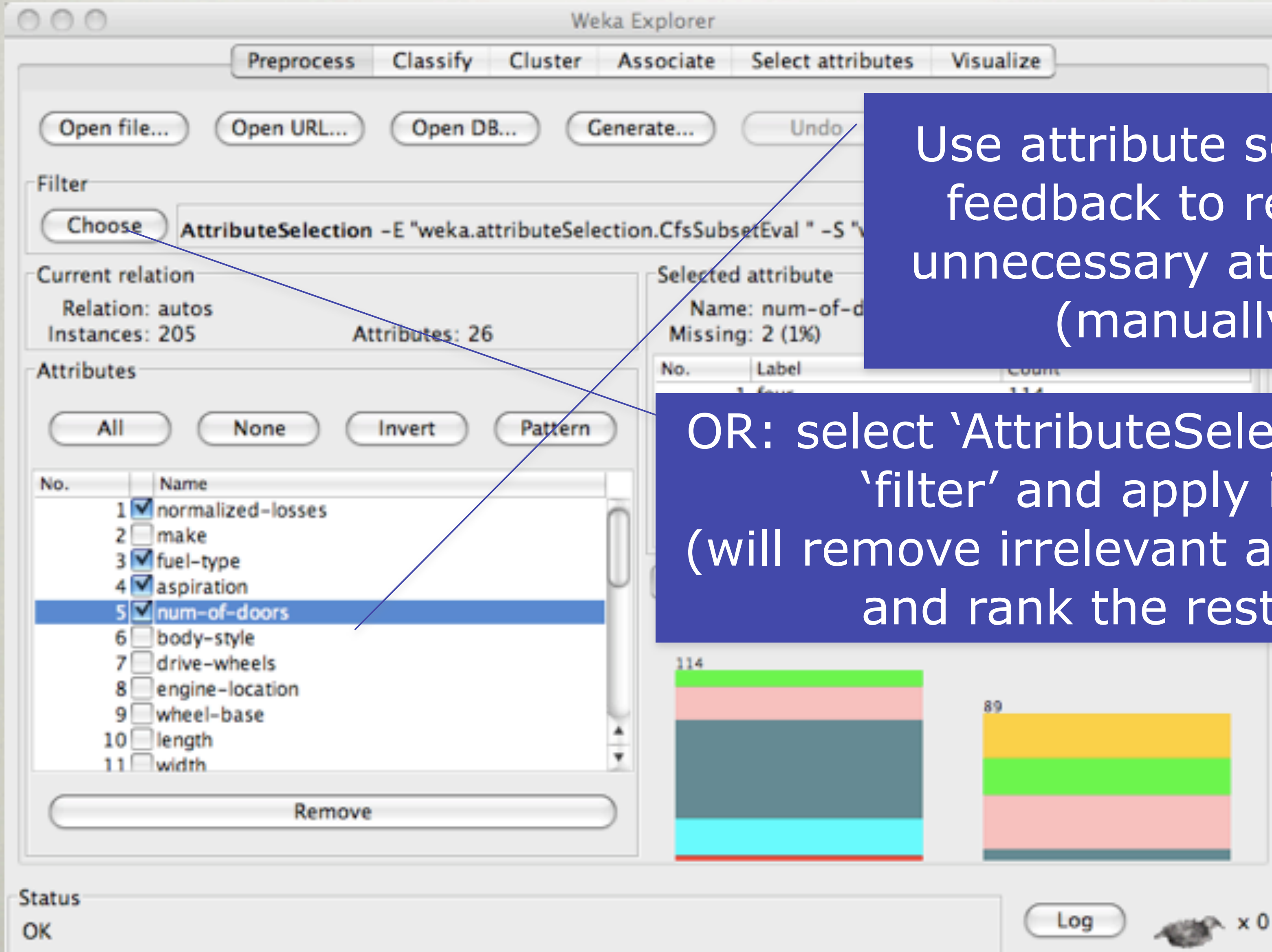
The 'Select attributes' tab



The 'Select attributes' tab



The 'Preprocess' tab



Use attribute selection feedback to remove unnecessary attributes (manually)

OR: select 'AttributeSelection' as 'filter' and apply it
(will remove irrelevant attributes and rank the rest)

Data Engineering

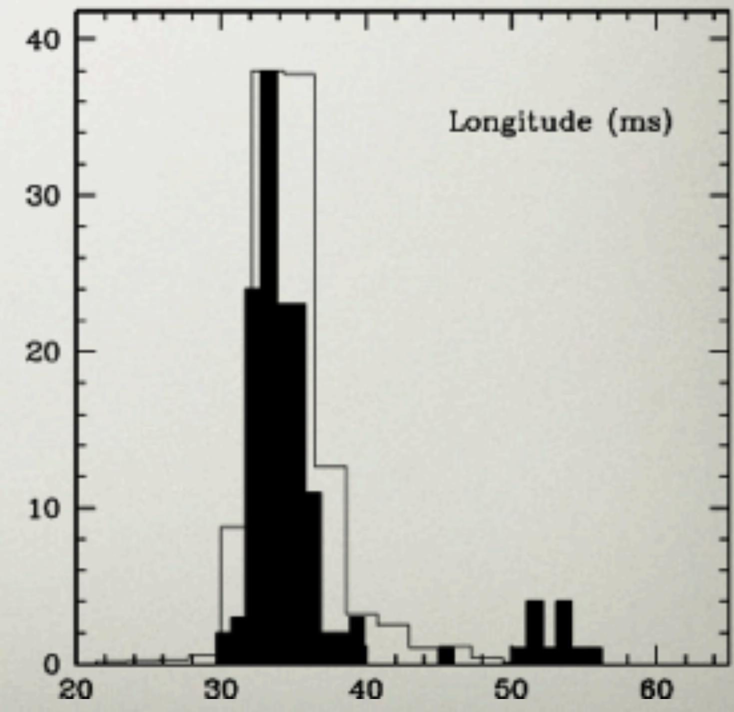
- Attribute selection (feature selection)
  - Remove features with little/no predictive information
- **Attribute discretization**
  - **Convert numerical attributes to nominal ones**
- Data transformations (feature generation)
  - Transform data to another representation
- Dirty data
  - Remove missing values or outliers

Attribute discretization

- Some learners cannot handle numeric data
  - ‘Discretize’ values in small intervals
  - Always looses information: try to preserve as much as possible
- Some learners can handle numeric values, but are:
  - Naive (Naïve Bayes assumes normal distrubution)
  - Slow (1R *sorts* instances before discretization)
  - Local (C4.5 discretizes in nodes, on less and less data)
- Discretization:
  - Transform into one *k*-valued discretized attribute
  - Replace with *k*−1 new **binary** attributes
    - values a,b,c: a→{0,0}, b→{1,0}, c→{1,1}

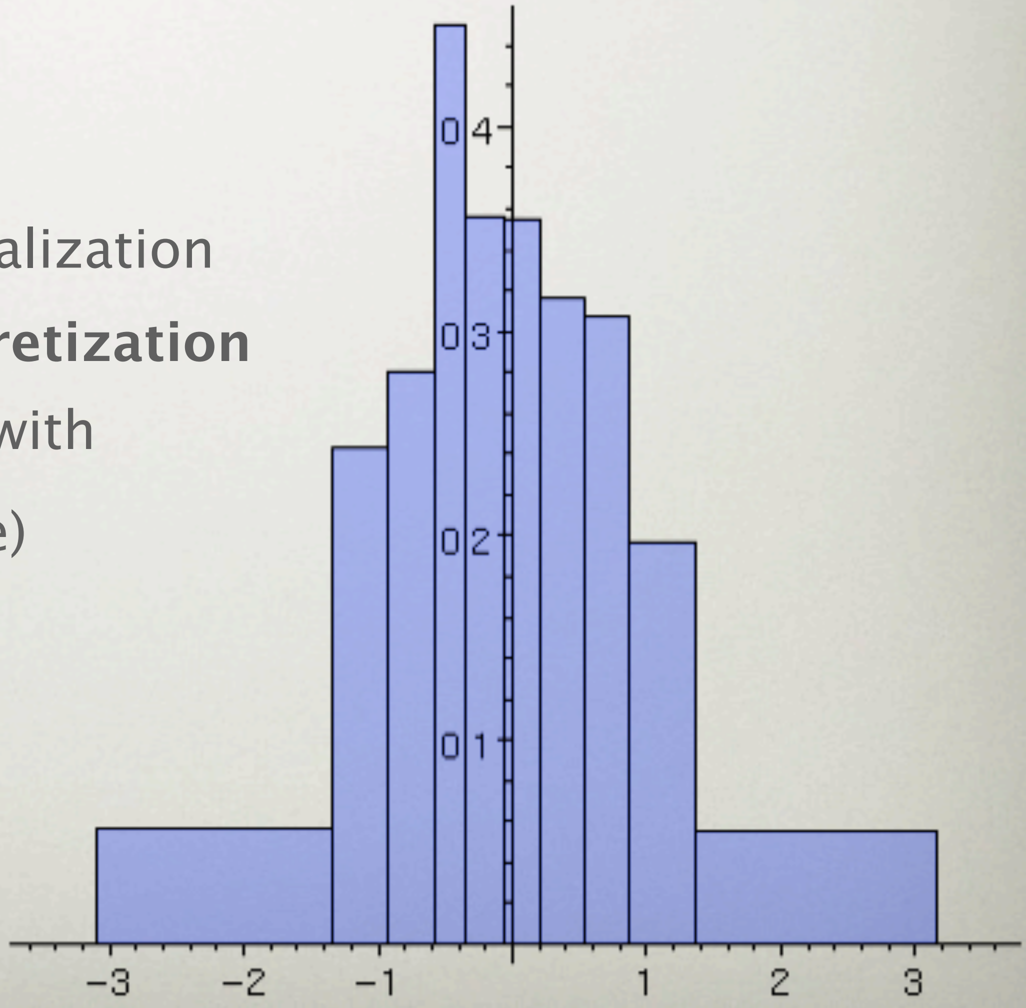
Unsupervised Discretization

- Determine intervals without knowing class labels
  - When clustering, the only possible way!
- Strategies:
  - **Equal-interval binning**: create intervals of fixed width
    - often creates bins with many or very few examples



Unsupervised Discretization

- Strategies:
  - **Equal-frequency binning**:
    - create bins of equal size
    - also called histogram equalization
  - **Proportional k-interval discretization**
    - equal–frequency binning with
    - # bins = sqrt(dataset size)



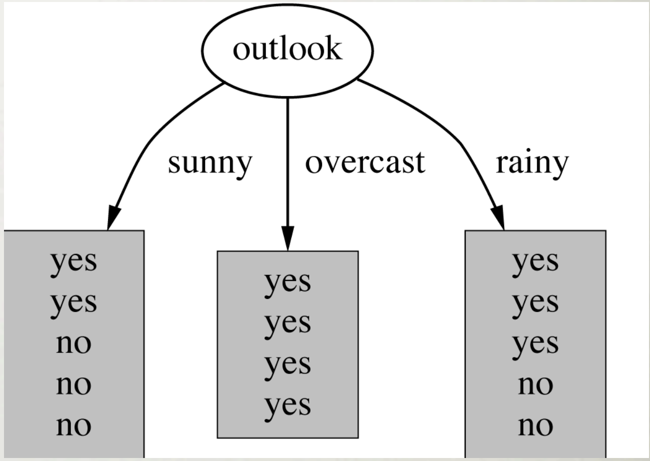
Supervised Discretization

- Supervised approach usually works better
  - Better if all/most examples in a bin have same class
  - Correlates better with class attribute (less predictive info lost)
- Different approaches
  - Entropy-based
  - Bottom-up merging
  - ...

Entropy-based Discretization

Split data in the same way C4.5 would: each leaf = bin

- Use entropy as splitting criterion



$$H(p) = -p\log(p) - (1-p)\log(1-p)$$

Outlook = Sunny:

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -2/5\log(2/5) - 3/5\log(3/5) = 0.971\text{ bits}$$

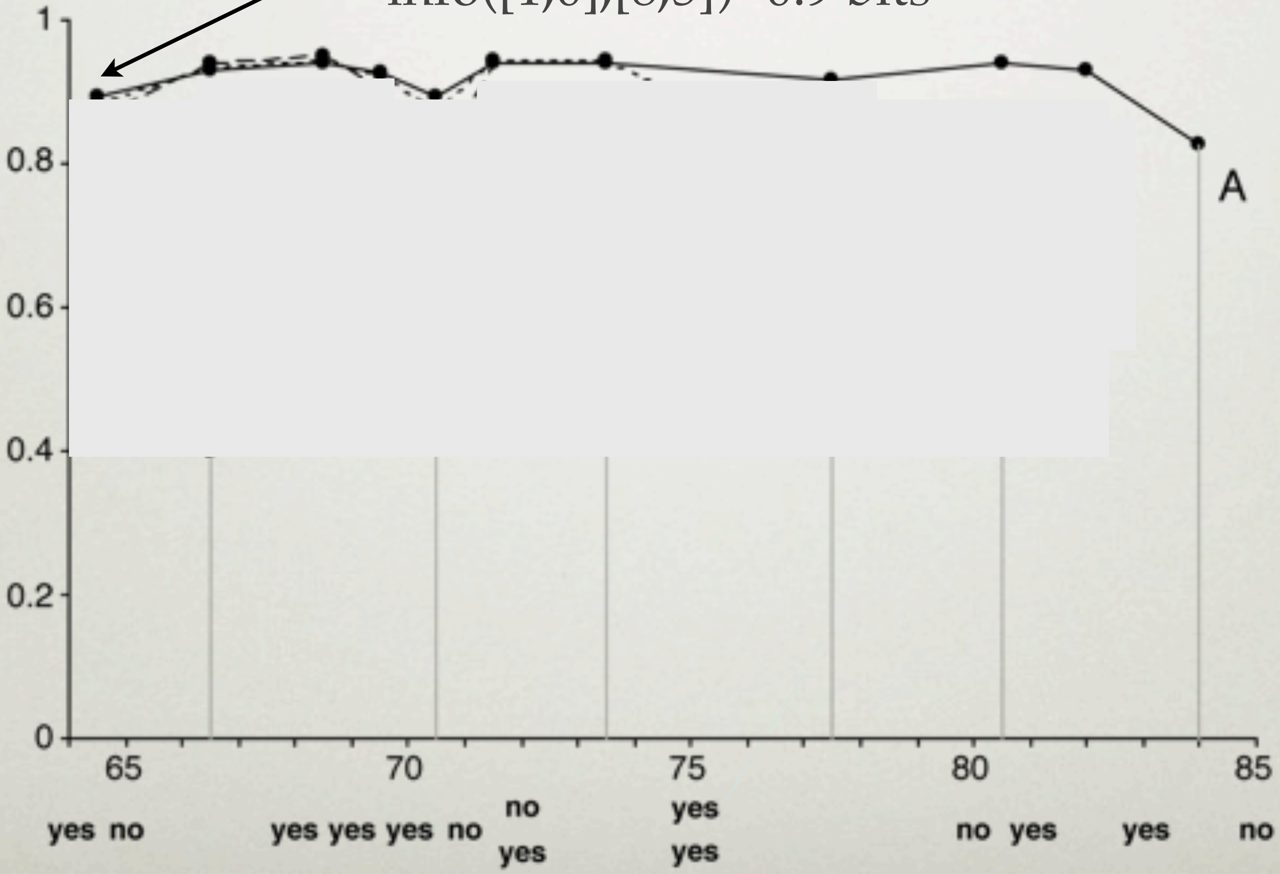
Expected information for outlook:

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$
$$= 0.693\text{ bits}$$

Example: temperature attribute

Temperature	64	65	68	69	70	71	72	72	75	75	80	81	83	85
Play	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

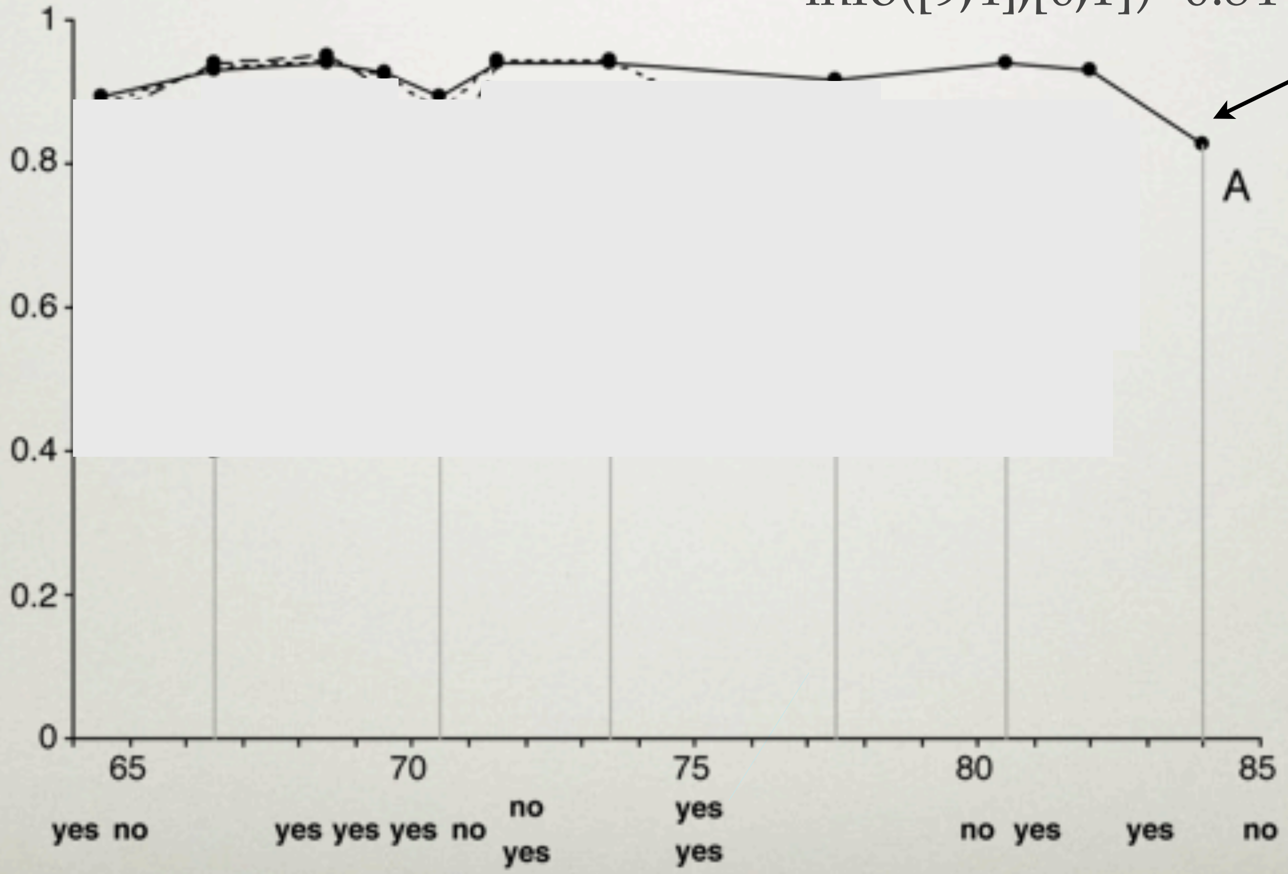
info([1,0],[8,5])=0.9 bits



Example: temperature attribute

Temperature	64	65	68	69	70	71	72	72	75	75	80	81	83	85
Play	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

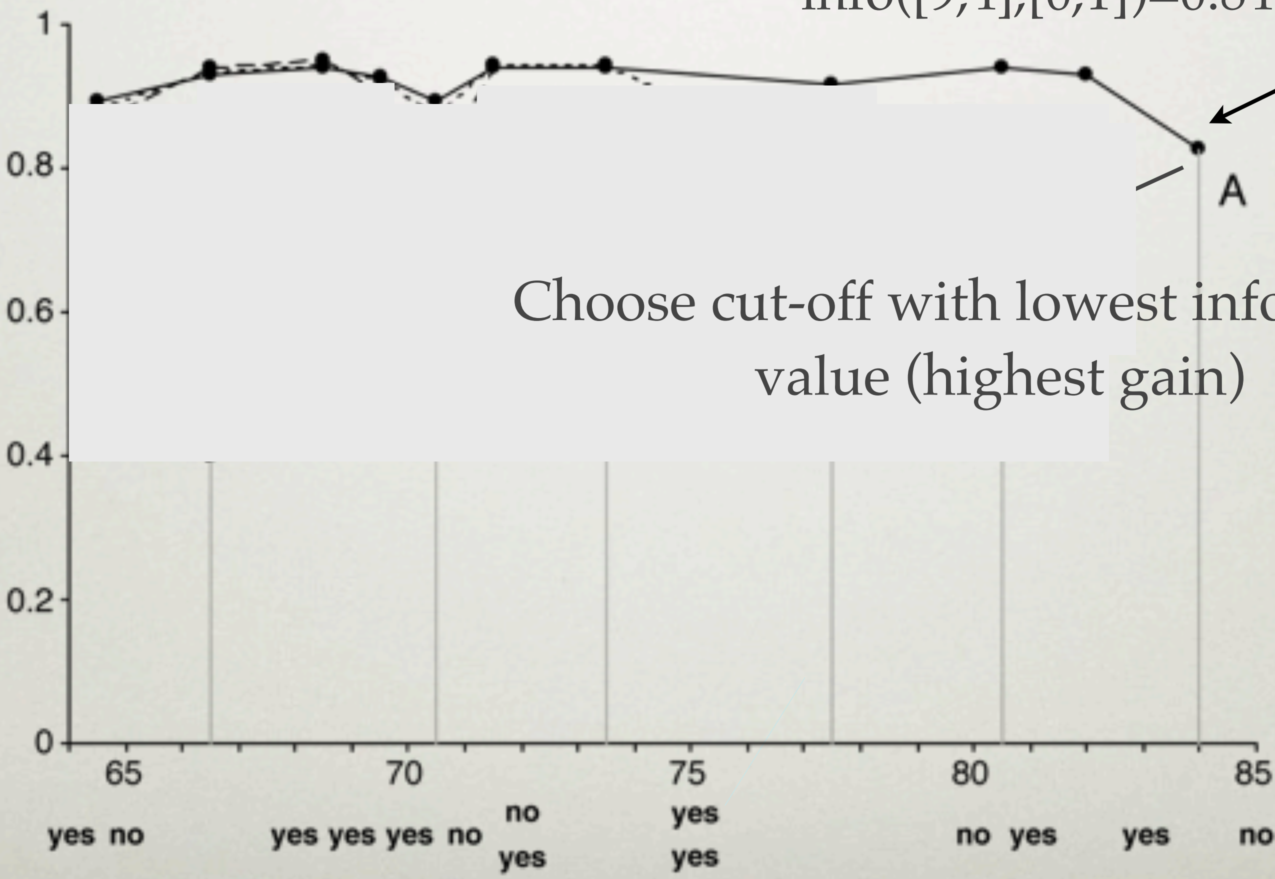
info([9,4],[0,1])=0.84 bits



Example: temperature attribute

Temperature	64	65	68	69	70	71	72	72	75	75	80	81	83	85
Play	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

info([9,4],[0,1])=0.84 bits

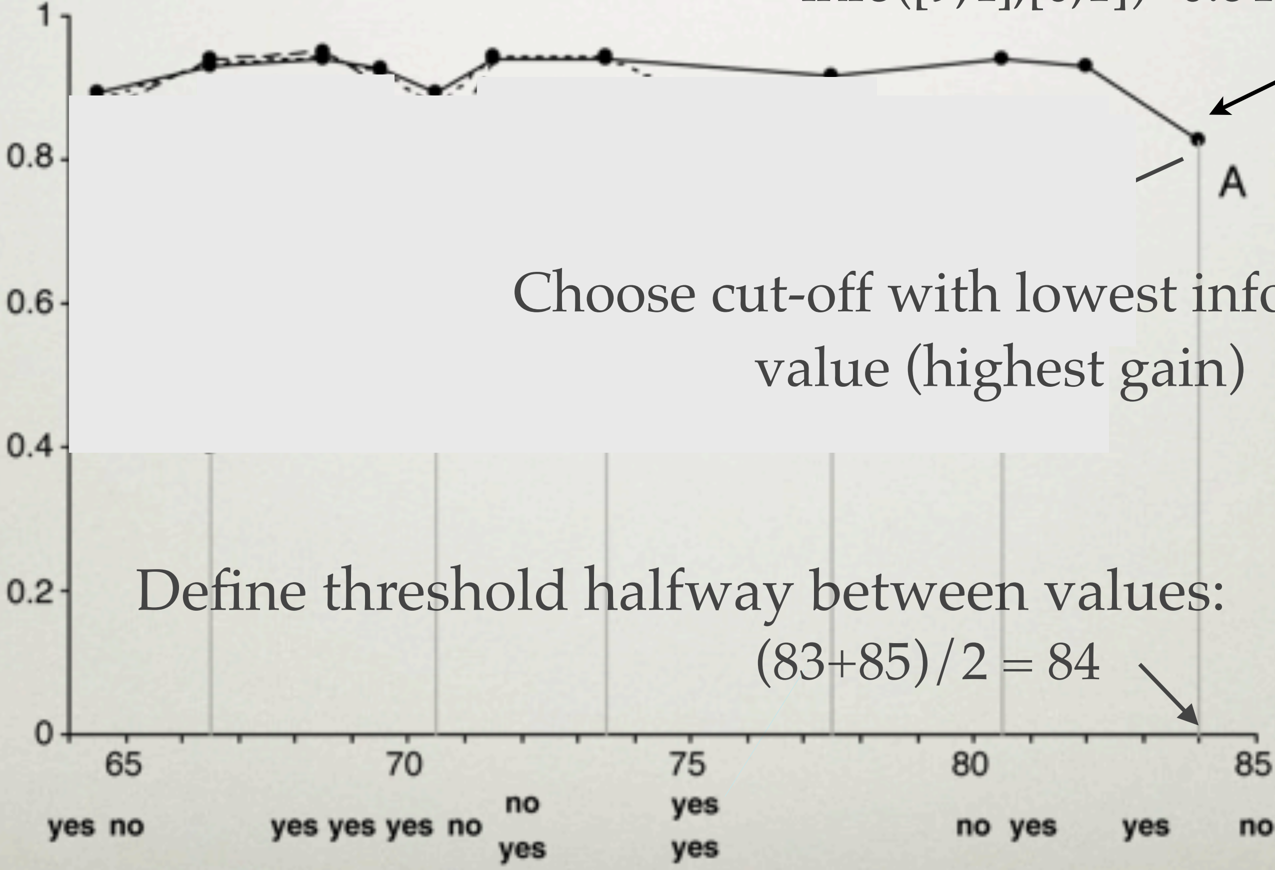


Choose cut-off with lowest information value (highest gain)

Example: temperature attribute

Temperature	64	65	68	69	70	71	72	72	75	75	80	81	83	85
Play	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

info([9,4],[0,1])=0.84 bits



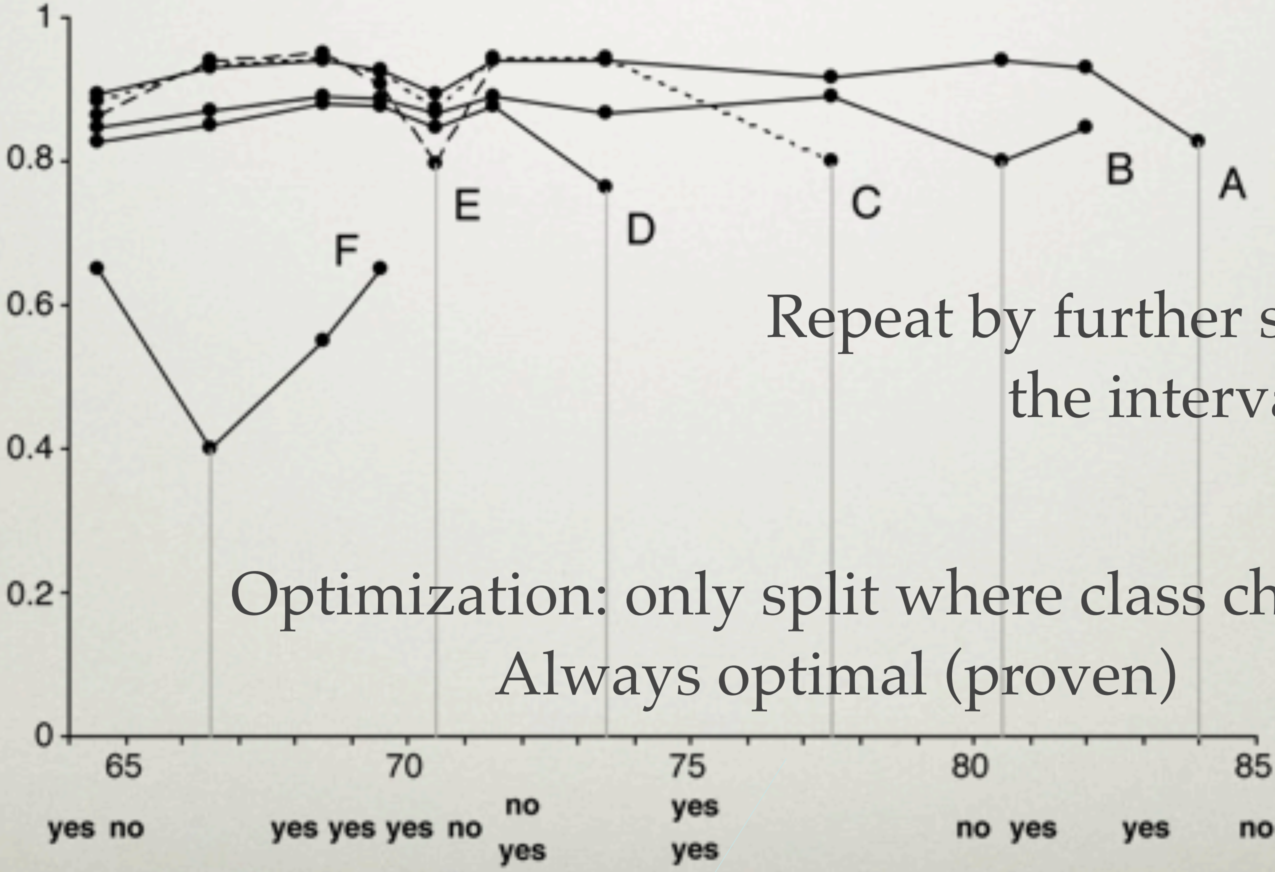
Choose cut-off with lowest information value (highest gain)

Define threshold halfway between values:

(83+85)/2 = 84

Example: temperature attribute

Temperature	64	65	68	69	70	71	72	72	75	75	80	81	83	85
Play	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No



Repeat by further subdividing the intervals

Optimization: only split where class changes

Always optimal (proven)

Entropy-based Discretization

Split data in the same way C4.5 would: each leaf = bin

- Use entropy as splitting criterion
- Use minimum description length principle as stopping criterion
  - Stop when description of attribute cannot be compressed more
    - Description of splitting points ($\log_2[N - 1]$ bits) + Description of bins (class distribution)
  - Short if few thresholds, homogenous (single-class) bins
  - Split worthwhile if information gain >

$$\frac{\log_2(N-1)}{N} + \frac{\log_2\left(3^k - 2\right) - kE + k_1E_1 + k_2E_2}{N}$$

Entropy E, number of classes k in original set (E,k), subset before threshold (E_1,k_1), after threshold (E_2,k_2)

Supervised Discretization: Alternatives

- Work bottom-up: each value in its own bin, then merge
  - Replace MDL by chi-squared test
  - Tests hypothesis that two adjacent intervals are independent of the class. If so, merge the intervals.
- Use dynamic programming to find optimum k-way split for given additive criterion
  - Requires time quadratic in the number of instances
  - Can be done in linear time if error rate is used (not entropy)

Make data numeric

- Inverse problem
- Some algorithms assume numeric features
  - e.g. kNN
- Classification
  - You could just number nominal values 1..k (a=0,b=1,c=2,...)
    - However, there isn't always a logical order
  - Replace attribute with k nominal values by k binary attributes ('indicator attributes')
  - Value '1' if example has nominal value corresponding to that indicator attribute, '0' otherwise: A→{1,0}, B→{0,1}

A
a
b

→

Aa	Ab
1	0
0	1

Make data numeric

- Regression
  - Value = average of all target values corresponding to same nominal attribute value

A	target
a	0.9
a	0.8
b	0.7
b	0.6

$\longrightarrow$

A′	target
0.85	0.9
0.85	0.8
0.65	0.7
0.65	0.6

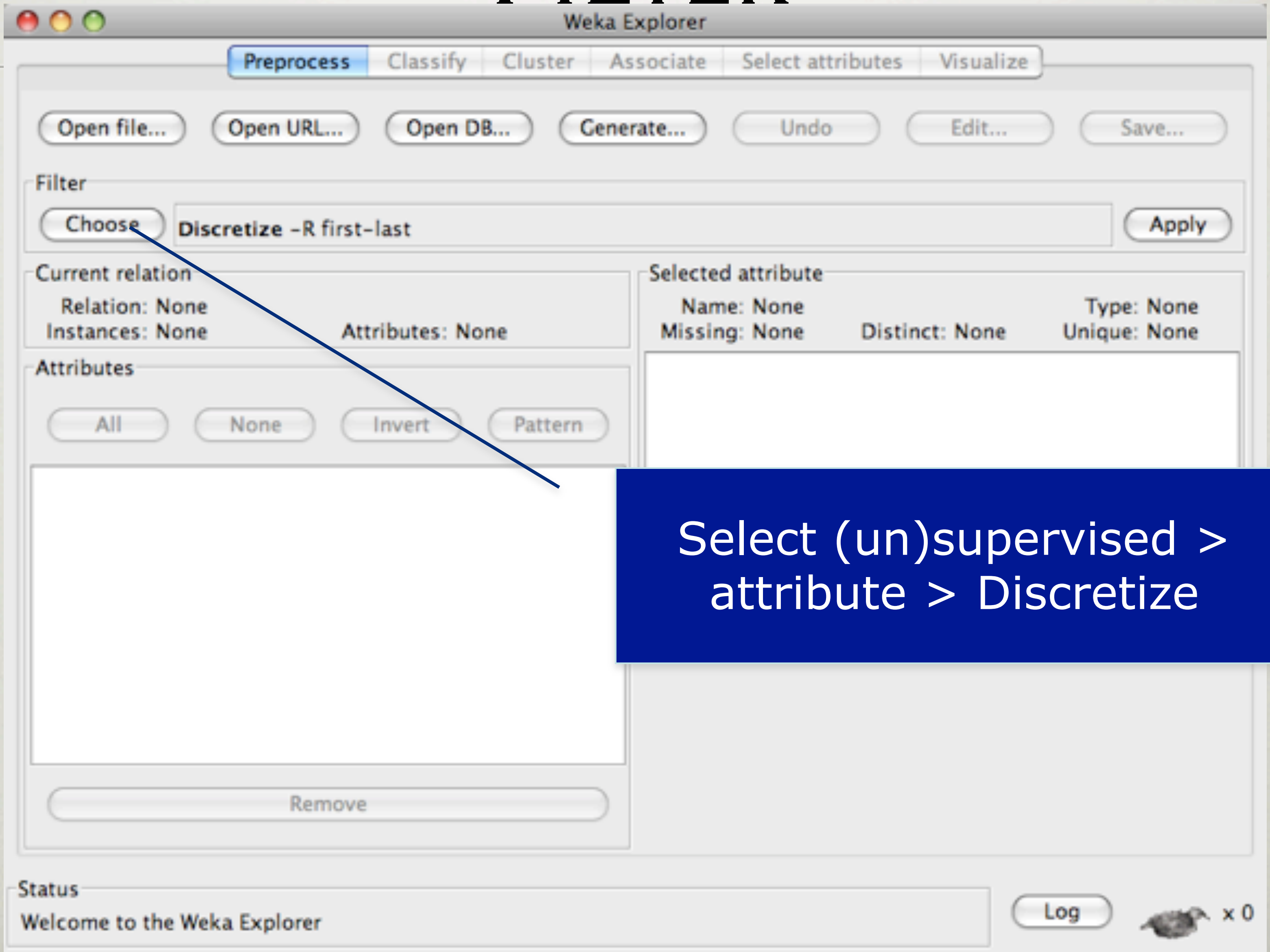
Discretization with Weka

- Discretization:
  - Unsupervised:
    - **Discretize**: Equal-width or equal-frequency
    - **PKIDiscretize**: equal-frequency with #bins=sqrt(#values)
  - Supervised:
    - **Discretize**: Entropy-based discretization

Discretization with Weka

- Nominal to numerical:
  - Supervised:
    - **NominalToBinary**: for regression (use average target value)
  - Unsupervised:
    - **MakeIndicator**: replaces nominal with boolean attribute
    - **NominalToBinary**: creates 1 binary attribute for each value

WEKA: Discretization Filter



Select (un)supervised > attribute > Discretize

Data Engineering

- Attribute selection (feature selection)
  - Remove features with little/no predictive information
- Attribute discretization
  - Convert numerical attributes to nominal ones
- **Data transformations (feature generation)**
  - **Transform data to another representation**
- Dirty data
  - Remove missing values or outliers

Data transformations

- Often, a data transformation can lead to new insights in the data and better performance
- Simple transformations:
  - Subtract two ‘date’ attributes to get ‘age’ attribute
  - If linear relationship is suspected between numeric attributes A and B: add attribute A/B
- Clustering the data
  - add one attribute recording the cluster of each instance
  - add k attributes with membership of each cluster

Data transformations

- Other transformations:
  - Add noise to data (to test robustness of algorithm)
  - Obfuscate the data (to preserve privacy)

Data transformations

- Convert text to table data
  - *Bag of words:*
    - Each instance is a document or string
    - Attributes are words, phrases, n-grams (e.g., \`to be')
    - Attribute values: term frequencies (f_{ij})
      - frequency of word i in document j

Document	f_i(to)	f_i(be)	f_i(or)	f_i(not)
\`To be or not to be'	2	2	1	1
\`Or not'	0	0	1	1

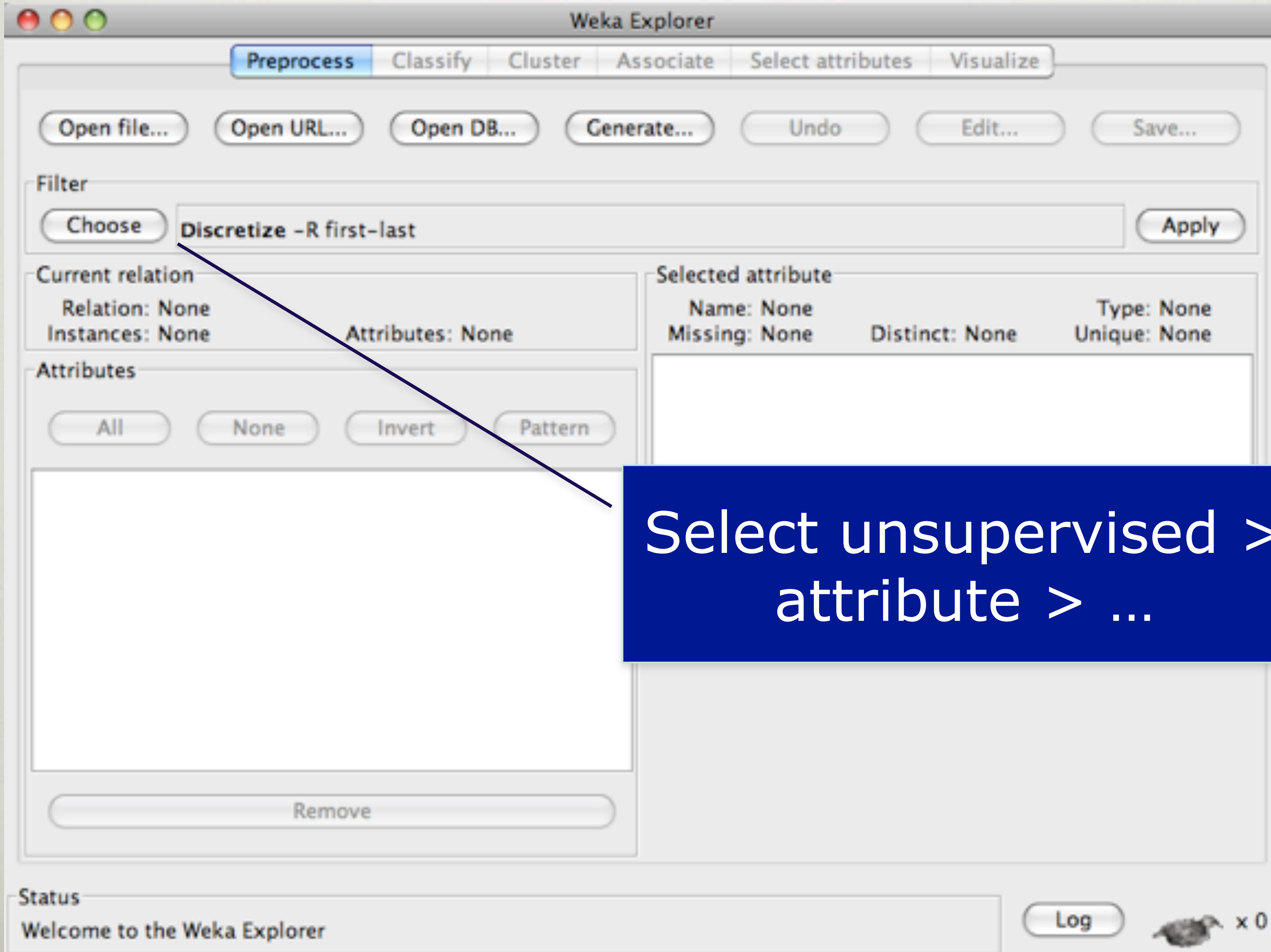
Data transformations

Document	f_i(to)	f_i(be)	f_i(or)	f_i(not)
`To be or not to be’	2	2	1	1
`Or not’	0	0	1	1

- Language-dependent issues:
  - Delimiters (ignore periods in ‘e.g.’?)
  - Stopwords (the, is, at, which, on, ...)
  - Low frequency words (ignore to reduce # features)
- Better alternatives: log(1+f_{ij}) or TFxIDF
  (*term frequency x inverse document frequency*)=

$$f_{ij} \log \frac{\#\, documents}{\#\, documents\_that\_include\_word\ i}$$

Data transformation filters



Select unsupervised > attribute > …

Some WEKA implementations

- Simple transformations:
  - **AddCluster**: clusters data and adds attribute with resulting cluster for each data point
  - **ClusterMembership**: clusters data and adds k attributes with membership of each data point in each of k clusters
  - **AddNoise**: changes a percentage of attribute's values
  - **Obfuscate**: renames attribute names and nominal/string values to random name

Some WEKA implementations

- Other transformations
  - **StringToWordVector**: produces bag of words (many options)
  - **RELAGGS**: propositionalization algorithm: converts relational data (e.g. relational database) to single table
  - **TimeSeriesDelta**: Replace attribute values with difference between current and past/previous instance
  - **TimeSeriesTranslate**: Replace attribute values with equivalent value in past/previous instance

Some WEKA implementations

- Also data projections (out of scope):
  - **PrincipalComponents**: does PCA transformation (constructs new (smaller) feature set to maximize variance per feature)
  - **RandomProjection**: Random projection to lower-dimensional subspace
  - **Standardize**: standardizes all numeric attributes to have zero mean and unit variance

Data Engineering

- Attribute selection (feature selection)
  - Remove features with little/no predictive information
- Attribute discretization
  - Convert numerical attributes to nominal ones
- Data transformations (feature generation)
  - Transform data to another representation
- **Dirty data**
  - **Remove missing values or outliers**

Some data `cleaning' methods in WEKA

- Unsupervised > Instance:
  - **RemoveWithValues**: removes instances with certain value and/or with missing values
  - **RemoveMisclassified**: removes instances incorrectly classified by specified classifier, useful for removing outliers
  - **RemovePercentage**: removes given percentage of instances
- Supervised > Instance:
  - **Resample**: produces random subsample, with replacement
  - **SpreadSubSample**: produces random subsample, with given spread between class frequencies, with replacement

Some data `cleaning' methods

- Unsupervised > Attribute:
  - **ReplaceMissingValues**: replaces all missing values for nominal /numeric attributes with mode/mean of training data