WPI – CS Department

Project II

CS252D - Knowledge Discovery and Data Mining

Amro Khasawneh

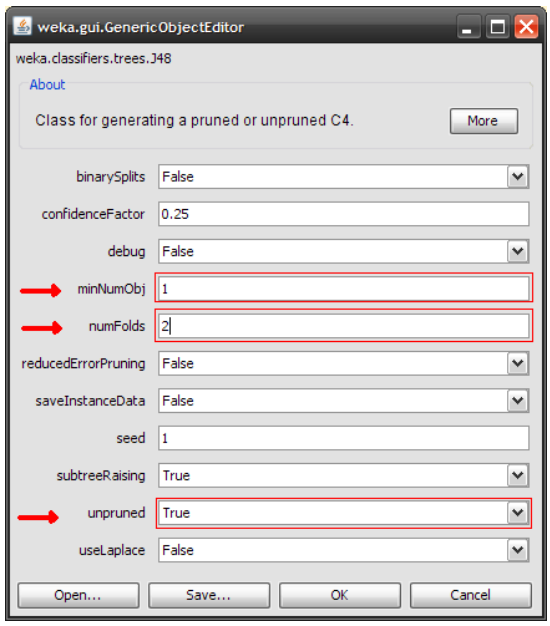
2/26/2008

PROJECT 2

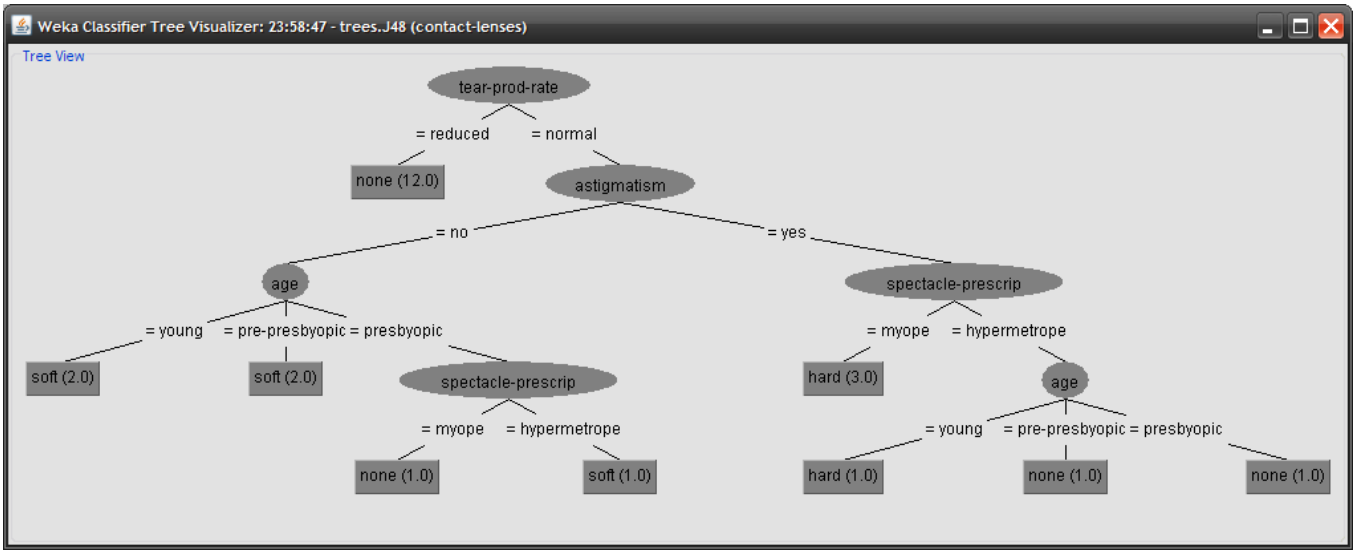
By: Amro Khasawneh

Part I: Homework part

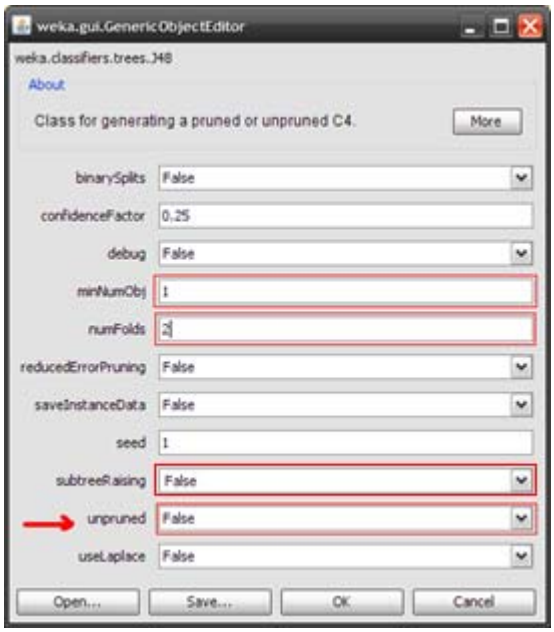
1) Running J4.8 with the specified parameters, results in the following un-pruned tree:




```
tear-prod-rate = reduced: none (12.0)
tear-prod-rate = normal
|   astigmatism = no
|   |   age = young: soft (2.0)
|   |   age = pre-presbyopic: soft (2.0)
|   |   age = presbyopic
|   |   |   spectacle-prescrip = myope: none (1.0)
|   |   |   spectacle-prescrip = hypermetrope: soft (1.0)
|   astigmatism = yes
|   |   spectacle-prescrip = myope: hard (3.0)
|   |   spectacle-prescrip = hypermetrope
|   |   |   age = young: hard (1.0)
|   |   |   age = pre-presbyopic: none (1.0)
|   |   |   age = presbyopic: none (1.0)

Number of Leaves  :    9
Size of the tree :    15

Time taken to build model: 0 seconds

=== Evaluation on training set ===
Correctly Classified Instances          24       100%
Incorrectly Classified Instances         0         0%
```



2) Now we enable pruning as showed below, and receive the following:

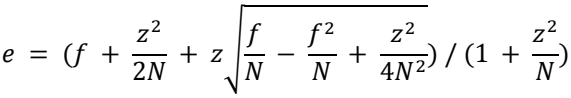



```
tear-prod-rate = reduced: none (12.0)
tear-prod-rate = normal
|   astigmatism = no: soft (6.0/1.0)
|   astigmatism = yes
|   |   spectacle-prescrip = myope: hard (3.0)
|   |   spectacle-prescrip = hypermetrope: none (3.0/1.0)

Number of Leaves  :    4
Size of the tree :    7

Time taken to build model: 0.03 seconds

=== Evaluation on training set ===
Correctly Classified Instances        22   91.6667 %
Incorrectly Classified Instances       2    8.3333 %
```

Between parentheses is the number of instances that reach that leaf (decimal because C4.5 uses fractional instances to handle missing values) followed by incorrectly classified instances.

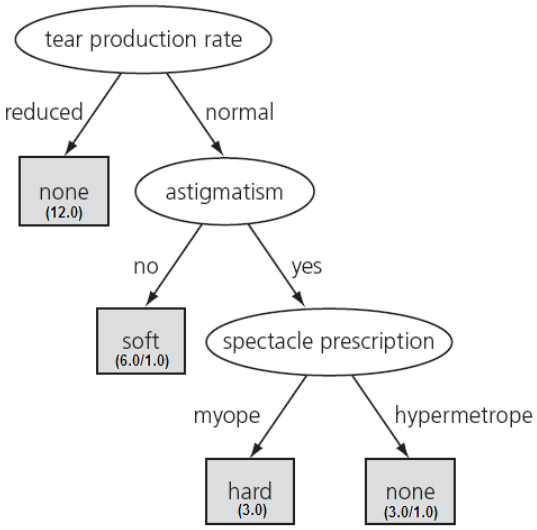
Note that we disabled subtree raising which is the other pruning technique that C4.5 uses.

Now by looking at the implementation of Weka, we try next to trace the pruning algorithm which takes place after the whole decision tree is generated.
It starts by calling the prune() function on the root node of the tree and proceeds recursively on its children, traversing the tree in a prefix order (parent, left, right). This process looks as if it proceeds from the leaves and works back up toward the root. Once a subtree is entirely traversed, an error estimate is calculated for its root as if it were a leaf node (using majority class) in addition to a weighted average error estimate for the children. These two are compared and pruning decision is made accordingly.
The formula for the error estimate is:

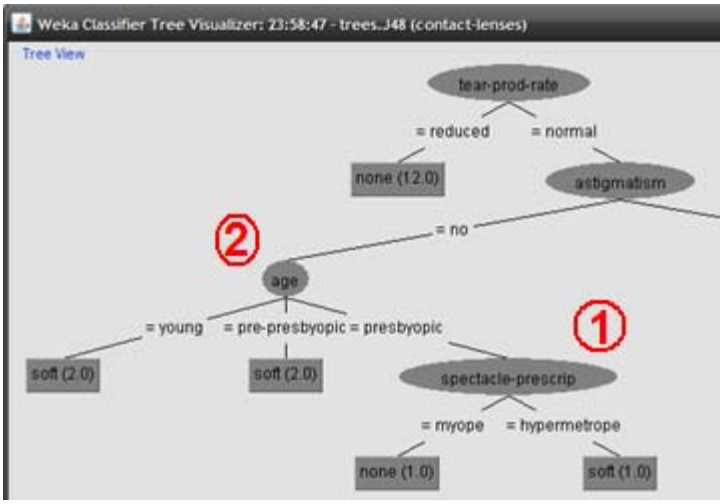


$$e = (f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}) / (1 + \frac{z^2}{N})$$

With confidence interval c=25% then z=0.69
and $f = E/N$, where E is the number of wrongly classified instances out of N all the instances that reached that node.
It is worth mentioning that Weka has some slight differences in that it computes $= (E + 0.5)/N$.

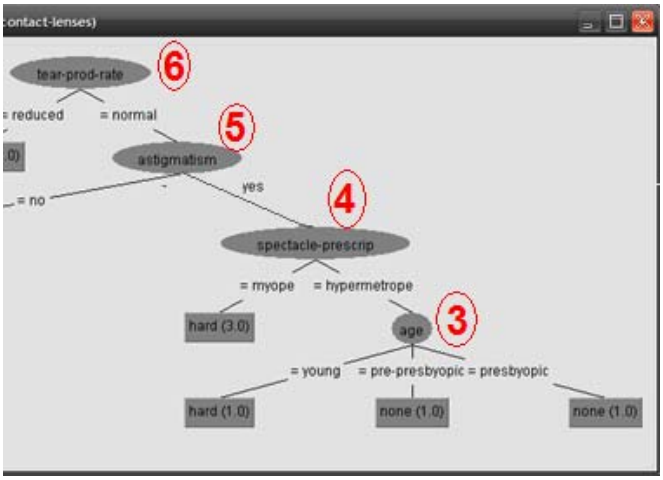


Now in our case, the first subtree considered is rooted with spectacle-prescription (numbered 1). We first look at the root node. It has N=2 instances that reach it one classified as none and the other as soft. Using the majority class it has E=1, therefore f=0.5 putting that into the equation yields: *e=0.719*
Now each of the children has one instance, so each has N=1, E=0, giving us *e= 0.322* each. Taking the weighted sum gives:
*e=1*0.322 + 1*0.322 = 0.644*
Now since the children has lower error estimate, we don't prune.



Now another difference in the way that Weka implements J4.5 is that the pruning evaluation still continues in this branch even though this subtree was not pruned.

Now on to the next subtree (marked 2), parent has N=6 with majority class=soft, then E=1, so f=1/6 yielding: *e=0.295*
The first two descendants are identical and has each N=2, E=0 so *e=0.192* for each. The third child which is the subtree we evaluated earlier, has error estimate equal to that of its children, namely e=0.644. Taking the weighted sum gives us e=2*0.192 + 2*0.192 + 2*0.644 = *2.056*
Now since the parent has smaller error, then we prune this subtree, and replace it by a single leaf node with class = soft.



Now reaching the other branch of the tree, and looking at subtree number 3, its root has N=3, majority class=none giving E=1, then f=1/3, therefore *e=0.532*
Children are all homogenous with one instance each, then e=1*0.322 + 1*0.322 + 1*0.322 = *0.966*
Comparing the two values, we decide to prune, and replace the subtree with a leaf node with class = none.

Next is subtree number 4. Root has N=6, E=2, f=1/3, then *e=0.473*. Children, first N=3, E=0 giving e=0.136. Second is the previously pruned one, with e=0.532. Weighted sum is e=3*0.136 + 3*0.532 = *2.004*, which is clearly larger, and then we don't prune.

The same goes for subtrees 5 and 6 which don't get pruned. Therefore we get the complete pruned C4.5 tree.

Part II: Projects part

DESCRIPTION OF THE ALGORITHMS:

For both datasets, we will run experiments using decision trees (ID3 and C4.5). Therefore let's look at the underlying algorithms for each data mining technique before we go to our datasets.

ID3 (Iterative Dichotomiser 3) is an algorithm used to generate a classification model from data. Given a set of instances consisting of a number of attribute/value pairs and their correct classification, we build a decision tree to classify each instance and predict future samples. ID3 expects the data to have certain requirements: the attributes must be of nominal type in addition to class attribute, and there can't be instances with missing values. The construction goes as follows:

- Choose as root node the attribute A with the lowest entropy (or the highest information gain) over the whole data
- For each possible value of A, we construct a child branch. Now we repeat the process of selecting an attribute over the new set of instances until we either get a homogenous split or run out of attribute (or if we set some other stopping criterion like a threshold on the minimum information gain), and we set that leaf node to the majority class of that split.

Now ID3 suffers from the fact it can't deal with neither numeric attribute nor missing values. And it often produces an over-fitting model of the training dataset. To overcome these problems, C4.5 algorithm was introduced as an extension of ID3 trees. C4.5 has made a number of improvements to ID3. Some of these are:

- Handling both continuous and discrete attributes: In order to handle continuous attributes, C4.5 finds the best split point and then splits the instances into those whose attribute value is above the threshold and those that are less than it.
- Handling training data with missing attribute values: C4.5 allows attribute values to be marked with "?" for missing. Missing attribute values are simply not used in gain and entropy calculations.
- Post-Pruning of the tree: C4.5 goes back through the tree once it's been created and attempts to remove branches that do not help by replacing them with leaf nodes using two techniques, subtree replacement and subtree raising.

CENSUS-INCOME DATASET:

DESCRIPTION OF THE DATASET:

Available through UC Irvine Machine Learning Repository, this dataset was extracted from the US Census Bureau database and used to make prediction about whether income exceeds $50K/yr based on census data. It contains a total number of 48842 instances split into two set training/testing sets with 32561 instances in the training dataset and 16281 for the testing set. It contains 14 attributes with 6 integers and 8 nominal attributes. The dataset has 3620 instances with missing values giving a percentage of 7.4% (2399 in train, 1221 in test). The class attribute is income a boolean nominal attribute with values >50k or <=50k.

OBJECTIVES:

Now looking at the raw data, we can formulate a few questions about the domain that we would want to be answered. For example:

- What is the most accurate predictor of income among those census data?
- To what degree does the level of education determine the income class?
- Is the race or gender a predictor of the income?
- How does age relate to the person's income?
- What working sector (private/public) profits the most in term of income?

PERFORMANCE METRIC & TRAINIG AND TESTING:

Note that we will use classification accuracy of the test set as a performance metric of the model we build over the dataset. An important detail to mention is that in order to use the separate test set supplied for testing, it must match the training set format –

any transformation to the attributes (removing, discretizing, replacing values) in the training date must be performed also on the test data for it to match. In case this is not feasible or undesired, we can always test on the original data but with a 10-fold cross-validation.

PREPROCESSING OF THE DATA:

By looking at the raw data, we observe that the attributes *education* and *education-num* are 100% correlated between each other which is what their description implies; starting with a value of 1 corresponding to preschool all the way to doctorate with a value 16. Therefore it makes sense to remove one of them as its kind of repetition, and in my opinion keeping the numeric attribute is better as it is more informative in the sense that it implies an explicit order on the attribute values rather than having an implicit meaning the way it is in the nominal attribute. Note that this applies to both data mining techniques used in this project.

Then when it comes to missing values, their percentage is relatively small considering we have a huge amount of instances. Since the ID3 model we are building can't deal with missing values, we can comfortably remove all instances containing missing values using the unsupervised instance filter with the parameter *matchMissingValues* set to YES , or even apply the unsupervised attribute filter *ReplaceMissingValues* to just replace them with the mode/mean. Three attributes are mainly the source of missing values, namely: *workclass (1836), occupation (1843), and native-country (583)*.
However C4.5 is capable of dealing with missing values therefore no specific handling is required. Although we can replace them with the mode/mean just as previously mentioned.

In addition, ID3 cannot handle numeric attributes. What we can do is discretize them either in a supervised way or in an unsupervised manner in which we can do tweak the number of bins or use an equal-frequency binning.

In terms of dimensionality reduction, first thing that comes to mind, is to use the almost automatic *correlated-based feature selection (CFS)* to find a smaller subset of attribute that can form a good predictor. An initial run of CFS with default parameters returns the following subset of attributes: *education-num, marital-status, relationship, capital-gain,* and *capital-loss*. Other re-runs with different parameters such as setting the search direction from forward to backward and increasing stopping criterion still returns the same subset (I even specified *ExhaustiveSearch* as the search method). This gives us a general idea about those attributes that are more descriptive than the others. (I have also tried *PrincipalComponents PCA*, but that would just render the data too unrecognizable! That is understandable since it is not well suited for decision trees.)

EXPERIMENTING:

The very first experiment to try is classifying with *ZeroR*. This gives us a sense of what the other experiments might want to compare to. The result is <=50k being correct almost 75%-76% of the time with both train and test sets. Two things we can say about that. One is that 75% is really noticeable. Second is that the training and testing sets have somewhat the same class frequency, which make us wonder if we could just test using cross-validation would give us the same accuracies.

Just as another basic experiment, we try the *OneR* classifier. The attribute used is capital-gain with a model accuracy of 81%. By now this attribute has shown up for the second time (after trying CFS) which says it is pretty predictive of the class.

Moving on to more sophisticated models, we want to make the data ready for ID3. First we try to replace missing values with mean/mode as explained in previous section then use supervised discretization. Running the ID3 classifier with a 10 cross-validation gives us a tree highly branched with an accuracy of 79% which is a poor improvement over the 75% of ZeroR. Now retrying the same thing but with removing the instances with missing values yields a similar result. Also doing the same things with unsupervised discretization gives even more poor result.

```
Scheme:       weka.classifiers.trees.Id3
Attributes:   education-num, marital-status, relationship
              capital-gain,  capital-loss,   class
Test mode:    10-fold cross-validation

relationship = Wife
|  education-num = '(-inf-8.5]'
|  |  capital-gain = '(-inf-57]'
|  |  |  capital-loss = '(-inf-1551.5]': <=50K
|  |  |  capital-loss = '(1551.5-1568.5]': null
|  |  |  capital-loss = '(1568.5-1820.5]': null
|  |  |  capital-loss = '(1820.5-1862]': null
|  |  |  capital-loss = '(1862-1881.5]': null
|  |  |  capital-loss = '(1881.5-1923]': null
|  |  |  capital-loss = '(1923-1975.5]': null
|  |  |  capital-loss = '(1975.5-1978.5]': >50K
|  |  |  capital-loss = '(1978.5-2168.5]': <=50K
|  |  |  capital-loss = '(2168.5-2176.5]': null
   ... << TRUNCATED >> ...
```

Since the preceding didn't go so well, let's try to apply CFS first ending up with the 5 attribute mentioned earlier then discretize in a supervised manner. This time the tree is still

too long however the accuracy is much better with 85.5% correctly classified instances (see figure on the right). I believe this is as close as it's going to be with an ID3 model. (Note that the CFS step got rid of all attributes causing missing values!). This concludes our experiments with ID3.

```
Time taken to build model: 0.24 seconds

=== Summary ===
Correctly Classified Instances       27858    85.5563 %
Incorrectly Classified Instances      4564    14.0168 %
=== Confusion Matrix ===
     a     b   <-- classified as
  4500  3280 |     a = >50K
  1284 23358 |     b = <=50K
```

From the previous experiment, we do get a sense of the shortages of the ID3 algorithm in dealing with missing values and numeric attributes, in addition to the fact that the tree gets too deep and the model is very over-fitting to the training data. For this reason, C4.5 extends Id3 with an important notion: pruning. So let's start looking at building a C4.5 model.

To get a sense out of it, we will run J4.8 with the default parameters over the original data. After 5.36 seconds we get a tree of size 710 and an impressive 84.8% accuracy. Clearly the size is way too large. We should remember that the default settings for J4.8 has both subtree raising and pruning enabled using error estimates with a confidence interval c=0.25 and 2 as the min number of instances in a leaf. Maybe by tweaking these values we can shorten the tree. Since we have a large train set, we could increase the min number of instances in a leaf to say 25, and specifying a smaller confidence interval c=0.20 which would insure a stricter pruning. Surprisingly after re-running the C4.5 algorithm with the new parameter we get a much smaller tree of size 174 which slightly better accuracy of 86%. Still we wish to decrease the size and make it less over-fitted. Now even increasing from 25 to 50 to 75 to 100 gets us a close accuracy but reaching a tree size of 74 with accuracy 85.9% which is much better. However there is a classic trade-off between better accuracy and smaller tree size that we should be aware of.
Now why don't we start this from another angle, by reducing the set of attributes first? So we do this using CFS which gave us from previous experiments good results. We end up with the 5 attributes which we are by now familiar with. Then we apply J4.8 to those. (we also apply CFS to the test set and save a temp copy of it to test against it). We set c=0.20 and min_num_instances_leaf=20. The result which came back in 1 second, is a tree of size 46 with an accuracy of 85.6%. This is in my opinion a better result than the first one because we are using fewer attributes, making the tree behaves better with new unseen instances, also the tree us considerably shallower with a max depth of 6. (I also tried using the *reducedErrorPruning* in C4.5 as the other test to decide whether to prune or not, and it came with some similar but inferior results).

```
Scheme:       weka.classifiers.trees.J48 -C 0.2 -M 20
Attributes:   6
              education-num,  marital-status, relationship
              capital-gain,   capital-loss,   class
Test mode:    user supplied test set: 16281 instances

J48 pruned tree
------------------
marital-status = Married-civ-spouse
|   capital-gain <= 5060
|   |   capital-loss <= 1762
|   |   |   education-num <= 12
|   |   |   |   capital-gain <= 4416
|   |   |   |   |   capital-gain <= 4101
|   |   |   |   |   |   capital-gain <= 3103
|   |   |   |   |   |   |   capital-gain <= 2993: <=50K (9255.0/2591.0)
|   |   |   |   |   |   |   capital-gain > 2993: >50K (63.0/5.0)
|   |   |   |   |   |   capital-gain > 3103: <=50K (177.0)
|   |   |   |   |   capital-gain > 4101: >50K (56.0/11.0)
|   |   |   |   capital-gain > 4416: <=50K (58.0)
|   |   |   education-num > 12
|   |   |   |   capital-gain <= 3103
|   |   |   |   |   capital-loss <= 625
|   |   |   |   |   |   relationship = Wife: >50K (345.0/117.0)
|   |   |   |   |   |   relationship = Own-child: <=50K (10.0/3.0)
|   |   |   |   |   |   relationship = Husband: >50K (2913.0/1015.0)
|   |   |   |   |   |   relationship = Not-in-family: <=50K (5.0/1.0)
|   |   |   |   |   |   relationship = Other-relative: <=50K (21.0/3.0)
|   |   |   |   |   |   relationship = Unmarried: >50K (0.0)
|   |   |   |   |   capital-loss > 625: <=50K (30.0/5.0)
|   |   |   |   capital-gain > 3103: <=50K (66.0/13.0)
|   |   capital-loss > 1762
|   |   |   capital-loss <= 1980: >50K (585.0/14.0)
|   |   |   capital-loss > 1980
|   |   |   |   capital-loss <= 2163: <=50K (63.0)
|   |   |   |   capital-loss > 2163
|   |   |   |   |   education-num <= 12: <=50K (58.0/21.0)
|   |   |   |   |   education-num > 12: >50K (62.0/2.0)
|   capital-gain > 5060: >50K (1209.0/14.0)
marital-status = Divorced
|   capital-gain <= 6849: <=50K (4323.0/348.0)
```

```
|   capital-gain > 6849: >50K (120.0/5.0)
marital-status = Never-married
|   capital-gain <= 7443: <=50K (10538.0/350.0)
|   capital-gain > 7443: >50K (145.0/4.0)
marital-status = Separated
|   capital-gain <= 4687: <=50K (998.0/46.0)
|   capital-gain > 4687: >50K (27.0/7.0)
marital-status = Widowed
|   capital-gain <= 4687: <=50K (964.0/62.0)
|   capital-gain > 4687: >50K (29.0/6.0)
marital-status = Married-spouse-absent: <=50K (418.0/34.0)
marital-status = Married-AF-spouse: <=50K (23.0/10.0)

Number of Leaves  :       28
Size of the tree :        46

Time taken to build model: 1 seconds

=== Evaluation on test set ===
Correctly Classified Instances       13945               85.652  %
Incorrectly Classified Instances      2336               14.348  %

=== Confusion Matrix ===
     a     b   <-- classified as
  2117  1729 |     a = >50K
   607 11828 |     b = <=50K
```

SUMMARY:

Now comparing the two final results from both ID3 and C4.5, we got somewhat similar accuracies of trees using the same subset of attributes (CFS) but with a greatly smaller pruned tree generated by the C4.5 algorithm.
And if we take a look back at the questions we asked before starting theses experiments, we see that most of them concentrated on the wrong attributes and that the most accurate predictors of income were the five attributes used in both ID3 and C4.5 above.

HEART DISEASE (STATLOG) DATASET:

DESCRIPTION OF THE DATASET:

Taken once again from UC Irvine Machine Learning Repository, this database contains 13 attributes (which have been extracted from a larger set of 75), and consists of 270 observations. It is considered somewhat simple dataset. The "goal" field refers to the presence of heart disease in the patient. There are no missing values.

Attribute Information:

- age
- sex
- chest pain type (4 values)
- resting blood pressure
- serum cholestoral in mg/dl
- fasting blood sugar > 120 mg/dl
- resting electrocardiographic results (values 0,1,2)
- maximum heart rate achieved
- exercise induced angina
- oldpeak = ST depression induced by exercise relative to rest
- the slope of the peak exercise ST segment
- number of major vessels (0-3) colored by flourosopy
- thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

Attributes types:

- Real: 1,4,5,8,10,12
- Ordered: 11
- Binary: 2,6,9
- Nominal: 7,3,13

Variable to be predicted: Absence (absent) or presence (present) of heart disease

OBJECTIVES:

We ask the following questions:

- What are the main indicators of presence of a heart disease?
- Does the patient's age have any influence on the prediction?
- Are the blood pressure and cholesterol level good predictors of heart disease?

PERFORMANCE METRIC & TRAINIG AND TESTING:

As before we will use classification accuracy as a performance metric of the model we build over the dataset. Most of the test will be using a 10-fold cross-validation.

PREPROCESSING OF THE DATA:

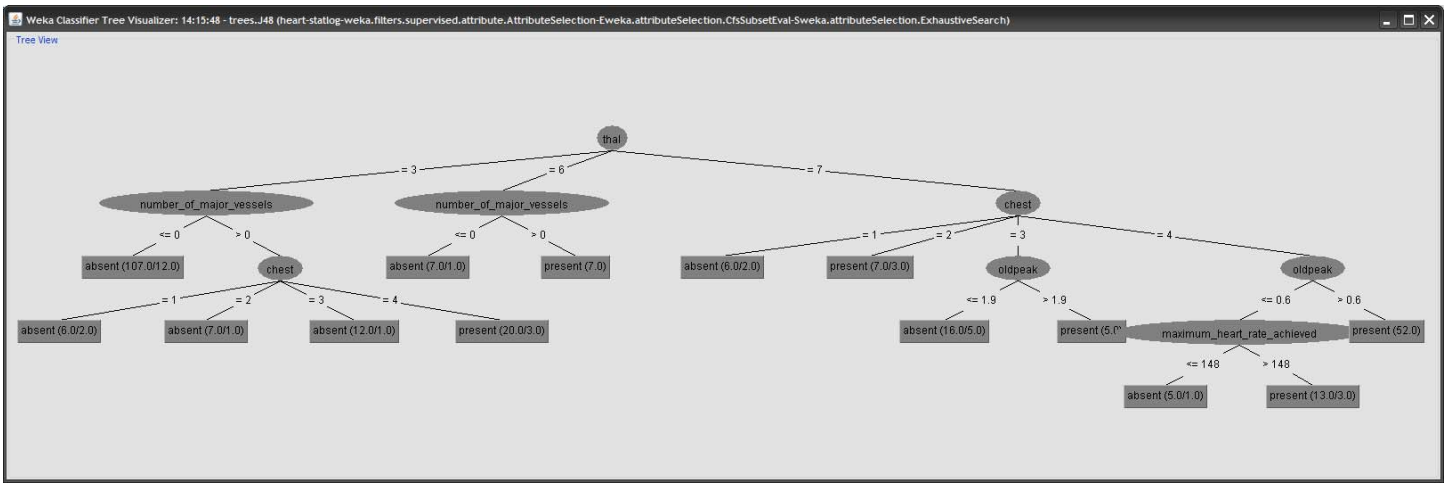
First step was of course to convert the data into .arff file format that the Weka system understands, which is a comma separated format with some description of the attributes at the top. Note that the data came originally as space separated. Also the class attribute values were relabeled from 1/ 2 to absent/present.
Besides that, the data doesn't really require any special preprocessing other than that necessary for making it compatible with ID3 decision trees (missing values and numeric attributes).

EXPERIMENTING:

Like before, we ran the *ZeroR* classifier to get a sense of the class distribution. Results are 55.5% (150 instances) correctly classified as absent and the other 120 present instances wrongly classified.
Then we try the *OneR* classifier. The attribute used is thal with a model accuracy of 76.3% on training instances and 71.1% on 10 fold cross-validation.

Next is the ID3. We first have to prepare the data. No missing values present, only numeric attribute have to be handled. A first try is to apply unsupervised discretization to numeric attributes with *findNumBins*=TRUE. Id3 algorithm over these results in a tree of 100% on training and 64% cross-validation. The tree is of medium length.
Another experiment was to try this time the supervised discretization over all attributes. Applying Id3 gave a much better result of 77.4% over a 10 fold cross-validation. Note that the original data's attribute of real are actually integers with almost no more than 50 distinct values. Therefore the discretization process picks upon that in a nice way.
Another experiment was to apply CFS to get a smaller subset of attributes then the supervised discretization. It gave a slightly improved accuracy of 79.2% with a subset of size 8.

Moving to C4.5, and applying it with default parameter values give a 93% over training and an 80% accuracy with cross-validation testing. The size of the tree is 43. Playing with the parameters by setting this time the reducedErrorPruning=TRUE, we get a tree size of 23 with accuracy 78.9%. To try something different, I applied CFS first, then ran J4.8 with minNumInstancesInLeaf=5 and c=0.20, I got the same 80.3% accuracy but with a tree size of 22. Note that we can get the tree size smaller, but that would affect the accuracy, and in this case missing on correctly identifying patients with heart disease.




```
SCHEME:       WEKA.CLASSIFIERS.TREES.J48 -C 0.2 -M 5
RELATION:     HEART-STATLOG-WEKA.FILTERS + SUPERVISED. ATTRIBUTESELECTION CFSSUBSET
INSTANCES:    270
ATTRIBUTES:   9
              SEX, CHEST,  MAXIMUM_HEART_RATE_ACHIEVED
              EXERCISE_INDUCED_ANGINA, OLDPEAK, SLOPE, NUMBER_OF_MAJOR_VESSELS, THAL, CLASS
TEST MODE:    10-FOLD CROSS-VALIDATION

J48 PRUNED TREE
------------------

THAL = 3
|   NUMBER_OF_MAJOR_VESSELS <= 0: ABSENT (107.0/12.0)
|   NUMBER_OF_MAJOR_VESSELS > 0
|   |   CHEST = 1: ABSENT (6.0/2.0)
|   |   CHEST = 2: ABSENT (7.0/1.0)
|   |   CHEST = 3: ABSENT (12.0/1.0)
|   |   CHEST = 4: PRESENT (20.0/3.0)
THAL = 6
|   NUMBER_OF_MAJOR_VESSELS <= 0: ABSENT (7.0/1.0)
|   NUMBER_OF_MAJOR_VESSELS > 0: PRESENT (7.0)
THAL = 7
|   CHEST = 1: ABSENT (6.0/2.0)
|   CHEST = 2: PRESENT (7.0/3.0)
|   CHEST = 3
|   |   OLDPEAK <= 1.9: ABSENT (16.0/5.0)
|   |   OLDPEAK > 1.9: PRESENT (5.0)
|   CHEST = 4
|   |   OLDPEAK <= 0.6
|   |   |   MAXIMUM_HEART_RATE_ACHIEVED <= 148: ABSENT (5.0/1.0)
|   |   |   MAXIMUM_HEART_RATE_ACHIEVED > 148: PRESENT (13.0/3.0)
|   |   OLDPEAK > 0.6: PRESENT (52.0)

NUMBER OF LEAVES  :       14
SIZE OF THE TREE :        22


TIME TAKEN TO BUILD MODEL: 0 SECONDS

=== SUMMARY ===
CORRECTLY CLASSIFIED INSTANCES         217               80.3704 %
INCORRECTLY CLASSIFIED INSTANCES        53               19.6296 %

=== CONFUSION MATRIX ===
   A   B   <-- CLASSIFIED AS
 127  23 |   A = ABSENT
  30  90 |   B = PRESENT
```

SUMMARY:

Once again, CFS attribute selection proved to be a solid technique resulting in a good model performance with less attributes. Also C4.5 showed to be superior to ID trees, giving similar results with much less deep trees.