GENERATING RULES FROM ITEM SETS

- First get all of the item sets

- Example:

```
Humidity = Normal, Windy = False, Play = Yes (4)
```

- Seven $(2^N - 1)$ potential rules

```
If Humidity=Normal and Windy=False then Play=Yes  4/4
If Humidity=Normal and Play=Yes then Windy=False  4/6
If Windy=False and Play=Yes then Humidity=Normal  4/6
If Humidity=Normal then Windy=False and Play=Yes  4/7
If Windy=False then Humidity=Normal and Play=Yes  4/8
If Play=Yes then Humidity=Normal and Windy=False  4/9
If True then Humidity=Normal and Windy=False
   and Play=Yes                                    4/12
```

ASSOCIATION RULES FOR WEATHE

- Rules with support > 1 and confidence=100%

	Rule	
1	Humidity=Normal Windy=False	==> Play=Yes
2	Temperature=Cool	==> Humidity=Norma
3	Outlook=Overcast	==> Play=Yes
4	Temperature=Cold Play=Yes	==> Humidity=Norma
...	...	...
58	Outlook=Sunny Temperature=Hot	==> Humidity=High

- In Total:
  3 rules with support four
  5 with support three
  50 with support two

RULES FROM THE SAME ITEM

- Item set

  Temperature = Cool, Humidity = Normal, Windy = Fals

- Resulting rules (all with 100% confidence):

  Temperature = Cool, Windy = False **==>** Humidity =
  Temperature = Cool, Windy = False Humidity = Nor
  Temperature = Cool, Windy = False, Play = Yes **==>** H

- Due to the following 'frequent' item sets:

  Temperature = Cool, Windy = False (2) Temperature
  Normal, Windy = False (2) Temperature = Cool, Wind
  (2)

FREQUENT ITEM SETS

- A *frequent* item set is an item set that meets a previously specified minimum support/coverage
- A *large* item set is the same as a frequent item set
- Use of *large* is historical

EFFICIENT GENERATION OF ITEM SETS

- Finding one-item sets is easy

- Basic idea: Use one-item sets to generate two-item sets, two-item sets to generate three-item-sets

- Theorems:
  - If {A,B} is a frequent item set, then {A} and {B} must be frequent.
  - If X is a frequent k-item set, then all $(k-1)$ item subsets of X must be frequent.

- Compute k-item set by merging $(k-1)$ item sets

EFFICIENT GENERATION OF ASSOCIATION RULES

- Many transactions contain may items
- There may be many possible items
- Data is sparse, many items are not purchased in supermarket trip
- There may be many transactions, too much for main memory
- Finding association rules requires a lot of search
- Good data structures and algorithms are needed.
  - Still a major research area

APRIORI in WEKA

1. Set minimum support to 100%
2. Set number of rules required
3. Set minimum confidence
4. Generate rules
5. If not time to stop
   Decrease support by 5%
   Go to 4
6. Stop if
   Enough rules have been generated
   Minimum confidence is reached
   Support reaches 10%

APRIORI in WEKA

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D
Relation:     cluster1.csv
Instances:    200
Attributes:   3
              Sex
              Student
              MovieType
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.1 (20 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 18
Generated sets of large itemsets:
Size of set of large itemsets L(1): 7
Size of set of large itemsets L(2): 10
Size of set of large itemsets L(3): 4
Best rules found:
 1. Student=y MovieType=action 41 ==> Sex=m 40
    <conf:(0.98)> lift:(1.82) lev:(0.09) [18] conv:(9.53)
 2. MovieType=action 86 ==> Sex=m 82
    <conf:(0.95)> lift:(1.78) lev:(0.18) [35] conv:(8)
 3. Student=n MovieType=action 45 ==> Sex=m 42
    <conf:(0.93)> lift:(1.74) lev:(0.09) [17] conv:(5.23)
 4. Sex=f Student=y 48 ==> MovieType=romance 44
   <conf:(0.92)> lift:(1.95) lev:(0.11) [21] conv:(5.09)
```

GENERATED ITEM SETS

```
Size of set of large itemsets L(1): 7
Large Itemsets L(1):
Sex=f 93
Sex=m 107
Student=n 97
Student=y 103
MovieType=action 86
MovieType=horror 20
MovieType=romance 94
Size of set of large itemsets L(2): 10
Large Itemsets L(2):
Sex=f Student=n 45
Sex=f Student=y 48
Sex=f MovieType=romance 82
Sex=m Student=n 52
Sex=m Student=y 55
Sex=m MovieType=action 82
Student=n MovieType=action 45
Student=n MovieType=romance 45
Student=y MovieType=action 41
Student=y MovieType=romance 49
Size of set of large itemsets L(3): 4
Large Itemsets L(3):
Sex=f Student=n MovieType=romance 38
Sex=f Student=y MovieType=romance 44
Sex=m Student=n MovieType=action 42
Sex=m Student=y MovieType=action 40
```

ASSOCIATIONS NOT ALWAYS USEFUL

```
Apriori
=======

Minimum support: 0.95 (4396 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 1

Generated sets of large itemsets:

Size of set of large itemsets L(1): 5

Size of set of large itemsets L(2): 9

Size of set of large itemsets L(3): 6

Size of set of large itemsets L(4): 1

Best rules found:

 1. mutton=f 4604 ==> salads=f 4598    <conf:(1)
 2. cigarette cartons=f 4590 ==> salads=f 4584    <conf:(1)
 3. cigarette cartons=f mutton=f 4567 ==> salads=f 4561    <conf:(1)
 4. brushware=f 4518 ==> salads=f 4512    <conf:(1)
 5. brushware=f mutton=f 4495 ==> salads=f 4489    <conf:(1)
 6. cigarette cartons=f brushware=f 4481 ==> salads=f 4475    <conf:(1)
 7. cigarette cartons=f brushware=f mutton=f 4458 ==> salads=f 4452    <conf:(1)
 8. casks white wine=f 4453 ==> salads=f 4447    <conf:(1)
 9. mutton=f casks white wine=f 4430 ==> salads=f 4424    <conf:(1)
10. cigarette cartons=f casks white wine=f 4416 ==> salads=f 4410 <conf:(1)
```

If they didn't buy mutton they didn't buy salads

MORE USEFUL OUTPUTx

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.
Relation:     supermarket
Instances:    4627
Attributes:   217
              [list of attributes omitted]
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.15 (694 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:
Size of set of large itemsets L(1): 44
Size of set of large itemsets L(2): 380
Size of set of large itemsets L(3): 910
Size of set of large itemsets L(4): 633
Size of set of large itemsets L(5): 105
Size of set of large itemsets L(6): 1

Best rules found:
 1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723
 2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696
 3. baking needs=t frozen foods=t fruit=t total=high 770 ==> bread and cake=t 705
 4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746
 5. party snack foods=t fruit=t total=high 854 ==> bread and cake=t 779
 6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725
 7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701
 8. biscuits=t fruit=t total=high 954 ==> bread and cake=t 866
 9. frozen foods=t fruit=t vegetables=t total=high 834 ==> bread and cake=t 757
10. frozen foods=t fruit=t total=high 969 ==> bread and cake=t 877
```

RULE METRICS

Confidence The percentage of times the consequent appears with antecedent.

Lift $\frac{confidence}{support}$

How much better than statistical independence.

Comes from direct marketing. If the response rate for all the data is 5% but rule finds a segment with a response rate of 20% the lift of the segment is 4.0 (20%/5%).

Leverage Based on statistical properties

Conviction Alternative measure

Support Percentage of transactions/records to which the rule applies.

RANDOM NUMBER GENERATORS

Have you noticed?

- Every time you run SimpleKmeans you get the same result
- BUT Kmeans is supposed to start with random initial cluster centres
- SO the result is likely to be different each time

Here's how random numbers are generated

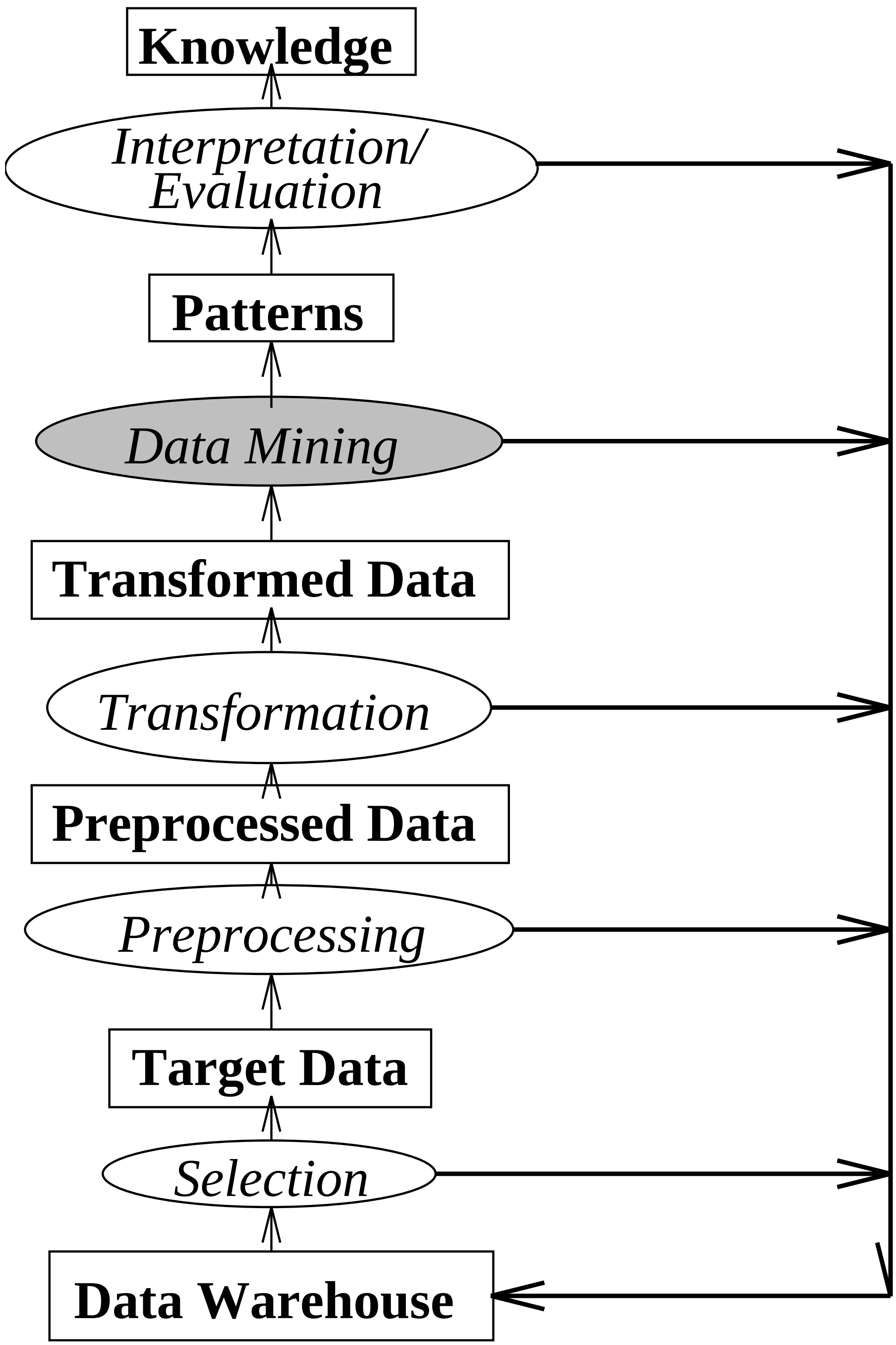
1. Start with a (random number) seed
2. Generate a random number from the seed
3. Generate the next random number from the current one
4. Go to 3

- This means every time you run the program you get the same sequence of random numbers
- If you want different random numbers, change the seed.
- Some applications set the seed from the clock so the numbers will be different every time.

COSC2110/COSC2111
Data Mining

Vic Ciesielski
Department of Computer Science
RMIT
vic.ciesielski@rmit.edu.au
14.08.16

KNOWLEDGE DISCOVERY IN DATA BASES



ATTRIBUTE/FEATURE SELECTION

- Some problems, eg micro array data, have many attributes and few examples
- Find the most important attributes
- Eliminate irrelevant or redundant attributes
- Dimensionality reduction
- Adding a random (i.e. irrelevant) attribute can significantly degrade J48's performance
- Why? Attribute selection based on smaller and smaller amounts of data. Chance of regularity in random data increases
- IBK very susceptible to irrelevant attributes
  - Number of training instances required increases exponentially with number of irrelevant attributes
  - Not all schemes have this problem, eg Naive Bayes

APPROACHES TO ATTRIBUTE SELECTION

- Manually by domain expert (Not always best)
- Scheme Independent (Filter)
  - Assess attribute based on general characteristics of the data
- Scheme Dependent (Wrapper)
  - Learning method is part of procedure
  - Search through combinations of attributes

SCHEME INDEPENDENT (FILTER)

– Use properties of a machine learning algorithm

  * Get decision tree. Any attributes not in the tree are irrelevant

  * Successively use OneR

  * Get linear model from SVM or logistic regression and pick attributes with highest coefficients

  * Get a linear model from linear regression and pick attributes with the highest weights

  * Evolve a genetic program classifier. Any attributes not in the program are irrelevant

– Use correlation

  * Select attributes that correlate well with class but not with each other (CfsSubsetEval, Correlation-based Feature Subset Evaluation)

  * Symmetric Uncertainty based on Entropy

SCHEME DEPENDENT (WRAPPER)

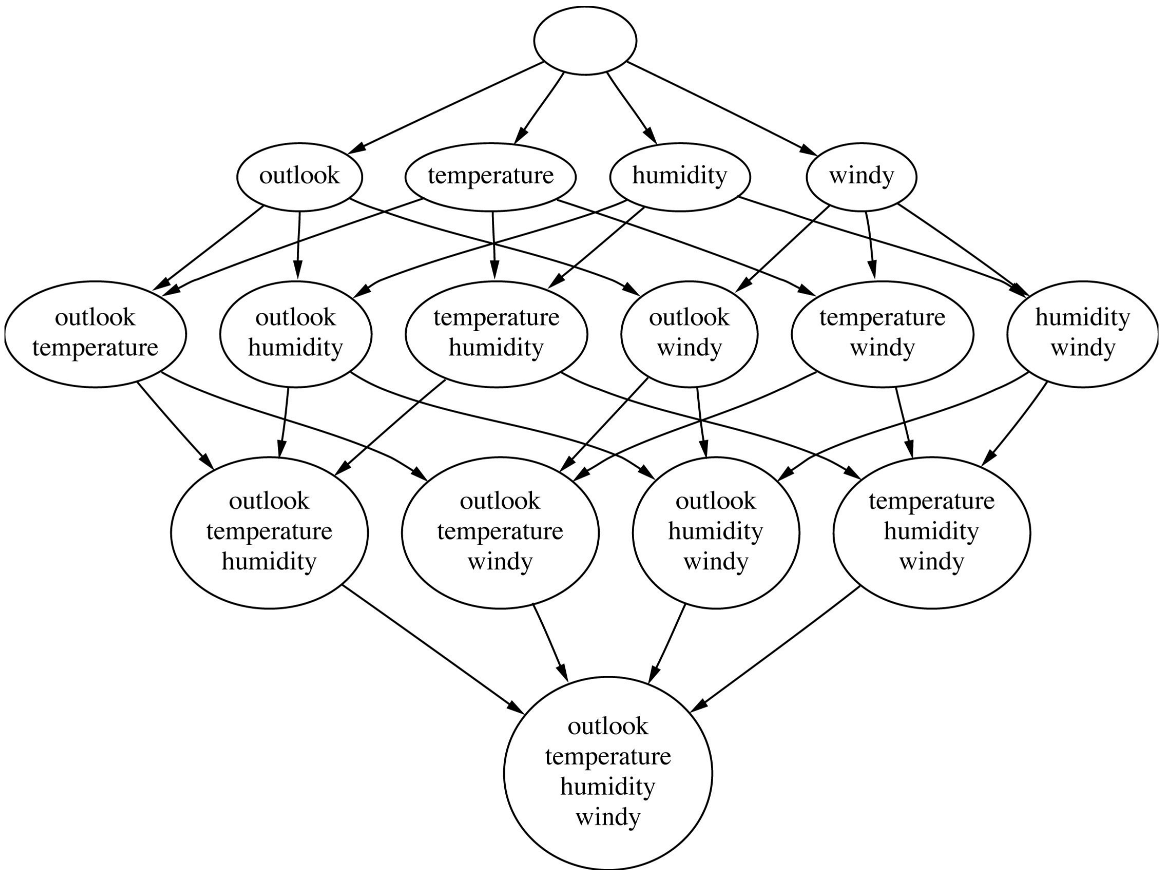
1. Pick a combination of attributes

2. Run the classifier, get accuracy, remember the best

3. If not time to stop go to 1 Stopping condition

   - Exhaustively examined all combinations
   - Examined combinations according to a heuristic search

ATTRIBUTE SEARCH



- Number of combinations is exponential in number of attributes, N
- $2^N - 1$ combinations
- Exhaustive search not possible

ATTRIBUTE SEARCH

- Forward selection (Greedy Search)
  1. Chosen set = empty
  2. For each unselected attribute
     Temporarily add to chosen set, get accuracy
     Add the best one to chosen set
  3. If unselected attributes remain go to 2
- Backward selection (Greedy Search)
  - Start will all attributes chosen
  - Eliminate the worst one
- Other AI search techniques are possible
  - Best First
  - Beam Search
  - Genetic Search
- Backward search gives more accurate classifiers
- Forward search gives more understandable classifiers

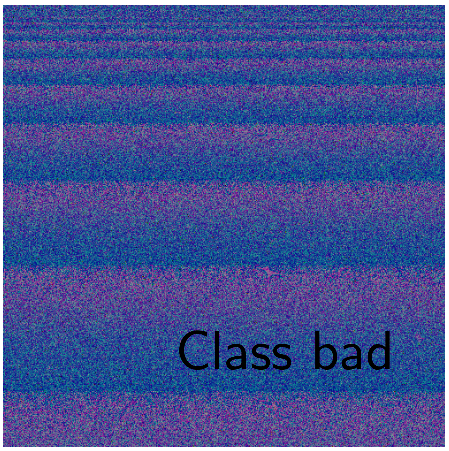
DETECTING ANOMALIES

- Anomaly: Data doesn't fit some expected pattern
  - Outlier
  - Error
- There are statistical and algorithmic methods
- Sometimes visualisation helps
- Data mining Methods
  - Automatic Approach, committee of different schemes:
    * Decision tree
    * Nearest Neighbour
    * Neural Network
  - Delete instances misclassified by all
  - But might sacrifice instances of small classes
  - Items in tiny clusters might be anomalies
- Needs domain expert

ATTRIBUTE SELECTION IN WEKA

- Both filter and wrapper methods
- Two components
  - Attribute evaluator
  - Search method
- Not all evaluators go with all search methods
- Some evaluators give a subset
- Some evaluators give a ranking of the full attribute set
- Attribute selection is not particularly useful if the number of attributes is small
- Attribute selection can be very effective when there are hundreds or thousands of attributes

EXAMPLE OF ATTRIBUTE SELECTION 1



- Which attributes (features) are important in telling the difference between bad and good?
  1. Compute a set of potentially useful features for each image
  2. Perform feature selection using all weka methods
  3. The most frequently occurring features can be associated with aesthetic value.

EXAMPLE OF ATTRIBUTE SELECTION 2

Feature	Description
F02	Earth Mover Distance from unsaturated grey (Colourfulness)
F01, F03 - F07	Average hue, saturation, brightness on all pixels and the pixels in the centre of the image
F08 - F19	Various wavelet functions used to compute levels of smoothness on different scales
F20 - F21	Image dimensions (width+height, width/height)
F22	The number of contiguous regions based on colour similarity larger than 1/100th of the total number of pixels in the image
F23 - F37	Average hue, saturation and brightness for each of the 5 largest contiguous regions of similar colours
F38 - F42	Size in pixels of each of the 5 largest regions of similar contiguous colours divided by the total number of pixels in the image
F43 - F44	Two variations on the measure of complimentary colours
F45 - F49	The location in the image of the centre of each of the 5 largest contiguous regions of similar colours
F50 - F52	Depth of field effect (emulating telephoto lens zoom) on each of the hue, saturation and brightness channels

MOST IMPORTANT ATTRIBUTES

CFS	Gain Ratio	Info Gain	OneR	Relief	Symmetric Uncert	Wrapper
F01	F30	F25	F25	F04	F30	F04
F04	F29	F41	F41	F07	F45	F18
F07	F27	F40	F26	F41	F39	F21
F13	F34	F39	F39	F25	F29	F31
F16	F28	F42	F40	F24	F35	F36
F25	F45	F44	F38	F03	F42	F41
F30	F33	F37	F31	F06	F27	
F37	F12	F43	F01	F01	F34	
F39	F39	F45	F04	F13	F37	
F40		F38	F07	F16	F25	
F42						
F45						

Number of occurrences

5 F39 5 F25 4 F41 4 F04 3 F45 3 F42 3 F40 3 F37 3 F30 3 F07 3 F01

- Golden Nugget. Only colour features are being used, no texture features.

EXAMPLE OF ATTRIBUTE SELECT

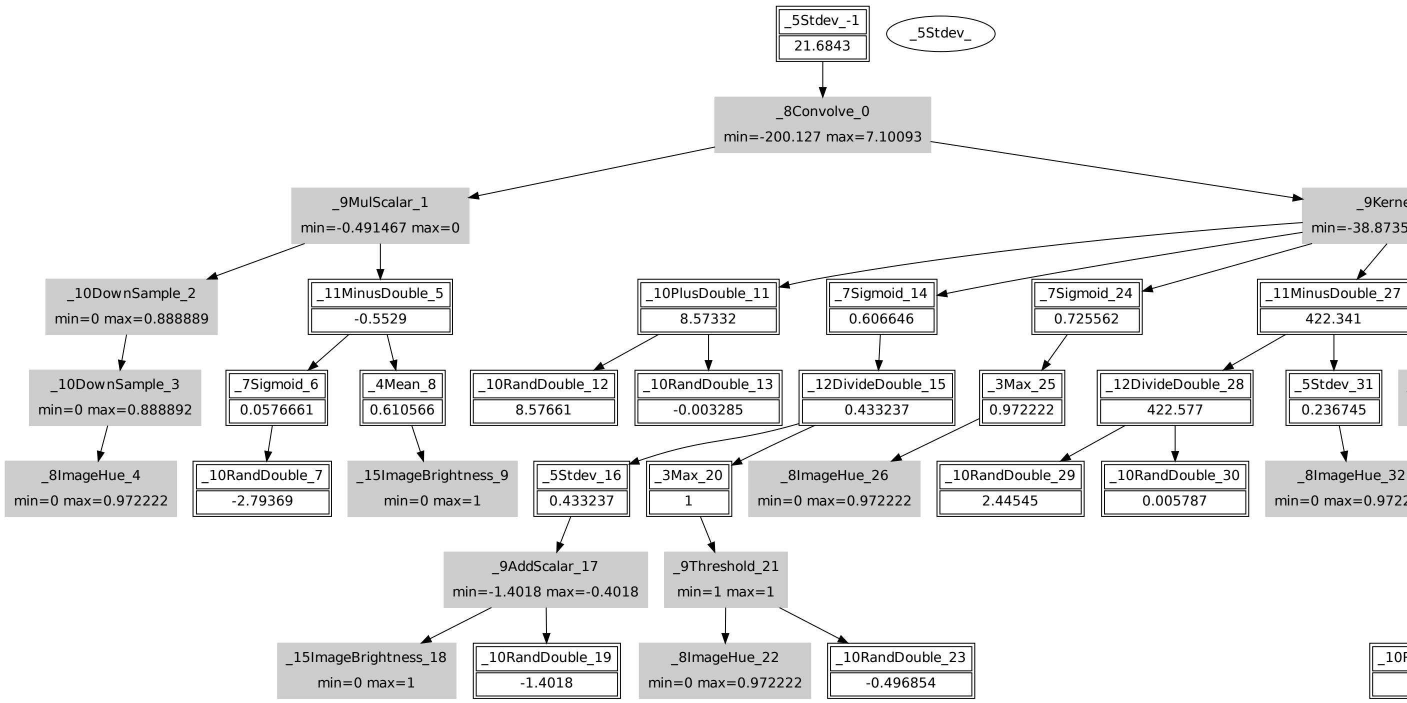
Classification accuracy of selected feature

Classifier	Full	CFS	GainRatio	InfoGain	OneR	Re
OneR	72	71	71	72	72	7
J48	87	88	75	88	87	8
Random Forest	91	92	77	88	88	9
SMO	89	83	55	79	79	8

Classification accuracy of EVOLVED featu

Classifier	Accuracy
OneR	80.3%
J48	89.7%
Random Forest	91.8%
SMO	89.9%

AN EVOLVED FEATURE



PREPARATION OF DATA FOR DATA MINING

- This lecture will be based on the files in:
  `/KDrive/SEH/SCSIT/Students/Courses`
  `/COSC2111/DataMining/`

  – `data/parking_duration_of_parking_event_vs_street_ID.csv`

    This is a file of 12,208,179 parking events in the city of Melbourne.

  – `data/parking-small.csv`

    This is a random subset of 10,000 events

  – `code-and-scripts/parking-time.sh`

    This is a script for taking the arrival date-time and generating useful features for data mining.

- Some typical data Arrival Time
  24/08/2012 11:34
  17/03/2012 13:07
  7/12/2011 19:50
  3/03/2012 14:36
  29/1/2012 12:26

PREPARATION OF DATA FOR DATA MINING

- Very bad
  - EXCEL or other spreadsheet
  - Your favourite editor
  - Interactive manual steps
- Why?
  - The procedure always needs to be done several times
  - Repeated manual steps introduce error
  - Little value in learning from erroneous data
- Very good
  - Data preparation script that can be executed repeatedly
    and independently verified for correctness
  - Uses mature utility programs

TOOLS FOR DATA MINERS

- Minimum requirement
  - cat, head, tail, cut, grep, pr, paste, sort, uniq, tr
  - Substitution with sed
  - Basic shell scripting
- To be an expert
  - Regular expressions
  - Advanced sed
  - Advanced shell scripting
  - awk or perl or python
- On a windows PC, install CYGWIN or equivalent

  Windows 10 has Ubuntu

Important Unix tools

`cat file1 file2 file3`
Concatenate files to standard output

`head -n 100 file`
Send the first 100 lines to stdout

`tail -n 10 file`
Send the last 10 lines to stdout

`cut -d',' -f7 file`
Send col 7 to stdout

`sed -e's/from/to/'|`
Stream editor: replace first occurrence in a line $from$ with to

`sed -e's+from+to+g'`
replace all occurrences $from$ with to

`paste -d, col1 col2 col3`
merge lines of files col1 col2 col3

`fgrep str file`
Get regular [fixed] expression
Send only lines that contain str to stdout

`egrep -e'RE' file`
Send only lines that contain the regular expressio
RE to stdout

```
tr ' ' ',' file
Translate characters
Change all occurrences of space to comma

tr -d'\r' file
Delete all occurrences of the return character

sort file
Sort

uniq file
Omit or count repeated lines

wc -l file
Word count, count lines (-l)
```